José L. Ayala Braulio García-Cámara
Manuel Prieto Martino Ruggiero
Gilles Sicard (Eds.)

# Integrated Circuit and System Design

## Power and Timing Modeling, Optimization, and Simulation

LNCS 6951

21st International Workshop, PATMOS 2011
Madrid, Spain, September 2011
Proceedings

Springer

Lecture Notes in Computer Science 6951

*Commenced Publication in 1973*
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

José L. Ayala
Braulio García-Cámara
Manuel Prieto  Martino Ruggiero
Gilles Sicard (Eds.)

# Integrated Circuit and System Design

## Power and Timing Modeling, Optimization, and Simulation

21st International Workshop, PATMOS 2011
Madrid, Spain, September 26-29, 2011
Proceedings

Springer

Volume Editors

José L. Ayala
Universidad Complutense de Madrid, Spain
E-mail: jayala@fdi.ucm.es

Braulio García-Cámara
Universidad de Cantabria, Santander, Spain
E-mail: garciacb@unican.es

Manuel Prieto
Universidad Complutense de Madrid, Spain
E-mail: mpmatias@dacya.ucm.es

Martino Ruggiero
Università di Bologna, Italy
E-mail: martino.ruggiero@unibo.it

Gilles Sicard
Laboratoire TIMA, Grenoble, France
E-mail: gilles.sicard@imag.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-24153-6 e-ISBN 978-3-642-24154-3
DOI 10.1007/978-3-642-24154-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: Applied for

CR Subject Classification (1998): C.4, I.6, D.2, C.2, F.3, D.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues



*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

PATMOS 2011 was the 21st in a series of international workshops on Power and Timing Modeling, Optimization and Simulation. The PATMOS meeting has evolved during the years into a leading scientific event where industry and academia meet to discuss power and timing aspects in modern integrated circuit and system design. Both universities and companies are invited to participate.

The objective of this workshop is to provide a forum to discuss and investigate emerging challenges in methodologies and tools for the design of upcoming generations of integrated circuits and systems, including reconfigurable hardware such as FPGAs. The technical program focuses on timing, performance and power consumption as well as architectural aspects with particular emphasis on modeling, design, characterization, analysis and optimization.

September 2011

José L. Ayala
Manuel Prieto

# Organization

PATMOS 2011 was organized by the department of Computer Architecture in Universidad Complutense de Madrid.

## Organizaing Committee

### General Chairs

| | |
|---|---|
| José L. Ayala | UCM, Spain |
| Manuel Prieto | UCM, Spain |

### Program Chairs

| | |
|---|---|
| Gilles Sicard | TIMA Lab, France |
| Martino Ruggiero | University of Bologna, Italy |

### Special Session Chairs

| | |
|---|---|
| Luis Piñuel | UCM, Spain |
| Edith Beigne | CEA-LETI, France |

### Local Arrangements Chairs

| | |
|---|---|
| Katzalin Olcoz | UCM, Spain |
| Theocharis Theocharides | University of Cyprus, Cyprus |

### Publication Chair

| | |
|---|---|
| Braulio García-Cámara | University of Cantabria, Spain |

### Finance Chair

| | |
|---|---|
| Fernando Rincón | UCLM, Spain |

### Institutional Relations

| | |
|---|---|
| David Atienza | EPFL, Switzerland |

## Steering Committee

| | | |
|---|---|---|
| Antonio J. Acosta | Enrico Macii | Christian Piguet |
| Nadine Azemard | Philipe Maurien | Dimitrios Soudris |
| Joan Figueras | Jose Monteiro | Diederik Verkest |
| Reiner Hartenstein | Wolfgang Nebel | Roberto Zafalon |
| Jorge Juan-Chico | Vassilis Paliouras | |

## Referees

A. Alvandpour
K. Amin
A. Arapoyanni
E. K. Ardestani
D. Atienza
J. L. Ayala
N. Azemard
P. Beerel
E. Beigne
D. Bertozzi
O. Billoint
F. Blisson
F. Campi
N. Chang
D. Chang
X. Chen
A. K. Coskun
S. Dabideen
R. Eilers
J. Figueras
E. G. Friedman
A. García-Ortiz
L. Guerra E. Silva
D. Guerrero
C. Goutis
E. Grass
O. Gustafsson
M. Guthaus
J. L. Güntzel
C. Ha
C. Hankendi
D. Helms
C. Hernandez
K. Hylla
S. Hu
R. Jakushokas
J. Juan
N. Julien
D. Kagaris
P. Knocke
A. Korotaeva
S. Kose
M. Krstic
K. Lee
V. Liberali
P. Maurine
S. Medardoni
J. Meng
M. Metzdorf
H. Michail
J. Monteiro
V. Moshnyaga
J. M. Moya
T. Murgan
M. Najibi
W. Nebel
D. Nikolos
A. Nuñez
V. G. Oklobdzija
M. Osama
V. Paliouras
G. Panic
D. Pandini
A. Papanikolaou
R. Patel
K. Pekmestzi
C. Piguet
M. Poncino
A. Reimer
R. Reis
S. Rosinger
M. Ruggiero
L. G. Salem
D. Sciuto
G. Sicard
D. Soudris
R. Van Leuken
I. Vaisband
J. I. Villar De Ossorno
M. Wahba
R. Wilson
A. Xie
E. Yahya
Z. Ye
H. Zakaria

## Sponsoring Institutions

Universidad Complutense de Madrid (UCM)
IEEE Council on Electronic Design Automation (CEDA)

# Table of Contents

# A Quick Method for Energy Optimized Gate Sizing of Digital Circuits

Mustafa Aktan[1], Dursun Baran[2], and Vojin G. Oklobdzija[1]

[1] Department of Electrical and Computer Eng., New Mexico State University, NM 88003
[2] Department of Electrical Eng., University of Texas at Dallas, TX 75080
{mustafa,dursun,vojin}@acsel-lab.com
http://www.acsel-lab.com

**Abstract.** Exploration of energy & delay trade-offs requires a sizing solution for minimal energy under operating delay and output load constraints. In this work, a simple method called Constant Stage Effort Ratio (CSER) is proposed for minimal energy solution of digital circuits with a given target delay. The proposed method has a linear run-time dependence on the number of logic gates that is exponential for the optimal solution. As sample cases, the proposed algorithm is applied to parallel VLSI adders with varying bit-widths at 65nm CMOS technology. CSER sizing algorithm provides more than 300x run-time improvement compared to energy optimal solution with a worst case difference of 10% in energy for a 128-bits Kogge-Stone adder.

**Keywords:** Circuit Sizing, Energy-Efficient Design, Energy-Delay Estimation, Switching Activity, VLSI.

## 1 Introduction

GATE sizing is a common technique used to explore energy & delay trade-offs of digital architectures. Using an RC delay model makes the problem a convex optimization problem [1]. For a convex optimization problem the local optimum is the global optimum, hence the problem can be solved optimally. Sapatnekar et al. [2] gives the exact solution of the gate sizing problem using convex optimization techniques.

Instead of using a general purpose convex optimizer, which becomes slow for large circuits, [3] makes use of the structure of the problem and gives an efficient solution using Lagrangian relaxation. Problem size grows linearly with circuit size. The method is fast but suffers from parameter initialization problems [4].

Technology scaling increases the integration capacity of the current designs. As the number of logic gates integrated in a single IC increases, the complexity of the sizing algorithm also increases. In [5] a sizing method based on convex optimization is proposed that can be applied to large scale circuits. Since the delay models constitute a certain modeling error there is no need to find the exact optimum with that model as it is time consuming. The optimizer stops whenever the solution is within a certain error margin which is equal to delay modeling error.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 1–10, 2011.

Logical Effort (LE) [6] intends to give the engineer an intuition on selecting among different circuit topologies from the maximum performance point of view in a back of the envelope manner. LE sizes for minimum delay under input-output load constraints without caring for energy. Furthermore, for multipath circuits LE requires the calculation of branching load which depends on the sizes. Hence, iteration is required which makes LE no more a back of the envelope solution.

This work is an attempt to develop an LE like sizing method for the problem: minimize energy for a given delay constraint. Based on the effort distribution property of a single path circuit (chain of gates) at minimum energy point, a method is developed that quickly sizes multi-path CMOS circuits. The sizing solutions provided by the method are within an acceptable margin as the delay and energy modeling errors are concerned. The paper is organized as follows: Section 2 is about the delay and energy models used in the optimization problem and estimation of sizing solutions obtained. Section 3 gives the definition and proposed solution of the delay constrained minimal energy optimization problem. The proposed method is compared in terms of accuracy and runtime to the convex optimization based optimal solution. Results are presented in Section 4. Section 5 concludes the work.

## 2 Energy-Delay Estimation

In order to generate energy-delay characteristic of each digital circuit, fast and accurate energy and delay estimations are required. Logical effort [6], [7] model can provide accurate delay estimation with a given sizing solution. In addition, Logical effort sizes the circuit for optimum delay under the input/output constraints. Similarly, an LE like model for energy estimation is presented in [7] by Oklobdzija.

### 2.1 Delay Model

The delay of driver logic can be estimated using Logical Effort [6] delay model as follows;

$$d = g \cdot \frac{C_{load}}{C_{in}} + p \tag{1}$$

where $d$ is the normalized delay, $g$ is a constant and called the logical effort of the gate which is a measure of the strength of the gate relative to the inverter of same input capacitance, $C_{in}$ is the input gate capacitance which is a measure of the strength (size) of the gate, and $C_{load}$ is the fanout load capacitance. The actual delay of the logic gate is $d \bullet \tau$ where $\tau$ is the unit sized inverter delay.

### 2.2 Energy Model

Similar to delay estimation, energy consumption of the driver logic can be estimated using the models presented in [7].

$$E = \left\{ E_p \cdot C_{in} + E_g \cdot C_{load} \right\} \alpha_{SW} + E_{lk} \cdot C_{in} \tag{2}$$

where $E_p$*(fJ/fF)* is the energy factor for the internal parasitic of the driver gate, $C_{in}$ is the input capacitance of the driver gate, $E_g$*(fJ/fF)* is the energy factor for the internal parasitic of the driver gate, $C_{load}$ is the output load capacitance, $\alpha_{Sw}$ is the switching activity rate at the output node and $E_{lk}$ is the leakage energy consumption.

Parameters $g$, $p$, $E_g$, $E_p$, and $E_{lk}$ are technology dependent that are extracted from the gate characterization. $C_{in}$ depends on the size of the driving gate and $C_{load}$ depends on the size of the driven gates. Switching activity rate ($\alpha_{Sw}$) depends on the activity rate at the inputs. Activity rate can either be found by simulation or probabilistic calculations [8].

## 3 Circuit Sizing Techniques

The objective function of the circuit sizing can be modified to find an optimal solution for a target parameter under some constraints. For example, the circuit sizing for minimal delay provides a solution for optimum delay subject to power/area budget. Similarly, the circuit sizing for minimal power/area provides a solution for optimum power/area subject to maximum delay target. In this section, the circuit sizing techniques for minimal delay and power/area are considered.

### 3.1 Circuit Sizing for Minimum Delay

Logical effort provides an optimum delay solution subject to output and input loads. In Fig. 1, a circuit block composed of $N$-stages drives an output load of $C_{\text{out}}$. For each gate $i$ in Fig. 1, the logical effort is $g_i$, and parasitic delay is $p_i$.

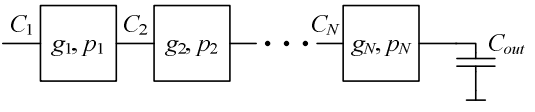



**Fig. 1.** A Circuit Block with $N$-Stages and a driving stage

Using logical effort delay model, the delay of the path given in Fig. 1 is:

$$D = \sum_{i=1}^{N} f_i + \sum_{i=1}^{N} p_i \tag{3}$$

where $f_i = g_i \frac{C_{i+1}}{C_i}$. Since the parasitic delays ($p_i$) in equation (3) are constant (hence their sum will be constant unless the structure of the path is changed) the total effort

$$F = \sum_{i=1}^{N} f_i \tag{4}$$

can be considered as the delay of the circuit. The minimum value of this delay is found using the logical effort method as

$$F_{\min} = N(GH)^{\frac{1}{N}} \tag{5}$$

where $G = \prod_{i=1}^{N} g_i$ and $H = \frac{C_{out}}{C_{in}}$. Note that the total effort is equally distributed among stages $f_i = f_{opt} = (GH)^{\frac{1}{N}}$ for $i = 1, 2, \ldots, N$.

### 3.2 Circuit Sizing for Minimum Energy

When sizing for minimum energy the objective of the sizing problem is changed to minimize weighted sum of capacitances ($\Sigma k_i C_i$) subject to delay and output load constraints. The weights ($k_i$) are mainly determined by switching activity. The problem is a constrained minimization problem which can be expresssed for the chain of gates of Fig. 1 as;

$$\text{Minimize } \sum_{i=1}^{N} k_i C_i$$

$$\text{Subject to } \sum_{i=1}^{N} f_i = F \ . \tag{6}$$

The problem in (6) can be solved optimally using convex optimization techniques. However, such a solution does not give insight to the engineer. Nevertheless, it is still possible to gather clues by analyzing the optimal solution and develop simple methods.

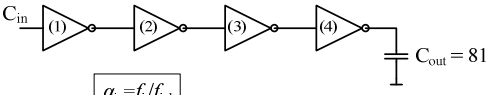
As we already know from Logical Effort sizing theory, for minimum delay (under a given input & output capacitance constraint) stage efforts are equal. Sizing for minimum energy under the delay constraint $F$ requires the sum of the stage efforts ($f_i$) be equal to $F$ ($\sum f_i = F$). Optimization for minimum energy basically re-distributes efforts so that energy is minimized. Fig. 2 shows the effort distribution at energy minimized design point for an inverter chain at various delay targets. The effort distributions are obtained by optimally solving (6). A relationship among stage efforts where each stage effort is expressed in terms of the previous stage's effort multiplied with a coefficient is

$$f_i = \alpha_i f_{i-1} \tag{7}$$

where $\alpha_i$ is called the stage effort ratio. We see that at the delay constrained minimal energy design point, efforts have the following property;

$$f_1 < f_2 < \ldots < f_N \ . \tag{8}$$

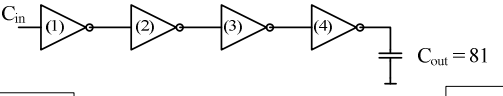
Note that the effort distribution is different than the LE design point distribution where all efforts are equal ($f_1 = f_2 = \ldots = f_N$). Unlike the equal distribution of LE, for which the stage efforts are found by simply dividing the total effort to the number of stages, (8) does not explicitly state the values of the efforts. Nevertheless, it gives a relationship between efforts: At minimum energy design point, stage efforts should increase from inputs to outputs.




$\alpha_i = f_i / f_{i-1}$

| $i$ | $f_i$ | $\alpha_i$ | $f_i$ | $\alpha_i$ | $f_i$ | $\alpha_i$ | $f_i$ | $\alpha_i$ | $f_i$ | $\alpha_i$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.9 | - | 0.8 | - | 0.8 | - | 0.7 | - | 0.6 | - |
| 2 | 1.6 | 1.9 | 1.5 | 1.8 | 1.3 | 1.8 | 1.1 | 1.7 | 0.9 | 1.6 |
| 3 | 2.9 | 1.8 | 2.5 | 1.7 | 2.1 | 1.6 | 1.7 | 1.5 | 1.1 | 1.3 |
| 4 | 6.6 | 2.3 | 5.1 | 2.0 | 3.8 | 1.8 | 2.5 | 1.5 | 1.4 | 1.3 |
| $\Sigma f_i$ | 12.0 | | 10.0 | | 8.0 | | 6.0 | | 4.0 | |

**Fig. 2.** Effort distribution at energy minimized solution




$f_1 < f_2 < \ldots < f_N$

$C_i = C_{i+1} / f_i$

| Effort distribution $f_1$ | $f_2$ | $f_3$ | $f_4$ | Delay $[\Sigma f_i]$ | $C_{in}$ | Energy $[\Sigma C_i]$ |
|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 12 | 1 | 40 |
| 1 | 3 | 2 | 6 | 12 | 2.3 | 25 |
| 1 | 2 | 3 | 6 | 12 | 2.3 | 22.5 |
| 6 | 3 | 2 | 1 | 12 | 2.3 | 137 |
| 1.5 | 1.5 | 3 | 6 | 12 | 2 | 23 |
| 0.9 | 1.6 | 2.9 | 6.6 | 12 | 3 | 22 |

Optimal solution

**Fig. 3.** Effort distribution versus total energy for an inverter chain

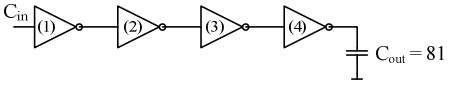
An example showing the effect of effort distributions on energy for a chain of inverters is given in Fig. 3. Note that the total effort is equal for all distributions. The last row is the effort distribution of the energy optimal solution. Note that the energy optimal distribution obeys equation (8). Rows 2, 3,and 4 distribute the same effort values but to different gates. The resulting energy ranges from 22.5 to 137 minimum sized inverter gate capacitance. The one that has least energy (row 3) obeys (8).

### 3.3 Constant Stage Effort Ratio (CSER) Method

We have shown that at the minimal energy design point the effort distribution is different than LE effort distribution. In LE all efforts are equal, as a consequence gate sizes grow geometrically to the outputs. At the minimal energy point, on the other hand, efforts do grow geometrically to the outputs. Size growth rate in LE is constant and equal to the effort value of each gate. However, at the minimal energy point the effort growth rate, what we call effort ratio, is not constant.

As the example of Fig. 2 has shown for each stage the effort ratio $\boldsymbol{\alpha_i}$ is different. However, for the sake of simplicity of the sizing method we can assume a constant

ratio $\alpha_i = \alpha$ between successive stages. In Fig. 4, sizing of the inverter chain is shown for different values of constant $\alpha$. The last row shows the effort distribution at the energy optimal design point which is obtained by optimally solving (6). The effort distribution with a constant effort ratio of 2 yields a very close result to the energy optimal distribution in terms of energy (less than 1% difference) at the same delay. Energy is measured in terms of total gate capacitance.




$f_2/f_1 = f_3/f_2 = f_4/f_3 = \alpha$ $\quad \Sigma f_i = 12$

| Effort ratio ($\alpha$) | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $C_{in}$ | Energy [$\Sigma C_i$] | |
|---|---|---|---|---|---|---|---|
| 1 | 3.0 | 3.0 | 3.0 | 3.0 | 1.0 | 40.0 | |
| 1.2 | 2.2 | 2.7 | 3.2 | 3.9 | 1.1 | 31.0 | |
| 1.6 | 1.3 | 2.1 | 3.3 | 5.3 | 1.7 | 23.8 | |
| 2 | 0.8 | 1.6 | 3.2 | 6.4 | 3.1 | 22.2 | CSER solution |
| 2.4 | 0.5 | 1.3 | 3.0 | 7.2 | 5.7 | 23.6 | |
| 2.8 | 0.4 | 1.0 | 2.8 | 7.8 | 1.0 | 28.0 | |
| - | 0.9 | 1.6 | 2.9 | 6.6 | 3.0 | 22.1 | Optimal solution |

**Fig. 4.** CSER effort distribution

The calculation of the gate efforts given a total effort $F$ and effort ratio $\alpha$ is done as follows. Assuming a constant effort ratio between succesive stages we have

$$\frac{f_2}{f_1} = \frac{f_3}{f_2} = \cdots = \frac{f_k}{f_{k-1}} = \cdots = \frac{f_N}{f_{N-1}} = \alpha \,. \tag{9}$$

Using equation (9) in equation (4) we get;

$$f_1 + \alpha f_1 + \cdots + \alpha^{k-2} f_1 + \alpha^{k-2} f_1 + \cdots + \alpha^{N-1} f_1 = F \,. \tag{10}$$

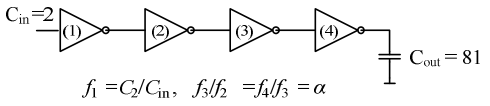
Solving for $f_1$ yields;

$$f_1 = F\,(1-\alpha)\,/(1-\alpha^N). \tag{11}$$

After finding $f_1$ the rest of the stage efforts are found by applying (7) from inputs to outputs.

In the example of Fig. 4, the input capacitance of the circuit is larger than the LE solution. Fig. 5 shows the CSER solution in case of fixed input capacitance. Stage efforts are calculated in the same manner using (7) and (11) except for the first stage. Since the input capacitance is fixed it is not assigned an effort but the effort is calculated from its load capacitance. This results in different total delay than target

delay as shown under the delay column for each solution in Fig. 4. The best CSER solution (α=1.7) is the one that has least energy meeting the delay target.




| Effort ratio (α) | $f_1$ | $f_2$ | $f_3$ | $f_4$ | Delay [$\Sigma f_i$] | Energy [$\Sigma C_i$] | |
|---|---|---|---|---|---|---|---|
| 1 | 1.5 | 3 | 3 | 3 | 10.5 | 41.0 | |
| 1.3 | 1.1 | 2.5 | 3.3 | 4.3 | 11.2 | 29.1 | CSER solution |
| 1.7 | 1.1 | 2.0 | 3.3 | 5.6 | 12.0 | 23.0 | |
| 2.1 | 1.3 | 1.5 | 3.2 | 6.6 | 12.6 | 20.7 | |
| 2.3 | 1.4 | 1.3 | 3.1 | 7.0 | 12.8 | 20.1 | |
| - | 1.3 | 1.8 | 2.9 | 6.0 | 12.0 | 22.7 | Optimal solution |

**Fig. 5.** CSER effort distribution in the case of fixed input capacitance

The examples in Fig. 4 and Fig. 5 show that it is possible to obtain different sizing solutions by applying different effort ratios. The goal is to find the solution with minimal energy satisfying the delay target. The sizing problem is converted to a search for a constant effort ratio α that yields minimum energy. The range for effort ratio (α) to be searched depends on the number of stages in the circuit. Experimental results have shown that for circuits with 4 or more stages, the optimal value of α is between 1 and 2.

### 3.4 Design Example

A more complicated design example with branching, multiple fan-in and fan-out is shown in Fig. 6. The same iterative method for finding the optimum α of the inverter chain example is used in this circuit. The results of the optimum energy solution and proposed sizing algorithm are given in Table 1.

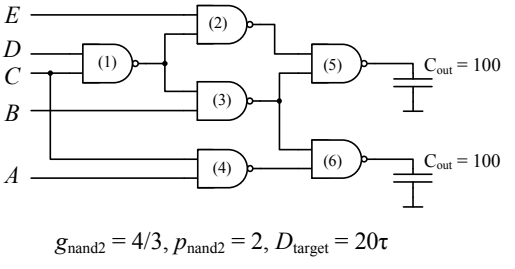



**Fig. 6.** Design example: ISCAS85 benchmark circuit C17

**Table 1.** Search steps for optimum α (C17 benchmark circuit)

| α | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | Delay [τ] | Energy [$\Sigma C_i$] | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.7 | 4.7 | 4.7 | 11.3 | 4.7 | 4.7 | 20 | 27.6 | |
| 1.4 | 3.2 | 4.5 | 4.5 | 9.7 | 6.3 | 6.3 | 20 | 21.6 | CSER solution |
| 1.8 | 2.3 | 4.2 | 4.2 | 8.5 | 7.5 | 7.5 | 20 | 19.5 | |
| 2.2 | 1.7 | 3.8 | 3.8 | 7.6 | 8.4 | 8.4 | 20 | 19.1 | |
| 2.6 | 1.4 | 3.5 | 3.5 | 6.9 | 9.1 | 9.1 | 20 | 19.5 | |
| 3.0 | 1.1 | 3.2 | 3.2 | 6.3 | 9.7 | 9.7 | 20 | 20.6 | Optimal solution |
| - | 1.7 | 4.0 | 4.0 | 7.7 | 8.3 | 8.3 | 20 | 19.1 | |

The effort distribution with a constant effort ratio of 2.2 yields a solution that consumes the lowest energy. The result of constant stage effort ratio algorithm is less than 1% different than the optimum energy sizing solution under the same delay target of 20τ. The energy consumption is estimated as the sum of capacitances. Note that the initial effort assigned to gate number 4 in Fig. 5 ($f_4$) is 3.8 (8.4/2.2). This is a multi-path circuit and not all paths are critical. Gates on non-critical paths are relaxed to reduce energy consumption. The initial assigned effort for gate 4 results in lesser delay than necessary. Therefore, the effort of gate number 4 ($f_4$) is increased to 7.6 to satisfy the delay target with the lowest energy consumption. For multi-path circuits, after initial effort distribution, gates on non-critical paths of the circuit are downsized to reduce the energy consumption of the circuit under the given delay constraint.

## 4 Results

In this section we give comparative results on sizing adder circuits using the proposed method and convex optimization based (optimal) method. The energy and delay model parameters for gates used in the circuits are extracted using technology characterization. The gate characterization is done at 65nm CMOS [9] technology at typical process corner. The delay of each logic gate is characterized under worst case single input switching condition. The summary of technology characterization is given in Table 2. The wire capacitance per length given in Table 2 is taken from the interconnect report of ITRS (International Technology Roadmap for Semiconductors) [10].

CSER method is applied to parallel adders to size them for minimal energy under the input-output loading constraint. The adders analyzed are Kogge-Stone (KS) [11], Han-Carlson (HC) [12], and Kogge-Stone with Ling recurrence (KS-L) [13]. For accurate energy estimation the activity factors for all nodes in the adders are calculated using the formulae given in [8]. Wire lengths are determined based on the bit pitch of register file.

The minimal energy sizing solution of 64-bit adders are given in Fig. 7. The worst case delay difference at same energy for CSER compared to optimum is 5% for all adder implementations. This shows that CSER can successfully be used to explore architectural trade-offs of various designs of circuits.

**Table 2.** Technology Characterization Summary

| Technology | 65nm CMOS |
|---|---|
| Power Supply [V] | 1.1 |
| [a]$W_{min}$ [nm] | 100 |
| FO4 [ps] | 20.85 |
| [b]INV Input Cap. [fF] | 0.526 |
| Wire Cap. [fF/μm] | 0.19 |
| Bit Pitch [μm] | 2 |
| Temperature [$^{o}$C] | 25 |

[a] Width of minimum sized transistor
[b] Input Capacitance of minimum sized inverter

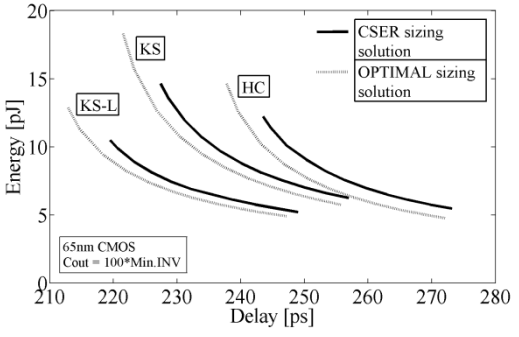



**Fig. 7.** Sizing performance of CSER compared to optimal solution tested on 64-bit adders

**Table 3.** Runtime comparison of CSER and energy optimal solution (65nm CMOS technology)

| Circuit | Gate # | $D_{opt}$ [*ps*] | $E_{opt}$ [*pJ*] | $E_{CSER}$ [*pJ*] | *Run Time Optimal* [*sec*] | *Run Time CSER* [*sec*] |
|---|---|---|---|---|---|---|
| 16-bitKS-L | 168 | 172 | 1.02 | 1.06 | 22.0 | 1.20 |
| 32-bitKS-L | 392 | 201 | 2.30 | 2.47 | 110 | 2.63 |
| 64-bitKS-L | 904 | 230 | 5.39 | 5.91 | 701 | 6.50 |
| 128-bitKS-L | 2056 | 263 | 13.7 | 15.2 | 5116 | 15.3 |

Table 3 provides a list of various length adders used to compare the runtime of CSER and optimal sizing algorithms. The size of the circuits range from 200 to 2000 gates. Both methods are executed on MATLAB. The runtime of the proposed technique has a linear dependence to number of gates. However, the runtime of optimal sizing algorithm increases exponentially with number of gates. The energy of the sizing solution generated by CSER is 10% off for the case of 128-bit KS2-L.

## 5 Conclusion

In this work, a simple gate sizing method for digital circuits is proposed. The method is able to size digital circuits for minimum energy under a delay constraint. To show

the effectiveness of the method in quick comparison of architectural trade-offs adder circuits are sized in 65nm CMOS technology. Sizing accuracy and runtime are evaluated for the proposed method. In comparison with the convex optimization based optimal method, proposed method has 300x runtime improvement for a circuit with 2000 gates with a sizing accuracy within 10% of optimal in terms of energy. The method is able to reflect architectural trade-offs in functional units enabling quick selection for the optimum architecture.

**Acknowledgements.** This work is supported by SRC Research Grant 2009-HJ-1836, Intel Corporation, and IBM Corporation.

## References

1. Fishburn, J., Dunlop, A.: TILOS: A Posynomial Programming Approach to Transistor Sizing. In: IEEE Int. Conf. Computer-Aided Design, Washington, DC, pp. 326–328 (1985)
2. Sapatnekar, S.S., Rao, V.B., Vaidya, P.M., Kang, S.M.: An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. IEEE Trans. Computer-Aided Design 12, 1621–1634 (1993)
3. Chen, C.-P., Chu, C.C.N., Wong, D.F.: Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation. IEEE Trans. Computer-Aided Design 18, 1014–1015 (1999)
4. Tennakoon, H., Sechen, C.: Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Pre-processing Step. In: Proc. IEEE/ACM Int. Conf. Comput. Aided Des., San Jose, CA, pp. 395–402 (November 2002)
5. Joshi, S., Boyd, S.: An Efficient Method for Large-Scale Gate Sizing. IEEE Trans. Circuits Syst. I 55(9), 2760–2773 (2008)
6. Sutherland, I.E., Sproull, R.F., Harris, D.: Logical Effort: Designing Fast CMOS Circuits. Morgan Kaufmann Publisher, San Francisco (1999)
7. Oklobdzija, V.G., Zeydel, B.R., Dao, H.Q., Mathew, S., Krishnamurthy, R.: Comparison of High-Performance VLSI Adders in Energy-Delay Space. IEEE Transaction on VLSI Systems 13(6), 754–758 (2005)
8. Baran, D., Aktan, M., Karimiyan, H., Oklobdzija, V.G.: Switching Activity Calculation of VLSI Adders. In: ASICON 2009, Changsha, China, October 20-23 (2009)
9. Predictive Technology Models, `http://www.eas.asu.edu/~ptm/`
10. International Technology Roadmap for Semiconductors, `http://www.itrs.net/Links/2007ITRS/2007Chapters/2007Interconnect.pdf`
11. Kogge, P.M., Stone, H.S.: A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. IEEE Trans. Computers C-22(8), 786–793 (1973)
12. Han, T., Carlson, D.A.: Fast Area-Efficient VLSI Adders. In: 8th IEEE Symposium on Computer Arithmetic, Como, Italy, pp. 49–56 (May 1987)
13. Ling, H.: High-Speed Binary Adder. IBM Journal of Research and Development 25(3), 156–166 (1981)

# Power Profiling-Guided Floorplanner for Thermal Optimization in 3D Multiprocessor Architectures

Ignacio Arnaldo, José L. Risco-Martín, José L. Ayala, and J. Ignacio Hidalgo

Universidad Complutense de Madrid, Spain
ignacioarnaldo@estumail.ucm.es, jlrisco@dacya.ucm.es,
jayala@fdi.ucm.es, hidalgo@dacya.ucm.es

**Abstract.** 3D integration has become one of the most promising techniques for the integration future multi-core processors, since it improves performance and reduces power consumption by decreasing global wire length. However, 3D integration causes serious thermal problems since the closer proximity of heat generating dies makes existing thermal hotspots more severe. Thermal-aware floorplanners can play an important role to improve the thermal profile, but they have failed in considering the dynamic power profiles of the applications. This paper proposes a novel thermal-aware floorplanner guided by the power profiling of a set of benchmarks that are representative of the application scope. The results show how our approach outperforms the thermal metrics as compared with the worst-case scenario usually considered in "traditional" thermal-aware floorplanners.

## 1 Introduction

In the last few years semiconductor industry has seen innumerable number of engineering advances that has permitted a logarithmic growth in the capability of integrated circuits (ICs).

Power density of the microprocessors is also increasing with every new process generation since feature size and frequency are scaling faster than the operating voltage [6]. As a result, there has been an increase in maximum chip temperatures because power density directly translates into heat. For example, Pentium 4 chips generate more heat than a kitchen hotplate. If a Pentium 4 chip is allowed to run without a proper cooling system, it catches fire [2]. Intels projections show that the heat generated by the processors will increase sharply in the coming years unless solutions to this problem can be found [1].

In the field of Multi-Processor Systems-on-Chip (MPSoCs), the 3D IC is gaining a lot of interest as a viable solution to help maintain the pace of system demands on scaling, performance, and functionality. The benefits include system-size reduction, performance enhancement due to shorter wire length, power reduction and the potential for hetero-integration. For complex systems with a large number of cores, only 3D technology is able to provide the required integration area while minimizing the inter-core communication delay.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 11–21, 2011.

Thermal aware floorplanning is finding an optimum floorplan by optimizing the cost function consisting of area, wire length, and temperature. The objective of the problem is to minimize the chip area, minimize the wire length, and minimize the maximum temperature of the chip. Thermal aware floorplanning can be used as one of the methods for decreasing the maximum temperature of the chip. Cooling of the blocks in a floorplan arises due to lateral spreading of heat through silicon blocks [26]. If a hot block is placed besides cooler blocks, lateral spreading of heat takes place. As a result, the temperature of the hot block is reduced.

A common limitation of the previous methods of 3D floorplanning is that they are focused on area and/or wire length minimization with or without thermal considerations. This can be a serious limitation as modern floorplanners often have to work with a fixed die size constraint, or with a fixed outline constraint in low-level design of hierarchical floorplanning flow [3]. However, all recently developed thermal-aware tools deploy temperature estimation techniques only on a single power profile representing power profiles of all inputs and all applications (e.g. using average or peak power profile). Different applications lead to different dynamic power profiles of the blocks. Most of the existing work use either average power or peak power per block of the applications for simulating temperature, without analyzing the impact of this assumption.

This work proposes for a first time an efficient thermal-aware 3D floorplanner for heterogeneous architectures of MPSoCs that uses as input the power traces obtained during an application power profiling phase. The design of this floorplanner is based on a genetic algorithm capable of obtaining optimal solutions, in a short time, for a large number of integrated processors and layers and with minimal overhead.

## 2 Related Work

Some initial works on thermal aware floorplanning [8] propose a combinatorial optimization to model our problem. However, the simplification of the considered floorplan and the lack of a real experimental framework motivated the further research on the area. Thermal placement for standard cell ASICs is a well researched area in the VLSI CAD community, where we can find works as [7].

In the area of floorplanning for microprocessor-based systems, some authors consider the problem at the microarchitectural level [26], showing that significant peak temperature reduction can be achieved by managing lateral heat spreading through floorplanning. Other works [17] use genetic algorithms to demonstrate how to decrease the peak temperature while generating floorplans with area comparable to that achieved by traditional techniques. [14] uses a simulated annealing algorithm and an interconnect model to achieve thermal optimization. These works have a major restriction since they do not consider multiple objective factors in the optimization problem, as opposed to our work. Other works [22] have tackled the problem of thermal-aware floorplanning with geometric programming but, in this case, the area of the chip is not considered constant.

Thermal-aware floorplanning for 3D stacked systems has also been investigated. Cong [10] proposed a thermal-driven floorplanning algorithm for 3D ICs, which is a natural extension of his previous work on 2D. In [15], Healy et al. implemented a multi-objective floorplanning algorithm for 2D and 3D ICs, combining linear programming and simulated annealing. Recent works as [11] also propose combinatorial techniques to tackle the problem of thermal-aware floorplanning in 3D multi-processor architectures.

In our work, we propose to use application profiling techniques to guide the floorplanner. A work by [23] shows that the power profile does not have major effect on the leakage power as long as the total power remains same. However, they do not consider the effect of power profile on temperature variation across different applications, especially the peak temperature of the blocks. Only a recent work [27] incorporates multiple power profiles in a thermal-aware floorplanner. However, this work is not devoted to MPSoC and could not be easily extended to 3D multi-processor stacks, where most traditional thermal-floorplanner fail to find an optimal solution.

## 3 Methodology

This work approaches the thermal-guided floorplanning problem for manycore heterogeneous architectures. The temperature of a given chip depends on physical factors such as the power dissipation of the processors, the size of the memories etc. but it also depends on the dynamic profile of the applications. One of our contributions is to consider energy profiles based on the simulation of real world applications. In fact, this problem is generally approached considering only the worst case scenario in terms of power dissipation.

### 3.1 Thermal Analysis

*Proposed architectures:* OVPsim [19] is a high level multiprocessor simulator for architectural exploration. In our case we are interested in the study of manycore heterogeneous architectures. The main elements composing our architectures are memories and SPARC, ARM CORTEX-A9 and POWERPC 440 9SF processors. We study two scenarios that differ from each other in the number and percentual distribution of cores. In the first architecture there are 30 cores with a small proportion of low-power processors: 20 SPARC, 5 CORTEX-A9 and 5 PPC440. The other platform is composed of a medium number of cores with an homogeneous distribution: 22 SPARC, 22 CORTEX-A9 and 22 PPC440 adding up a total of 66 cores. In both cases, there is a shared memory common to all the processors (used for the inter-processor communication) and a local memory for each of the cores. The size of the local memories must be small, as manycore architectures with big local memories are not feasible.

*Benchmarks:* We work with ParMiBench [20] which is composed of parallel versions of typical applications. We select 11 applications grouped into six different categories: Calculus, Network, Security, Office, Multimedia and Mixed. Therefore we have 6 different benchmarks corresponding to different kinds of applications

that will exhibit very different execution profiles. As a result the power dissipation profiles obtained are different from one benchmark to another. These benchmarks are simulated on bare machines. When an operating system is considered, the power consumption, and hence the temperature tends to be homogeneous due to the execution of the kernel. Therefore, we decide to avoid this overhead in our power profiles. The benchmarks need to be adapted to be run by OVPsim (mapping the compiled code into specific memory regions, replacing the system calls etc.). To obtain the profiling statistics for a given benchmark, we assign a parallel application and a shared memory region to each of the groups of processors working together. Figure 1 shows a task distribution example where 7 groups of six processors compute a Dijkstra shortest path algorithm and two groups of 12 processors compute a Patricia algorithm. The IDs in this table correspond to processor IDs. The processors in the same row form a group of processors that will execute a given parallel application. With these task distributions, we aim to simulate scenarios that could happen on a real platform.

| Application | SPARC id | CORTEX-A9 id | PPC440 id |
|---|---|---|---|
| pat1 | 0,1,2,3 | 22,23,24,25 | 44,45,46,47 |
| pat2 | 4,5,6,7 | 26,27,28,29 | 48,49,50,51 |
| dijk1 | 8,9 | 30,31 | 52,53 |
| dijk2 | 10,11 | 32,33 | 54,55 |
| dijk3 | 12,13 | 34,35 | 56,57 |
| dijk4 | 14,15 | 36,37 | 58,59 |
| dijk5 | 16,17 | 38,39 | 60,61 |
| dijk6 | 18,19 | 40,41 | 62,63 |
| dijk7 | 20,21 | 42,43 | 64,65 |

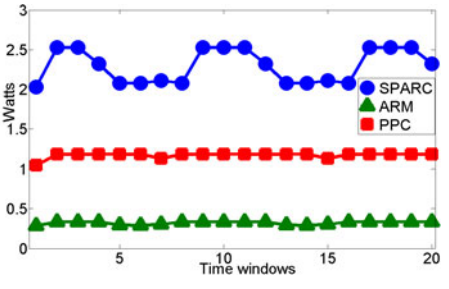



**Fig. 1.** Task distribution of the Network benchmark for the 66 core architecture and a common power consumption pattern caused by synchronization

To compute the power dissipated by each of the processors and memories of the studied architectures, we perform simulations of the 2 platforms and 6 benchmarks (12 simulations). We split the simulation of a given benchmark in time slices called windows and study the evolution of the dissipated power versus time for every element. For each of these windows we count the executed instructions and the idle cycles per processor, as well as the read and write accesses per memory (local and shared). The simulations have to be long enough to reach a stationary state. Therefore, we obtain at least the statistical data of 100 windows. We fix the window period to 128ms. This value is chosen to be long enough to reduce the impact on performance of the profiling phase, but short enough to capture the dynamic behavior with the required accuracy.

*Memories:* We fix the size of the local memories to $512KB$ and the size of the shared memory to $4MB$ and $8MB$ for the 30 and 66 cores platforms respectively. We consider both the local and shared memories to be direct mapped SRAM memories, with a block size 64 bytes and a transistor size of 45 nm. We obtain the energy consumption and area values with the CACTI software [16]. We approximate the energy per write access value with $E_w = E_r * 1.5$.

*Processors:* To obtain realistic power profiles of heterogeneous platforms, we have chosen three processors with a different computing power. In fact, we are specially interested in understanding the effect of synchronization and communication in the temperature of the chip. Figure 1 shows a typical power dissipation pattern of three different processors working together. We can see clearly how the activity of the SPARC core changes periodically over time, waiting for slower processors. We assume that the energy consumption of a given processor depends on its working frequency and its state. We consider two states: active or idle. To compute the power densities of the processors, we need their areas and a power consumption value for both the active and idle states. In [24] we find that the power consumption of the SPARC is 4W at 1.4GHz. In the case of the CORTEX-A9, we find that 0.4W is the estimated power dissipation working at 830 MHz while the PPC440 9SF dissipates 1.1W at 667MHz (see [4] and [18]). We approximate the power dissipated in the idle state with $P_{idle} = P_{active}/10$. We consider the following areas: $3.24mm^2$, $1.5mm^2$ and $6.2mm^2$ for the SPARC, CORTEX-A9 and PPC440 respectively (see [24], [4] and [18]).

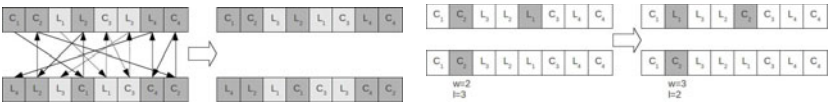
### 3.2 Multi-objective Genetic Algorithm

*Introduction to MOGA:* Most of the algorithms presented for the 3D thermal aware floorplanning problem are based on a Mixed Integer Linear Program (MILP) [15], [21], Simulated Annealing (SA) [10], [15] or Genetic Algorithm (GA) [28]. MILP has proven to be an efficient solution. However, when MILP is used for thermal aware floorplanning, the (linear) thermal model must be added to the topological relations and the resultant algorithm becomes too complex [13], specially as the problem size (number of cores and memories, in our case) increases. Regarding SA and GA, the main issue is based on the representation of the solution. Some common representations are polish notation [5], combined bucket array [10] and O-tree [28]. Most of these representations do not perform well, because they were initially developed to reduce area. In the thermal aware floorplanning problem, hottest elements must be placed as far as possible in the 3D IC. In this work, we have developed a straightforward Multi-Objective Genetic Algorithm (MOGA) based on NSGA-II [12], which tries to minimize maximum temperature and total wire length while fulfilling all the topological constraints. In this work, as opposed to traditional floorplanning problems in 2D, the area of the chip is not targeted as we consider a fixed die size. In order to apply MOGAs to our problem, a genetic representation of each individual has first to be found. Furthermore, an initial population has to be created, as well as defining a cost function to measure the fitness of each solution. For an overview of MOGAs the reader is referred to [9].

*Genetic representation and operators:* Every block $i$ in the model $B_i(i = 1, 2, \ldots, n)$ is characterized by a width $w_i$, a height $h_i$ and a length $l_i$ while the design volume has a maximum width $W$, maximum height $H$, and maximum length $L$. We define the vector $(x_i, y_i, z_i)$ as the geometrical location of block $B_i$, where $0 \leq x_i \leq L - l_i$, $0 \leq y_i \leq W - w_i$, $0 \leq z_i \leq L - h_i$. We use

$(x_i, y_i, z_i)$ to denote the left-bottom-back coordinate of block $B_i$ while we assume that the coordinate of left-bottom-back corner of the resultant IC is $(0, 0, 0)$.

In order to apply a MOGA correctly we must guarantee that all the chromosomes represent real and feasible solutions to the problem and ensure that the search space is covered in a continuous and optimal way. To this end, we use a permutation encoding [9], where every chromosome is a string of labels, that represents the block placement sequence. Figure 2 depicts the genetic operators used in our MOGA on a floorplanning problem. A chromosome in Figure 2.a is formed by 8 blocks: 4 cores $C_i (i = 1, 2, 3, 4)$ and 4 memories $L_i (i = 1, 2, 3, 4)$.



(a) Cycle crossover (b) Swap mutation or rotation

**Fig. 2.** MOGA operators: Cycle crossover and two mutation operators (swap or rotate)

In every cycle of the optimization process (called generation) two chromosomes are selected by tournament. To this end, we pick two chromosomes at random from the whole population and we select the best of these. This task is repeated twice in order to obtain two chromosomes (called parents, see Figure 2). Next, as Figure 3.2 depicts, we apply the cycle crossover: starting with the first allele of chromosome A $(C_1)$, we look at the allele at the same position in chromosome B. Next, we go to the position with the same allele in A, and add this allele to the cycle. Then, we repeat the previous step until we arrive at the first allele of A. Finally, we put the alleles of the cycle in the first child on the positions they have in the first parent, and take next cycle from second parent. Finally, mutation can be executed in two different ways, both with the same probability (see Figure 3.2). As a result, some blocks are chosen and swapped, and others are rotated 90 degrees.

*Fitness function:* Each chromosome represents the order in which blocks are being placed in the design area. Every block $B_i$ is placed taking into account all the topological constraints, the total wire length, and the maximum temperature in the chip with respect to all the previously placed blocks $B_j : j < i$. In order to place a block $i$, we take the best point $(x_i, y_i, z_i)$ in the remaining free positions. To select the best point we stablish a dominance relation taking into account the $m$ following objectives in our multi-objective evaluation.

The first objective is determined by the topological relations among placed blocks. It represents the number of topological constraints violated (no overlapping between placed blocks and current area less or equal than maximum area). The second objective is the wire length. The wire length is approximated as the Manhattan distance between interconnected blocks. The following objectives $(3, 4, \ldots, m)$ are a measure of the thermal impact, each one based on a

power consumption based on our profiles. To compute the thermal impact for every power consumption we cannot use an accurate thermal model, which includes non-linear and differential equations. In a classical thermal model, the temperature of a unitary cell of the chip, depends not only on the power density dissipated by the cell, but also on the power density of its neighbors. The first factor refers to the increase of the thermal energy due to the activity of the element, while the second one is related to the diffusion process of heat [25]. Taking this into account, we use the power density of each block as an approximation of its temperature in the steady state. This is a valid approximation because the main term of the temperature of a cell is given by the power dissipated in the cell, the contribution of its neighbors does not change significantly the thermal behavior. Thus, our remaining objectives can be formulated as:

$$J_{k\in 3..m} = \sum_{i<j\in 1..n} (dp_i^{k-2} * dp_j^{k-2})/(d_{ij}) \quad (1)$$

where $dp_i^p$ is the power density of block $i$ for power consumption $p$, and $d_{ij}$ is the Euclidean distance between blocks $i$ and $j$.

In our case, we must obtain up to 600 different power consumptions (100 time windows $\times$ 6 applications). Obviously, 600 objectives is too high for a MOGA, since it will converge too slow. However, we discuss how to reduce this number of objectives in the next section.

## 4 Experimental Setup

The experimental work will analyze the thermal optimization achieved by the floorplanner in the two different scenarios presented in section 3.1 (30 and 66 cores architectures). The floorplanner will place the processors, the local and shared memories of the 3D manycore platforms in 3 and 4 layers respectively.

We compare four different floorplans obtained with our algorithm. We propose as baseline a performance optimized floorplan targeting only the wire length (BAS). To obtain the thermally optimized floorplans, it is not possible to take directly into account all the data retrieved from our simulations. In fact, if the power dissipation of every element for each benchmark and time window was considered, the floorplanner would target 600 objectives and would hardly converge. Therefore, to obtain the other three floorplans, we consider different power metrics computed with the data retrieved from the simulation of 100 time windows for the 6 different execution profiles. The first of the remaining configurations is obtained considering the mean power dissipation for each element and profile. Therefore, the floorplanner looks for feasible solutions that minimize six thermal objectives (one per profile) and the wire length. In a similar way, we obtain another configuration taking into account the highest power consumption per element and profile (WOR); once again the floorplan obtained is the result of an optimization of eight objectives. Finally, a weighted sum of the power consumptions of the different profiles is considered for each element (WSM).

In this last case, only three objectives are targeted: feasibility, a thermal objective and the wire length. We run the multi-objective genetic algorithm with a population of 100 individuals and 250 generations. These parameters are fixed according to previous research. The crossover probability $p_c$ is fixed to 0.90 and the mutation probability $p_m$ to $1/\#blocks$ (see [12]). We consider a fixed area equal to the total sum of the areas of the different elements. This value is increased in a 15% as no solutions are found with less area. The configurations obtained are chosen among a front of non-dominated solutions returned by the floorplanner. Finding the best solution of the non-dominated front is out of the scope of this paper and is postponed to a future work.

## 5 Results

In this section, we compare the thermal profiles of the different configurations explained in the previous section. To evaluate them, we propose two experiments. In the first one, the thermal simulator evaluates each of the floorplans with the highest power dissipation for every element. This case corresponds to the worst scenario. The metrics considered for the analysis of the experimental results are the mean and maximum temperature of the chip and the maximum thermal gradient. These metrics are usually found in thermal-related analysis. In Figure 3 we present the thermal profiles of the four different configurations for each of the six benchmarks. The results show that our power profiling-guided floorplanner produces thermally optimized configurations. We can see a better overall thermal behaviour for the WSM in the case of the 30 architecture. On the other hand, the AVG configuration outperforms the others in the 66 cores scenario. The hotspots found in the performance optimized floorplans proposed as baseline justify the thermal optimization presented in this paper. For example, we can appreciate that the dramatic peak temperature of 423.16K found in the baseline for the second benchmark is reduced to 380.46K in the WSM configuration in the 30 core platform. We illustrate this example in Figure 4 where we show the thermal maps of the hottest layer of the BAS and WSM configurations. In the baseline configuration the floorplanner tends to place the processors near their local memories while in the other the hottest elements (the SPARC cores) are separated as much as possible, generally placed in the borders of the chip. Vertical heat spread is also taken into account, hence the floorplanner avoids placing cores above the others. In both cases the shared memory is placed in the second layer to minimize the wire length.

In the second experiment, we evaluate the floorplans in a more realistic way, the thermal behaviour of the different configurations is simulated for 25 time windows with the power dissipation values obtained from our execution profiles. Three metrics are considered in this case to compare the different configurations. The mean of the maximum temperatures of the different time windows is given as well as its standard deviation. This metric is a good indicator of the existence of hotspots in the studied chip. We also compute the overall mean temperature of the chip and the mean of the maximum thermal

| | Tmax | | | | | |
|---|---|---|---|---|---|---|
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | 386.75 | 423.16 | 382.27 | 365.89 | 389.83 | 427.97 |
| AVG | 355.39 | 393.83 | 366.30 | 348.59 | 357.19 | 396.65 |
| WOR | 354.39 | 382.31 | 351.64 | 340.19 | 352.63 | 385.46 |
| WSM | 358.46 | 380.46 | 348.41 | 339.93 | 359.86 | 384.21 |
| | Tmean | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | 340.45 | 351.79 | 336.71 | 331.48 | 340.69 | 354.29 |
| AVG | 334.78 | 346.71 | 333.10 | 327.74 | 335.11 | 349.10 |
| WOR | 334.64 | 346.60 | 332.60 | 326.79 | 334.99 | 348.90 |
| WSM | 334.55 | 346.41 | 332.17 | 326.84 | 334.73 | 348.54 |
| | Tgrad | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | 82.86 | 116.65 | 77.69 | 61.65 | 85.56 | 119.77 |
| AVG | 42.65 | 80.85 | 56.79 | 40.58 | 44.59 | 82.85 |
| WOR | 41.86 | 69.49 | 43.82 | 34.05 | 40.33 | 71.48 |
| WSM | 45.28 | 66.97 | 39.45 | 32.58 | 46.88 | 69.97 |

| | Tmax | | | | | |
|---|---|---|---|---|---|---|
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | 362.69 | 428.45 | 422.70 | 416.88 | 372.84 | 405.20 |
| AVG | 347.57 | 389.76 | 386.13 | 382.59 | 349.32 | 381.53 |
| WOR | 345.27 | 393.04 | 389.61 | 384.88 | 356.73 | 382.20 |
| WSM | 349.35 | 396.22 | 390.26 | 386.22 | 352.87 | 384.17 |
| | Tmean | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | 334.33 | 353.67 | 348.45 | 346.53 | 334.64 | 342.91 |
| AVG | 329.72 | 348.80 | 343.74 | 341.69 | 330.85 | 339.70 |
| WOR | 329.95 | 349.29 | 344.25 | 342.22 | 330.96 | 339.78 |
| WSM | 330.26 | 349.45 | 344.05 | 342.32 | 331.25 | 340.09 |
| | Tgrad | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | 56.61 | 121.58 | 117.44 | 112.47 | 67.25 | 99.95 |
| AVG | 40.59 | 81.71 | 78.93 | 76.15 | 43.53 | 75.43 |
| WOR | 35.88 | 83.42 | 81.72 | 76.34 | 48.48 | 74.81 |
| WSM | 40.50 | 86.69 | 81.83 | 78.80 | 45.51 | 76.46 |

**Fig. 3.** Thermal metrics of the 30 (left) and 66 (right) cores platforms

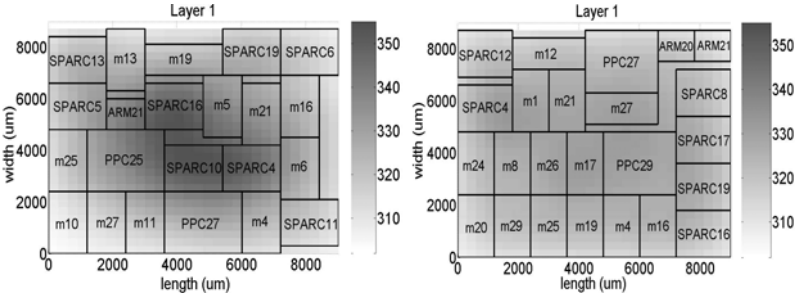



**Fig. 4.** Thermal map of the hottest layer of the 30 cores BAS(left) and WSM(right) configurations

gradients as well as their respective standard deviations. In Figure 6 we present these metrics for the six studied benchmarks and the four different configurations. Comparing to the baseline, we can see that in all of the cases our floorplanner reduces the peak and mean temperatures and the thermal gradient. Therefore, not only the temperature of the chip is reduced but it is more evenly distributed. In the 30 cores scenario, the WOR configuration has the best thermal behavior for peak temperatures while the WSM has the lowest mean temperature. In the second scenario, we can see that the AVG configuration clearly outperforms the others. To illustrate this idea, we show in Figure 5 the evolution of the peak temperature of the different 66 cores configurations during the 25 simulated windows for the fifth benchmark. We can see that with an

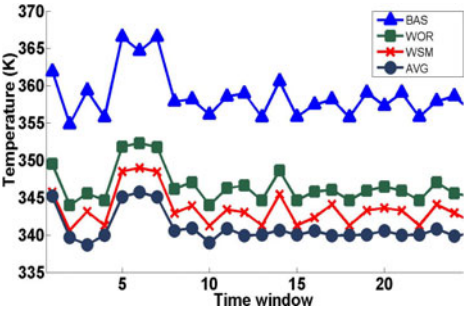



**Fig. 5.** Peak Temperature evolution of the 66 cores platform

| | Tmax | | | | | |
|---|---|---|---|---|---|---|
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | $366.6_{9.6}$ | $364.4_{5.1}$ | $369.8_{6.6}$ | $314.4_{10.6}$ | $377.6_{3.9}$ | $361.4_{8.9}$ |
| AVG | $346.4_{3.5}$ | $357.9_{2.6}$ | $356.5_{6.7}$ | $309.6_{8.1}$ | $348.3_{2.3}$ | $344.5_{3.7}$ |
| WOR | $340.9_{6.5}$ | $339.9_{3.2}$ | $343.7_{3.5}$ | $312.0_{5.8}$ | $346.1_{2.3}$ | $339.7_{3.8}$ |
| WSM | $343.2_{7.5}$ | $338.7_{4.2}$ | $340.9_{3.1}$ | $311.6_{5.9}$ | $352.3_{2.7}$ | $341.4_{5.8}$ |
| | Tmean | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | $330.6_{5.3}$ | $325.1_{3.3}$ | $331.1_{2.6}$ | $307.6_{4.9}$ | $335.2_{1.7}$ | $328.0_{4.7}$ |
| AVG | $326.6_{4.3}$ | $323.6_{2.5}$ | $328.1_{2.4}$ | $306.7_{4.4}$ | $330.2_{1.4}$ | $324.8_{3.7}$ |
| WOR | $325.9_{4.8}$ | $322.2_{2.7}$ | $327.4_{2.2}$ | $306.8_{4.1}$ | $330.1_{1.4}$ | $324.8_{3.7}$ |
| WSM | $325.9_{4.8}$ | $322.0_{2.7}$ | $327.2_{2.2}$ | $306.9_{4.1}$ | $329.9_{1.4}$ | $324.6_{3.7}$ |
| | Tgrad | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | $63.3_{9.3}$ | $61.8_{4.6}$ | $66.0_{6.3}$ | $13.6_{9.9}$ | $74.3_{3.7}$ | $57.9_{8.2}$ |
| AVG | $38.8_{2.2}$ | $54.7_{3.5}$ | $48.7_{6.1}$ | $7.5_{6.8}$ | $37.8_{1.8}$ | $36.3_{3.6}$ |
| WOR | $33.8_{3.0}$ | $36.5_{1.8}$ | $37.2_{3.1}$ | $10.0_{5.0}$ | $35.2_{1.8}$ | $32.1_{3.8}$ |
| WSM | $35.6_{4.0}$ | $35.7_{2.4}$ | $33.5_{2.3}$ | $9.6_{4.8}$ | $40.7_{2.2}$ | $33.2_{4.8}$ |

| | Tmax | | | | | |
|---|---|---|---|---|---|---|
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | $350.4_{5.7}$ | $375.7_{4.8}$ | $410.2_{3.6}$ | $373.3_{0.7}$ | $358.7_{3.2}$ | $368.0_{4.7}$ |
| AVG | $337.8_{4.5}$ | $352.4_{3.0}$ | $379.0_{0.8}$ | $365.1_{4.3}$ | $340.9_{2.0}$ | $360.1_{9.3}$ |
| WOR | $335.2_{4.8}$ | $354.3_{3.7}$ | $382.3_{0.8}$ | $367.6_{5.0}$ | $346.6_{2.4}$ | $361.9_{8.7}$ |
| WSM | $336.0_{7.7}$ | $353.8_{3.2}$ | $382.6_{1.0}$ | $367.8_{5.0}$ | $343.5_{2.4}$ | $361.6_{8.9}$ |
| | Tmean | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | $326.0_{4.4}$ | $328.6_{1.8}$ | $343.2_{1.5}$ | $325.9_{1.6}$ | $328.6_{1.4}$ | $325.2_{1.1}$ |
| AVG | $322.2_{4.2}$ | $325.7_{1.5}$ | $339.1_{1.1}$ | $324.5_{1.0}$ | $325.4_{1.2}$ | $323.8_{0.8}$ |
| WOR | $322.5_{4.1}$ | $326.0_{1.5}$ | $339.5_{1.2}$ | $324.8_{1.1}$ | $325.5_{1.2}$ | $323.9_{0.8}$ |
| WSM | $322.5_{4.3}$ | $325.3_{1.6}$ | $339.3_{1.1}$ | $324.9_{1.1}$ | $325.8_{1.2}$ | $324.0_{0.8}$ |
| | Tgrad | | | | | |
| | Bench1 | Bench2 | Bench3 | Bench4 | Bench5 | Bench6 |
| BAS | $46.4_{4.8}$ | $72.8_{5.2}$ | $105.7_{3.3}$ | $71.4_{1.0}$ | $53.9_{3.1}$ | $65.8_{5.3}$ |
| AVG | $32.2_{3.9}$ | $48.3_{3.3}$ | $72.7_{0.7}$ | $62.5_{5.0}$ | $36.0_{1.8}$ | $57.0_{10.0}$ |
| WOR | $28.0_{3.7}$ | $50.7_{4.4}$ | $75.5_{0.7}$ | $65.1_{6.2}$ | $39.6_{2.2}$ | $58.8_{10.0}$ |
| WSM | $29.4_{6.5}$ | $49.6_{3.6}$ | $75.3_{1.2}$ | $65.7_{5.9}$ | $37.1_{2.2}$ | $58.8_{10.3}$ |

**Fig. 6.** Thermal metrics of the 30 (left) and 66 (right) cores platforms

increase in the number of cores, the AVG is the best metric since it reflects a more realistic behavior due to a greater number of statistical sources. On the other hand, considering the WOR configuration leads to overestimate the temperature of the chip leading to non optimal floorplans. Future research expects to find a lower number of meaningful windows to reduce the overhead of profiling.

## 6 Conclusion

This work has proposed an efficient approach that incorporates power-profiling information to guide a thermal-aware floorplanner for 3D multi-processor architectures. The implementation of the tool with genetic algorithms has provided thermal-optimized floorplans as compared with a baseline heterogenous system. Also, the integration of power-profiling information in the floorplanner has shown how our floorplanner is able to outperform the thermal metrics obtained by thermal-aware floorplanners (obtaining up to 6 °C saving for the peak temperature) with a minimum impact on performance.

**Acknowledgements.** Ignacio Arnaldo is supported by Spanish Government Avanza Competitividad I+D+I: TSI-020100-2010-962 proyect. The work has also been supported by Spanish Government grants TIN 2008-00508 and MEC CONSOLIDER CSD00C-07-20811.

## References

1. Electrothermal monte carlo modelling of submicron hfets (2004), `http://www.nanofolio.org/research/paper03.php`
2. Intel tries to keep it cool (2004), `http://www.pcworld.idg.com.au/article/108386/inteltrieskeepitscool`
3. Adya, S., et al.: Fixed-outline floorplanning: enabling hierarchical design. IEEE Transactions on VLSI Systems 11(6), 1120–1135 (2003)
4. ARM: `http://www.arm.com/products/processors/cortex-a/cortex-a9.php`

5. Berntsson, J., Tang, M.: A slicing structure representation for the multi-layer floorplan layout problem. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 188–197. Springer, Heidelberg (2004)
6. Borkar, S.: Design challenges of technology scaling. IEEE Micro 19(4) (1999)
7. Chen, G., et al.: Partition-driven standard cell thermal placement (2003)
8. Chu, C., et al.: A matrix synthesis approach to thermal placement. IEEE Transactions on CADICS 17(11), 1166–1174 (1998)
9. Coello, C., et al.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer, Dordrecht (2002)
10. Cong, J., et al.: A thermal-driven floorplanning algorithm for 3D ICs (2004)
11. Cuesta, D., Risco, J.L., Ayala, J.L., Atienza, D.: 3D Thermal-Aware Floorplanner for Many-Core Single-Chip Systems. In: IEEE Latin-American Test Workshop (2011)
12. Deb, K., et al.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
13. Ekpanyapong, M., et al: Thermal-aware 3D microarchitectural floorplanning. Tech. rep., Georgia Institute of Technology Center (2004)
14. Han, Y., et al.: Simulated annealing based temperature aware floorplanning (2007)
15. Healy, M., et al.: Multiobjective microarchitectural floorplanning for 2D and 3D ICs. IEEE Transactions on CADICS 26(1), 38–52 (2007)
16. HPlabs: `http://www.hpl.hp.com/research/cacti/`
17. Hung, W.L., et al.: Thermal-aware floorplanning using genetic algorithms. In: ISQED, pp. 634–639 (March 2005)
18. IBM, C.D.C.: Complex soc design (2009)
19. Imperas: `http://www.ovpworld.org`
20. Iqbal, S., et al.: ParMiBench - an open-source benchmark for embedded multiprocessor systems. CAL 9(2), 45–48 (2010)
21. Li, X., et al.: A novel thermal optimization flow using incremental floorplanning for 3D ICs. In: ASPDAC, pp. 347–352. IEEE Press, Los Alamitos (2009)
22. Li, Y., et al.: Temperature aware floorplanning via geometry programming. In: IEEE International Conference on CSE Workshops, pp. 295–298 (July 2008)
23. Liu, Y., et al.: Accurate temperature-dependent integrated circuit leakage power estimation is easy. In: DATE, pp. 1526–1531 (2007)
24. OpenSPARC (2007), `http://www.opensparc.net/pubs/preszo/07/n2isscc.pdf`
25. Paci, G., et al.: Exploring temperature-aware design in low-power MPSoCs. International Journal of Embedded Systems 3(1), 43–51 (2007)
26. Sankaranarayanan, K., et al.: A case for thermal-aware floorplanning at the microarchitectural level. JILP 7(1), 8–16 (2005)
27. Singhal, L., et al.: Statistical power profile correlation for realistic thermal estimation. In: ASP-DAC, pp. 67–70. IEEE Computer Society Press, Los Alamitos (2008)
28. Tang, M., et al.: A memetic algorithm for VLSI floorplanning. IEEE Transactions on Systems, Man, and Cybernetics, Part B 37(1), 62–69 (2007)

# A System Level Approach to Multi-core Thermal Sensors Calibration

Andrea Bartolini, MohammadSadegh Sadri, Francesco Beneventi,
Matteo Cacciari, Andrea Tilli, and Luca Benini

University of Bologna, DEIS
via Risorgimento 2
40136 Bologna, Italy
{a.bartolini,mohammadsadegh.sadr2,francesco.beneventi,
matteo.cacciari,andrea.tilli,luca.benini}@unibo.it
http://www-micrel.deis.unibo.it

**Abstract.** Many-cores systems on chip provide the highest performance scaling potential due to the massive parallelism, but they suffer from thermal issues due to their high power densities. Thermal sensors and feedback strategies are used to reduce these threats but sensor accuracy directly impact control performance. In this paper we propose a novel technique to calibrate thermal sensors. Our approach can be applied to general multi-core platforms since it combines stress patterns and least-square fitting to perform thermal sensor characterization directly on the target device. We experimentally validate our approach on the Single Chip Cloud (SCC) prototype by Intel.

## 1 Introduction

Upcoming many-cores platforms stress the limits of Moore's law. High performance translates in high power densities, that, combined with high spatial parallelism and workload variations, produces non-uniform power dissipation that translates in not-uniform silicon die thermal distributions. This leads to localized degradation, acceleration of chip aging and increasing cooling costs. To help in studying and counteracting these looming threats, leading silicon manufacturers have started to deliver many-core prototypes to push researchers toward creating solutions anticipating next-generation product challenges.

As the number of cores integrated on a single die increases, an increasing number of accurate thermal sensors is required to reliably maximize performance under a thermal envelope[1,2]. Unfortunately, due to process variations[3,4], sensors differ from the nominal ones[5] and thus need to be carefully calibrated before they can be used. This process requires complex and expensive infrastructures, such as thermal cameras and special packages, which are usually not available to the end-users[8].

Whereas different thermal sensors design strategies have been exploited to limit their sensitivity to process variation and operating conditions, these solutions usually came with an area and power overhead [5]. To avoid that, in recent years software techniques have been proposed to use *a priory* knowledge of the probability distribution of the process variation combined with the knowledge of the thermal behavior to improve sensor

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 22–31, 2011.

accuracy[6]. These information are not always available and usually require as reference an initial nominal calibration limiting their applicability to uncalibrated thermal sensors[7,9].

The Single-Chip Cloud Computer (SCC) experimental processor [9] is a 48-core 'concept vehicle' created by Intel Labs as a platform for many-core software research. It has 24 dual-core tiles arranged in a 6x4 mesh. Each core is a P54C cpu. The entire system is controlled by a board management microcontroller (BMC) that initializes and shuts down critical system functions. It is commonly connected by PCI-Express cable to a PC acting as a Management Console (MCPC). Each tile integrates two thermal sensors based on ring oscillators, one positioned in proximity of the router and the other positioned close to the bottom core L1 cache. The BMC includes a power sensor capable of measuring the full SCC chip power consumption.

Unfortunately the built-in thermal sensors are not characterized and thus provide as output only a counter value proportional to the temperature. Due to process variations, each sensor under the same operating conditions has different offset and temperature sensitivity thus it is hard to use these uncalibrated sensors to drive advanced closed-loop thermal-control policies. Moreover thermal sensors output is also affected by noise, adding another degree of complexity on their usage.

In this paper we present a technique to overcome the limitations of uncalibrated thermal sensors. Our approach is based on a combination of stress patterns and least square fitting to extract the thermal sensor characterization directly from the multi-core devices. We evaluate the proposed approach using SCC as use case.

The rest of paper is organized as follows: Section 2 describes the basics of thermal transients and sensors whereas 3 describes the thermal sensor calibration technique. Section 4 describes the performance and behavior of the SCC thermal sensors. Section 5 shows the results and performance of the calibration procedure.

## 2 Background

As introduced in Section 1 several techniques have been proposed to use the knowledge on thermal transients behavior with power consumption spatial distribution to estimate the temperature in different locations of a die from the surrounding thermal sensors and to filter out the noise on the thermal sensors output. In a multi-core scenario, the temperature distribution across die area depends on the chip floorplan, its thermal environment, and the power consumption of the cores. The latter has been shown to be related to the operating point/performance level and workload characteristics, such as instruction type, density and data locality [11]. In the following, we focus solely on the relationship between power and temperature, with the ultimate goal of performing calibration of thermal sensors directly from power and temperature measurements.

According to the granularity usually required for thermal control, we can assume uniform power and temperature distributions in each core area. Then, recalling Fourier's law, the temperature of each core can be assumed dependent on its own dissipated power, ambient temperature and adjacent cores temperatures (boundary conditions). This assumption is actually straightforward for continuous time models only. When discrete-time models are considered, a larger coupling among cores has to be considered

to account for the "chain of interactions" taking place during the blind intervals between sampling periods. Recalling again Fourier's Law, the coupling among two cores will be inversely related to their distance and directly related to the sampling time period. Hence, the "equivalent neighborhood" of a core depends on the floorplan combined with the adopted sampling time. When the number of cores is small, we can assume that all cores lie in the same neighborhood, and the thermal behavior is described by a linear matrix equation:

$$T[n+1] = A \cdot T[n] + B \cdot [P[n]; T_{AMB}] \tag{1}$$

Equation 1 describes the transient thermal behavior in discrete time, where $n$ is the time sample index. We assume a system with $C$ cores, hence $T[]$ and $P[]$ are $C$-dimensional vectors, and $A, B$ are $C \times C$ square matrices. In steady state, we have $T[n+1] = T[n] = T$, and we therefore obtain:

$$T = SG \cdot P + \quad \cdot T_{AMB} \tag{2}$$

where $SG = (I - A)^{-1} \cdot B$ is the *steady-state gain* matrix, which links the power consumed by cores to their temperature in steady state and is a vector of one. If $SG$ is known, the temperature distribution can be computed from power consumption measurements by simple vector-matrix product. In practical terms this is a very useful equation, as it gives a way to predict the asymptotic temperature, given a constant power consumption level.

Moreover [8] thermal sensors usually show a linear dependency of the measured quantity (resonant frequency $f_{TS}$ for ring-oscillators ones[5]) with the absolute temperature $T_i$. As consequence by counting the number of cycles expired in a fixed time interval we can express the cycles count ($TS_i$) as a linear function of the real temperature.

$$TS_i = A_i + B_i \cdot T_i \tag{3}$$

$$T_i = A'_i + B'_i \cdot TS_i \tag{4}$$

where $A'_i = -A_i/B_i$ and $B'_i = 1/B_i$. Thus calibrating the thermal sensors means to determine the values of $A'$,$B'$ for all the sensors in the chip.

## 3 System Level Thermal Sensors Calibration

If any additional information about chip building materials and the packaging thermal properties is available (Thermal resistance and capacitance) one simplistic approach to calibrate the thermal sensors is to use off-chip temperature measurements at references under minimum activity ($T_{AMB\_MIN}$) and maximum activity ($T_{AMB\_MAX}$), and then probe for all the thermal sensors the output value ($TS_i$) under the two different reference cases ($TS_{i\_MIN}, TS_{i\_MAX}$)[10,8]. Then we can characterize each sensor and find the $A_{REF_i}$ and $B_{REF_i}$ for each core by linearly interpolating these two points as described in Eq.5.

$$\begin{cases} A_{REF_i} = TS_{i\_MIN} - TS_{i\_MAX} \cdot \frac{TS_{i\_MAX} - TS_{i\_MIN}}{T_{AMB\_MAX} - T_{AMB\_MIN}} \\ B_{REF_i} = \frac{TS_{i\_MAX} - TS_{i\_MIN}}{T_{AMB\_MAX} - T_{AMB\_MIN}} \end{cases} \tag{5}$$

This approach assumes all sensors temperatures to be equal both under minimum and maximum processing utilization. Whereas this approximation is reasonable at low utilization, it leads to severe approximation errors at maximum utilization since it does not consider spatial temperature gradients caused by the not-uniform thermal dissipation and the package thermal resistance. To account for these effects we developed a new characterization method that takes advantage of the linear relation between temperature and power (in steady-state condition) and discovers it by linear regression.

Indeed if we consider the thermal transients expired (steady-state) and we consider a stable workload homogeneously distributed along the multi-core surface (same load/task executing in all the core), from equations (4) and (2) we can write the thermal sensor output as direct function of full chip power consumption and ambient temperature[1]. Indeed by calling $K = GS \cdot \quad$ and $P_{CORE} = P_{TOT}/\#core$, where $\quad$ is a vector of one, $K$ is a vector in which each element ($K_i$) is the row-wise sum of $GS$ and $P_{CORE}, P_{TOT}$ are respectively the core and full chip power consumption, we can write for each thermal sensor:

$$\begin{aligned} TS_i &= A_i + B_i \cdot T_i = A_i + B_i \cdot (K_i \cdot P_{CORE} + T_{AMB}) \\ &= A_i + B_i \cdot K_i \cdot P_{CORE} + B_i \cdot T_{AMB} \end{aligned} \tag{6}$$

The equation above (6) can be formulated for each thermal sensor ($i$) as a least square problem where the unknown parameters are $A_i, B_i, B_i \cdot K_i$ and the input data are $TS_i$, $P_{CORE}, T_{AMB}$. Indeed if we generate a cloud of N tuples $\{TS, P_{TOT}, T_{AMB}\}$ by stressing all the cores of the target multi-core together at different frequencies and workloads we can write:

$$\begin{cases} Y_i = \left\{TS_{i,\#h}\right\}, h = 1, \cdots, N \\ X_i = \left\{1, P_{CORE,\#h}, T_{AMB,\#h}\right\}, h = 1, \cdots, N \\ \Theta_i = \{\alpha_i, \beta_i, \gamma_i\}, \alpha_i = A_i, \beta_i = B_i \cdot K_i, \gamma_i = B_i \\ Y_i = X_i \cdot \Theta_i => \Theta_i = X_i^{\dagger} \cdot Y_i \end{cases} \tag{7}$$

Where $N$ is the number of different stress pattern applied to the system, $X_i^{\dagger}$ is the pseudo-inverse of $X_i$ and $\Theta$ contains the target calibration coefficient $A_i, B_i$.

## 4 SCC Test Case

As early introduced, the SCC chip represents today one of the most advanced HW platform available for many-core research. Moreover it features a number of uncalibrated thermal sensors. In this section we will describe and characterize the behavior and performance of the SCC thermal sub-system. As introduced in Section1 SCC integrates in each tile two built-in thermal sensors. The first thermal sensor is placed close to the router and the second one is placed near the L1 cache of the bottom core. Each thermal sensor is composed of two ring oscillators and the sensor output (TS) is the difference of the two oscillators clock counts over a specific time window $t_W$. The difference is proportional to the local die temperature($T$) [10].

[1] This measurement infrastructure are typical in today end user set-up.

For each tile, the time window ($t_W$) can be programmed through a per-tile control register[2][12] as a number of tile clock cycle(NCC) $t_W[s] = NCC/f_{TILE}$. Thus it needs to be updated each time the tile frequency changes.

$$TS = (A + B \cdot T) \cdot t_W \tag{8}$$

where $B < 0$ ($TS$ decreases with the temperature rising) and $A > 0$ ($TS$ is always positive and in the thousands) are different for each sensors.

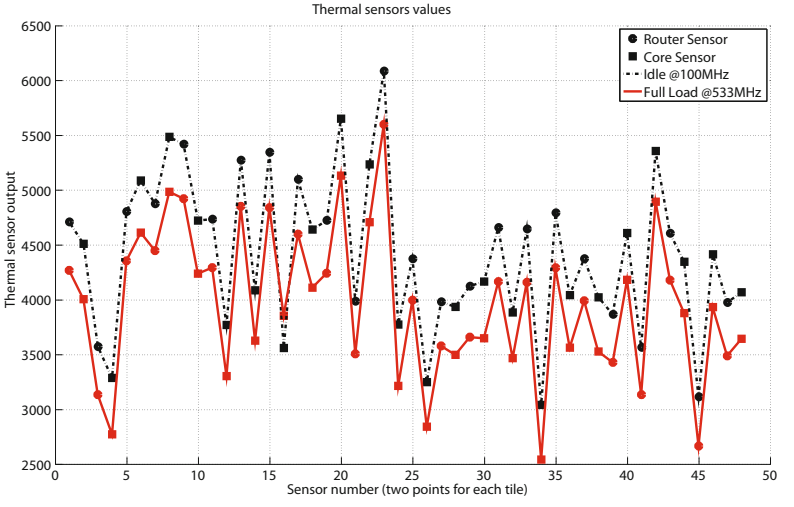



**Fig. 1.** Raw thermal sensor values

We first evaluate the spatial variation of the thermal sensor values under two different homogeneous stress conditions. Fig.1 shows the results of this test. In the x-axis are reported the thermal sensors moving from the bottom-left corner to the top-right one of SCC (odd ones refer to router sensors whereas even ones to core sensor ). The dashed line shows the thermal sensors output when all the cores are in idle state (no task allocated) and are running at the smallest possible clock frequency (100MHz): the coldest point. Instead, the solid line shows the thermal sensors output when all the cores are executing a power virus[3] while running at higher frequency (533MHz). In the coldest point we can assume the real silicon temperature to be roughly constant across the chip area. However we can notice a strong variation in the raw sensor values ($> 50\%$). Moreover, it is notable that idle and full-load plots are very similar in shape but for the hotter case (all core busy) all the sensors output are significantly lower, as expected because sensor count decreases when teperature is high. However it must be noted that the variations due to temperature differences between minimum and maximul load conditions

[2] CRB Sensor Register (rw).

[3] *cpuburn* power virus by Robert Redelmeier: it takes advantage of the internal architecture to maximize the CPU power consumption.

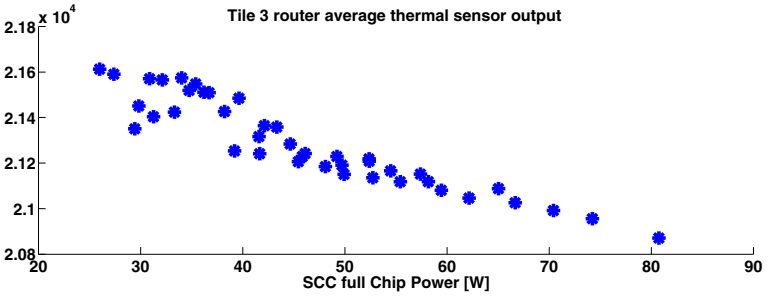



**Fig. 2.** SCC thermal sensors output vs. full chip power consumption

is less than 20% compared to the uncalibrated spatial sensors variation. This clearly highlights the strong need of the thermal sensor calibration step.

We perform a second test with the goal of evaluate the relation between SCC power and thermal sensor output while executing different benchmarks and while running all the tiles together at different frequency levels. Fig.2 shows the results, we can recognize that sensor values are linear with the power consumption. This support the fundamental assumption of our proposed calibration approach and will be used in the following section to construct the least squares input data-set.

## 5 Experimental Results

In this section we discuss the performance of the presented calibration method. In section 5.1 we describe our data collection infrastructure. In section 5.2 we show the performance of our sensor calibration technique when applied to the SCC multi-core whereas in section 5.3 we make a comparison between characterization results with a Hotspot [13] model of SCC.

### 5.1 XTS Framework

To obtain the thermal sensors values we have modified the SCC standard linux image to access at each schedule tick the tile thermal sensors and the per-core performance counters. Inside the kernel we implement a double buffer system to save the sensors entries generated for each core at the speed of the scheduler kernel tick. In user space we implement an application that synchronizes through a kernel driver with the internal double buffer system and flushes it as soon one is full. Thanks to this, we can trace sensors and performance counters with a sampling time of 10ms to 1ms[4].

### 5.2 Sensors Calibration

We use SCC as test bench for the presented calibration methodology. As described in section 3 we first create the data set by applying different synthetic benchmarks

[4] Accordingly to the linux kernel internal configuration and current tile frequency.

and power viruses to all the cores together at different $f_{tile}$. For all these workloads and configurations we wait for the end of the thermal transient and we save the full chip power consumption, the board temperature and all the thermal sensors outputs. As described in (7) we collect them together and we solve the least square problem to generate the calibration coefficients for all the sensors.

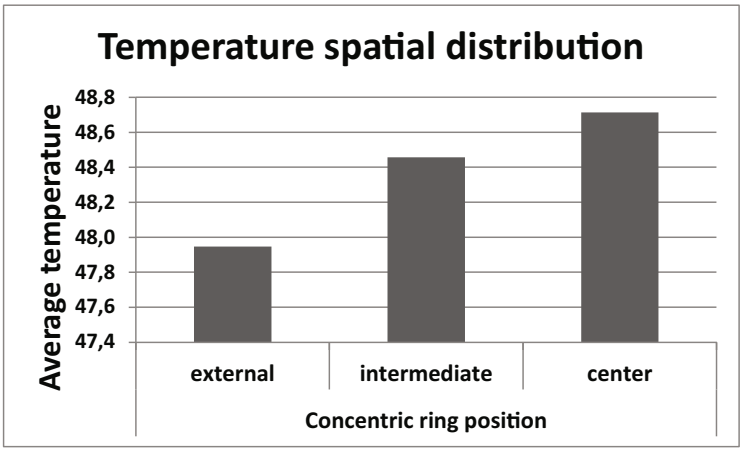



**Fig. 3.** Distribution of temperature on chip along different concentric rings of cores under full load at 533Mhz

We first use the calibrated sensors to highlight the spatial thermal behavior of SCC. Indeed, Fig.3 shows the average temperature over three different rings of cores in the SCC floorplan, under full load. "External" bar refers to the outer ring of sensors (24 sensors), "intermediate" bar refers to the immediate inner ring of sensors (16 sensors) whereas "center" bar refers to the four central tiles (8 sensors). From the plot we can notice the presence of a thermal gradient between the center cores and the boundary ones that show average lower temperatures.

Secondly we use the $A, B$ calibrated sensors in comparison with the $A_{REF_i}, B_{REF_i}$ calibration described by Eq.5 to evaluate the performance of proposed solution. Fig.4 shows with solid line the board temperature when all the core of SCC pass from idle to full load whereas with dots are reported the calibrated output for the sensor showing the lower temperature. Circles refer to the proposed calibration whereas stars refer to the reference one (REF). From the figure we can first notice that the ambient around SCC is exposed to a significant temperature drift due to the heat dissipated by the SCC chip. This gives us a lower bound for the real thermal cycle happening inside the SCC die.

Thirdly comparing all the three curves together we can notice that our approach improves upon the reference one (REF) since it is able to account for the package thermal resistance, showing internal higher temperature compared to the ambient one. Moreover from the same figure we can notice that the thermal transient is characterized by different time constants, indeed it looks like to be the response of a system composed

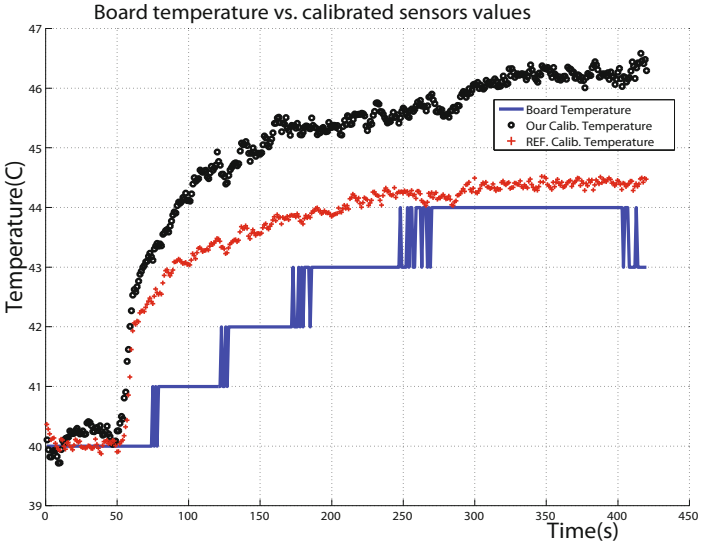



**Fig. 4.** Difference between our calibration and the reference one under full load at 533Mhz

by multiple poles with a very long settling tail, probably it is caused by a "partial" pole-zero cancellation. From the same plot we can see that ambient temperature react significantly slower compared to the internal one suggesting the presence of an high thermal capacitance due to the package and the heat-sink.

### 5.3 Hotspot Model Comparison

To evaluate our calibration results we have developed a thermal model for SCC using HotSpot[13] thermal simulator. This model is capable of predicting SCC chip temperature values at different on-die locations according to input power to the chip. The tuning of the thermal model has been performed automatically by selecting the heat-sink's thermal convection resistance that minimize the mean square error between hotspot estimated temperature and real ones under two extremes SCC load: all core in idle at 100MHz and all core running a power virus and at 533MHz. We tune two different HotSpot model: one using as extremes the temperature obtained by the SCC thermal sensors calibrated with the proposed approach and the other using the reference calibration (REF). We then compare the real thermal map of SCC in the two calibration approaches (our and REF) against the corresponding Hotspot model outcome under the same SCC stress configuration. These is done for different load pattern each one obtained by imposing a different workload and a different frequency value to all the core of the SCC chip. Both in real HW and in the Hotspot simulations.

Fig.5 (a) shows the results of these tests. On the y-axis is reported the percentage of occurrences of a given mean square error between the hotspot simulation and the real thermal map for all the tests made (8 frequency level x 18 workloads). As can be seen for

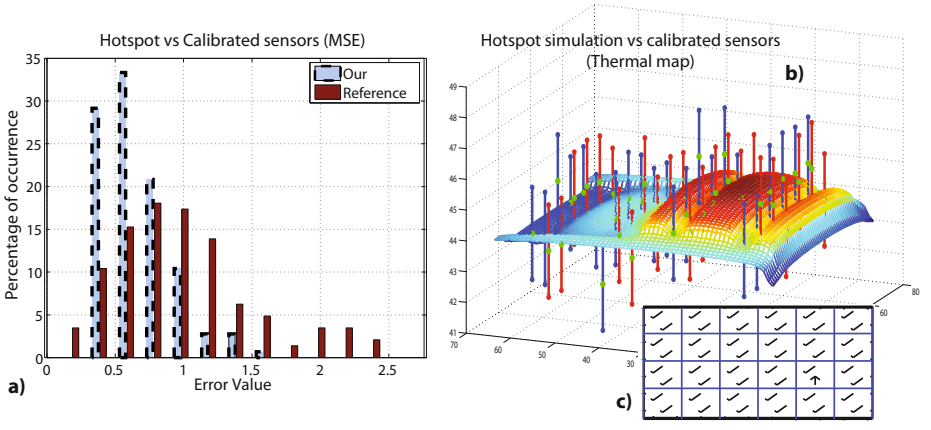



**Fig. 5.** hotspot simulation vs real measurements

our calibration the MSE is always less than 1.5 degree with the majority of them below 1 degree. In contrast, the reference calibration when compared to hotspot simulation produces errors bigger than 2 degrees. In Fig.5 (b) we compare the SCC thermal map with sensor calibrated with our approach with the hotspot simulation under a mixed workload configuration. The surface is the output temperature of the thermal model and bars represent the calibrated sensors outputs with a tolerance range ($\pm 1.5C$). During this test half of the chip is under full loads and half of the chip is idle. Fig.5 report the sensors outcome compared to the modeled one. The ticks show when the model output is within the sensor error tolerance whereas the arrows represent when the model output is lower or higher. From this set of results we notice that the calibrated thermal sensors measurements captures the thermal trends highlighted by the hotspot model with good accuracy.

## 6 Conclusions

In this paper we have presented a calibration approach for the thermal sensors typical for recent and future multi-core architectures. As test case we studied the SCC thermal sensors performance and behavior. Our results highlight significant spatial and temporal noise on the output values. To manage these non uniformities in uncalibrated sensors we developed a novel procedure to extract the relationship between thermal sensors output and absolute temperature using data-regression. We then used our calibration to highlight the thermal performance of SCC. We compared it with Hotspot thermal simulation of a floorplan similar to SCC showing a tolerance error below the 1.5 degrees.

**Acknowledgements.** This work was supported, in parts, by Intel Corp., Intel Labs Braunschweig and the EU FP7 Projects Pro3D (GA n. 248776) and Therminator (GA n. 248603).

## References


1. Bartolini, A., Cacciari, M., Tilli, A., Benini, L.: A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multi-cores. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), March 14-18, pp. 1–6 (2011)
2. Sharifi, S., Simunic Rosing, T.: Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. Trans. Comp.-Aided Des. Integ. Cir. Sys. 29(10), 1586–1599 (2010)
3. Humenay, E., Tarjan, D., Skadron, K.: Impact of process variations on multicore performance symmetry. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2007), pp. 1653–1658. EDA Consortium, San Jose (2007)
4. Paterna, F., Acquaviva, A., Caprara, A., Papariello, F., Desoli, G., Benini, L.: An efficient on-line task allocation algorithm for QoS and energy efficiency in multicore multimedia platforms. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), March 14-18, pp. 1–6 (2011)
5. Remarsu, S., Kundu, S.: On process variation tolerant low cost thermal sensor design in 32nm CMOS technology. In: Proceedings of the 19th ACM Great Lakes Symposium on VLSI (GLSVLSI 2009), pp. 487–492. ACM, New York (2009)
6. Zhang, Y., Srivastava, A.: Accurate Temperature Estimation Using Noisy Thermal Sensors. In: Proc. of Design Automation Conference, pp. 472–477 (2009)
7. Revision Guide for AMD NPT Family 0Fh Processors. AMD Publication, #33610, p. 38 (October 2006)
8. IBM, Calibrating the Thermal Assist Unit in the IBM25PPC750L Processors. PowerPC Embedded Processors Application Note (October 6, 2001)
9. Howard, J., et al.: 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. In: Solid-State Circuits Conference, ISSCC (2010)
10. Intel Labs "Using the Sensor Registers", Revision 1.1, `http://communities.intel.com/community/marc`
11. Bartolini, A., Cacciari, M., Tilli, A., Benini, L., Gries, M.: A Virtual Platform Environment for Exploring Power, Thermal and Reliability Management Control Strategies in High-performance Multicores. In: GLSVLSI (2010)
12. Intel Labs, "SCC External Architecture Specification (EAS)", Revision 1.1, `http://communities.intel.com/community/marc`
13. Huang, W., Stan, M.R., Skadron, K.: Parameterized physical compact thermal modeling. IEEE Transactions on Components and Packaging Technologies 28(4), 615–622 (2005)

# Improving the Robustness of Self-timed SRAM to Variable Vdds

Abdullah Baz, Delong Shang, Fei Xia, Alex Yakovlev, and Alex Bystrov

Microelectronic System Design Group, School of EECE, Newcastle University
Newcastle upon Tyne, NE1 7RU, England, United Kingdom
{Abdullah.baz,delong.shang,fei.xia,alex.yakovlev,
a.bystrov}@ncl.ac.uk

**Abstract.** The most efficient power saving method in digital systems is to scale Vdd, owing to the quadratic dependence of dynamic power consumption. This requires memory working under a wide range of Vdds in terms of performance and power saving requirements. A self-timed 6T SRAM was previously proposed, which adapts to the variable Vdd automatically. However due to leakage, the size of memory is restricted by process variations. This paper reports a new self-timed 10T SRAM cell with bit line keepers developed to improve robustness in order to work in a wide range of Vdds down to 0.3V under PVT variations. In addition, this paper briefly discusses the potential benefits of the self-timed SRAM for designing highly reliable systems and detecting the data retention voltage (DRV).

**Keywords:** Self-Timed SRAM, Robustness, Energy Harvesting, Low Power Systems, DRV, Handshake protocols, Open Loop Control, Closed Loop Control, Reliable System, Fault model, Fault Detection, PVT variations.

## 1 Introduction

At present, there is a scarcity of memory systems that work robustly over a wide range of supply voltage. Generally, this requirement is imposed either by the power delivery or management technique employed in low power systems [2]. The supplied voltage might be either unstable, e.g. with voltage droops, or completely nondeterministic, e.g. in the case of energy scavengers. Systems may also have different working modes by design, e.g. DVS (Dynamic Voltage Scaling). The range of Vdd change may be large, from the nominal voltage of the technology down to the Data Retention Voltage (DRV) of memory banks in order to accomplish tasks with the highest performance whenever power is available and save as much energy as possible without losing data whenever the system is idle [4]. For such a wide range of voltage the latency of memory access varies significantly. [5] shows that the working frequency of 128 cells per column SRAM changes from 0.6GHz at 1V to 0.01GHz at 0.4V. This means that the difference in timing in such a wide voltage range is more than 60 times.

SRAM memory banks mostly work based on an open loop control model, in which timing assumptions are used to manage the pre-charge, word line selection, and

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 32–42, 2011.

reading/writing sub-operations. For example, synchronous systems use different phases of clock signals to control these sub-operations. The low phase of the clock signal is used during pre-charge and the high phase is used during word line selection and reading/writing. Here the clock frequencies must reflect the worst case timing assumption of the memory banks.

Because of significant latency differences in memory banks under a wide range of Vdds [1][2], there are three possible solutions to design a memory sub-system working in a wide range of Vdds: 1) different clock frequencies are required for different Vdds; 2) the lowest clock frequency associated with the lowest Vdd is used for the whole range of Vdds; 3) bundled-data method, bundling with a delay element. There exist a number of problems which were explored in [1][2].

Unlike the open loop control model, closed loop control uses feedbacks coming from the SRAM itself to manage the pre-charge, word line selection and reading/writing. Such a mechanism is possible if completion signals can be obtained from the SRAM and connected to asynchronous handshake protocols.

Recently, a fully self-timed 6T SRAM was proposed [1][2], which uses closed loop control by monitoring bit lines during reading and writing. The SRAM regulates the data flow based on the actual speed of the circuit under any operating condition. The SRAM then works fast under the nominal voltage and slow under low voltage. Based on [1], this kind of SRAM can work down to 0.2V in UMC 90nm CMOS technology. However, when process variations are considered, the working range of the SRAM becomes severely restricted for memories of reasonable size [2]. For example, for SRAM with 256 cells per bit line, the working range under the worst case variations is from 1V to 0.6V.

The main contributions of this paper include: 1) exploring in detail the reasons why process variations reduce the working range and demonstrating the limitations in supporting large bit lines and working robustly below 0.4V; 2) designing a new self-timed 10T SRAM and its controller; 3) introducing bit line keeper techniques to asynchronous SRAM design which in combination with 2) improves variation robustness.

Closed loop, unlike open loop, control has potential deadlocks. This paper also proposes an idea for building a highly reliable memory system by making use of such deadlocks.

The remainder of this paper is organized as follows: Section 2 highlights the problems of the existing self-timed 6T SRAM memory sub-system designs. In section 3, a new self-timed 10T SRAM and its controller are described. Section 4 shows the further improvement of the robustness of the SRAM by introducing bit line keepers. Section 5 demonstrates a reasonably sized memory bank based on the 10T SRAM cells. Section 6 gives a brief discussion on using the self-timed SRAM to build highly reliable systems, and finally Section 7 concludes with discussions.

## 2 Problems in Existing Fully Self-timed SRAM Memory

In self-timed SRAM, the completion of writing and reading is detected by monitoring the bit lines [1][2] because the bit lines are shared between the memory cells in the same column and the peripheral circuits used to detect the completion of the operation. This avoids having completion detection circuit in each cell which is not practical for area, power and performance reasons [6][7][8].

The beauty of the existing self-timed 6T SRAM is in combining a cheap standard gates based sensing logic and an intelligent control.

Without considering process variations, this technique works under a wide range of supply voltage, from 1V down to 0.2V [1] for UMC 90nm CMOS technology. It however stops working below 0.4V under process and/or temperature variations. Fig. 1 clearly shows that the data has already been written into the cell. However, the WAck signal is not generated as the BL cannot be charged to high enough to trigger the logic gates. A deadlock happens. As shown in the figure the BL voltage is charged to about 0.18V while the threshold voltage is more than 0.2V.

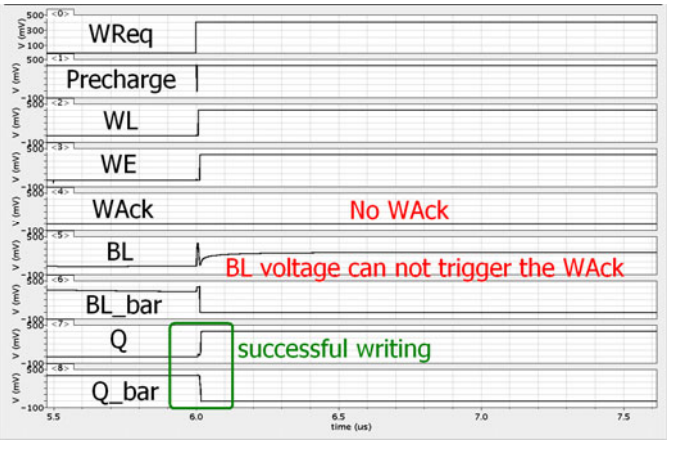



**Fig. 1.** Experiment of successful writing in self-timed SRAM without acknowledgment under 400mV

Intuitively a Sense Amplifier (SA) can be used to amplify the difference in the bit lines to trigger the acknowledgment signal. However an SA needs an enable signal which may still fall into the same situation and it is a power hungry circuit. Paper [7] clearly states that an SA is not suitable for asynchronous memory.

These problems may be caused by bit line leakage and/or n-type transistors not suitable for propagating logic one. The following experiment is used to demonstrate how these problems can be caused by leakage.

As the number of cells per column increases, the leakage from un-accessed cells increases [14]. This reduces the bit line voltage and then worsens the problem of generating the acknowledgment signals.

The worst case leakage happens when all cells in the column store the same data and then a different data is written into one of the memory cells, for example, all 1s are stored in a column, and then 0 is written into one cell of the column. There is a fight between the cell and all the other cells in the column, the cell being written is trying to charge the bit line to trigger the acknowledgment signal while the leakage from all other cells discharge it. By simulating this worst case scenario we have measured the bit-line voltage for different number of cells per bit line after successful writing. The results are depicted in Fig. 2 (a) for the typical corner (TT), and (b) for

the worst case leakage which happens in the FF corner [8]. The red curve in each figure expresses the minimum voltage needed to trigger the acknowledgment while the black curves represent the maximum bit-line voltage after flipping the cell for different bit-line configurations. Hence all values above the red curve mean successful generation of acknowledgment while all values below it mean failure.

The figures illustrate the effect of an increasing number of cells and process variations on increasing the leakage and hence failing the completion detection mechanism. For 64, 128 and 256 cells per column the completion can be detected down to 0.3V, 0.33V, 0.4V for the typical-typical corner and 0.48V, 0.52V, 0.60V for the worst case process variations. It is worth mentioning that as the system temperature increases the transistors become more leaky [8] which worsens the problem and increasingly leaves the memory in a deadlock situation as no feedbacks are generated. Even if this type of deadlock is detected and resolved, the next request still results in no acknowledgment. This motivates us to propose a new design in the next section.

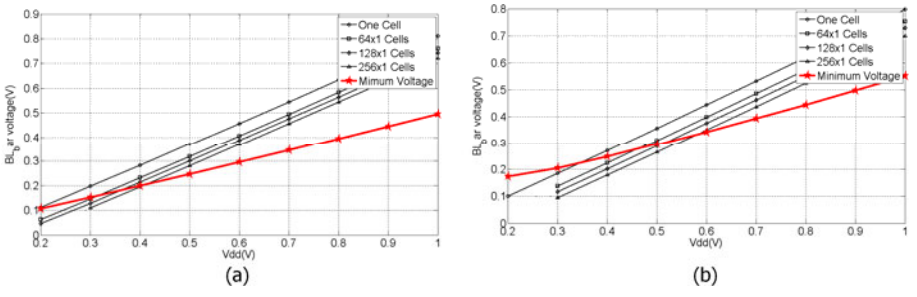



**Fig. 2.** The bit-line voltage after successful writing for different number of cells per column under the (a) typical corner and (b) fast corner (worst case leakage)

## 3 Proposed New Self-timed SRAM for Robustness

The proposed new SRAM structure is shown in Fig. 3 (a), which includes two extra bit lines and four extra transistors (N5-N8). Two transistors are on each side and stacked which can reduce leakage as well. The two new bit lines are used for reading, and the two existing bit lines are used for writing only. This structure is the same as the 10T SRAM proposed in [9]. The main original purpose for the 10T SRAM is to separate the reading and writing in order to tolerance noise. This property is inherited here. In addition, unlike the 6T SRAM propagating logic one to charge one of the bit lines through n-type transistors for completion detection, the new structure dynamically builds a path to ground based on the stored value to discharge one of the bit lines for completion detection.

The size of the 10T cell is about 67% bigger than the 6T. Its write-ability and holding stability are similar to the 6T's while it has no reading disturbance [9]. Hereafter, we will use RWL and WWL for reading and writing word lines and (RBL & RBL_bar) and (WBL & WBL_bar) for reading and writing bit lines respectively.

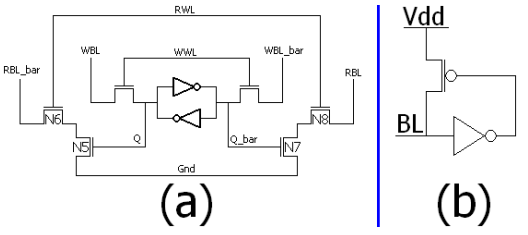



**Fig. 3.** Proposed (a) 10T SRAM cell and (b) bit line keeper

To be fully self-timed, similar to the self-timed 6T SRAM, reading is arranged in two steps. The first is pre-charging which pre-charges the four bit lines (over threshold). The second is reading which opens one of the two stacked n-type transistors by the reading word line (RWL) and Q/Q_bar to discharge one of the bit lines (RBL and RBL_bar). The bit line being discharged indicates that the data is ready for reading.

Writing is arranged in four steps unlike the three steps in the 6T, here assuming zero is stored and one is going to be written to flip the cell, 1) pre-charging, the same as the pre-charging in reading, 2) opening the writing word line (WWL) which discharges one of the writing bit lines (similar to the normal reading in 6T), in this example WBL is discharged, 3) enabling the write-driver which discharges WBL_bar and then flips the memory cell, and 4) enabling RWL which results in discharging RBL_bar provided that the cell is flipped and one is generated from the other side to open the transistor N5, hence guaranteeing the writing operations. These are described in STG form in Fig. 4, (a) for writing and (b) for reading.

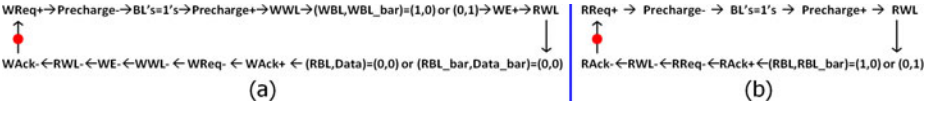



**Fig. 4.** STG specifications for (a) writing and (b) reading operations

Delay-Insensitive (DI) circuits are the safest asynchronous circuits as they work correctly regardless of delays in both the wires and gates [12]. However, DI is hard to implement or even impossible in some cases. Thus the best trade-off is the Quasi-Delay-Insensitive (QDI) or Speed Independent (SI) circuits [10]. Fig. 5 shows an SI solution mapped from the STG specifications above. The circuit is designed by using Petrify [16] and optimized manually.

The implementation is similar to the controller of the self-timed 6T SRAM. Modifications include gate 5 which is used to generate the WAck signal. Only when either both Data and RBL are low or both Data_bar and RBL_bar are low, the acknowledgement signal for writing is generated. Fig. 6 shows the waveforms obtained from the controller for reading (a) and writing (b). The numbers 1-4 are associated with the reading and writing steps mentioned above.

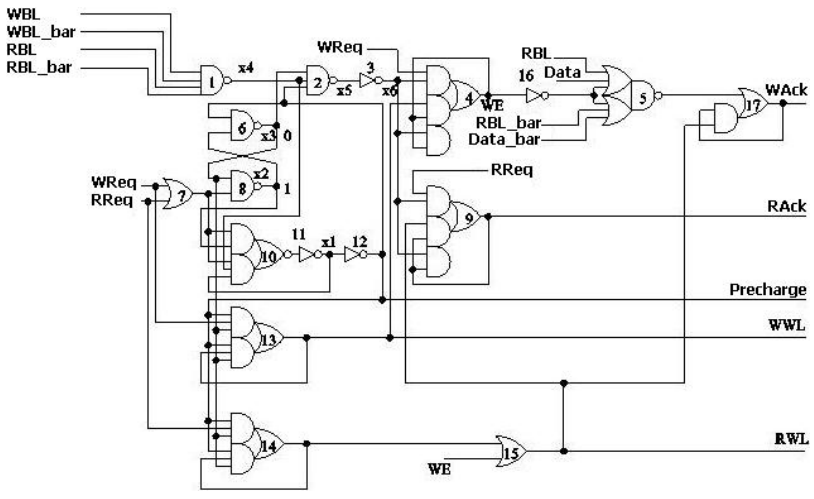



**Fig. 5.** Proposed SI controller

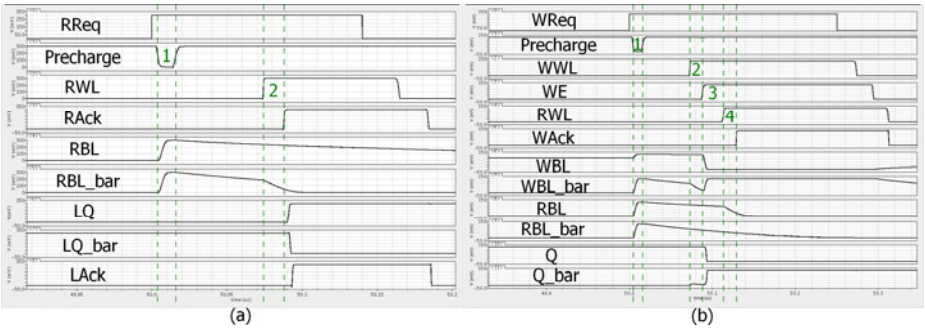



**Fig. 6.** Waveforms of the proposed SRAM under 300mV for (a) reading and (b) writing

## 4 Robustness Improvements by Leakage Reduction

A bank containing 256X128 such 10T SRAM cells is robust just above 0.4V for all process corners and temperature variations. However at 0.3V it fails at the FF corner and high temperature. For example, gate1 is faster than its typical delay if it is in the FF corner. And then it is triggered by leakage. The latency depends on the corners and the temperature. This timing failure is shown in Fig. 7 (a), in which WE is generated earlier than WWL and then a wrong WAck is generated.

This happens because leakage violates the control protocols. When all four bit lines are fully charged, according to the protocols, WWL is expected and then one of the bit lines can be discharged. However, one of the bit lines is discharged wrongly by leakage. This triggers gate1, generates the WE signal, and causes a wrong operation.

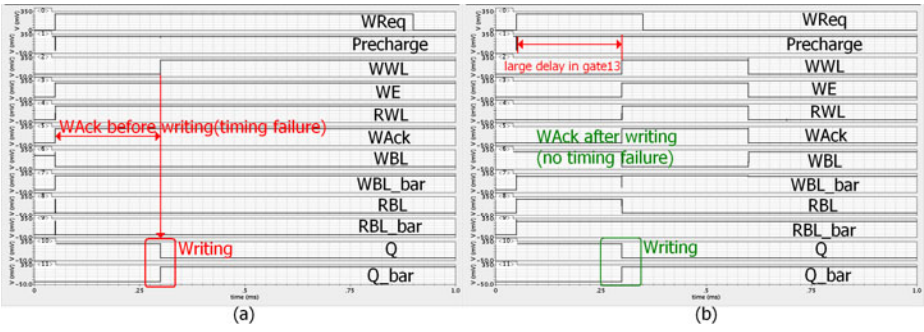



**Fig. 7.** Simulation results for writing operation under 300mV (a) without keeper (shows timing failure) (b) with keeper (no failure)

Leakage therefore must be suppressed or at least compensated. Here we proposed to use the bit line keeper as shown in Fig. 3 (b) [9] to maintain the charges in the bit line and suppress that timing failure. Using this bit line keeper the problem has been addressed as shown in Fig. 7 (b). With Vdd at 0.3V, whatever the delay of gate13 all signals are delayed until the actual writing occurs.

This bit line keeper is smaller than the 6T SRAM cell and hence for a bit-line with 256 cells its area overhead in less than 0.4%.

The keeper was also tried with on the self-timed 6T SRAM. The 6T SRAM is about 2 times faster than the 10T SRAM for reading, almost the same speed for both 6T and 10T SRAMs for writing over 500mV, and the 6T SRAM is about 60% of the performance of the 10T SRAM at 300mV. However, when taking into account the process variations, the 6T SRAM with the keeper still failed with Vdd below 400mV.

## 5 SRAM Bank Realizations and Analysis

Bundling is the use of one column to track the timing of a multi-column bank which has been proposed in the literature not only for asynchronous but also for synchronous SRAM [11] which has proved its reliability across a wide range of PVT variations [11]. In all previous work [2][8][11] the authors found that the timing of any column depend on the data stored in its cells and the data being accessed. Therefore a redundant column which is made slowest is added to each bank for bundling. In [2][8] the authors worsened the bundling column's timing by hard-coding its stored data to reflect the worst case leakage while the authors of [2] worsened their bundling column by flipping its data content for every write for maximum latency.

A bundled 32kbit (256x128) SRAM is developed in this work. The far end normal data column is used as the bundling column instead of a redundant worst-case one. We have tested this memory under all process corners (TT, FF, SS, SNFP and FNSP), temperature changing from -55°C up to 125°C, and supply voltage varying from 1V down to 0.3V.

Fig. 8 shows simulation results for writing at 0.3V. The black part of each bar represents the writing time from raising the request signal to the cell being flipped while the gray part reflects the completion detection time from flipping the cell to the acknowledgment being generated. Under all operating conditions, writing acknowledgment of the bundling column is generated in the safe time after all cells are flipped.

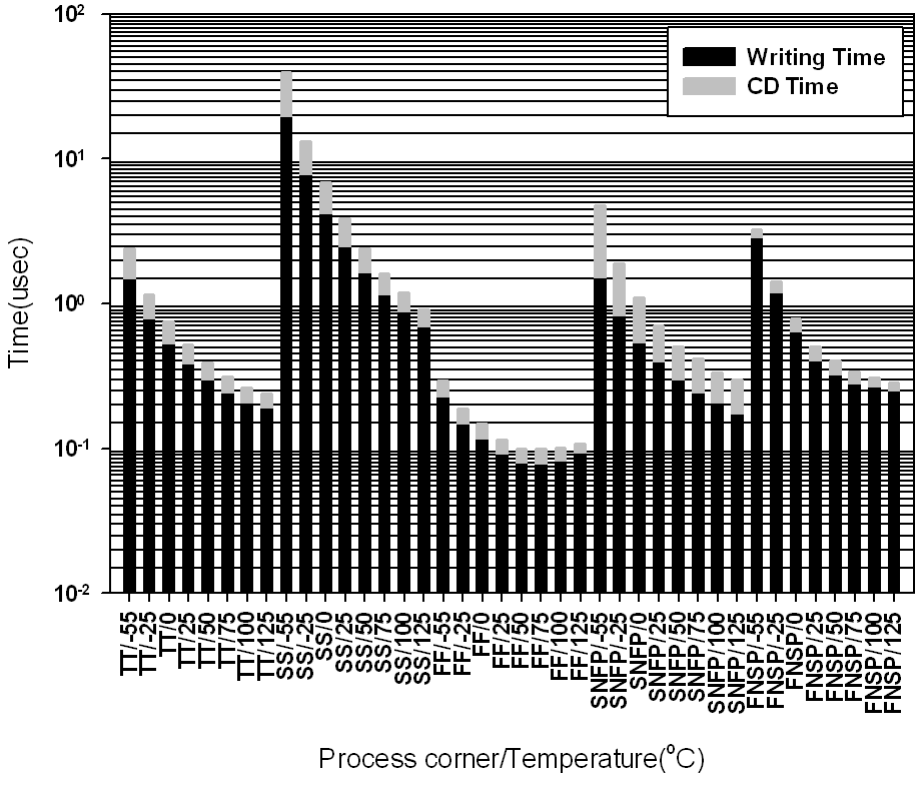



**Fig. 8.** SRAM bank writing time with the last column as bundler at 300mV

The omission of a redundant bundling column is possible because the additional bit line keepers, stacked n-type transistors and the modified controller are heavier than the controller for the 6T, and can cover the latency caused by all process corners and temperature variations as well as latency inconsistencies across different data values.

However, the heavy controller increases cost. For example, a 1Kb memory with keepers pays less than 1% penalty on performance and 10% on power consumption.

## 6 Making Use of Memory Deadlocks in Highly Reliable Systems

Currently memory subsystems dominate the system chip area (predicted to occupy about 94% by 2014) [13]. Memory reliability is very important, especially in low power and energy harvesting systems, as lowering Vdd may reduce reliability.

Process variations have different impacts on different components of a memory sub-system. For instance, the presence of important leakage currents in sub-90nm technology is generally considered as a main concern in terms of power consumption and more recently in terms of fault modelling [4]. Mostly variations affect the threshold voltage and eventually the leakage and latency. The proposed SI SRAM covers the latency related faults. Here we further investigate the faults caused by leakage. We refer to all these leakage related faults as leakage faults.

In a highly reliable system, fault detection is crucial. The earlier a fault is detected, the better a system is. Normally the faults are detected by a series of writing and reading and/or some dedicated hardware. As the SI SRAM uses closed loop control, all operations are controllable. Section 2 showed a writing fault which causes a deadlock eventually. The same problem can happen in reading. During reading, after precharging, if the leakage current is high, it will discharge the bit lines. As a result, RAck is generated wrongly. These leakage faults can be detected either from deadlocks or from wrong relationships in control signals.

[15] shows a low cost method to check asynchronous handshake protocols at run time. It can detect both deadlocks and wrong relationships.

Data retention voltage (DRV) is an important parameter of a highly reliable system for low power operations. For idle memory banks, lowering the supply to the DRV enables dramatic reduction of standby power. However the DRV depends on the extremes of local mismatch variation [4] and design-time decisions may need to be so conservative as to render this technique useless.

DRV-tracking methods exist [4]. One is to bias the array supply to twice the threshold voltage of the bit-cell devices. Although this is sufficient to preserve data, it is not a necessary condition. Further reduction is possible. Another is the canary cell method which must be calibrated against a known DRV distribution. Most recently a new efficient method was proposed in [4]. It includes a DRV column, which is added to a memory bank as a sensor, and a DRV prediction algorithm, which reads the contents of the DRV sensor and processes results under different voltage skews generated by a voltage skew generator. The DRV is detected by running a series of writing and reading operations, such as March, on the sensors. In addition to the dedicated sensors, it requires active testing which takes more time and consumes more power.

On the other hand, with the self-timed SRAM, when Vdd is reduced to a certain point, the controller stops working properly because of the simple sensing mechanism as discussed in the previous sections. From the experiments shown in Sections 2 and 4, it is clear that in this situation the data is still written into a cell correctly, but the sensing mechanism stops working. This Vdd is higher than the DRV, but lower than or equal to the minimum energy per operation point, which for 90nm technology is generally 0.3-0.4V. As a result, such self-timed SRAM can be regarded as self-sensing for usable DRV in a low-power context. A deadlock or conflict detector (e.g. [15]) can then be used to generate a warning signal to raise system Vdd.

## 7 Conclusions and Future Work

This paper investigates in detail problems in the self-timed 6T SRAM and highlights their limitations to support large bit lines across a wide range of PVT variations.

A new self-timed 10T SRAM is proposed and designed, which outperforms the 6T solution with a better ability to support large bit lines with the support of new bit-line keepers at the expense of a 67% bigger area. This allows memory systems to potentially work under supply voltages which are nondeterministic or naturally or deliberately variable as required in most energy conscious systems nowadays. The new SRAM is implemented in UMC 90nm CMOS technology under the Cadence toolkits, and demonstrated through Spectre analogue simulation across all process corners. As

an example, a 32Kb memory bank is implemented and the new SRAM together with the bit line keepers improves the robustness down to 0.3V under PVT variations.

As closed loop control is used in the self-timed SRAM designs, all operations of the memory sub-system are controllable. This provides a potential opportunity to build highly reliable systems. This will be our next future research topic.

A primary idea in this SRAM cell is to avoid using n-type transistors to propagate logic one. The 10T SRAM cell may not be the only solution fitting this description but the most conveniently available. We plan to further explore other types of cells.

We have fabricated the self-timed 6T SRAM, and will fabricate the 10T SRAM as well to verify our designs and prove the measurements.

**Acknowledgement.** This work is supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/G066728/1 "Next Generation Energy-Harvesting Electronics: Holistic Approach," website: www.holistic.ecs.soton.ac.uk. The first author is also supported by a scholarship from Umm Al-Qura University in the Kingdom of Saudi Arabia.

## References

[1] Baz, A., Shang, D., Xia, F., Yakovlev, A.: Self-Timed SRAM for Energy Harvesting Systems. In: van Leuken, R., Sicard, G. (eds.) PATMOS 2010. LNCS, vol. 6448, pp. 105–115. Springer, Heidelberg (2011)

[2] Baz, A., Shang, D., Xia, F., Yakovlev, A.: Self-Timed SRAM for Energy Harvesting Systems. Journal of Low Power Electronics 7(2), 274–284 (2011)

[3] Gupta, S.K., Raychowdhury, A., Roy, K.: Digital Computation in Subthreshold Region for Ultralow-Power Operation: A Device–Circuit–Architecture Codesign Perspective. Proceedings of the IEEE 98(2), 160–190 (2010)

[4] Qazi, M., Stawiasz, K., Chang, L., Chandrakasan, A.P.: A 512kb 8T SRAM Macro Operating Down to 0.57V With an AC-Coupled Sense Amplifier and Embedded Data-Retention-Voltage Sensor in 45nm SOI CMOS. IEEE Journal of Solid-State Circuits 46(1), 85–96 (2011)

[5] Kulkarni, J.P., Roy, K.: Ultralow-Voltage Process-Variation-Tolerant Schmitt-Trigger-Based SRAM Design. IEEE Transactions on Very Large Scale Integration (VLSI) Systems (not published yet)

[6] Sit, Wing-Yun, V., Choy, C.-S., Chan, C.-F.: A Four-Phase Handshaking Asynchronous Static RAM Design for Self-Timed Systems. IEEE Journal of Solid-State Circuits 34(1), 90–96 (1999)

[7] Dama, J., Lines, A.: GHz Asynchronous SRAM in 65nm. In: Proceedings of Asynchronous Circuits and Systems. In: 15th IEEE Symposium on ASYNC 2009, May 17-20, pp. 85–94 (2009)

[8] Chang, M.-F., Yang, S.-M., Chen, K.-T.: Wide Vdd Embedded Asynchronous SRAM With Dual-Mode Self-Timed Technique for Dynamic Voltage Systems. IEEE Transactions on Circuits and Systems I: Regular Papers 56(8), 1657–1667 (2009)

[9] Noguchi, H., Okumura, S., Iguchi, Y., Fujiwara, H., Morita, Y., Nii, K., Kawaguchi, H., Yoshimoto, M.: Which is the Best Dual-Port SRAM in 45-nm Process Technology?— 8T, 10T single end, and 10T differential —. In: IEEE International Conference on Integrated Circuit Design and Technology and Tutorial, 2008. ICICDT, June 2-4, pp. 55–58 (2008)

[10] Martin, A.J.: The Limitations to Delay-Insensitivity in Asynchronous Circuits. In: Dallyed, W.J. (ed.) Advanced Research in VLSI, pp. 263–278. MIT Press, Cambridge (1990)
[11] Amrutur, B.S., Horowitz, A.: A Replica technique for wordline and sense control in low power SRAM's. IEEE Journal of Solid-State Circuits 33(8), 1208–1219 (1998)
[12] Sparsø, J., Furber, S.: Principles of Asynchronus Circuit Design: A System Perspective. Kluwer Academic Publishers, Boston (2001)
[13] Chen, Q., Mahmoodi, H., Bhunia, S., Roy, K.: Modeling and Testing of SRAM for New Failure Mechanisms due to Process Variations in nanoscale CMOS. In: Proceedings of the 23rd IEEE VLSI Test Symposium 9VTS 2005 (2005)
[14] Dilillo, L., A1-Hashimi, B.M., Rosinger, P., Girard, P.: Leakage Read Fault in Nanoscale SRAM:Analysis, test and Diagnosis. In: Proceedings of the International Design and Test Workshop, Duday, November 19-20 (2006)
[15] Shang, D., Yakovlev, A., Burns, F., Xia, F., Bystrov, A.: Low Cost Online Testing of Asynchronous Handshakes. In: Proceeding of European Testing Symposium (ETS), May 21-25 (2006)
[16] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. IEICE Transactions on Information and Systems E80-D(3), 315–325 (1997)

# Architecture Extensions for Efficient Management of Scratch-Pad Memory

José V. Busquets-Mataix, Carlos Catalá, and Antonio Martí-Campoy

Department of Computer Engineering, Universidad Politécnica de Valencia,
Camino de Vera s/n, 46022-Valencia, Spain
{vbusque,amarti}@disca.upv.es, ccatala@grupoazahar.com

**Abstract.** Nowadays, many embedded processors include in their architecture on-chip static memories, so called scratch-pad memories (SPM). Compared to cache, these memories do not require complex control logic, thus resulting in increased efficiency both in silicon area and energy consumption. Last years, many papers have proposed algorithms to allocate memory segments in SPM in order to enhance its usage. However, very few care about the SPM architecture itself, to make it more controllable, more power efficient and faster. This paper proposes architecture extensions to automatically load code into the SPM whilst it is fetched for execution to reduce the SPM updating delays, which motivates a very dynamic use of the SPM. We test our proposal in a derivation of the Simplescalar simulator, with typical embedded benchmarks. The results show improvements, on average, of 30.6% in energy saving and 7.6% in performance compared to a system with cache.

**Keywords:** Embedded processors, memory architecture, scratch-pad memory.

## 1 Introduction

In recent years, the commercial popularity of mobile embedded devices such as phones, PDAs, cameras, MP4 players, etc. has attracted strong economic interests. As a consequence, much research effort has been accomplished to increase the computing power of such devices to be able to incorporate as much functionality as possible. However this increment in performance has been not accompanied by a equivalent increment in battery technology. Despite the great step forward due to *lithium-ion* batteries, since then, larger energy consumption requires larger battery size. Consequently, while battery technology slowly advances, the effort should be made to reduce the energy consumption of mobile embedded devices.

Compared to general purpose computing, there is an important key characteristic in these devices that may be exploited: most of the workload is somewhat fixed and known at design time. Therefore, some techniques may be used to allocate code and data objects to a lower stage in the memory hierarchy (i.e. SPM).

In general computing, caches have played a decisive role in providing the memory bandwidth required by processors. In fact, they became as one important technique to

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 43–52, 2011.

reduce the famous "memory bottleneck". The caches memory has a very dynamic and unpredictable behavior, capable of adapting its contents to any unknown workload. However, it is not energy efficient because it requires tag memory and the hardware comparison logic. Some authors have becoming to identify the memory subsystem as the energy bottleneck of the entire system [3].

High energy consumption of the cache, and predictable workload in embedded computing, has led the SPM memory to emerge as an efficient alternative to caches. In addition to its energy efficiency, it is fully predictable, playing an important role in real-time systems. The disadvantages are derived from the fact that the SPM is basically a small and fast memory mapped into the address space of main memory. Therefore, its operation must be done explicitly by mapping memory objects by the linker and loader, or by programming.

To do so, many approaches have been presented this decade to carefully select the contents to be stored in SPM to improve energy and/or performance. Orthogonal to them, this paper presents an original approach to reduce the overhead resulting from updating the SPM contents at run time.

The contributions of the paper are new architectural extensions to dynamically control the SPM. The difference among others solutions is that SPM loading is done on the fly whilst code is fetched from memory for execution, with minimum time and energy cost. This fact will enable allocating techniques to dynamically adapt the contents of the SPM to the program run at a reduced delay cost. Moreover, these techniques will trade-off to favor frequent updates, adapting the SPM contents to the program flow in a more effective and precise way

Our proposal only requires small changes on the processor design. Our interest is to obtain a realistic solution, simple enough to be implemented in a real world.

This paper is structured as follows. Section 2 reviews the related work. Section 3 proposes the overall architecture of our approach. The experimental setup is explained in section 4. Section 5 discusses the experimental results obtained. Finally, section 6 concludes the paper.

## 2 Related Work

In the literature, there are many works that focus on reducing the energy consumption and/or increasing performance by means of the effective use of SPM memories. These papers present the SPM as worthy alternative to cache memories, when energy and not only performance is important.

Many of these studies present a range of techniques on the allocation of code in the SPM which can be divided into two types. First, those of a static approximation where the contents of the SPM are assigned in advance and remain unchanged during program execution [1], [2], and [4]. Second, the ones that perform a dynamic update of the SPM contents at run time: Egger et al. [5] [6] and [7], Hyungmin Cho et al. [8], Janapsatyat et al. [9], Steinke et al. [10], Polleti et al. [11] , Lian Li et al. [13] and Doosan et al. [19]. The latter have the advantage of adapting the contents of the SPM to the program run but at the cost of periodically reloading the SPM contents.

There are some papers that propose hardware extensions to better control the SPM [9], [11], [12], and [13]. In [9] Janapsatyat et al. introduce a special set of instructions at compile time in a number of key points using a heuristic algorithm, which trigger a hardware controller that manages the flow of data to the SPM. To the best of our knowledge, this is the technique that better approximates to ours. However, the main advantage of our solution is that it requires fewer instructions and less control logic to operate.

Some papers propose the use of DMA to reduce the cost of copying data from main memory to the SPM [11], [19]. The main difference to our proposal is the larger die size and energy cost of this approach by using the DMA.

## 3 Architecture Extensions

Our proposal is based on a number of changes in the processor architecture, which may be classified in two categories. The first contain small changes required in the processor hardware design in order to support our approach. Second, three new instructions have been added to the instruction set. Below we explain these changes.

### 3.1 Hardware Design

Basically, the memory hierarchy is composed by the SPM and main memory. However, in order to take advantage of the spatial locality, we have added a prefetching buffer (see figure 1). This buffer behaves like a small cache memory with only one line in size. It will help in reduce the energy power and latency for those sequential fragments of code that are not selected to reside in SPM.

The SPM will be updated dynamically at run time on the fly to contain loops and functions that are executed frequently. No explicit load instructions are needed. The processor has three execution modes: *memory* mode, *SPM* mode, and *SPM function* mode. Changes among modes are controlled by three specific instructions.

In *memory* mode, instructions are brought to instruction decoder from main memory through the one-line-buffer (OLB) to exploit spatial locality. In *SPM* and *SPM function* mode, instructions are fetched from SPM memory. However, before the instructions may be used from SPM, they should be loaded to SPM from memory. This mechanism may be compared to a cache miss.

These modes require some hardware changes. We add a second program counter, so called SPM_PC and a tag register containing the memory address of the first instruction in SPM. A refinement of the technique proposes that the SPM may be split in independent partitions or blocks. Each one will include a tag register (as shown in figure 2). This schema may be used to hold in SPM different functions and/or loops at the same time. However, this architecture is not comparable to cache, since SPM partitions are much larger than cache lines, and consequently, there are only few tag registers in SPM compared to cache. We also need a small tag controller for comparison and update, and a mechanism to invalidate the whole SPM partition in one cycle.

### 3.2 Architectural Issues

To deal with the executions modes proposed in former section, three new instructions have been added to the processor architecture: *SPM_start*, *SPM_call_start* and *SPM_end*. Both *SPM_start* and *SPM_call_start* include an immediate field containing the SPM block number to be used. This block is selected by the programmer. These instructions are inserted into the original code by the compiler or programmer to tell the processor which pieces of code are selected to execute from SPM. For instance, when a loop is selected, two instructions are inserted to mark the bounds of the code: *SPM_start* at the beginning of the loop code, and *SPM_end* at the end.

When a *SPM_start* instruction is executed, the following actions take place: first, processor changes to *SPM* running mode. Second, the counter SPM_PC is initialized to the beginning of the SPM block (explicitly chosen by the *SPM_start*). Next, the physical address of the instruction *SPM_start* is compared to the tag register corresponding to the chosen SPM block. In case both addresses are equal, it means the contents of the SPM block correspond to the instructions in main memory that follow the *SPM_start*. Thus, the code is fetched from SPM. Both SPM_PC and PC counters are incremented simultaneously to point to two instances of the same instruction, one in main memory, and a copy in SPM. This schema will allow continuing execution from main memory once the end of the SPM code is reached (either by reaching the end of SPM block, or reaching *SPM_end* instruction). This allows dealing with loops larger than the SPM block. Knowing SPM size, loops may also spread several SPM blocks.

If tag comparison misses, the running code is not in the SPM block. The new starting address is copied to the tag register and the SPM block contents are invalidated. Next, the instructions are fetched from main memory to perform both, SPM load, and execution. This operation may be compared to a cache cold start. For the second and following pass of the loop, instructions are fetched from SPM.

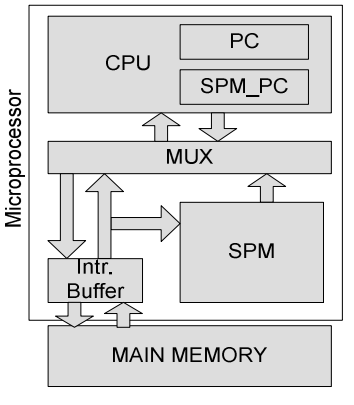



**Fig. 1.** Architecture

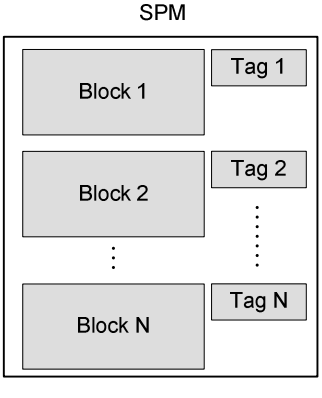



**Fig. 2.** SPM blocks

The former mechanism works well for loops and pieces of code frequently used. However, we propose an additional instruction, *SPM_call_start,* to deal with functions that the source code is not available to the programmer (i.e. functions in compiled libraries). It works as follows:

When the PC reaches a *SPM_call_start*, processor changes to *SPM_function* mode. Instructions are fetched from memory until a "call to function" instruction is found. Once the jump is taken, the target address (beginning of the function) is compared to the tag of the selected SPM block. The process that follows is somewhat similar to the one described for the *SPM_start* instruction. There is only one additional difference. The processor mode is changed to *memory* mode, once the end of the function is reached (return instruction). Therefore, it is not necessary the insertion of the *SPM_end*. This process may be seen in figure 3. Any call instruction to allocated functions, must be preceded by a *SPM_call_start*. Otherwise, the functions will be executed normally from main memory, without any SPM benefit.

There are also some particular cases that require a detailed explanation. The SPM is loaded at the same time instructions are brought from memory for execution. Therefore, it is possible to have some instructions only in memory until a given iteration requires their execution (i.e. *if then else* structure inside the loop). The processor must realize whether a location in SPM contains a valid instruction. In cache, this is solved at line by line basis, through costly tag comparison. Our approach cope with this issue performing a quick erase (invalidate) of the entire SPM block. This is accomplished in one processor cycle by a special hardware attached to the SPM memory addressing circuitry. The approach takes care to avoid any overhead in controlling the SPM. See figure 4 for the overall fetching process.

The SPM is mapped in the same physical address space than main memory. A key benefit of our approach is that it does not require virtual memory manager, allowing its use in medium to small embedded processor. However, we have to take special care of any flow change (*jump, call*). For a piece of code that is brought from memory to SPM, for the point of view of the architecture, the instructions are changing their physical addresses.

The main structures that require jump instructions are *loop* and *if then else*. Both of them use branch instructions with relative offset, thus no absolute addressing is necessary. Regarding function calling, the returning address is stored in stack. The processor have to select either pushing the SPM_PC or the PC depending on where is placed the call instruction. When the return is executed, the program flow returns to the caller code, irrespective it is in SPM or in memory. The processor has to switch automatically between *memory* and *SPM* modes.

## 4 Experimental Setup

In order to compare the cache against the SPM, we have used the simulator *Vatios* [14]. *Vatios* is a simulator based on the popular *SimpleScalar* framework [15]. Similarly to *Wattch* simulator [16], *Vatios* adds a model to calculate the energy consumption of both entities, memory and processor.

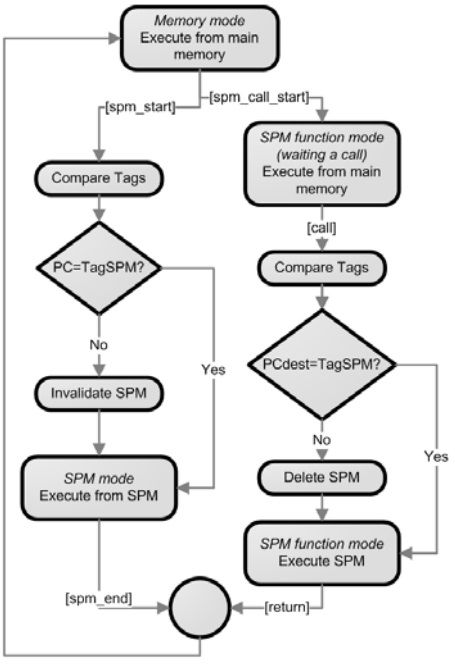



**Fig. 3.** Fetching process

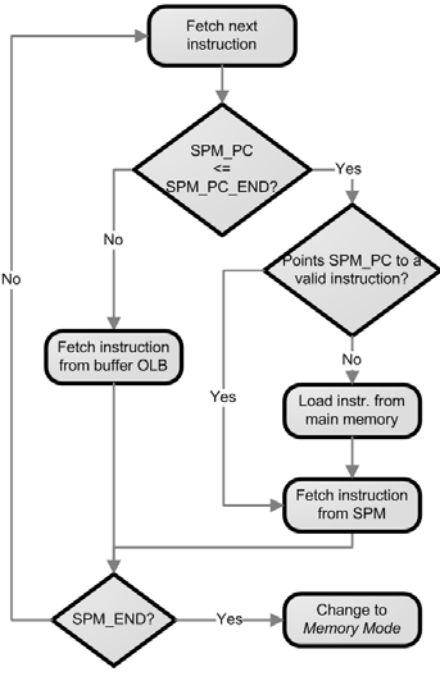



**Fig. 4.** Instruction fetch in SPM mode

*Vatios* presents a series of advantages in the calculation of the energy consumption with respect to *Wattch*. To calculate the energy consumption of the SPM and the cache, *Vatios* is based on the energy model called *Cacti* [17]. The SPM energy efficiency are due basically to the reduced control circuitry compared to the cache. To allow a fair comparison, both memories have been simulated using the same manufacturing technology.

Attending to the intended target architecture of this technique, we have selected realistic benchmarks. In particular, we have chosen a collection of programs selected by the The Mälardalen WCET research group [18]. They are representative programs for embedded systems, and mainly intended to be used in WCET analysis tools. We have selected the following:

**Bsort100:** Bubblesort program. Tests the basic loop constructs, integer comparisons, and simple array handling by sorting 100 integers.

**Cnt:** Counts non-negative numbers in a matrix. Nested loops, well-structured code.

**Compress:** Data compression program. Adopted from SPEC95 for WCET-calculation. Only compression is done on a small buffer containing totally random data.

**Cover:** Program for testing many paths. A loop containing many switch cases.

**Expint:** Series expansion for computing an exponential integral function. Inner loop that only runs once, structural WCET estimate gives heavy overestimate.

**Fdct:** Fast Discrete Cosine Transform. Many calculations based on integer array elements.

**Fir:** Finite impulse response filter (signal processing algorithms) over a 700 items long sample. Inner loop with varying number of iterations, loop-iteration dependent decisions.

The processor architecture of the simulator has been modified to incorporate the proposed approach. Since the simulator version that we used only offers a cache memory, we have implemented the SPM from scratch. We have considered the speed and energy models for this kind of memory. The decoder unit has accommodated the new instructions and, the necessary additional registers (SPM program counter) have been added. We have validated the correctness of the implemented extensions by exhaustive running of real workload.

The SPM control instructions have been inserted into the code by heuristics. We have not used any automatic allocation technique. The hot spots in the program can be easily identified by profiling. Benchmarks that are focused to WCET have help in this task.

The experimental process is as follows: first, from the C source code of benchmarks, we have added the SPM control instructions. The resulting code is compiled with *sslittle-gcc*. The binary programs are simulated by the *Sim-Vatios* simulator to obtain a trace. This trace is used by the tool *Power-Vatios* to obtain the energy consumption.

Regarding the hardware configuration, the benchmarks are simulated over three different cache or SPM sizes: 128, 256 and 512 bytes. The cache is direct mapped. The SPM has only one block. This is due the fact that the programs considered do not have concurrent hot spots. Therefore, the optimal configuration is a larger and unique block, but it may be updated frequently, thanks to the reduced overhead of the approach.

## 5 Experimental Results

The obtained results are depicted in figure 5 to 10. We can see that for 128B (Figures 5 and 6) the SPM performs better in both performance and energy consumption across all benchmarks. The SPM provides an average improvement in performance by 17%, and 29% in energy consumption with respect to the cache. The better results are displayed for the *fir* benchmark in which our approach obtains an improvement in performance of 44% and energy consumption 53%.

For 256B sizes (Figures 7 and 8) for the eight benchmarks used, only *expint* has better performance using a cache. This program consists of two nested loops where the outer loop cannot be entirely placed into the SPM and the most inner loop is executed only under certain circumstances. This makes that the cache takes advantage for this case. Summarizing, the SPM 256B has a 9% improvement in performance and 31% in energy consumption with respect to the cache.

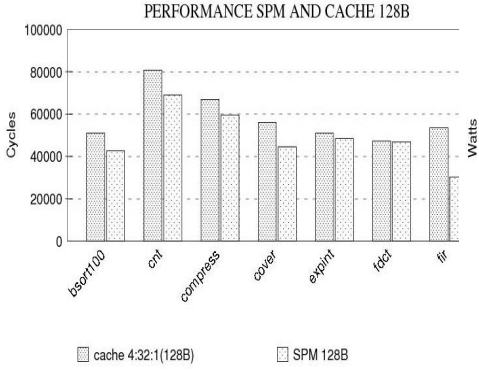



**Fig. 5.** Perfomance 128B

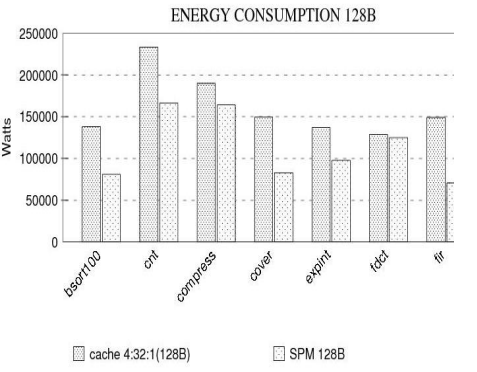



**Fig. 6.** Energy comsumption 128B

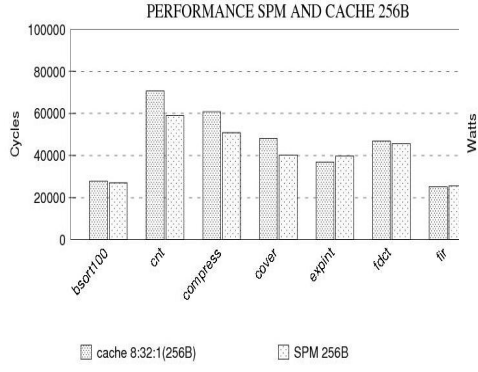



**Fig. 7.** Performance 256B

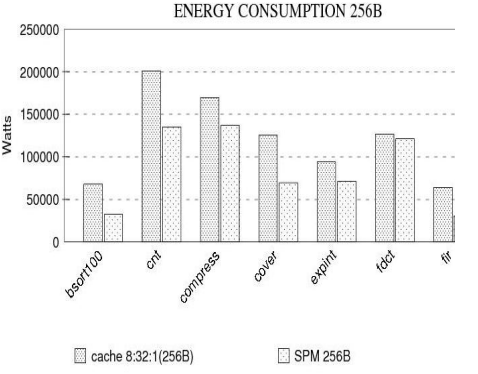



**Fig. 8.** Energy consumption 256B

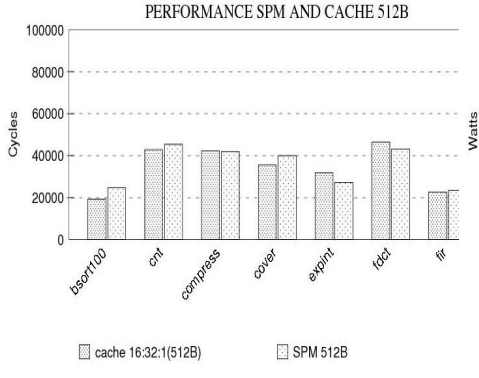



**Fig. 9.** Performance 512B

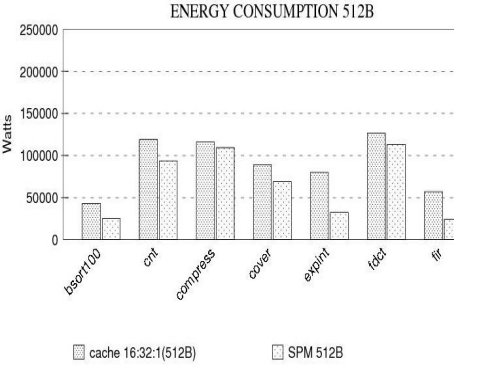



**Fig. 10.** Energy Consumption 512B

Finally, for SPM and cache sizes of 512B (Figures 9 and 10) we note that with respect to the performance, both of them behave similarly, but the cache shows the best results in five of the eight benchmarks. These differences are not significant showing, in average, a 3% in performance loss for the SPM with respect to the cache. The differences are larger in terms of energy consumption, but in this case the SPM outperform the cache in all benchmarks, presenting an average of 32% improvement with respect to the cache.

In general, we can observe that SPM slightly beats the cache in performance (7.6% on average), but largely reduces the energy consumption by 30.6% on average. It is important to mention that this results would be even better for SPM, if we were used a cache of the same silicon die size than the SPM. For simplicity reasons, we have compared directly both structures with same byte sizes. Many other works in the literature use the more fair comparison over the same die size.

## 6 Conclusions and Future Work

This paper has presented an original approach to better control the scratch-pad memory in embedded processors in order to reduce energy consumption. The key idea is to reduce as much as possible the overhead resulting from updating the SPM contents at run time. This allows allocating techniques to dynamically adapt the contents of the SPM to the workload execution, maximizing the number of hot spots that may be loaded into SPM.

The technique is orthogonal and complementary to many solutions presented to allocate objects in SPM. Those solutions may benefit and increase the effectiveness adopting the proposed architectural extensions.

The proposed technique has been compared to a instruction cache over a typical workload for embedded systems. On average, compared to a processor with an on-chip instruction cache of the same byte size, our approach improves performance by 7,6% and reduces energy consumption by 30,6%. For certain workloads, our approach has reached an increment of 44% in performance, and a reduction in power around 53%.

This paper has exploited the reductions on energy of the SPM. Future work will focus on the predictable nature of the SPM to exercise our technique in order to obtain better worst case execution times for real-time systems.

**Acknowledgments.** This research was sponsored by local Government “Generalitat Valenciana” under project GV07/ 2007/122.

## References

1. Banakar, R., Steinke, S., Lee, B.-S., Balakrishnan, M., Marwedel, P.: Scratchpad memory: design alternative for cache on-chip memory in embedded systems. In: CODES 2002, pp. 73–78 (2002)
2. Verma, M., Wehmeyer, L., Marwedel, P.: Cache-Aware Scratchpad Allocation Algorithm. In: DATE 2004, pp. 1264–1269 (2004)

3. Verma, M., Marwedel, P.: Advanced memory optimization techniques for low-power embedded processors, pp. I-XII, 1–188. Springer, Heidelberg (2007)
4. Nguyen, N., Dominguez, A., Barua, R.: Memory allocation for embedded systems with a compile-time-unknown scratch-pad size. In: CASES 2005, pp. 115–125 (2005)
5. Egger, B., Kim, C., Jang, C., Nam, Y., Lee, J., Min, S.L.: A dynamic code placement technique for scratchpad memory using postpass optimization. In: CASES 2006, pp. 223–233 (2006)
6. Egger, B., Lee, J., Shin, H.: Scratchpad memory management for portable systems with a memory management unit. In: EMSOFT 2006, pp. 321–330 (2006)
7. Egger, B., Lee, J., Shin, H.: Dynamic scratchpad memory management for code in portable systems with an MMU. ACM Trans. Embedded Comput. Syst. 7(2) (2008)
8. Cho, H., Egger, B., Lee, J., Shin, H.: Dynamic data scratchpad memory management for a memory subsystem with an MMU. In: LCTES 2007, pp. 195–206 (2007)
9. Janapsatya, A., Parameswaran, S., Ignjatovic, A.: Hardware/software managed scratchpad memory for embedded system. In: ICCAD 2004, pp. 370–377 (2004)
10. Balakrishnan, M., Marwedel, P., Wehmeyer, L., Grunwald, N., Banakar, R., Steinke, S.: Reducing Energy Consumption by Dynamic Copying of Instructions onto Onchip Memory. In: ISSS 2002, pp. 213–218 (2002)
11. Poletti, F., Marchal, P., Atienza, D., Benini, L., Catthoor, F., Mendias, J.M.: An integrated hardware/software approach for run-time scratchpad management. In: DAC 2004, pp. 238–243 (2004)
12. Li, L., Gao, L., Xue, J.: Memory Coloring: A Compiler Approach for Scratchpad Memory Management. In: IEEE PACT 2005, pp. 329–338 (2005)
13. Lee, L.H., Moyer, B., Arends, J.: Instruction fetch energy reduction using loop caches for embedded applications with small tight loops. In: ISLPED 1999, pp. 267–269 (1999)
14. Victorio, J.A., Torres Moren, E.F., Yúfera, V.V.: Vatios: Simulador de Procesador con Estimación de Potencia. XVIII Jornadas de Paralelismo, Zaragoza (2007)
15. Burger, D., Austin, T.M.: The SimpleScalar Tool Set Version 2.0. Technical Report 1342, Computer Sciences Department. University of Wisconsin–Madison (May 1997)
16. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: a framework for architectural-level power analysis and optimizations. In: ISCA 2000, pp. 83–94 (2000)
17. Tarjan, D., Thoziyoor, S., Jouppi, N.: CACTI 4.0, P. HPL-2006- 86 20060606
18. The Mälardalen WCET research group. The Mälardalen WCET benchmarks homepage, `http://www.mrtc.mdh.se/projects/wcet/benchmarks.html`
19. Cho, D., Pasricha, S., Issenin, I., Dutt, N.D., Ahn, M., Paek, Y.: Adaptive Scratch Pad Memory Management for Dynamic Behavior of Multimedia Applications. IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD) 28(4), 554–567 (2009)

# Pass Transistor Operation Modeling for Nanoscale Technologies*

Panagiotis Chaourani[1], Ilias Pappas[1], Spiros Nikolaidis[1], and Abdoul Rjoub[2]

[1] Physics Department, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
[2] Jordan University of Science and Technology, Computer Eng. Dept., 22110 Irbid, Jordan

**Abstract.** This paper studies the operation of the pass transistor structure taking into account secondary effects which become intense in nanoscale technologies. The different regions of operation are determined and the differential equation which describes the pass transistor operation is solved analytically. Appropriate approximations about the current waveforms are used simplifying the modeling procedure without significant influence on the accuracy. The evaluation of the model was made through comparison with HSpice simulation results and by using three different technologies: CMOS 65 nm, CMOS 32 nm and CMOS 32nm with high-K dielectric.

**Keywords**: pass transistor, modeling, timing simulation, operation analysis.

## 1 Introduction

Pass transistor is very often being used in digital designs, not only as a stand-alone device, but, also, as part in simple logic gates (XOR) or more complex circuits, like multiplexers and full adders [1]. Lately, pass transistors are used in the FPGA systems for the implementation of the Look–Up–Tables (LUTs), the switch and connection boxes and the tristate buffers. The main advantages of the pass transistor use are the reduction of the total silicon area and the power consumption caused by the minimization of the parasitic capacitance.

Many analytical models have been development, in the last years, for the CMOS inverter [2], the logic gates [3] and the pass transistor [4]. These models were referred to submicron MOSFETs and they were derived by solving analytically the differential equations which describe the circuits operation. These differential equations result by applying the Kirchhoff's law for the currents at the output nodes of the circuits. Based, up to now, on submicron devices, the used current models have not taken into account the effects rising from the scaling of the technology in the nano-scale regime. The modern MOSFET technologies (below 90 nm) impose intensively the existence of short–channel effects, like the channel length modulation, causing a significant degradation of the accuracy of the existing models for nanoscale technologies. Moreover, as the channel length and the thickness of the oxide of the transistors are reduced, the impact of the

* This work has been supported by FP7, ERA-WIDE JEWEL project 266507, 2010 -2013, funded by European Union.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 53–62, 2011.

leakage current is important because it becomes comparable to the drain current of the transistor. Therefore, the leakage current is expected to influence the behavior of the pass transistor and the other circuits.

In this paper, the operation of the pass transistor will be studied for charging the load capacitance by applying a ramp signal at its gate input. The driving current model that will be used takes into account the carrier velocity saturation effect and the channel length modulation effect having a significant impact for the nanoscale technologies. The influence of the leakage current in the behavior of the pass transistor will be studied by applying the proposed models in two technologies with different leakage current characteristics.

## 2 Pass Transistor Structure

The topology of the pass transistor that will be studied is shown in fig. 1. The pass transistor is selected to be an n-type MOSFET which is more commonly used. The load capacitance ($C_{out}$) corresponds to the input capacitance of the next logic level plus the parasitic capacitance of the connection lines. Finally, the capacitance $C_{GS}$ represents the Gate-to-Source parasitic capacitance of the pass transistor.

The operation of the pass transistor is based on driving the next logic level. This means that node A is set to logic "0" (low) or "1" (high) and by applying an input ramp at the gate of the pass transistor (node B), the load capacitance is charged or discharged, through the pass transistor, depending on the condition of node A.

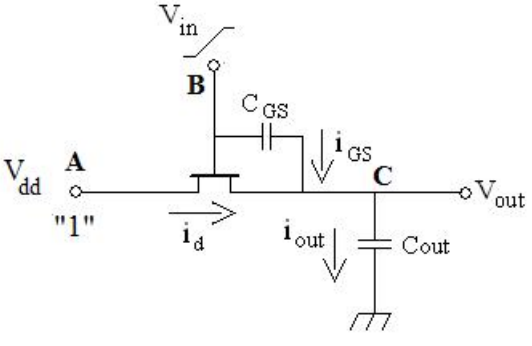



**Fig. 1.** Basic pass transistor topology with an input ramp at the gate of the pass transistor and by including the gate-to-source capacitance ($C_{GS}$)

The operation condition of setting node A to "0" corresponds to an inverter with a ramp voltage at its input which will cause the n-type MOSFET to turn ON and the output capacitance to be discharged. When node A is settled at "1", the pass transistor operates constantly at saturation region ($V_{DS} \geq V_{GS} - V_{TH}$, where $V_{TH}$ is the threshold voltage of the transistor) and the output capacitance will be charged with the drain current of the pass transistor. This condition will be described in the following study. Both cases have been solved analytically and the behavior of the pass transistor is well described.

As the MOSFET technologies were scaled down, the impact of some effects are strengthen, like that of the channel length modulation. This is obvious from the DC

output characteristics of the devices. In the existing models, describing the behavior of the pass transistor, such effects were not taken into account posing limitation in their application in nano-scale technologies.

In order to describe the transistor's drain current, the alpha–power law model [5] was chosen. This model takes into account the carrier velocity saturation effect and a term which describes the channel length modulation has been added. Therefore, the drain current of the transistor in the saturation region is expressed by the equation

$$I_{out} = k_s(V_{GS} - V_{TH})^{\alpha}(1 + \lambda V_{DS}) \tag{1}$$

where $k_s = W\mu C_{ox}/2L$ is the transconductance of the transistor in the saturation region, W is the channel width, L is the channel length, μ is the carrier mobility, $C_{OX}$ is the gate oxide capacitance, α is the velocity saturation index and λ is the channel length modulation coefficient. These parameters can be calculated from measured/simulated output characteristics of the devices. Parameter *a* is close to 1 for these technologies and this approximation is used to simplify the proposed model without significant influence on the accuracy.

The transistor's source node is not connected to the ground, as the bulk node, resulting in the appearance of the body–effect [6]. The expression of the body–effect is rather complicated, therefore the threshold voltage is expressed by its first order Taylor series approximation around $V_{SB} = 0$, as:

$$V_{TH} = V_{THO} + bV_{SB} \tag{2}$$

where $V_{THO}$ is the zero-bias threshold voltage and $V_{SB}$ is the source-to-bulk voltage. The analysis has shown that the deviation between the threshold voltage derived from Taylor series and the complicated body–effect expression is less than 3.9 %, for $V_{GS}$ = 1 V, making this approximation appropriate for our analysis.

By applying the Kirchhoff's current law at node C, the differential equation which describes the circuit operation is:

$$C_{out}\frac{dV_{out}}{dt} = k_s(V_{GS} - V_{TH})^{\alpha}(1 + \lambda V_{DS}) + C_{GS}\left(\frac{dV_{in}}{dt} - \frac{dV_{out}}{dt}\right) \tag{3}$$

In the saturation region, the parasitic capacitance $C_{GS}$ is calculated by the expression: $C_{GS} = 2WLC_{OX}/3$. Different operating conditions result for the pass transistor according to the evolution of the input signal. Equation (3) has to be solved for every region of operation, separately.

## 3 Modeling the Pass Transistor Operation

The input ramp is described by the expression

$$V_{in} = \begin{cases} 0, & t \le 0 \\ \frac{V_{dd}}{\tau}t, & 0 < t \le \tau \\ V_{dd}, & t > \tau \end{cases} \tag{4}$$

where τ is the time interval needed for the input ramp to make a transition.

During the circuit operation, as the output capacitance is charged, two cases for the input ramps are distinguished, namely fast and slow input ramps. The two cases are determined by the parameter τ and the characteristics of the circuit. The output capacitance is charged with the transistor's drain current and the current that flows from the parasitic capacitance $C_{GS}$. As the output current ($I_{out} = I_d + I_{Cgs}$) increases, the output voltage also increases. If eq. (5) is satisfied, the output current will become constant and, therefore, a plateau will appear (slow input ramp). However, if the increase rate of the input ramp is high enough, depending on the characteristics of the circuit (W, $C_{out}$), eq. (5) may never be satisfied resulting in a continuous increase of the output current. In this case no plateau will appear (fast input ramp).

$$\frac{dV_{in}}{dt} = (1+b)\frac{dV_{out}}{dt} \tag{5}$$

The existence of the plateau or not is shown in fig 2. In one condition, the rise time of the input ramp is small (τ = 0.005ns) and no plateau exists and in the other condition, the rise time is high (τ = 0.05ns) and the output current is constant for a particular region of operation. From fig. 2 it is shown that the output current increases almost linearly, while decreases almost exponentially for the last region of operation.

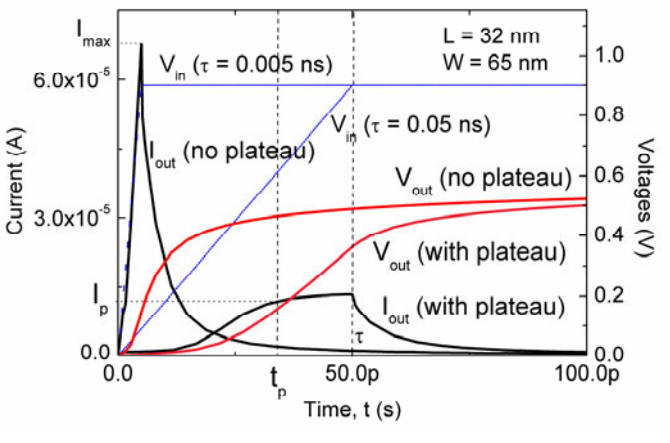



**Fig. 2.** Circuit response for two different conditions: i) input ramp having rise time τ = 0.05ns with current plateau and ii) input ramp with rise time τ = 0.005 ns without current plateau

### 3.1 First Region of Operation

At the beginning of the input ramp rising ($t < t_{th}$), no conduction path has been formed in the transistor channel because $V_{GS} < V_{TH}$. In this case, the output capacitance can, only, be charged through the parasitic capacitance $C_{GS}$. The differential equation that describes this situation is $I_{out}=I_{Cgs}$:

$$C_{out}\frac{dV_{out}}{dt} = C_{GS0}\left(\frac{dV_{in}}{dt} - \frac{dV_{out}}{dt}\right)$$

$$\frac{dV_{out}}{dt} = \frac{\kappa V_{dd}}{\tau} \tag{6}$$

where the $C_{GS0}$ is the parasitic capacitance between gate and source when the transistor operates in the cut-off region.

By assuming that the output capacitance is completely discharged at the beginning ($V_{out\,(t=0)} = 0$), the solution of the above differential equation is

$$V_{out1} = \frac{\kappa V_{dd}}{\tau} t \tag{7}$$

The differential Eq. (6) is valid until the input voltage becomes equal to the threshold voltage. This will occur at time $t_k$. Time $t_\kappa$ can be calculated, by the condition $V_{GS}=V_{TH}$:

$$V_{in} - V_{out} = V_{THo} + bV_{out}$$

$$t_k = \frac{V_{TH0}\tau}{V_{dd}[1-\kappa(1+b)]} \tag{8}$$

Finally, for this region, the output current will be equal to

$$I_{out1} = C_{out}\frac{\kappa V_{dd}}{\tau} \tag{9}$$

### 3.2 Second Region of Operation

#### A. Fast Input Ramp

As the input ramp voltage increases ($t_{th}<t<\tau$), a conduction path is formed at the transistor and the output capacitance is charged with the transistor's drain current and the $I_{Cgs}$. The output current increases almost linearly with time until a maximum value $I_{max}$ at $t = \tau$. Thus, the output current can be approximated as:

$$I_{out} = I_d + I_{Cgs}$$

$$I_{out} = \frac{I_{max}-I_{out1}}{\tau-t_k}(t - t_\kappa) + I_{out1} \tag{10}$$

The output voltage ($V_{out}$) results as the integration of the current:

$$V_{out2} = V_{out1}(t_k) + \frac{1}{C_{out}}\left[\frac{I_{max}-I_{out1}}{2(\tau-t_k)}(t - t_k)^2 + I_{out1}(t - t_k)\right] \tag{11}$$

i. Calculation of $I_{max}$

The value of the output current at $t = \tau$ is given by:

$$I_{out}(\tau) = k_s(V_{GS} - V_{TH})^a(1 + \lambda V_{DS}) + C_{GS}\left(\frac{dV_{in}}{dt} - \frac{dV_{out2}(\tau)}{dt}\right)$$

$$I_{max} = k_s\left[V_{dd} - (1 + b)V_{out2}(\tau)\right][1 + \lambda V_{dd} - \lambda V_{out2}(\tau)] \tag{12}$$

#### B. Slow Input Ramp

For the slow input ramp, the current increases up to a value $I_p$ at time $t_p<\tau$ and then remains constant (plateau value) until time $\tau$. The output current will be

$$I_{out} = \frac{I_p - I_{out1}}{t_p - t_k}(t - t_k) + I_{out1} \tag{13}$$

where $I_p$ is the value of the plateau and $t_p$ is the time that the plateau is appeared.

The output voltage will be given by the expression

$$V_{out2} = V_{out1}(\tau_k) + \frac{1}{C_{out}}\left[\frac{I_p - I_{out1}}{2(t_p - t_k)}(t - t_k)^2 + I_{out1}(t - t_k)\right] \tag{14}$$

i. Calculation of $I_p$

When time is equal to $t_p$, the value $I_p$ will be

$$\frac{dV_{in}}{dt} = (1 + b)\frac{dV_{out}}{dt}$$

$$I_p = \frac{C_{out}V_{dd}}{\tau(1+b)} \tag{15}$$

ii. Calculation of $t_p$

The value of the $I_p$ can be expressed as:

$$I_p = k_s(V_{GS} - V_{TH})^a(1 + \lambda V_{DS}) + C_{GS}\frac{dV_{in}}{dt} - C_{GS}\frac{dV_{out}}{dt}$$

$$I_p = k_s\left(\frac{V_{dd}}{\tau}t_p - (1 + b)V_{out2}(t_p) - V_{TH0}\right)\left(1 + \lambda V_{dd} - \lambda V_{out2}(t_p)\right) -$$

$$bC_{GS}\frac{dV_{out2}(t_p)}{dt} \tag{16}$$

By solving eq. (16), the value of $t_p$ is calculated. If $t_p<\tau$, the slow input ramp will occur, otherwise, the fast input ramp will occur.

### 3.3 Third Region of Operation

#### A. Fast Input Ramp

When $\tau<t<\infty$, the output current has to turn to zero, otherwise the output capacitor charge (Q), and the output voltage, will tend to be infinity instead of a finite value. The form of the output current can be approximated according to

$$I_{out} = \frac{v}{t^2} \tag{17}$$

The output voltage will be given by the expression

$$\int_{V_{out}}^{V_f} dV_{out3} = \frac{1}{C_{out}}\int_t^{\infty}\frac{v}{t^2}dt$$

$$V_{out3} = V_f - \frac{v}{C_{out}t} \tag{18}$$

where $V_f$ is the value of the output voltage when $t \rightarrow \infty$. Theoretical, this voltage value is equal to the voltage needed to turn the transistor off. Therefore, the value of $V_f$ results by the condition $V_{GS}=V_{TH}$:

$$V_f = V_{off} = \frac{V_{dd} - V_{TH0}}{1+b} \tag{19}$$

Simulation results have shown that there is a significant difference between the model and HSPICE results for the third region of operation. According to them the output voltage do not have a finite value at the end of the third region of operation but increases constantly towards Vdd with a decreased rate which never turn to zero. This is caused because of the weak inversion effect. Therefore, Eq. (19) has to be redefined. The expression of $V_f$ has to take into account the gate width W, the output capacitance $C_{out}$ and the time constant τ. By applying the necessary calculation, the derived expression of $V_f$ is

$$V_f = V_{off} + \frac{I_{tot}}{C_{out}+C_{GS}} \left( \ln \frac{W}{W_o} \right) \Delta t \tag{20}$$

where $W_o$ is the channel width for which the output voltage becomes equal to $V_{off}$ and $I_{tot}$ is the total current when $V_D = V_G = V_{dd}$, $V_s = V_{off}$ and $W = W_o$

### A. Slow Input Ramp

For the slow input ramp, the third region of operation begins at time $t_p$ until time τ. In this region the output current is constant and equal to the plateau value. Therefore,

$$V_{out3} = \frac{1}{C_{out}} \int I_p dt$$

$$V_{out3} = \frac{1}{C_{out}} I_p (t - t_p) + V_{out2}(t_p) \tag{21}$$

Furthermore, for the slow input ramp, a fourth region of operation exists which is similar to the third region of operation for the fast input ramp.

## 4 Simulation Results

The evaluation of the proposed analysis and the study of the leakage current influence were made through comparisons between the proposed model and HSpice simulation results. For the HSpice simulations, a CMOS 65 nm, a CMOS 32 nm and a CMOS 32nm with high–k gate dielectric technologies were chosen (BSIM4 Predicted Technology Models). The characteristics of the pass transistor ($V_{dd}$, $V_{TH0}$, $k_s$ and λ) were extracted from the specifications of each technology. In Table 1, the values of all the parameters for each CMOS technology are presented.

**Table 1.** Parameters' values for each used CMOS technology

| Parameter | 65nm technology | 32nm technology | High – k 32nm technology |
|---|---|---|---|
| W (nm) | 130 | 65 | 65 |
| L (nm) | 65 | 32 | 32 |
| $V_{TH0}$ (mV) | 423 | 508.8 | 355.8 |
| λ | 0.04 | 0.02 | 0.06 |
| $k_s$ (x $10^{-4}$ A/V) | 2.216 | 2.066 | 2.94 |
| $C_{out}$ (fF) | 1 | 0.5 | 0.5 |

Figure 3 shows the comparison between the model and HSpice simulation results for the three selected technologies. The solid lines represent the HSpice simulation results, while the symbols represent the model results. The good agreement between the model and the HSpice results indicates the correctness of the proposed model. Furthermore, the smallest deviation was measured for the 32nm technology with high-K dielectric and this is caused due to the limitation of the leakage current. The high–K dielectric minimizes the leakage current of the transistor, especially the one resulting from the gate tunneling effect. Therefore, the theoretical value of the output current ($I_{out} = I_d + I_{Cgs}$) is not affected significantly by the leakage current.

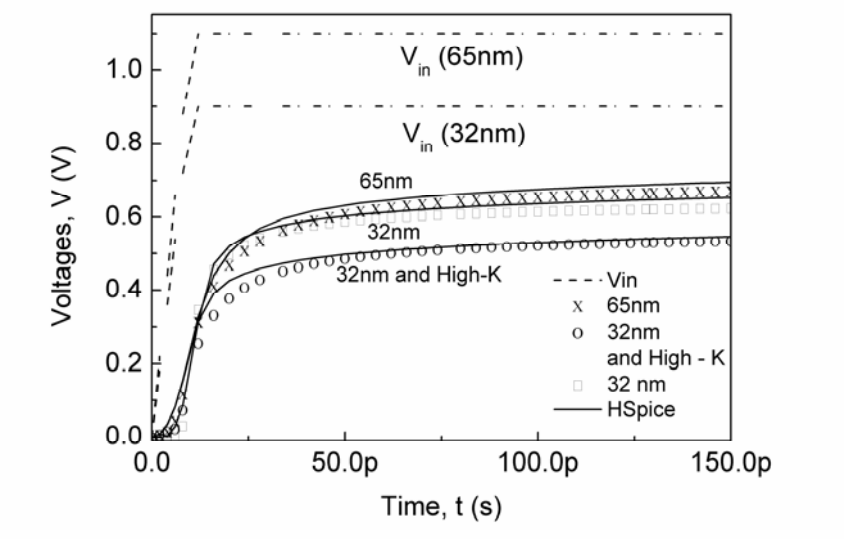



**Fig. 3.** Comparison between the model results (symbols) and the HSpice simulation results (solid lines) for the three selected technologies (65nm, 32nm, 32nm with high–K dielectric)

For having a better evaluation of the proposed model, the comparison was extended for various values of the circuit parameters, like the pass transistor width, the output capacitance value and the rise time of the input ramp. The results of the comparison are shown in Table 2 for the 65 nm technology and in Table 3 for the 32 nm and 32nm with high–K technology. In all conditions there is a very good agreement between the results, showing the accuracy of the model.

**Table 2.** Comparison between model and HSpice simulation results for various values of the circuit parameters for 65nm technology

| $C_{out}$ = 0.5fF, W = 65nm τ (ns) | Mean Deviation % | $C_{out}$ = 0.5fF, τ = 0.01ns W (nm) | Mean Deviation % | τ = 0.01ns, W = 65nm $C_{out}$ (fF) | Mean Deviation % |
|---|---|---|---|---|---|
| 0.005 | 3.95 | 90 | 5 | 0.5 | 4.2 |
| 0.01 | 4.9 | 130 | 4.9 | 1 | 4.9 |
| 0.05 | 7.6 | 300 | 1.95 | 2 | 5.7 |

**Table 3.** Comparison between model and HSpice simulation results for various values of the circuit parameters for 32nm and 32 nm with high-K dielectric technology

| $C_{out}$ = 0.5fF, W = 32nm τ (ns) | Mean Deviation % (32nm) | Mean Deviation % (32nm, high-K) | $C_{out}$ = 0.5fF, τ = 0.01ns W (nm) | Mean Deviation % (32nm) | Mean Deviation % (32nm, high-K | τ = 0.01ns, W = 65nm $C_{out}$ (fF) | Mean Deviation % (32nm) | Mean Deviation % (32nm, high-K |
|---|---|---|---|---|---|---|---|---|
| 0.005 | 7.5 | 6.4 | 65 | 5.4 | 5.4 | 0.2 | 3.4 | 7.9 |
| 0.01 | 5.4 | 5.8 | 90 | 7.2 | 4.1 | 0.5 | 5.4 | 5.8 |
| 0.05 | 22 | 17.3 | 160 | 6.7 | 6.7 | 1 | 8.2 | 5.2 |

In Table 4, the comparison between the propagation delay measured with the model's results and that of HSPICE simulations are shown, for 32nm and 32nm with high-K technology. The propagation delay in measured from the Vdd/2 time point of the input signal to Vdd/2 time point of the output signal. As it can be seen from the comparison, the error in the 32nm technology with high–K dielectric is smaller than the ordinary 32nm technology. This can be explained by the impact of the leakage current on the load capacitance charge. The high-K dielectric minimizes the gate tunneling leakage current leading to a very good correlation between the theoretical values of the model and the simulated ones. It is shown that for ordinary nanoscale transistors the gate tunneling leakage current plays an important role in their behavior and it has to be taken into account for more accurate results.

**Table 4.** Comparison of the propagation delay between the model and HSpice simulation results, for 32nm and 32nm with high-K dielectric technologies

| Ordinary 32nm | | | | 32nm with high-K | | | |
|---|---|---|---|---|---|---|---|
| $C_{out}$ = 0.5fF, W = 32nm τ (ns) | Propagation delay $t_d$ HSpice (ps) | Propagation delay $t_d$ Model (ps) | % error | $C_{out}$ = 0.5fF, W = 32nm τ (ns) | Propagation delay $t_d$ HSpice (ps) | Propagation delay $t_d$ Model (ps) | % error |
| 0.005 | 15.1 | 18.2 | 17 | 0.005 | 4.9 | 5.5 | 10.9 |
| 0.01 | 21.5 | 27 | 20 | 0.01 | 7.5 | 8 | 6.25 |
| 0.05 | 32 | 54 | 40.7 | 0.05 | 21 | 21.5 | 2.3 |
| Ordinary 32nm | | | | 32nm with high-K | | | |
| $C_{out}$ = 0.5fF, τ = 0.01ns W (nm) | Propagation delay $t_d$ HSpice (ps) | Propagation delay $t_d$ Model (ps) | % error | $C_{out}$ = 0.5fF, τ = 0.01ns W (nm) | Propagation delay $t_d$ HSpice (ps) | Propagation delay $t_d$ Model (ps) | % error |
| 65 | 21.5 | 27 | 20 | 65 | 7.5 | 8 | 6.2 |
| 90 | 16.5 | 18 | 8.3 | 90 | 6.5 | 6 | 7.7 |
| 160 | 11 | 18 | 39 | 160 | 5 | 5 | 0 |
| Ordinary 32nm | | | | 32nm with high-K | | | |
| τ = 0.01ns, W = 65nm $C_{out}$ (fF) | Propagation delay $t_d$ HSpice (ps) | Propagation delay $t_d$ Model (ps) | % error | τ = 0.01ns, W = 65nm $C_{out}$ (fF) | Propagation delay $t_d$ HSpice (ps) | Propagation delay $t_d$ Model (ps) | % error |
| 0.2 | 10 | 14.5 | 25.6 | 0.2 | 5 | 4.5 | 10 |
| 0.5 | 21.5 | 27 | 20 | 0.5 | 7.5 | 8 | 6.25 |
| 1 | 28 | 36.5 | 23 | 1 | 9.5 | 11 | 13.6 |

## 5 Conclusion

In this paper, the operation of the pass transistor structure was studied. The channel length modulation effect was taken into account resulting in the validation of the model to nano-scale technologies. The evaluation of the model was made through comparison between the model and HSpice results by using three different nanoscale technologies. Furthermore, the significance of the leakage current was testified by comparing the model results of an ordinary 32nm technology with a 32nm technology with high-K dielectric which minimizes the gate tunneling leakage current.

## References

1. Zimmermann, R., Fichtner, W.: Low power logic styles: CMOS versus pass-transistor logic. IEEE Journal of Solid-State Circuits 32, 1079–1090 (1997)
2. Bisdounis, L., Nikolaidis, S., Koufopavlou, O.: Analytical Transient Response and Propagation Delay Evaluation of the CMOS Inverter for Short-channel Devices. IEEE Journal of Solid-State Circuits 33(2), 302–306 (1998)
3. Chatzigeorgiou, A., Nikolaidis, S., Tsoukalas, I.: A Modeling Technique for CMOS Gates. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 18(5), 557–575 (1999)
4. Nikolaidis, S., Nikolaidis, T.: Analyzing the Operation of the Basic Pass Transistor Structure. International Journal of Circuit Theory and Applications 35, 1–15 (2007)
5. Sakurai, T., Newton, A.R.: Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. IEEE Journal of Solid State Circuits 25, 584–594 (1990)
6. Rabaey, J.M., Chandrakasan, A., Nikolic, B.: Digital Integrated Circuits; A design perspective, 2nd edn. Prentice Hall, Englewood Cliffs (2003)

# Timing Modeling of Flipflops Considering Aging Effects

Ning Chen, Bing Li, and Ulf Schlichtmann

Institute for Electronic Design Automation, Technische Universitaet Muenchen,
Arcisstr. 21, 80333 Munich, Germany
{ning.chen,b.li,ulf.schlichtmann}@tum.de

**Abstract.** Traditionally, the timing of a flipflop is modeled by a single constraint pair of setup and hold times. For timing verification of digital circuits both timing constraints should not be violated. Furthermore, the interdependency of these two quantities is exploited and multiple constraint pairs are taken as valid setup and hold times. STA Tools can be easily constructed by the propagation of arrival times. In this paper, we present a comprehensive study of flipflop timing behavior and extend the timing modeling by explicitly building the functional relationship between clock-to-q delay and timing parameters at flipflop data input in order to break the timing boundaries and thus allow interdependency of different computation stages to be analyzed at gate level. Aging effects HCI and NBTI are also considered in the modeling to pave the way for aging analysis.

**Keywords:** Timing, Modeling, Flipflop.

## 1 Introduction

For static timing analysis (STA) at gate level, delay models for individual gates are applied. In the commonly used nonlinear delay models, all pin-to-pin delays and output slews of a combinatorial gate are stored in two dimensional look-up tables with input slew and output load as indexes. For the sequential cells such as flipflops, setup and hold times are specified at data input with clock slew and data slew as indexes into the look-up tables while the clock-to-q delay and output slew are specified at data output with clock slew and output load as indexes [1].

In STA, the propagation of arrival times is run once where the starting points are the primary inputs of the circuit under analysis or the data outputs of flipflops and the ending points are the primary outputs of the circuit and the data inputs of flipflops. For a circuit synthesized using a standard cell library, the output load of each gate is known after placement and routing. The clock slew is determined by the clock distribution network together with the input capacitance of sink clock pins. In STA, the propagation of arrival times begins by calculating the clock-to-q delay and data output slew of each flipflop. Both are known values which are obtained using look-up tables. During the propagation, signal slews

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 63–72, 2011.

are calculated locally. At the end of timing paths, the setup and hold times are viewed as fundamental properties of a flipflop determined by the clock slew and calculated data input slew and checked against violations.

Recent research on flipflop timing behavior reveals that the valid pair of setup and hold times is not unique [2,3]. Instead, all the pairs on a constant degradation curve of clock-to-q delay can be taken as valid pairs as in [3,4], due to the compensation effect between the time intervals of stable data before and after the triggering edge of clock signal. This effect is viewed as the interdependency of setup and hold times.

This non-linear constant degradation curve can be obtained by SPICE simulation, although the simulation overhead is high due to the large number of transient simulations. A method for fast calculation is proposed in [5,6]. With the help of this curve, the STA with one single constraint pair of setup time and hold time can be easily extended for the case of multiple pairs by choosing the minimum suitable setup time to determine the minimum clock period of the circuit. A piecewise linear approximation of the constant degradation curve is used in [4] and provides conservative results due to the convexity of this non-linear curve.

In this paper, we present a comprehensive study and investigate different factors which have influence on flipflop timing. Based on the observation, we propose a new modeling method by building the clock-to-q delay into a functional relationship with the timing parameters at flipflop data input. This can be viewed as a generalization of traditional modeling method using one or multiple constraint pairs. With this modeling, new challenges and opportunities for timing analysis arise because the timing boundaries naturally set by flipflops are broken and the timing of different computation stages depends on each other. Two aging effects Hot Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) are also considered to facilitate aging analysis.

The remainder of this paper is organized as follows. In Section 2 we explain the basics for timing modeling of flipflops and introduce the traditional modeling method using one or multiple constraints pairs. In Section 3 we give general remarks about the performances and factors related to flipflop timing. In Section 4 the individual factors affecting flipflop timing are analyzed and a new modeling method is proposed. Finally, Section 5 draws the conclusion.

## 2 Modeling Basics

The flipflop that we have taken for investigation comes from a standard cell library in 90nm technology of an industry partner. All the parasitic resistors and capacitors extracted after layout are contained in the flipflop netlist. For Spice simulation, the tool Spectre from Cadence is used.

In order to obtain a full picture of flipflop timing, we first give the definition of setup skew and hold skew which originates from the alignment of clock, data input and data output signals as shown in Fig. 1. These signals are labeled as as clk, d and q respectively. Setup skew is the time difference between the starting point of data and triggering clock edge while hold skew is the one between triggering clock edge and the ending point of data. Compared with setup time and

hold time, both skews are not viewed as fundamental properties of flipflops. For example, the setup skew changes correspondingly as the clock period changes. The clock-to-q delay is the time difference between the triggering clock edge and flipflop data output.

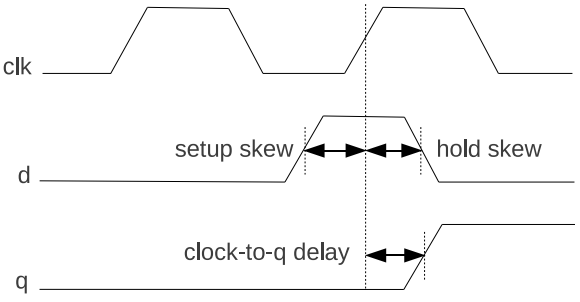



**Fig. 1.** Flipflop Modeling

Contrarily, setup time and hold time are normally viewed as fundamental properties of flipflops which set the time range to allow data passing through flipflops safely. They are specific skews which guarantee the functionality of flipflops. In Fig. 2, the dependency of clock-to-q delay on the two individual skews are shown. When one skew is taken for consideration, the counterpart skew is set to be large enough. As can be seen, the clock-to-q delay increases as the setup skew decreases and the same happens by hold skew. In order to obtain the setup time and hold time, the clock-to-q delay is allowed to degrade by a certain ratio, e.g. by 10%, compared with the nominal delay defined when both skews are large enough [6]. The corresponding setup and hold skews are chosen as setup and hold times respectively.

Obviously, this pair of setup and hold times ($Pair_\infty$) is over-optimistic, mainly due to the setting of the counterpart skew. As shown in Fig. 3 with four different counterpart skew values, the effect of one skew on clock-to-q delay is also related to the specific value of the counterpart skew. With the reduction of counterpart skew, the clock-to-q delay generally increases.

It is exactly at this point that the interdependency of setup time and hold time comes into play [4]. Due to the compensation effect of setup skew and hold skew, different pairs of these two quantities can be taken as valid pairs of setup and hold times based on the fact that they lead to the same ratio of degradation of clock-to-delay with respect to the nominal delay. The constant degradation curve with degradation ratio 10% is shown in Fig. 4. The diamond symbol under the curve is to show the location of $Pair_\infty$.

On this curve, different points can be defined [4]. The Minimum Setup-Hold Pair (MSHP) is the pair where the sum of setup skew and hold skew is the minimum while the Effective Setup Pair (ESP) and Effective Hold Pair (EHP) are user specified boundaries. Due to the convexity of the constant degradation

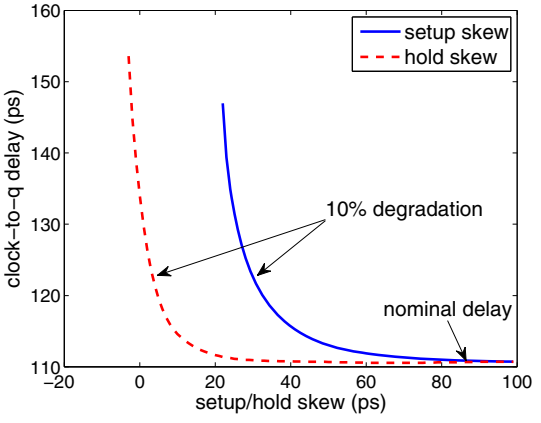
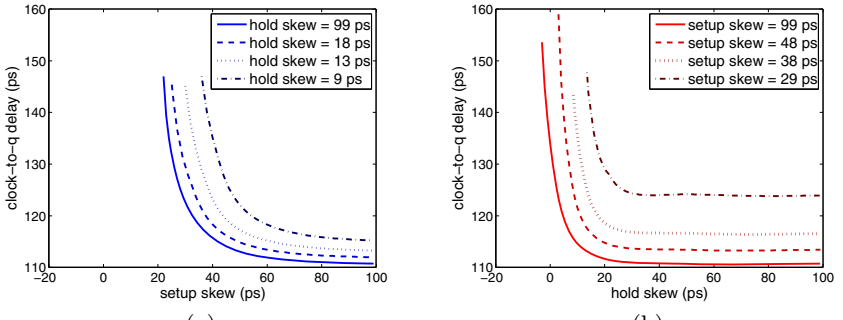



**Fig. 2.** Clock-to-q Delay on Setup/Hold Skew




(a) (b)

**Fig. 3.** Influence of Counterpart Skew on Clock-to-q Delay

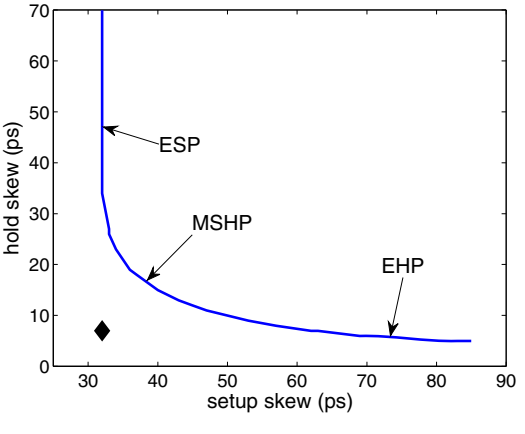



**Fig. 4.** Constant Degradation Curve

curve, further simplification is undertaken in [4] to use a piecewise linear approximation by connecting ESP, MSHP and EHP directly. The STA tool can be easily extended for using multiple pairs, because the only change is the choice of setup and hold times.

## 3 General Remarks

The timing of flipflops at data output can be represented by following two performances.

- clock-to-q delay
- output slew

The possible factors which have influence on the two performances above, are as follows.

- setup skew
- hold skew
- output load
- clock slew
- data slew at starting edge
- data slew at ending edge
- aging effects
- input data pattern
- environmental variation
- process variation

The traditional characterization of flipflops adopts the approach to define setup time and hold time as two additional performances based on the constant degradation curve of clock-to-q delay. This sets flipflops as natural boundaries for the propagation of arrival times and ignores the interdependency of timing beyond the boundaries. The main purpose of this paper is to build a new modeling to break the timing boundaries and to model the fact that in real circuits the clock-to-q delay changes dynamically depending on the input data pattern and the alignment of data with clock. The same cycle stealing effect as in latch based circuits [7] is to be expected.

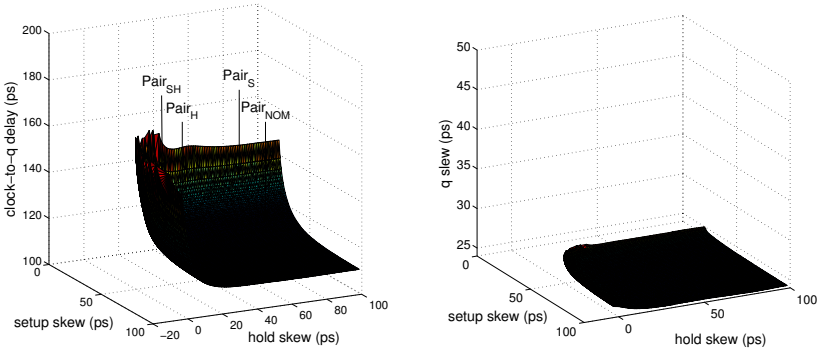
Of the factors mentioned above, the handling of different input data patterns as well as environmental variation and process variation is not included in this paper. But the proposed modeling can be easily extended for these factors.

## 4 Flipflop Modeling

In this section, a new modeling method for flipflop timing behavior is proposed in Subsection 4.1 based on simulation data. The effects of load and data/clock slew are investigated in Subsection 4.2. Aging effects are considered in Subsection 4.3.

### 4.1 Setup/Hold Skew

The surface of clock-to-q delay with respect to different setup/hold skew pairs is shown in Fig. 5(a). The value range in z-axis is set to be about one nominal delay to avoid optical misleading. As can be seen, the setup and hold skews have direct influence on the clock-to-q delay. It is not sufficient just to consider one specific degradation curve. The full delay surface should be modeled.




(a) (b)

**Fig. 5.** Performance Surface

To model this functional relationship explicitly, the following analytical function is used in this paper.

$$q = f_0(s,h) = a_0 + \frac{a_1}{s - s_0} + \frac{a_2}{h - h_0} \tag{1}$$

where $q$ is the clock-to-q delay, $s$ the setup skew and $h$ the hold skew. The constants $a_0$, $a_1$, $a_2$, $s_0$, $h_0$ are obtained by nonlinear fitting. $s_0$ and $h_0$ represent also the lower bounds for $s$ and $h$ respectively. This form is the best and simplest form that we have found with minimal fitting error. The proposed modeling approach is not limited to a specific function form. Its main strength lies in the direct modeling of the dependency of clock-to-q delay on setup skew and hold skew to facilitate analysis of the timing dependency in different computation stages.

The surface for output slew is shown in Fig. 5(b). As can be seen, the effect of skew on output slew is small. The traditional characterization based on clock slew and output load can further be used.

Based on Eq. (1), the interdependency of clock-to-q delays at different flipflops is built. In order to determine the clock-to-q delay $q$ for the current flipflop under consideration, all the clock-to-q delays of its fanin flipflops need to be known first in order to obtain the setup skew $s$ and hold skew $h$ of the current flipflop by using maximum and minimum calculation respectively. Furthermore, considering the common structure of loops where a path starts from the data output of a flipflop and traverses combinational circuits or other flipflops and comes back

into the data input of the source flipflip again, the determination of minimal clock period of the circuit can not be finished by using only one propagation run. The same cycle stealing effect as observed in latch based circuits [7] is to be expected.

By using the robust nonlinear fitting in Matlab based on totally 7741 sample points of the delay surface, the standard deviation of the absolute fitting error is about 9.6 ps. Generally, the fitting error becomes larger as the setup/hold skew becomes very smaller and thus the clock-to-q delay increases sharply. In Fig. 6 the error depending on an upper bound of clock-to-q delay relative to nominal delay is shown. For example, by 1.2 times the nominal delay, the error is about 1.6 ps. With respect to circuit path delay, this error can be safely neglected. A specific upper bound can be defined by the user to avoid a large fitting error on one side and the metastability issues as described in [8] on the other side.

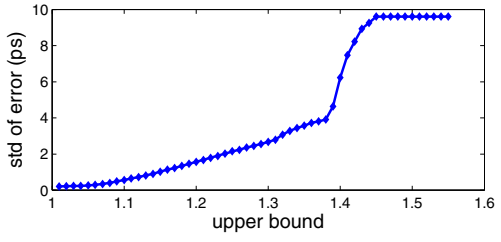



**Fig. 6.** Standard Deviation of Absolute Fitting Error

### 4.2 Load and Data/Clock Slew

For investigating the effects of load and data/clock slew on clock-to-q delay, We define four pairs of setup and hold skews representing different regions as shown in Fig. 5(a) where the three pairs $Pair_H$, $Pair_{SH}$ and $Pair_S$ are on the constant degradation curve in Fig. 4 while $Pair_{NOM}$ corresponds to the nominal delay.

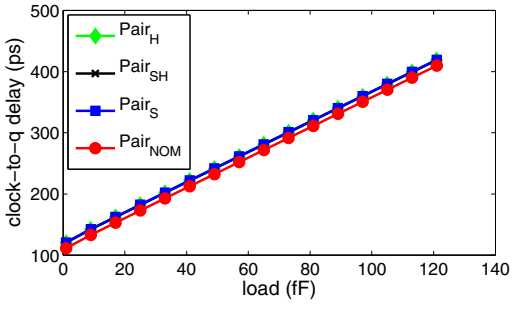



**Fig. 7.** Load Effects

The effects of load are shown in Fig. 7 for all four pairs. The curves for the pairs $Pair_H$, $Pair_{SH}$ and $Pair_S$ overlap with each other, because they correspond to the same delay degradation. For each of these four pairs, the clock-to-q delay has

a linear relationship with the load. Since all four curves are almost in parallel, the effects of load can be modeled as an additive term. We extend Eq. (1) to the following form to include the load effects.

$$q = f_1(s, h, c) = f_0(s, h) + c_0 \cdot c \tag{2}$$

where $c$ is the load and $c_0$ the fitting constant. In our experiment, the standard deviation of absolute fitting error for load is 0.7 ps.

The effects of clock/data slew are shown in Fig. 8 where both slews are defined as the time range between 10% and 90% of the whole voltage swing. In Fig. 8(a), the effects of clock slew on four skew pairs are different. There exists a linear relationship on $Pair_{NOM}$ while on other three pairs not. For characterization, a look-up table can be built with setup skew and hold skew as indexes to determine a multiplicative factor $c_{l_c}$ related with clock slew. An alternative to determine $c_{l_c}$ is to use a general nonlinear function as follows to describe the relationship for each region of setup skew and hold skew.

$$c_{l_c} = (l_{c0} + l_{c1} \cdot l_c^{l_{c2}}) \tag{3}$$

where $l_c$ is the clock slew and $l_{c0}$, $l_{c1}$, $l_{c2}$ the fitting constants.

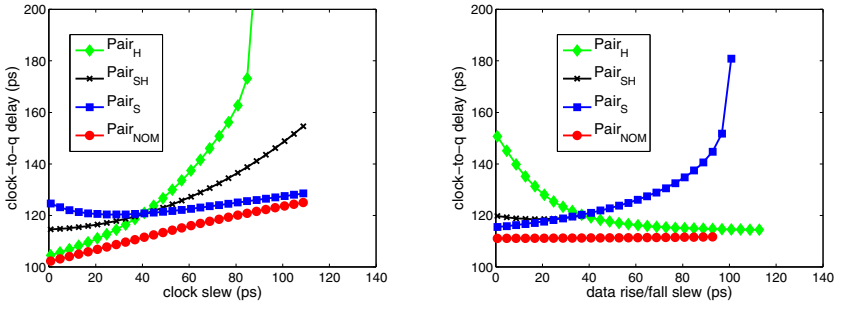



**Fig. 8.** Slew Effects

The effect of data slew is shown in Fig. 8(b). As long as the flipflop works in the nominal region around $Pair_{NOM}$, there is no influence of data slew on clock-to-q delay. Around $Pair_S$ and $Pair_H$, where the data slew of starting edge $l_{ds}$ and of ending edge $l_{de}$ take effect respectively, the influence is nonlinear and can be modeled by a multiplicative factor $c_{l_d}$ just like in the case of clock slew. This factor can be based on look-up tables or formulated in the following form,

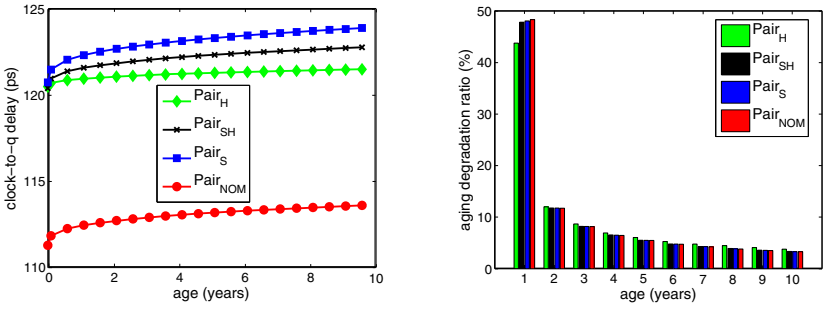
$$c_{l_d} = (l_{d0} + l_{d1} \cdot l_d^{l_{d2}}) \tag{4}$$

where $l_d$ is the data slew and $l_{d0}$, $l_{d1}$, $l_{d2}$ the fitting constants.

For an optimized circuit, where the path delays are relatively balanced, the flipflops along the critical paths are working in the region around $Pair_S$ while others around $Pair_{NOM}$. In this respect, the effect of data slew is a serious concern. However, the data slews of real circuits are normally limited in a narrow value range where $c_{l_d}$ can be approximated by a constant value.

### 4.3 Aging

Hot Carrier Injection (HCI) [9] and Negative Bias Temperature Instability [10] are the two most important aging effects. For aging simulation of flipflops, we consider HCI for N-type transistors while both HCI and NBTI for P-type transistors and use the tool Relxpert from Cadence. The aging models are obtained from the same industry partner as the flipflop. A period of ten years is taken into account. The results corresponding to the four pairs of setup and hold skews are shown in Fig. 9(a) respectively.




(a) (b)

**Fig. 9.** Aging Effects

As can be seen from Fig. 9(a), the clock-to-q delay increases nonlinearly with the increase of aging time, but with decreasing degradation rate. With different pairs of setup and hold skews, the degradation rates are different. Based on this observation, we can add a multiplicative aging factor $c_t$ by using look-up tables or a fitting function written as,

$$c_t = t_0 + t_1 \cdot t^{t_2} \tag{5}$$

where $t$ is the age, $t_0$, $t_1$ and $t_2$ are fitting parameters dependent on setup skew $s$ and hold skew $h$. Generally, the feasible area of setup skew and hold skew can be divided into different regions to simplify this dependency. In our experiment, the fitting errors using Eq. (5) for each of the four pairs is very small and the function can be safely used.

The degradation rate during the time is shown in Fig. 9(b), where the y-axis represents the aging degradation ratio in each year. This ratio $r_i$ is defined as,

$$r_i = \frac{q_i - q_{i-1}}{q_{10} - q_0}, 1 \leq i \leq 10 \tag{6}$$

where $q_i$ is the clock-to-q delay in $i$th year. As can be seen, a large percentage of the degradation occurs in the first years.

Putting all effects together, we can write the modeling function as follows.

$$q = (a_0 + \frac{a_1}{s - s_0} + \frac{a_2}{h - h_0} + c_0 \cdot c) \cdot c_{l_c} \cdot c_{l_d} \cdot c_t \tag{7}$$

## 5 Conclusion

In this paper, a comprehensive study of flipflop timing behavior is presented. Based on simulation data, a new modeling method is proposed which puts the clock-to-q delay into a functional relationship with setup and hold skews at data input. This facilitates the timing information going beyond the flipflop boundary. The load is built into the model based on the linear relationship while for clock/data slew it is proposed to use a multiplicative factor based on look-up tables or analytical forms. Aging effects are presented in the paper to make aging analysis possible.

**Acknowledgement.** This research was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89).

## References

1. The open source liberty library modeling format specification, `http://www.opensourceliberty.org/`
2. Lang, A., Bergler, S.: Method and apparatus for circuit verification of meeting setup and hold time requirements (Verfahren und Vorrichtung zum Ueberpruefen einer Schaltung auf Einhaltung von Setup- und Holdzeiten). Patent DE102004044668A1 (2004)
3. Salman, E., Dasdan, A., Taraporevala, F., Kucukcakar, K., Friedman, E.G.: Pessimism reduction in static timing analysis using interdependent setup and hold times. In: ISQED 2006, pp. 159–164 (2006)
4. Salman, E., Dasdan, A., Taraporevala, F., Kucukcakar, K., Friedman, E.G.: Exploiting setup-hold-time interdependence in static timing analysis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26(6), 1114–1125 (2007)
5. Srivastava, S., Roychowdhury, J.: Interdependent latch setup/hold time characterization via Euler-Newton curve tracing on state-transition equations. In: DAC 2007, pp. 136–141 (2007)
6. Srivastava, S., Roychowdhury, J.: Independent and interdependent latch setup/hold time characterization via Newton-Raphson solution and Euler curve tracking of state-transition equations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(5), 817–830 (2008)
7. Sakallah, K.A., Mudge, T.N., Olukotun, O.A.: Analysis and design of latch-controlled synchronous digital circuits. In: DAC 1990, pp. 111–117 (1990)
8. Gabara, T., Cyr, G., Stroud, C.: Metastability of CMOS master/slave flip-flops. In: CICC 1991, pp. 29.4/1–29.4/6 (1991)
9. Wu, L., Fang, J., Yan, H., Chen, P., Chen, A.I.-H., Okamoto, Y., Yeh, C.-S., Liu, Z., Iwanishi, N., Yonezawa, N.K.H., Kawakami, Y.: Glacier: A hot carrier gate level circuit characterization and simulation system for vlsi design. In: ISQED 2000, p. 73 (2000)
10. Kumar, S.V., Kim, C.H., Sapatnekar, S.S.: An analytical model for negative bias temperature instability. In: ICCAD 2006, pp. 493–496 (2006)

# Iterative Timing Analysis Considering Interdependency of Setup and Hold Times

Ning Chen, Bing Li, and Ulf Schlichtmann

Institute for Electronic Design Automation, Technische Universitaet Muenchen,
Arcisstr. 21, 80333 Munich, Germany
{ning.chen,b.li,ulf.schlichtmann}@tum.de

**Abstract.** The interdependency of setup and hold times of flipflops in digital circuits needs to be considered in order to obtain more accurate results of timing analysis. In this paper, an iterative STA method is developed based on a new modeling of flipflop timing behavior. Two basic problems are solved: whether a circuit can work at a given clock period, and how the minimal clock period is determined. Experimental results show that a reduction of the clock period by 3.3% can be achieved compared to traditional STA method.

**Keywords:** Iterative Timing Analysis, Interdependency.

## 1 Introduction

In a synchronous digital circuit, sequential cells such as flipflops coordinate the data/state transfer between different computation stages and play a central role. For the timing analysis of these circuits, setup and hold times are extracted for the sequential cells by the characterization process. Traditional Static Timing Analysis (STA) method is then applied to evaluate the circuit performance and to ensure that no violation of setup and hold times should occur.

In common practice the setup and hold times are characterized independently. However, recent research has shown that these two quantities are actually interdependent. This observation appeared first in [1]. In [2] and [3] the interdependency is represented by multiple constraint pairs of setup and hold times where a smaller setup time corresponds to a larger hold time and vice versa. All these constraint pairs, when becoming active, lead to a constant ratio of increase or degradation of clock-to-q delay, compared with the case when they are far away from active. These pairs are obtained using SPICE simulation and all are taken as valid to be integrated into a standard STA flow. In [4] and [5], the constant degradation curve used to define these multiple pairs is obtained directly by solving the system state transition equations instead of direct SPICE simulations. This method leads to much faster characterization. The idea of interdependency has been extended to statistical timing analysis in [6].

The use of constant degradation curve of clock-to-q delay for flipflop characterization avoids the timing dependency of different computation stages as seen in latch-based circuits [7]. The timing is only checked on the directed paths

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 73–82, 2011.

starting from the primary inputs or flipflop data outputs and ending at primary outputs or flipflop data inputs. However, if the clock-to-q delay is modeled as a variable which allows fluctuation as observed in reality, the assumption of independence of different computation stages is not valid any more. A new timing analysis method is to be developed in this respect.

In this paper, we have developed an iterative method for timing analysis to check whether a circuit can work at a certain clock period. Based on this iterative method, an algorithm using binary search has been designed to obtain the minimal clock period. The same cycle stealing effect as in latch-based circuits could be observed because of the compensation of unbalanced stage delays. The metastability problem as described in [8] is solved by setting an upper bound to the maximal clock-to-q delay allowed. The experimental results on the ISCAS89 benchmark synthesized using an industry standard cell library show the clock period can be reduced on average by 4.6% or 3.3% in comparison with the result of traditional STA based on one single constraint pair or multiple constraint pairs in [2] respectively.

The remainder of the paper is organized as follows. In Section 2 a new flipflop modeling is presented where the clock-to-q delay is formulated by an analytical function. In Section 3 the iterative timing analysis method based on the modeling is explained in detail where the basic problems are formulated and solutions for them are presented. Experimental results are shown in Section 4. Finally, Section 5 draws the conclusion.

## 2 Flipflop Modeling

Setup and hold times are normally characterized as fundamental properties of the flipflop under consideration. In traditional library characterization each quantity is obtained separately by looking up a two dimensional table with clock slew and data input slew as indexes while the clock-to-q delay is looked up in another table with clock slew and data output load as indexes. As can be seen, the clock-to-q delay is modeled without considering the alignment of data input with clock.

In order to obtain a full view of timing behavior of flipflop, the setup and hold skews are first defined as shown in Fig. 1.

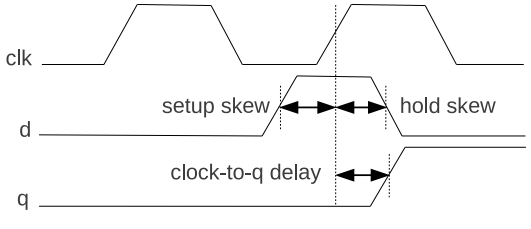
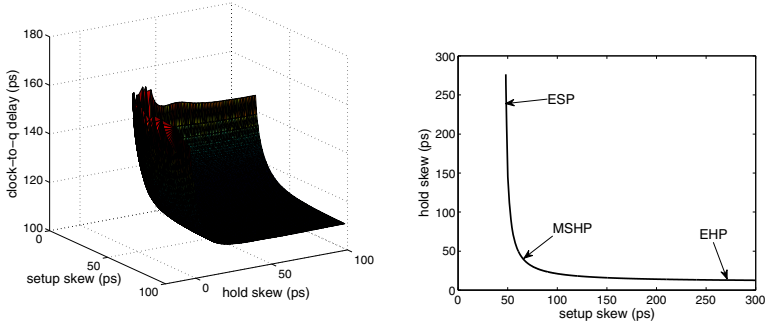



**Fig. 1.** Flipflop Modeling

The setup skew is the time difference between the starting point of data input and the triggering clock edge while the hold skew is the one between the

triggering clock edge and the ending point of data input. The clock-to-q delay is the delay from the triggering clock edge to the time point when the data is available at the output of the flipflop.

To see how the clock-to-q delay is affected by setup skew and hold skew, the flipflop is simulated by SPICE. The surface of the clock-to-q delay is obtained as shown in Fig. 2(a) with setup skew as x-axis, hold skew as y-axis and clock-to-q delay as z-axis. As can be seen, the clock-to-q delay has a constant value when the setup skew and hold skew are large enough. This delay is called the nominal clock-to-q delay.




(a) (b)

**Fig. 2.** Clock-to-q Delay Surface and Constant Degradation Curve

If the setup skew or hold skew becomes smaller, the clock-to-q delay increases. The 10% degradation curve is shown in Fig. 2(b) where all the pairs of setup skew and hold skew on this curve correspond to 1.1 times the nominal clock-to-q delay. Three special pairs are marked on the curve. The Minimal Setup-Hold Pair (MSHP) is taken where the sum of setup skew and hold skew is the minimum while the Effective Setup Pair (ESP) and the Effective Hold Pair (EHP) are user defined boundaries. The common characterization takes the MSHP as setup time and hold time while in [2] all the pairs between ESP and EHP are accepted as valid pairs of setup time and hold time.

The use of constant degradation curve puts aside the fact that there exists a functional dependency of clock-to-q delay on the timing parameters such as setup and hold skews at the data input. Setup and hold times are only artificial guard bands to guarantee the correct flipflop behavior. They are not fundamental properties of flipflops.

To model the functional relationship between clock-to-q delay and setup/hold skew explicitly, an analytical form is obtained while also taking the capacitive load and clock slew into account. This analytical function is shown as follows,

$$\begin{aligned} q &= f(s,h,c,l) \\ &= (a_0 + \frac{a_1}{s-s_0} + \frac{a_2}{h-h_0} + c_0 \cdot c) \cdot (l_0 + l_1 \cdot l^{l_2}) \end{aligned} \tag{1}$$

where $q$ is the clock-to-q delay, $s$ the setup skew, $h$ the hold skew, $c$ the capacitive load connected at the data output and $l$ the clock slew. The constants $a_0$, $a_1$, $a_2$, $c_0$, $s_0$, $h_0$, $l_0$, $l_1$ and $l_2$ are obtained by nonlinear fitting. According to our investigation, the influence of data input slew on clock-to-q delay can be ignored.

## 3 Iterative Timing Analysis

In this section, the iterative timing analysis method based on Eq. (1) is explained in detail. The new problem is first discussed in Subsection 3.1. The iterative method for STA with known clock period is presented in Subsection 3.2. In Subsection 3.3 the method to find the minimal clock period for a circuit is explained.

### 3.1 New Problem

In traditional STA, the clock-to-q delay of each flipflop in a circuit is obtained separately depending on its clock slew and capacitive load. Only one propagation run starting from primary inputs and flipflop outputs to primary outputs and flipflop data inputs is needed. However, with the new modeling from Eq. (1), the clock-to-q delays of different flipflops are interdependent.

The setup skew $s_j$ and hold skew $h_j$ of flipflop $j$ are calculated as

$$s_j = T - \max_{i \in A_j}(q_i + \bar{d}_{ij}) \tag{2}$$

$$h_j = \min_{i \in A_j}(q_i + \underline{d}_{ij}) \tag{3}$$

where $A_j$ is the set of all preceding fanin flipflops and primary inputs of $j$. $\bar{d}_{ij}$ and $\underline{d}_{ij}$ are logic delays from $i$ to $j$ for maximum and minimum calculations respectively. If $i$ corresponds to a primary input, then $q_i$ is set to a user defined value. Substitute Eq. (2) and Eq. (3) into Eq. (1), the clock-to-q delay of $j$ is obtained as

$$q_j = f(s_j(q_{i_1}, q_{i_2}, ..., q_{i_n}), h_j(q_{i_1}, q_{i_2}, ..., q_{i_n}), c_j, l_j) \tag{4}$$

where $s_j$ and $h_j$ have been written as functions of the clock-to-q delays of the preceding fanin flipflops, $n$ is the number of fanin flipflops in $A_j$.

In Eq. (4), the interdependency of the clock-to-q delays of different flipflops in a circuit is formulated. The existence of feedback loops commonly observed in a sequential circuit complicates this issue further, where, for example, the data output of a flipflop traverses a combinational circuit or even further flipflops and goes into the data input of the source flipflop again.

The equation system (2)-(4) actually corresponds to the real circuit behavior where the clock-to-q delays of individual flipflops fluctuate dynamically depending on the current alignment of data input with clock. This behavior is reflected by the solution of the equation system above.

### 3.2 Analysis for Constant Clock Period

The first problem to be solved is whether the circuit can work at a given clock period $T$. Due to the interdependency of individual clock-to-q delays as described in Subsection 3.1, the problem can not be solved easily. Therefore, an iterative algorithm (shown in Alg. 1) has been developed. The basic idea is to iterate the propagation of arrival times until all clock-to-q delays fulfill the equation system above.

The algorithm starts with setting the arrival time for each primary input $j$ to specified values $\tau_j$ and the initial value of each clock-to-q delay as $Q_j^\star$ corresponding to the nominal delay. For flipflop $j$, all the fanin flipflops and eventually primary inputs are stored in the set $A_j$. The variable $k$ is used as the iteration index and $K$ is the specified maximal number of iterations. Inside each iteration, the setup skew and hold skew are updated according to Eq. (2) and (3). The clock-to-q delay is calculated by Eq. (4). The setup skew and hold skew are checked against the lower bounds $s_0$ and $h_0$ required by Eq. (1) respectively while the clock-to-q delay of each flipflop $j$ is checked against the nominal one $Q_j^\star$ multiplied by a user specified ratio $\theta$ as a upper bound to take the metastability issue into account. If the bounds are not guaranteed, the algorithm breaks and the circuit is considered as failed. If not failed, the new clock-to-q delay is then compared with the one from the last iteration. If the difference is larger than a specified threshold value $\delta$, the system is still not converged. This is signified by the flag *state*. The iteration goes forward until the system is converged or the maximal iteration number is reached.

If each clock-to-q delay does not stabilize to a specific value, we check whether the delay series from totally $K$ iterations for each flipflop is periodic. If it is true, then the system is still thought as converged. The check of periodic sequence is not inside the iteration loop because generally the algorithm converges very fast and $K$ is a relatively small number. This can also be moved to be inside the iteration loop.

### 3.3 Determination of Minimal Clock Period

After we have developed the method to verify the circuit at a given clock period, the next problem to solve is to determine the minimal clock period $T_{min}$ at which the circuit can work. A natural approach is to decrease the clock period $T$ from an initial value gradually with a certain rate until the circuit can not work any more. At each $T$, Alg. 1 is used to check whether the circuit can still work.

The initial value $T_{init}$ of the clock period can be, for example, the result obtained using traditional STA.

$$T_{init} = \max_{i,j}(\theta_0 \cdot Q_i^\star + \bar{d}_{ij} + S_j) \tag{5}$$

where $Q_i^\star$ is the nominal clock-to-q delay of flipflop $i$, $\theta_0$ is a user specified value and normally 1.1, $\bar{d}_{ij}$ is the logic delay from flipflop $i$ to $j$ for maximum calculation, $S_j$ is the setup time of flipflop $j$.

**Alg. 1.** Iterative STA Method

```
1  foreach primary input j do
2  |  q_j^(0) = τ_j;
3  end
4  foreach flipflop j do
5  |  q_j^(0)=Q_j^⋆;
6  |  find A_j;
7  end
8  LOOP: for k = 1 to K do
9  |  set state as CONV;
10 |  foreach flipflop j do
11 |  |  s_j^(k) = T − max_{i∈A_j}(q_i^(k−1) + d̄_ij);
12 |  |  h_j^(k) = min_{i∈A_j}(q_i^(k−1) + d_ij);
13 |  |  q_j^(k) = f(s_j^(k), h_j^(k), cap_j);
14 |  |  if (s_j^(k) ≤ s_0) ∨ (h_j^(k) ≤ h_0) ∨ (q_j^(k) ≥ θ · Q_j^⋆) then
15 |  |  |  set state as FAIL;
16 |  |  |  break LOOP;
17 |  |  end
18 |  |  if |q_j^(k) − q_j^(k−1)| ≥ δ then
19 |  |  |  set state as NCONV;
20 |  |  end
21 |  end
22 |  if state = CONV then
23 |  |  break;
24 |  end
25 end
26 if state = NCONV then
27 |  foreach flipflop j do
28 |  |  if q_j^(k) is nonperiodic then
29 |  |  |  set state as FAIL;
30 |  |  |  break;
31 |  |  end
32 |  end
33 |  set state as CONV;
34 end
35 return state;
```

The obvious disadvantage of this approach is the large computation overhead if $T_{min}$ is far away from $T_{init}$. To solve this problem, a binary search method is proposed as shown in Alg. 2.

This algorithm starts with defining the search range of clock period where $T_{start}$ is set to 0.0 and $T_{end}$ to $T_{init}$. We update the range by checking whether the circuit can work at the middle point of $T_{start}$ and $T_{end}$ by Alg. 1. If the range is small enough to be below a threshold value $\delta$, the algorithm is terminated.

**Alg. 2.** Find $T_{min}$

```
1  set T_start as 0.0;
2  set T_end as T_init;
3  while true do
4  |  T_current = (T_start + T_end)/2;
5  |  get state by calling Alg. 1 with T_current;
6  |  if state = CONV then
7  |  |  T_end = T_current;
8  |  else
9  |  |  T_start = T_current;
10 |  end
11 |  if |T_end − T_start| ≤ δ then
12 |  |  T_min = T_end;
13 |  |  break;
14 |  end
15 end
```

## 4 Experimental Results

The proposed method was implemented in Java and applied to the ISCAS89 benchmark on a linux machine running at 3.0GHz. All the circuits were synthesized with a 90nm standard cell library from an industry partner using Cadence RTL Compiler. For comparison we implemented the traditional STA in two versions, firstly with setup time defined by MSHP and secondly by Multiple Constraint Pairs (MCP) as in [2]. We compare the minimal clock period achieved with our method with both of them. The results are shown in Table 1. The circuits are sorted using the number of flipflops as shown in the third column.

### 4.1 Comparison of Minimal Clock Period

In Table 1, $T_{MSHP}$ is the minimal clock period by traditional STA using MSHP to define setup time while $T_{MCP}$ the minimal clock period obtained in [2]. The result of our Iterative Timing Analysis (ITA) method is shown as $T_{ITA}$. By all of these experiments, the arrival times for primary inputs are set to 0.0. The user specified ratio $\theta$ is set to 1.2.

The ratio of the minimal clock period of ITA with respect to that of MSHP or MCP is defined as follows,

$$r_{MSHP|MCP} = \frac{T_{ITA}}{T_{MSHP|MCP}} \tag{6}$$

where $T_{MSHP|MCP}$ corresponds to either $T_{MSHP}$ or $T_{MCP}$. The corresponding two ratios are shown as $r_{MSHP}$ and $r_{MCP}$ in Table 1. The geometric means of the two ratios show that a reduction of clock period by 4.6% and 3.3% can be achieved respectively.

**Table 1.** Experimental Results

| index | circuit | #FF | $T_{MSHP}$ (ns) | $T_{MCP}$ (ns) | $T_{ITA}$ (ns) | $r_{MSHP}$ | $r_{MCP}$ | $t_{ITA}$ (s) |
|---|---|---|---|---|---|---|---|---|
| 1 | s27 | 3 | 0.401 | 0.406 | 0.364 | 0.908 | 0.897 | 0.01 |
| 2 | s820 | 5 | 0.697 | 0.694 | 0.716 | 1.028 | 1.032 | 0.31 |
| 3 | s832 | 5 | 0.742 | 0.726 | 0.732 | 0.986 | 1.008 | 0.22 |
| 4 | s1488 | 6 | 0.863 | 0.866 | 0.861 | 0.998 | 0.994 | 0.23 |
| 5 | s1494 | 6 | 0.813 | 0.792 | 0.754 | 0.928 | 0.953 | 0.34 |
| 6 | s386 | 6 | 0.601 | 0.587 | 0.573 | 0.953 | 0.976 | 0.05 |
| 7 | s510 | 6 | 0.710 | 0.704 | 0.699 | 0.986 | 0.993 | 0.07 |
| 8 | s208 | 8 | 0.452 | 0.446 | 0.475 | 1.052 | 1.066 | 0.04 |
| 9 | s298 | 14 | 0.578 | 0.578 | 0.535 | 0.925 | 0.925 | 0.06 |
| 10 | s641 | 14 | 0.794 | 0.775 | 0.746 | 0.939 | 0.962 | 0.01 |
| 11 | s713 | 14 | 0.726 | 0.706 | 0.664 | 0.915 | 0.941 | 0.07 |
| 12 | s344 | 15 | 0.661 | 0.640 | 0.604 | 0.914 | 0.944 | 0.07 |
| 13 | s349 | 15 | 0.680 | 0.659 | 0.620 | 0.911 | 0.940 | 0.07 |
| 14 | s420 | 16 | 0.491 | 0.484 | 0.484 | 0.986 | 1.000 | 0.08 |
| 15 | s1196 | 18 | 0.787 | 0.793 | 0.763 | 0.969 | 0.962 | 0.04 |
| 16 | s1238 | 18 | 0.762 | 0.766 | 0.727 | 0.954 | 0.949 | 0.06 |
| 17 | s382 | 21 | 0.694 | 0.675 | 0.640 | 0.923 | 0.949 | 0.07 |
| 18 | s400 | 21 | 0.683 | 0.664 | 0.629 | 0.921 | 0.947 | 0.08 |
| 19 | s444 | 21 | 0.664 | 0.666 | 0.630 | 0.949 | 0.946 | 0.07 |
| 20 | s526 | 21 | 0.699 | 0.683 | 0.647 | 0.925 | 0.947 | 0.09 |
| 21 | s953 | 29 | 0.838 | 0.843 | 0.797 | 0.951 | 0.946 | 0.06 |
| 22 | s838 | 32 | 0.526 | 0.507 | 0.492 | 0.935 | 0.969 | 0.16 |
| 23 | s1423 | 74 | 1.009 | 0.995 | 0.959 | 0.950 | 0.963 | 0.38 |
| 24 | s9234 | 125 | 0.979 | 0.973 | 0.951 | 0.971 | 0.978 | 0.43 |
| 25 | s5378 | 160 | 0.805 | 0.798 | 0.778 | 0.967 | 0.975 | 0.41 |
| 26 | s13207 | 426 | 1.035 | 1.042 | 0.999 | 0.965 | 0.959 | 1.04 |
| 27 | s15850 | 442 | 1.083 | 1.065 | 1.038 | 0.958 | 0.974 | 2.12 |
| 28 | s38584 | 1233 | 1.007 | 1.000 | 0.980 | 0.973 | 0.980 | 5.84 |
| 29 | s38417 | 1462 | 1.052 | 1.032 | 0.999 | 0.950 | 0.968 | 7.38 |
| 30 | s35932 | 1728 | 1.015 | 0.996 | 0.971 | 0.957 | 0.975 | 1.84 |
| geometric mean | | | | | | 0.954 | 0.967 | |

The percentage of reduction is also shown in Fig. 3 with the indexes in the first column of Table 1 as x-axis. Because this ratio is dependent on the threshold value $\theta$ used in Alg. 1, we have also drawn the results when $\theta$ is set to 1.1. Combined with the definition of setup time with respect to either MSHP or MCP, four different curves are obtained.

As can be seen in Fig. 3, the reduction is almost independent on the choice of setup time when the threshold value $\theta$ is set to 1.1 as shown in the curves marked with "MSHP (1.1)" and "MCP (1.1)". This represents the pure cycle stealing effect under the new flipflop modeling. The unbalance of logic path delays in different computation stages facilitates the benefit of cycle stealing which leads to a possibly higher working frequency of the circuit. Increase of the threshold value $\theta$ will increase the reduction percentage of our proposed method

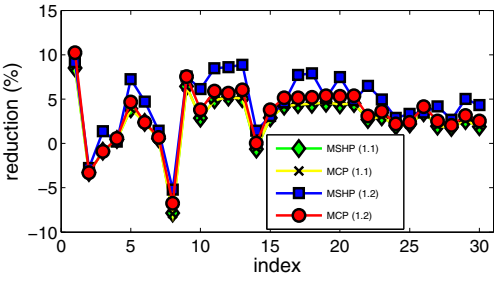



**Fig. 3.** Reduction Percentage

by allowing more upward potential of clock-to-q delay on individual flipflops as shown in the lines marked with "MSHP (1.2)" and "MCP (1.2)". But this threshold can not be set arbitrarily large due to the metastability issues [8].

As shown in Fig. 3, there are cases where the reduction pecentage is a negative number. This means, if we apply $T_{MSHP}$ or $T_{MCP}$ for Alg. 1, the circuit fails. In our experiments, actually two analytical functions as presented in Eq. (1) are used for the data pattern 0 and 1 respectively. The propagated state vector of the last iteration will be directly used in the current one. It happens that a certain input pattern calculated during the iteration causes a worse path found. In traditional STA, propagation of arrival times is only based on successive local maximum or minimum and thus such paths can not be found.

### 4.2 Runtime

The characterization cost of flipflop depends on the number of samples considered. For the purpose of this paper, we have run an extensive set of samples in order to obtain the analytical function in Eq. (1). This can be accelerated by observing that the surface in the nominal region is flat or further by parallel simulation.

The runtime for finding the minimal clock period is shown in the last column of Table 1 labeled as $t_{ITA}$. As can be seen, the proposed method can finish in seconds for all the benchmark circuits. Actually, each iteration in Alg. 1 corresponds to one traditional STA run. So the total runtime of ITA depends on the convergence rate of Alg. 1 and Alg. 2. The number of flipflops is not the only factor affecting the runtime. Different internal circuit structures can also lead to different convergence rate and thus runtime.

## 5 Conclusion

In this paper, an iterative timing analysis method is presented based on a new modeling of flipflop timing behavior. Two basic problems solved are the circuit verification under a given clock period and the determination of minimal clock period. Compared with traditional STA, a reduction of the clock period by 3.3% can be achieved.

**Acknowledgement.** This research was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89).

## References

1. Lang, A., Bergler, S.: Method and apparatus for circuit verification of meeting setup and hold time requirements (Verfahren und Vorrichtung zum Ueberpruefen einer Schaltung auf Einhaltung von Setup- und Holdzeiten). Patent DE102004044668A1 (2004)
2. Salman, E., Dasdan, A., Taraporevala, F., Kucukcakar, K., Friedman, E.G.: Exploiting setup-hold-time interdependence in static timing analysis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26(6), 1114–1125 (2007)
3. Salman, E., Dasdan, A., Taraporevala, F., Kucukcakar, K., Friedman, E.G.: Pessimism reduction in static timing analysis using interdependent setup and hold times. In: ISQED 2006, pp. 159–164 (2006)
4. Srivastava, S., Roychowdhury, J.: Interdependent latch setup/hold time characterization via Euler-Newton curve tracing on state-transition equations. In: DAC 2007, pp. 136–141 (2007)
5. Srivastava, S., Roychowdhury, J.: Independent and interdependent latch setup/hold time characterization via Newton-Raphson solution and Euler curve tracking of state-transition equations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(5), 817–830 (2008)
6. Hatami, S., Abrishami, H., Pedram, M.: Statistical timing analysis of flip-flops considering codependent setup and hold times. In: GLSVLSI 2008, pp. 101–106 (2008)
7. Sakallah, K.A., Mudge, T.N., Olukotun, O.A.: Analysis and design of latch-controlled synchronous digital circuits. In: DAC 1990, pp. 111–117 (1990)
8. Gabara, T., Cyr, G., Stroud, C.: Metastability of CMOS master/slave flip-flops. In: CICC 1991, pp. 29.4/1–29.4/6. (1991)

# Ultra Compact Non-volatile Flip-Flop for Low Power Digital Circuits Based on Hybrid CMOS/Magnetic Technology

Gregory Di Pendina[1], Kholdoun Torki[1], Guillaume Prenat[2], Yoann Guillemenet[3], and Lionel Torres[3]

[1] CMP – Circuits Multi-Projets, 46 Avenue Felix Viallet, Grenoble, France
{gregory.dipendina,kholdoun.torki}@imag.fr
[2] SPINTEC, CEA, CNRS, UJF, INPG; CEA/INAC
17 Rue des Martyrs, 38054 Grenoble Cedex, France
guillaume.prenat@cea.fr
[3] University of Montpellier 2 – LIRMM – UMR CNRS
161 Rue ADA, 34 392 Montpellier, Montpellier, France
{yoann.guillemenet,lionel.torres}@lirmm.fr

**Abstract.** Complex systems are mainly integrated in CMOS technology, facing issues in advanced process nodes, in particular for power consumption and heat dissipation. Magnetic devices such as Magnetic Tunnel Junction (MTJ) have specific features: non-volatility, high cyclability (over $10^{16}$) and immunity to radiations. Combined with CMOS devices they offer specific and new features to designs. Indeed, the emerging hybrid CMOS/Magnetic process allows integrating magnetic devices within digital circuits, modifying the current architectures, in order to contribute to solve the CMOS process issues. We present a high performance innovative non-volatile latch integrated into a flip-flop which can operate at high speed. It can be used to design non-volatile logic circuits with ultra low-power consumption and new functionalities such as instant startup. This new flip-flop is integrated as a standard cell in a full Magnetic Process Design Kit (MPDK) allowing full custom and digital design of hybrid CMOS/Magnetic circuits using standard design tools.

**Keywords:** Non-volatile, Latch, Flip-Flop, Magnetic Tunnel Junction, Low Power, Full Custom Design, Digital Design, Process Design Kit, Standard Cell.

## 1 Introduction

For almost 40 years, the development of electronic circuits is evolving according to more or less Moore's law: speed and density double every 18 months. But in advanced technology nodes, this trend tends to get out of breath. Indeed, due to the small dimensions of the devices and the high speed operations, the power consumption of logic circuits becomes larger and larger, resulting in heat dissipation and reliability issues. Several techniques have been implemented to decrease the power consumption of logic circuits [1] (clock and power gating, dynamic voltage

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 83–91, 2011.

frequency scaling...). Power gating consists in cutting off the power supply of unused blocks of a circuit to reduce the standby power consumption. With volatile memories, this technique requires copying the data into non-volatile or very low leakage memory, resulting in delays and dynamic power consumption. Using a non-volatile flip-flop (NVFF) allows cutting off the power supply without any additional operation and with very low area overhead, allowing an efficient instant on/off policy. In this way it allows the circuit to be stopped and restarted at once on demand with full performance, leading to the concept of "normally off electronics". This also improves the circuit reliability in particular against power failures. Radiation immunity is a further advantage of this hybrid CMOS/Magnetic technology. In this paper, we present a compact and high performance non-volatile latch integrated into a flip-flop and a full Magnetic Process Design Kit (MPDK) allowing full custom and digital design of hybrid CMOS/Magnetic circuits. The first part deals with the magnetic technology, the second part is dedicated to the description of the non-volatile latch and finally, we present the MPDK embedding the latch into a flip-flop as a standard cell to be used in a conventional CMOS design flow.

## 2 Technology and Device

### 2.1 Magnetic Tunnel Junction Using Thermally Assisted Switching Method

A Magnetic Tunnel Junction (MTJ) is a nano-structure basically composed of two ferro-magnetic layers separated by an oxide layer as shown in Fig.1. The magnetization of one of the magnetic layer, called the hard layer, is pinned and acts as a reference, while the magnetization of the second layer, called the soft layer, can be switched by an external magnetic field, or a current. The resistance of the MTJ depends on the relative magnetization of the two layers (Tunnel Magneto-Resistance, TMR): the resistance in the parallel state (RP) is lower than the resistance in the antiparallel state (RAP). These two values typically differ by a factor 2 to 3. The MTJ pillar has typically an elliptical shape. Due to shape anisotropy, the long axis of the ellipse constitutes an easy axis of magnetization. In the absence of external solicitation, the magnetization of the soft layer lies in one direction or the opposite one along this axis. In practice, for memory and logic applications, the magnetization has two stable states, parallel or antiparallel to the hard layer magnetization, with an hysteretic behavior. In this case, the value stored in the MTJ is represented by its magnetic configuration and the associated resistance value. Writing an information in the MTJ consists in orienting the magnetization of the storage layer either parallel or antiparallel to that of the reference layer. In the first MRAM generation [2], this is performed by a pulse of magnetic field generated by a pulse of current flowing in a current line located above or below the MTJ. This approach suffers from selectivity and scalability problems. The Thermally Assisted Switching (TAS) approach relies on the strong dependency of the magnetic properties upon temperature: a pulse of current is first applied throughout the MTJ to heat it above a so-called "blocking" temperature. It can then be easily switched by a low-magnetic field. Subsequently, the heating current is switched off so that the MTJ cools down to ambient temperature. This heating and cooling process only takes about 20ns. It insures an excellent retention of the information in standby for more than 10 years. This technology is being developed by the MRAM manufacturer CROCUS-Technology.

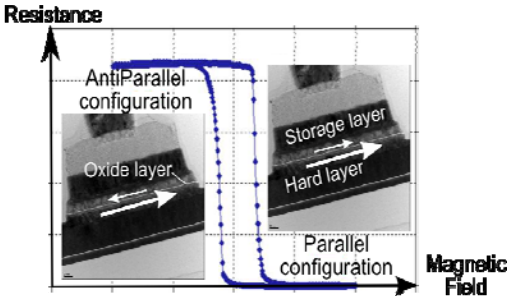



**Fig. 1.** MTJ Device Principle

### 2.2 Technology and Post Process of the Hybrid CMOS/Magnetic Process

The hybrid technology presented here has been developed in the framework of a French national project SPIN [3]. The magnetic devices are fabricated at CEA-LETI and CROCUS-Technology, in post-process above the STMicroelectronics 130nm CMOS process. Fig. 2 shows a cross-section of the technology: it integrates the whole standard CMOS process, the magnetic layers with the MTJ itself, top and bottom electrodes plus vias to connect the MTJ to the CMOS layers. An interesting feature of this post process is its full compatibility with any standard CMOS process.

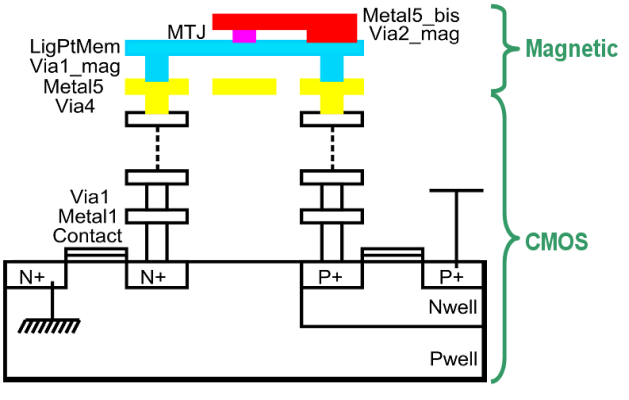



**Fig. 2.** Hybrid CMOS/Magnetic Process Cross Section

## 3 Innovative Cell and Tools

### 3.1 4 Transistor Loadless Non-volatile Cell

An innovative non-volatile SRAM cell, made by Black and Das (B&D), has been proposed in [4]. It has been adapted to the TAS technology in [5] and integrated into a non-volatile flip-flop in [6] and [7]. The SRAM cell has been modified adding two MTJ acting in differential mode: one MTJ is in parallel state while the other is in antiparallel state. Thus, two logic values are coded in the cell: one in the magnetic

part and one in the latch itself. They are independent since it is possible to write the magnetic information without affecting the latch and vice-versa. This cell requires 9 transistors instead of 6 in a classical SRAM cell, two of them being dedicated to heating the junction. The B&D can be used as any CMOS latch. A simple pulse of voltage on a so-called "Autozero" (Az) additional transistor allows transferring the value of the MTJ into the latch.

Several years ago, on previous generation processes, the 6 transistor (6T) SRAM were modified by replacing 2 transistors by 2 resistor loads above the 4 remaining transistors in order to improve the density, yielding a 4T SRAM. To operate correctly, the device requires high resistance values to ensure good logic levels and to limit the power consumption. Moreover, the resistance's area increases with advanced technology nodes compared to transistors area, so that the size of the resistors becomes prohibitive. In addition, the 4T SRAM suffers from intrinsic static power consumption and is sensitive to noise and soft errors because of the high resistance values. To solve these problems, a loadless 4T SRAM cell has been proposed in [8], in which the access transistors are also used as load transistors: when they are "on", they are connected to the bit lines and allow the access to the SRAM cell for read and write operations. When they are "off", they are connected to the power supply and maintain the output logic levels "Q" and "Q_" by leakage currents. To operate correctly, the leakage current at "off" state has to be higher in the access transistors than in the latch transistors, to ensure logic levels to be compatible with the technology. This is performed by using low-threshold access transistors and high-threshold latch transistors.

We propose here a new non-volatile cell based on the 4T loadless SRAM. Two MTJ are added in the 4T SRAM cell as illustrated on Fig. 3.

The main innovation of this cell is that it uses the same path to write the MTJs and to read their state. Writing the MTJs is performed in two cycles, with the access transistors connected to vdd: first, one access transistor is activated for example applying a low level on the heat2 control signal. This saturates the N1 transistor. A low level on heat1 control signal allows the generation of the heating current for the MTJ1. The writing magnetic field can then be applied to write to the MTJ. This operation is repeated to write to MTJ2. In standby state, the access transistors are connected to vdd, but in "off" state. To restore the value of the magnetic part into the latch, P1 and P2 are on, connected to the power supply again and an Az pulse is applied: the latch operates as a sense amplifier to read the difference of resistances in the two branches. In non-magnetic operations, data can be read or written directly to the latch. In this case, P1 & P2 are connected to the bit lines and activated at the same time. Since the resistance of the MTJs is comparable to the "on" resistance of the transistors, their presence does not disturb the writing/reading operations. The heating resulting from the current passing through the MTJs is not a problem since no magnetic field is applied at this time.

The main advantage of this approach is that it requires only 5 transistors instead of 9 in the classical TAS B&S cell. Moreover, since the heating current is proportional to the surface of the junction, it decreases very quickly with the technology node. It is possible to use minimum size transistors to generate the heating current from 65nm CMOS technology and below, which means that a SRAM can be made fully non-volatile with a very low area overhead.

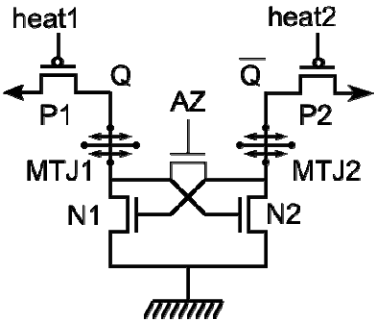



**Fig. 3.** 5 Transistor Loadless Non-volatile Cell

This compact innovative non volatile cell could be used in Magnetic RAM (MRAM) architectures which would enable to reduce the memory area in the design or to increase the storage capability for a given area. However we present below how this cell can be implemented in a flip-flop to make it non volatile. This innovative cell is used as the first latch of a flip-flop, while the second latch is a classical 6T SRAM.

This cell can be used as a classical register for high-speed operation. It also offers the possibility to backup its content into the magnetic part at any time and to restore it with an Az pulse shorter than 1ns as illustrated in Fig. 4, to restart the logic on a known state.

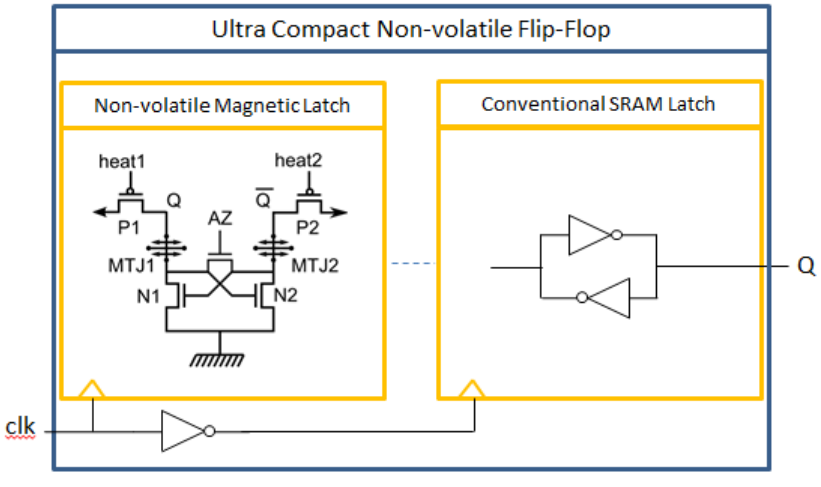



**Fig. 4.** Ultra Compact Non-volatile Flip-Flop

The total write time of this NVFF is typically around 40ns, driven by the heating/cooling rates, themselves depending on the materials. This allows a high frequency and low-power backup of the data in the non-volatile part in comparison with Flash for example. For applications which do not require a very high operation speed, it is even possible to write the MTJ at each clock cycle, resulting in a fully non-volatile logic circuit. This cell can be used for example to ease the power gating technique: in operation, the cell acts as a standard latch. When the block has to be deactivated, the

active data of the latch can be copied in the magnetic part, and the power supply totally switched off using ultra low leakage transistors. Thus, the standby consumption is lower than the technics consisting in lowering the voltage for SRAM data retention. When the block has to be used again, a simple Az pulse allows for immediate recovery of the data, to retrieve the state of the block.

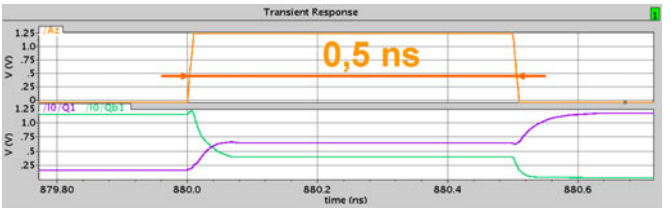



**Fig. 5.** Flip-flop Reading Operation Simulation with TAS method

### 3.2 Full Custom Flow and Tools

Concerning the front-end, an electrical model of the MTJ has been developed [9]. The result is a dynamic library which can be loaded by an electrical simulator (for instance Spectre). The use of this model is very similar to that of the bsim model of the transistor: the model is a generic representation of the MTJ, which contains a given number of parameters. Some of these parameters are chosen independently for each instance. It is typically the case for geometrical parameters (size, shape...). They can be chosen by the designer when instantiating the device, according to the post process capabilities. The other parameters are linked to the technology and cannot be modified by the designer. They are provided by the manufacturer in a corner file. This compact model enables one to run an electrical simulation of a design including both standard CMOS devices such as transistors, resistors, capacitors and so on, and the MTJ with its specific signals. A symbol view is available to design a full custom circuit using MTJs. In Fig. 5, (a) is the heating control signal, (b) is the signal to control the generation of the magnetic field, (c) represents the temperature, (d) the voltage across the MTJ and (e) its resistance. During the period (1), a pulse of current is applied to the MTJ, resulting in a temperature increase. Then, a magnetic field is applied. After a delay (2) corresponding to the intrinsic switching duration, the resistance changes, as a consequence of the soft layer magnetization switching whose dynamic behavior is detailed in the inset. Then (3) is the end of the heating phase and (4) the cooling phase during which the temperature returns to the standby value. Fig. 5 (e) clearly shows the effect of temperature and voltage on the electrical resistance.

Concerning the back-end design flow, all necessary technology files have been developed and implemented in the design kit: specific layers have been added to insert the MTJs coupled with their connections. This is fully integrated with CMOS layers in the Cadence Layer Selection Window (LSW). A parameterized cell (PCell) has been developed to ease the MTJ design and to enable interactive parameter modifications when running the simulations and adjusting the full custom design. Regarding the verification, all MTJ's specific design rules have been implemented in the Design Rule Checks (DRC) with 2 different switches, offering the possibility to run the DRC either on both CMOS and magnetic layers or on the magnetic layers

only. This gives the designers some flexibility. A full extraction is also possible with this design kit which enables one to run Layout Versus Schematic (LVS) checks on the entire design, including magnetic tunnel junctions. As a consequence, a netlist including all devices and parasitics can be extracted from the layout for post layout simulations and validation before manufacturing.

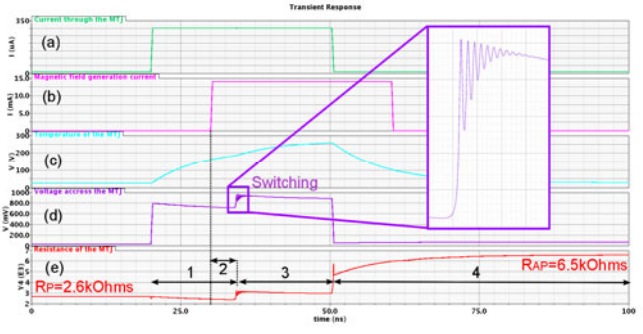



**Fig. 6.** MTJ Writing Operation Simulation with TAS method

### 3.3 Digital Flow and Tools

A complex digital architecture can be made of standard volatile CMOS blocks and non-volatile CMOS/Magnetic blocks. Each of them needs to be described in a Hardware Description Language (HDL) before the synthesis. In order to be able to run digital simulation for advanced and complex designs, a full Verilog description of the NVFF has been developed. It takes into account the behavior and also the timing checks for the MTJ's heating duration, writing and reading. Indeed, the TAS method has very strict intrinsic timing constraints which need to be taken into account to ensure the simulation reflects the real MTJ behavior, as we have in full custom simulations using the compact model.

To design non volatile blocks using the innovative flip-flop standard cell described above, the synthesis uses a specific library containing only registers made of CMOS and MTJs. The connections between all the magnetic signals such as Heat1, Heat2 and Az are defined in the HDL description where all these signals need to be declared and interconnected to manage the MTJ writing and reading. As a consequence, the netlist extracted from the synthesis includes the NVFF and both connections of CMOS signals (D, Clk, Q, Rst, Set …) and magnetic signals. The abstract view of the NVFF is given in order to place and route a full block composed of standard cells, using standard tools. Once the place and route job is completed, only the write signal paths need to be placed over all the rows composed of non volatile flip-flops and connected to current generators to generate the switching field. Since these specific cells can be neither flipped nor mirrored because of the importance of the write field direction, MTJs are placed only every 2 rows. To complete final simulations, an SDF file including the NVFF internal timings can be extracted from the placed and routed layout to back annotate the netlist to run more accurate simulations. Fig. 6 shows the layout of a digital filter described in HDL format, simulated with Mentor Graphics /

Modelsim, synthesized under Synopsys / Design Compiler, placed and routed with Cadence / Encounter. This filter includes about 1100 standard cells in which about 290 are non volatile flip-flops described above.

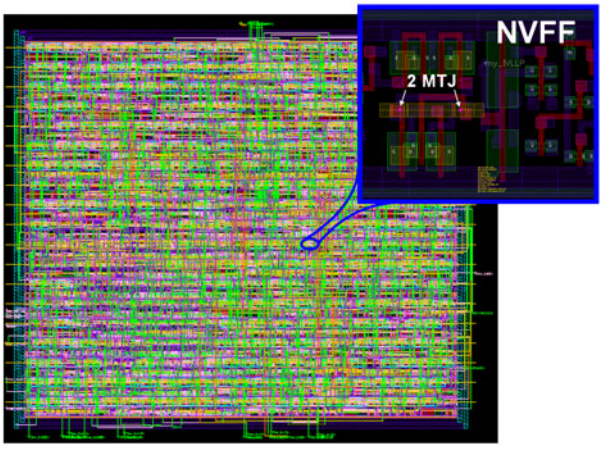



**Fig. 7.** Automatically Placed and Routed Circuit using Non-volatile Flip-Flop Standard Cell

## 4 Conclusion

Non-volatile technologies and in particular the CMOS/Magnetic are more and more considered as a way to circumvent the limits of advanced CMOS circuits, combined with existing design techniques or technologies. This requires evaluating the use of these new devices in complex systems, and probably adapting the existing techniques and architectures correspondingly. Such studies can only be carried out using tools that integrate these devices in standard design suites. The proposed full MPDK has been developed and used to realize a demonstrator which is currently under fabrication within the SPIN research project. This design kit is compatible with design suites supported by most of the foundries and can be easily improved with the evolution of the technology. It could be adapted to fit accurate characterization results and made available for any designer interested in the design of hybrid Magnetic/CMOS circuits.

## References

1. Hu, Z., Buyuktosunoglu, A., Srinivasan, V., Zyuban, V., Jacobson, H., Bose, P.: Microarchitectural techniques for power gating of execution units. In: Proceedings of International Symposiumon Low Power Electronics and Design (ISLPED) (August 2004)
2. Andre, T.W., Nahas, J.J., Subramanian, C.K., Garni, B.J., Lin, H.S., Omair, A., Martino, W.L.: A 4-mb 0.18-µm 1t-1mtj toggle mram with balanced three input sensing scheme and locally mirrored unidirectional write drivers. IEEE Journal of Solid State Circuits 23(1), 301–309 (2005)

3. http://www.lirmm.fr/SPIN
4. Black Jr., W.C., Das, B.: Programmable logic using giant-magneto-resistance and spindependent tunneling devices. J. Appl. Phys. 87(9), 6674–6679 (2000)
5. Guillemenet, Y., Torres, L., Sassatelli, G.: A Non-Volatile Run-Time FPGA structures using Thermally Assisted Switching MRAMs. Journal IET Computers and Digital Techniques 4(3), 211–226 (2010), doi:10.1049/iet-cdt.2009.0019
6. Lakys, Y., Zhao, W., Klein, J.-O., Chappert, C.: Low power, High Reliability Magnetic Flip-Flop. Electronics Letters 46(22), 1493, 2 pages (2010)
7. Sakimura, N., Sugibayashi, T., Nebashi, R., Kasai, N.: Nonvolatile Magnetic Flip-Flop for Standby-Power-Free SoCs. IEEE Journal of Solid State Circuits 44(8) (August 2009)
8. Noda, K., Matsui, K., Takeda, K., Nakamura, N.: A loadless CMOS four-transistor SRAM cell in a 0.18-$\mu$m logic technology. IEEE Transactions on Electron Devices 48(12), 2851–2855 (2001), doi:10.1109/16.974716
9. ElBaraji, M., Javerliac, V., Guo, W., Prenat, G., Dieny, B.: Dynamic compact model of thermally assisted switching magnetic tunnel junctions. Journal of Applied Physics 106(12), 123906 (2009)

# Performance-Driven Clustering of Asynchronous Circuits

Georgios D. Dimou[1], Peter A. Beerel[1,2], and Andrew M. Lines[1]

[1] Fulcrum Microsystems Inc., Calabasas, CA, USA
{gdimou,lines}@fulcrummicro.com
[2] Ming Hsieh Dept. of Elec. Eng. University of Southern California, Los Angeles, CA, USA
pabeerel@usc.edu

**Abstract.** This paper describes a novel approach for generating asynchronous circuits from HDL specifications by clustering the synthesized gates into asynchronous pipeline stages while preserving liveness, meeting throughput constraints, and minimizing area. The method enables a form of automatic re-pipelining in which the throughput of the resulting design is not limited to the clock frequency or the level of pipelining in the original RTL specification. The method is design-style agnostic and is thus applicable to many asynchronous design styles.

## 1 Introduction

Many ASIC design flows for asynchronous semiconductor circuits translate netlists generated using synchronous synthesis tools from often legacy RTL specifications. This approach is practical because it largely reuses mature sophisticated tools that have been proven in the synchronous design arena, but at the same time can limit the advantages of the resulting asynchronous circuits.

In particular, many such flows implement an asynchronous netlist with the same level of pipelining defined in the synchronous netlist. These flows include a coarse-grain pipelining using Phased Logic [11], de-synchronization techniques [5], and Null Convention Logic [9]. The advantage of this approach is that the circuit is largely identical to the original synchronous counterpart. However, because the level of pipelining is determined by the synchronous counterpart, the performance characteristics of the asynchronous circuit are similar to their synchronous counterpart and offer no significant advantage. In contrast, other asynchronous flows target fine-grain pipelined circuits in which each gate is converted into an asynchronous pipeline stage with its own handshake control. This includes fine-grain version of Phased Logic [10], as well as Weaver [12] which uses QDI templates. These flows can achieve significantly higher throughput than their synchronous counterparts, but because of the large amount of handshaking circuitry, suffer a large area penalty.

We assert that the optimum use of asynchronous circuits often requires a middle-of-the-road approach in which the original netlist is partitioned into pipeline stages that are large enough to minimize control overhead while small enough to achieve the desired target performance. Efforts to consider this during high-level synthesis have been considered as a form of operator chaining [6], however, we believe this paper

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 92–101, 2011.

provides the first approach to cluster synthesized asynchronous netlists into pipeline stages that targets given throughput constraints while minimizing control area. In doing so, it highlights a unique pitfall in clustering asynchronous gate-level netlists – the potential of creating deadlock - as well as introduces constraints to avoid doing so. Experimental results on a *multi-level domino* dual-rail template (MLD) [7] are described as well as results on other templates [2][8] are summarized.

## 2 Background – Performance Metrics

In the absence of a global clock, the performance of an asynchronous circuit is characterized using different metrics. In particular, an asynchronous pipeline stage is characterized using forward and backward latencies as well its local cycle time [1]. The *forward latency (FL)* of a pipeline stage is the time between the arrival of a new token(s) and the production of valid output token(s), assuming the stage is initially idle. The *local cycle time (LCT)* of a pipeline stage is the time between the arrival of a token and the time that the stage has reset and is ready to receive the next token. The *backward latency (BL)* of a pipeline stage is the difference between the LCT and the FL and it can be perceived as the time it takes for a *bubble* – or empty position in the pipeline – to propagate backwards in the pipeline [3]. The performance of an asynchronous pipelined circuit, consisting of many pipeline stages, is measured using the *global cycle time (GCT)* which is defined as the long term average time it takes the circuit to process a token. The inverse of the GCT is the circuit's *throughput*.

In addition, we define the *algorithmic cycle time (ACT)* of the circuit which is the maximum for all cycles of the sum of the forward latencies of all the pipeline stages in the cycle divided by the number of tokens (data) that are in the cycle at any time [3]. This is a lower bound for the GCT and thus the global cycle time cannot be improved beyond this value. In particular, the ACT will be lower than the GCT when the circuit's performance is dictated not only by how fast data can propagate forward, but how fast the pipeline stages reset to accept new tokens, i.e., pipeline stalls. A general solution to this problem is to introduce additional pipeline stages into the design to improve the performance by a process historically called *slack matching* [1].

## 3 Clustering

Our approach for creating asynchronous circuits begins with a netlist created from standard logic synthesis. We start with the synthesized netlist and convert each gate to a pipeline stage. We then merge these primitive pipeline stages, into larger stages referred to as *clusters*. The goal is to create as large clusters as possible without affecting the functionality of the circuit or reducing its performance below designer-given performance constraints.

In particular, clustering is a sequence of *local moves* defined in terms of the merging of two neighboring nodes. Clustering is a general approach to form clusters because all partitions can be achieved with a sequence of basic two-way merges and has the benefit that it is easier to characterize and study than arbitrary partitioning. Since every cluster will ultimately need to have its own control unit as well as C-

element trees for multiple fanins/fanouts, every local move results in a drop in total area. The goal is to cluster the circuit into pipeline stages that achieve the minimum overall area while hitting a target performance. However practically this must take into consideration not only the clustering process, but also the effects of the slack matching process that typically follows. Moreover, before even considering different optimization algorithms, we identified more fundamental issues that must be addressed before area optimality. In particular, the focus of this work is maintaining correctness and performance during clustering. For brevity, we do not show the proofs to the theorems; they can be found in [7].

### 3.1 Circuit Model

We abstract the circuit as a weighted directed graph $G = (V, E, h, m)$, where $V$ is the set of nodes in the netlist. $V = PI \cup PO \cup CL \cup TB$, where $PI$ is the set of primary inputs, $PO$ is the set of primary outputs, $CL$ is the set of combinational gates and $TB$ is the set of asynchronous flip-flops called token buffers that upon reset (or startup) are initialized with a token (data) on its output. All four sets $PI, PO, CL, TB$ are mutually disjoint. $E$ is the set of directed edges $E \subseteq (VxV)$. We use the notation $e_{i,j}=(v_i,v_j)$ for an edge in $E$ to simplify our notation for a directed edge that starts from node $v_i$ and ends in node $v_j$. We assume $E$ does not contain any self-loops $e_{i,i}$. We define a function $h: E \rightarrow \Re^+$ to map an edge onto a positive real number that represents the *forward latency* of the edge. We also define the function $m : E \rightarrow \{0,1\}$, such that $m(e_{i,j}) = 1$ when $v_i \in TB$ and 0 otherwise.

We define a path $p_{i,j}$ as a sequence of edges in $E$, the first edge in the sequence starting from node $v_i$, and the last edge in the sequence ending in node $v_j$ and such that for all other edges in the sequence, their starting point is the ending point of the previous edge in the path and their ending point is the starting point of the next edge in the path. We assume that a path goes through each node once (i.e., is a simple path). We define a cycle as a path $p_{i,i}$ that starts and terminates at the same node $v_i$. We define $P_G$ as the set of all paths that exist in $G$.

The target performance metric is defined in terms of the *target cycle time* (TCT) of the circuit, denoted as $\tau_{goal}$. We define the *algorithmic cycle time* (ACT) denoted as $\tau_{alg}$ of the circuit, which as mentioned above is a lower bound of $\tau_{goal}$, as follows

$$\tau_{alg} = \max_{v_i \in V : \exists p_{i,i} \in P_G} \left\{ \frac{\sum_{e_{j,k} \in p_{i,i}} h(e_{j,k})}{\sum_{e_{j,k} \in p_{i,i}} m(e_{j,k})} \right\}. \quad (1)$$

In order to define constraints that identify when $\tau_{alg}$ meets $\tau_{goal}$, we define the weight of an edge as follows:

$$w_{i,j} = w(e_{i,j}) = h(e_{i,j}) - m(e_{i,j}) * \tau_{goal}, \quad (2)$$

and assume that the value of $\tau_{goal}$ is generally much larger than the forward latency of a token buffer. In addition, the weight of a path – as an extension of the edge weight –

is defined as the sum of the weights of all edges in the path sequence. Moreover, we define the distance between two nodes $d_{i,j}$ as the maximum weight of all paths from $v_i$ to $v_j$ and as negative infinity if no paths exist from $v_i$ to $v_j$.

Lastly, we define the transitive fanout $TFO(v_i)$ of a node $v_i$ as the set of all nodes in the graph reachable from $v_i$, and combinational transitive fanout $CTFO(v_i)$ of $v_i$ as the set of all nodes in the graph reachable from $v_i$ with a path that does not include a token buffer. The transitive fanin $TFI(v_i)$ and combinational transitive fanin $CTFI(v_i)$ are similarly defined.

### 3.2 Local Moves

A *local move* is a function on the graph $G = (V, E, h, m)$ that produces a modified graph $G' = (V', E', h', m')$. It takes two nodes $v_i$, $v_j$ in $V$ and replaces them in $V'$ with a unified new node $v'_k \in V'$ that implicitly contains the contents of both nodes (including the instances and wires internal to the pipeline stages that correspond to the original nodes $v_i$ and $v_j$). The rest of the nodes of $V$ are preserved in $V'$.

If both $e_{i,j}$ and $e_{j,i}$ are in $E$ the move is not allowed because this case would generate a self-loop in the graph. Otherwise, when combining nodes $v_i$ and $v_j$ into the new node $v'_k$ in $V'$, $v_i$ and $v_j$ are replaced by $v'_k$ in all edges in $E'$. This means that the new node $v'_k$ has the combined fanin and fanout of $v_i$ and $v_j$, except for any edges between $v_i$ and $v_j$ are absorbed in the new node $v'_k$.

We define the forward latency of the edges in the new graph using the weight function $h'(e_{m,k})$. If edge $e_{i,j} \in E$ when $v_i$ and $v_j$ merge into the new node $v'_k$ in $V'$ then $h'(e_{m,i}) = h(e_{m,i}) + h(e_{i,j})$. An example of this is shown in Fig. 1. Otherwise, the forward latency remains unchanged. The motivation of this definition is that the forward latency of an edge is conservatively assumed to be proportional to the maximum depth of logic that the corresponding individual input signals traverse through the destination pipeline stage. This is applicable for templates that use completion detection as well as bundled data protocols in which the delay lines that correspond to each communication channel are proportional to the corresponding logic and not combined into a single worst-case delay line for the entire node.

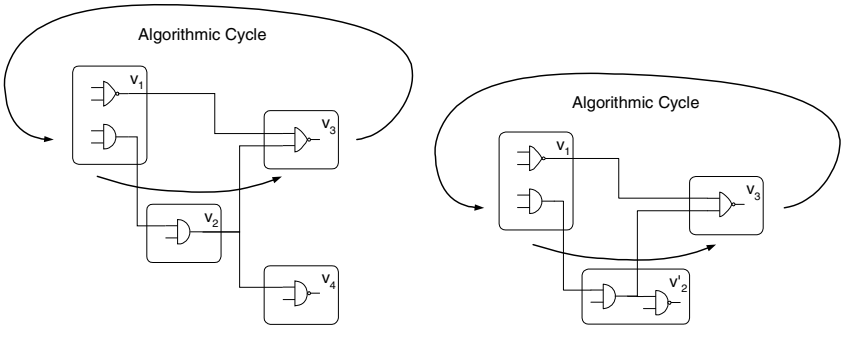



**Fig. 1.** Example of a local move increasing the ACT of the circuit

### 3.3 Ensuring Liveness

The handshaking nature of asynchronous circuits introduces the extra consideration that we preserve circuit liveness. We say a circuit is *live* if every cycle in the circuit has at least one data token [3]. This is guaranteed during the design process by ensuring every cycle in the design contains at least one token buffer [2]. Arbitrary clustering, however, can generate new cycles that violates this principle and this must be prevented. Our first theorem identifies criterion can prevent all moves that create a circuit that is not live.

**Theorem 1.** Let a local move merge two nodes $v_i, v_j \in V$ in graph $G = (V, E, h, m)$ into node $v'_k \in V'$ in the graph $G' = (V', E', h', m')$. The graph $G'$ is non-live iff in the modified graph $G^* = (V, E - \{e_{i,j}, e_{j,i}\}, h, m)$, $v_i \in CTFO(v_j)$ or $v_j \in CTFO(v_i)$.

The intuition behind this theorem is as follows. If a node $v_i$ belongs to the *CTFO* of another node $v_j$ it means that there is a path connecting the two nodes that does not include a token buffer. Merging two such nodes would create a loop in the design that does not have a token in it and therefore the circuit would not be live. Moreover, if $v_i$ and $v_j$ are not in each others CTFO then merging them creates no such loop.

Unfortunately, the gate-level netlist can be complex and every local move that gets executed may generate new paths. It is consequently computationally intensive to perform this check for every available local move after each executed move. On the other hand, for the performance criteria that we will describe next, we need to maintain the pair-wise distances of all nodes in the graph. This motivated the test for liveness in terms of node distances as follows:

**Theorem 2.** Let a local move merge two nodes $v_i, v_j \in V$ in graph $G = (V, E, h, m)$ into node $v'_k \in V'$ in the graph $G' = (V', E', h', m')$. Graph $G'$ will be live if in the modified graph $G^* = (V, E - \{e_{i,j}, e_{j,i}\}, h, m)$, $d_{i,j} \leq 0$ and $d_{j,i} \leq 0$

Note that this distance test is only a sufficient condition for liveness to be preserved. In particular, the reverse of Theorem 2 is not always true, meaning that $d_{j,i} > 0$ does not guarantee that the resulting graph is not going to be *live*. This is because it could be true that $d_{j,i} > 0$ even when the path includes a token buffer, as shown in Fig. 2. Notice that in the example shown $d_{2,9} = 2$, but the path includes $v_5$ is a token buffer thus $v_9 \notin$ CTFO($v_2$) and the circuit remains live. However, this weakness of Theorem 2 can be ignored because as we will show next when we present the performance criteria, we would never execute a move where $d_{j,i} > 0$.

### 3.4 Maintaining Performance

The goal of our performance analysis is to define criteria to guarantee that our local clustering move maintains the performance (throughput) of the original circuit in terms of the algorithmic cycle time of the circuit. Extensions to address circuit latency can be found in [7]. We first describe distance criteria that determine whether the graph satisfies the performance target.

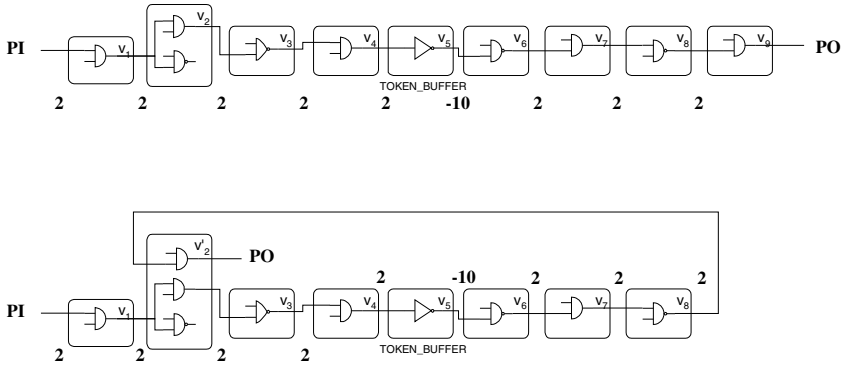



**Fig. 2.** An example where clustering creates a new loop in the graph

**Lemma 1.** In a live graph $G$ the $\tau_{goal} \le \tau_{alg}$ iff $\forall v_i \in V$ it is true that $d_{i,i} \le 0$.

From definition (2), in order to have a violation, there must be an algorithmic cycle in the graph that has a total forward latency that is larger than $\tau_{goal}$ times the number of token buffers in the cycle. But since each token buffer in the cycle contributes a negative $\tau_{goal}$ to the overall distance of the path, the largest value that the distance of the path could have and still meet this requirement is 0.

We now present our distance criteria for guaranteeing throughput during clustering.

**Theorem 3.** Let a local move merge two nodes $v_i, v_j \in V$ in graph $G = (V, E, h, m)$ into node $v'_k \in V'$ in the graph $G' = (V', E', h', m')$. If $G$ is live and $\tau_{alg} \le \tau_{goal}$, the local move will not create a path violating this constraint iff in the modified graph $G^* = (V, E - \{e_{i,j}, e_{j,i}\}, h, m)$ the following distance relationships are satisfied $d_{i,j} + a \le 0$, $d_{j,j} + a \le 0$, $d_{j,i} + b \le 0$ and $d_{i,i} + b \le 0$ where

$$a = \begin{cases} 0, & if \ \neg\exists e_{j,i} \in E \\ w(e_{j,i}), & if \ \exists e_{j,i} \in E \end{cases} \quad \text{and} \quad b = \begin{cases} 0, & if \ \neg\exists e_{i,j} \in E \\ w(e_{i,j}), & if \ \exists e_{i,j} \in E \end{cases}.$$

The intuition behind this theorem is that if an edge *e* exists that connects the two merging nodes and is absorbed during the move, it increases the weight of incoming edges in the combined node by *w*(*e*). So, we avoid introducing any new violations by ensuring all created cycles between the two do not cause a violation. For example, the cycle created in Fig. 2 causes a violation because we can see that now for every node along the cycle it is true that $d_{i,i} = 2$, which violates the condition of Lemma 1. Theorem 3 would prevent such moves from being executed and causing a violation as one can see that before the move is executed $d_{2,9} = 2$, so it is expected to cause a violation.

## 4 Implementation

We implemented a simple steepest-decent greedy algorithm for area-minimizing clustering, illustrated in Fig. 3. The algorithm evaluates at every step which local

move maximizes a gain function and executes the move before re-evaluating. The gain function is weighted in such a way that it will first favor local moves that would reduce any LCT that is larger than the requested TCT of the circuit. As a second priority it favors moves that reduce the most area. Notice that a large constant is used in order to favor LCT improvements over area improvements.

The node distances are initialized using the Floyd-Warshall algorithm with worst-case complexity $\Theta(|V|^3)$. However, this complexity is prohibitively high to run after every local move. Moreover, the Floyd-Warshall algorithm does not take advantage of the fact that only a small number of edge weights typically change after each local move. We thus created a fast local update routine that is run after each local move. Unfortunately, we have not been able to find a reasonable worst case complexity of this algorithm, but in practice it reduces runtimes dramatically compared to Floyd-Warshall.[1]

This algorithm was incorporated into the software tool called ClockFree that is part of the recently developed ASIC flow for asynchronous circuits called Proteus [2]. ClockFree is capable of handling multiple design templates including QDI, Multi-Level Domino (MLD) [6], Multi-Level Static Single-Track Full Buffer (ML-SSTFB) [8] and others. It also performs slack matching and netlist and testbench generation. For QDI and MLD area evaluation was done using production libraries [2].

```
clustering () {
  generate all moves;
  for all moves m {
    calc_gain_metric(m);}

  while moves exist, with gain >= 0 {
    best_move = select_best_move();
    n = execute(best_move);
    Update(n);
    delete all invalid moves;
    generate new moves;
    for all new and affected moves m {
      calc_gain_metric(m);}
  }
}
calc_gain_metric(move m) {
    float A = 0;
    for all edges e {
        A += max{0, LCT(e) - TCT};
    float B = old area - new area;
    m.gain = 1000 * A + B;
}
```

```
Update(newNode){
    Reinitialize_distance(newNode)
    Add newNode to searchlist

    While searchlist not empty {
        remove node from searchlist;
        update_distances_from_node(node);
        if any distance changed
            for each s ∈ FI(node) not in searchlist
                add s to searchlist;
    }
}
Reinitialize_distance(s) {
    for all n
        if n ∈ FO(s)
            d(s,n) = w(e_s,n);
        else
            d(s,n) = - ∞;
}
update_distances_from_node(s) {
    for all n ∈ FO(s)
        for all m ∈ G
            d(s,m) = max{d(s,m),d(s,n)+d(n,m)};
}
```

**Fig. 3.** Pseudocode for the steepest-descent clustering and update distance algorithms

## 5 Experimental Results

In order to show the benefits of our formulation and the effectiveness of the discussed criteria we setup an experiment using the Multi-Level Domino (MLD) template [7] for asynchronous circuits. Each pipeline stage in this template consists of multiple

[1] An alternative local update algorithm with worse run-times but known worst-case complexity of $O(BN^2)$, where B is the branching factor at a node is described in [7].

levels of dual-rail domino logic, a specialized asynchronous controller, and C-element trees to merge the requests from all inputs and acknowledgements from all outputs. Finally, each stage has as a pre-charged completion-detection tree to generate the validity of the logic. We chose this template because it allows arbitrarily large pipeline stages and can tradeoff between control overhead and performance.

For our experiments we compare the attained performance of final circuits using three variants of clustering for each of the netlists. The first one applies no rules to the greedy clustering. The second one only applies the liveness rule described in Theorem 1. The third one adds the TCT constraints of Theorem 3 and also uses the expressions in Theorem 2 for liveness to reduce the computational complexity. We set our performance target to 26 transitions which with our 65nm library that targets an average of 50ps/transition is estimated to yield throughputs of ~770MHz.

As shown in Table 1, 12 of 19 circuits produced deadlocked circuits without use of our liveness criteria. It also shows that the version without performance criteria was not able to reach the desired performance for many cases and in fact in some cases the performance was far worse than any other method due to large loops that were introduced by clustering. Finally, the last approach always managed to achieve performance that was limited by the LCT.

Even in applications that target maximum performance, our clustering algorithm provides significant area benefits. In [2], clustering is applied to a high-performance QDI template and comparisons to the un-clustered circuits were made. Both versions achieved the same performance of 18 transitions per cycle (~1.1GHz in a 65nm process). For the clustered version up to four logic gates were allowed per cluster and clustering was not allowed on special type of gates, including token buffers, scan, conditional or buffer stages. Even with such restrictions, clustering demonstrated on average a 22.7% savings in control area, 14% reduction in total area as well as 10% improvement in total latency over the un-clustered netlist [2]. This is a significant improvement over flows using un-clustered fine-grain pipelining (e,g, [12]).

Another important benefit of the proposed approach is that it is performance-aware. Starting from the same synthesized netlist the designer can chose the desired performance target and clustering will adjust the amount of pipelining in the design automatically. In other words, clustering enables a much larger tradeoff between performance and area without having to redesign or even re-synthesize a netlist. Table 2 shows the results of two MLD designs for varying performance targets. The area savings are due to both clustering and slack-matching savings. Overall on average a 37.5% increase in TCT gives us a 55% reduction in overall area, 66% reduction in total control area and 72% reduction in number of logic pipeline stages. Note that as clustering increases the runtime grows, but all examples finish in less than 9 minutes on an Intel Core 2 Duo desktop computer.

Finally in [8] the benefits of our clustering algorithm are shown using a newly proposed multi-level single-track full-buffer template. Several circuits were tested for the same input netlist with varying performance targets. In the examples presented there, reducing the performance target by 20% enabled area savings of approximately 50%. In each case the asynchronous circuit is re-pipelined from the same original synchronous circuit according to the performance specification without having to redefine the circuit HDL or even the synthesized netlist. This kind of performance-driven re-pipelining is unique in our approach and one of its biggest advantages.

**Table 1.** Performance

| | Area Guided | | Liveness | | Liveness & TCT | |
|---|---|---|---|---|---|---|
| **Design** | **LCT** | **GCT** | **LCT** | **GCT** | **LCT** | **GCT** |
| c3540 | 28 | DL | 32 | 32 | 34 | 34 |
| MAC16 | 28 | 68 | 32 | 32 | 34 | 34 |
| MAC32 | 32 | 72 | 32 | 32 | 32 | 32 |
| s1196 | 34 | 34 | 34 | 34 | 32 | 32 |
| s1238 | 28 | DL | 32 | 32 | 34 | 34 |
| s13207 | 28 | DL | 32 | 32 | 30 | 30 |
| s1423 | 28 | DL | 32 | 50 | 32 | 32 |
| s1488 | 30 | 30 | 30 | 30 | 32 | 32 |
| s15850 | 30 | DL | 38 | 64 | 34 | 34 |
| s526 | 26 | DL | 32 | 32 | 28 | 28 |
| s5378 | 32 | 32 | 32 | 32 | 32 | 32 |
| s641 | 26 | DL | 32 | 32 | 28 | 28 |
| s713 | 26 | DL | 32 | 32 | 32 | 32 |
| s820 | 28 | DL | 32 | 32 | 30 | 30 |
| s832 | 32 | 32 | 32 | 32 | 32 | 32 |
| s838 | 26 | DL | 32 | 32 | 30 | 30 |
| s9234 | 32 | 32 | 32 | 32 | 32 | 32 |
| s953 | 28 | DL | 32 | 32 | 32 | 32 |
| SISO | 34 | DL | 36 | 76 | 46 | 46 |

**Table 2.** Performance vs. Area

| | MAC16 | | | | MAC32 | | | |
|---|---|---|---|---|---|---|---|---|
| **TCT** | **Total Area ($um^2$)** | **CTRL Area ($um^2$)** | **Logic Stages** | **Runtime (sec)** | **Total Area ($um^2$)** | **CTRL Area ($um^2$)** | **Logic Stages** | **Runtime (sec)** |
| **32** | 156407 | 94371.8 | 572 | 210 | 104806 | 62535.2 | 305 | 100 |
| **34** | 139505 | 80299 | 420 | 249 | 95142.5 | 54291.5 | 203 | 145 |
| **36** | 117645 | 63258.6 | 185 | 453 | 104911 | 61779.5 | 143 | 307 |
| **38** | 110170 | 59526.1 | 68 | 497 | 106792 | 62963.7 | 142 | 259 |
| **44** | 60422.4 | 24671.2 | 60 | 547 | 53513.9 | 25030.7 | 137 | 229 |

## 6 Summary and Conclusions

We proposed a new performance-driven approach that uses clustering to group gates into larger pipeline structures that preserves liveness and does not compromise user-defined performance targets. We take advantage of the fact that asynchronous circuits are data driven in nature which allows clustering to redefine the pipeline stages irrespective of the pipeline stages defined at the RTL without altering the behavior of the circuit. Thus clustering enables automatic re-pipelining based on performance-driven criteria that can create a circuit that meets the desired performance without changes to the RTL. This makes the design easier and saves designers time and effort that would otherwise be required in adjusting their pipeline stages to meet the required performance.

Our approach is based on an implementation-agnostic mathematical model that makes this flow applicable to a variety of different asynchronous design styles. This paper shows results for one form of multi-level domino templates and others have shown results for QDI and single-track templates [2][8].

**Acknowledgements.** This work was done jointly between Fulcrum Microsystems Inc. and the University of Southern California. Fulcrum Microsystems provided funding as well as engineering resources, which include the QDI and MLD libraries.

## References

1. Beerel, P.A., Lines, A., Davies, M., Kim, N.H.: Slack matching asynchronous designs. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC 2006), pp. 184–194 (2006)
2. Beerel, P.A., Dimou, G.D., Lines, A.: Proteus: An ASIC flow for GHz Asynchronous Designs. IEEE Design & Test (September/October 2011)
3. Beerel, P.A., Ozdag, R.O., Ferretti, M.: A Designer's Guide to Asynchronous VLSI. Cambridge University Press, Cambridge (2010)
4. van Berkel, K., Kessels, J., Roncken, M., Saeijs, R., Schalij, F.: The VLSI-Programming Language Tangram and Its Translation into Handshake Circuits. In: Proc. European Conference on Design Automation (EDAC), pp. 384–389 (1991)
5. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: De-synchronization: Synthesis of Asynchronous Circuits from Synchronous Specification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (October 2006)
6. Venkataramani, G., Goldstein, S.C.: Operation Chaining Asynchronous Pipelined Circuits. In: Proceedings of the International Conference on Computer Aided Design (ICCAD) (November 2007)
7. Dimou, G.: Clustering and Fanout Optimizations of Asynchronous Circuits. PhD. Thesis. University of Southern California (May 2009)
8. Golani, P., Beerel, P.A.: An area-efficient multi-level single-track pipeline template. In: Design, Automation & Test in Europe Conference & Exhibition (DATE) (March 2011)
9. Kondratyev, A., Lwin, K.: Design of asynchronous circuits by synchronous CAD tools. IEEE Design and Test of Computers 19(4), 107–117 (2002)
10. Reese, R.B., Thornton, M.A., Traver, C.: A fine-grain Phased Logic CPU. In: Proceedings of the IEEE Computer Society Annual Symposium on VLSI (February 2003)
11. Reese, R.B., Thornton, M.A., Traver, C.: A coarse-grain phased logic CPU. IEEE Transactions on Computers (July 2005)
12. Smirnov, B., Taubin, A., Su, M., Karpovsky, M.G.: An Automated Fine-Grain Pipelining Using Domino Style Asynchronous Library. In: Proc. of 5th International Conference on Application of Concurrency to System Design, ACSD (2005)

# Power/Performance Exploration of Single-core and Multi-core Processor Approaches for Biomedical Signal Processing

Ahmed Yasir Dogan[1], David Atienza[1], Andreas Burg[2], Igor Loi[3], and Luca Benini[3]

[1] Embedded Systems Lab. (ESL) - EPFL; Lausanne - 1015, Switzerland
{ahmed.dogan,david.atienza}@epfl.ch
[2] Telecommunications Circuits Lab. (TCL) - EPFL; Lausanne - 1015, Switzerland
andreas.burg@epfl.ch
[3] UNIBO-Micrel Lab, Viale Risorgimento 2, 40136, Bologna, Italy
{igor.loi,luca.benini}@unibo.it

**Abstract.** This study presents a single-core and a multi-core processor architecture for health monitoring systems where slow biosignal events and highly parallel computations exist. The single-core architecture is composed of a processing core (PC), an instruction memory (IM) and a data memory (DM), while the multi-core architecture consists of PCs, individual IMs for each core, a shared DM and an interconnection crossbar between the cores and the DM. These architectures are compared with respect to power vs performance trade-offs for a multi-lead electrocardiogram signal conditioning application exploiting near threshold computing. The results show that the multi-core solution consumes 66% less power for high computation requirements (50.1 $MOps/s$), whereas 10.4% more power for low computation needs (681 $kOps/s$).

**Keywords:** WBSN, ECG, Parallel Processing, Near Threshold Computing.

## 1 Introduction

Personal health systems monitor metabolic functions, such as heart and respiratory rates, blood oxygen, and carbon dioxide levels to detect and diagnose potential health problems. Wireless body sensor networks (WBSNs) are the enabling technology for such personal health systems [1]. A WBSN for health monitoring consists of a number of light-weight sensor nodes attached to the human body, where each node is responsible for processing a specific low rate physiological signal. For instance, one of the most important physiological signals is the electrocardiogram (ECG), which is typically acquired at sampling rates between 125 Hz and 1 kHz to capture the often important details of the waveform. In order to monitor the heart rate for extended periods of time (up to multiple days or weeks), an ultra low power design with embedded biomedical

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 102–111, 2011.

signal processing for feature extraction on the sensor node is necessary [2] to reduce the costly signal storage or transmission to the essence. The corresponding algorithms, consist mostly of low-effort computations and can thus be optimized to run in real time on typical embedded low-power microcontroller. For example, Rincon et al. [3] showed how delineation of ECG signals using the relatively complex wavelet transform algorithm can be realized on a commercially available WBSN sensor node with limited computation capability.

Unfortunately, despite the reasonable required computational effort, achieving low-power consumption remains a pressing issue since devices are expected to operate on a single battery for long periods of time. An effective technique to reduce power consumption is supply voltage scaling, potentially all the way to sub-threshold operation. In the literature, voltage scaling and its limitations and disadvantages such as performance loss, the risk of functional failure, performance variability, etc., have been extensively analyzed [4,5,6,7] and various low-power architectures have been presented. For example, Chen et al. [8] proposed a sensor platform capable of nearly-perpetual operation by using harvesting from solar cells. The proposed single processor architecture has an ARM Cortex M3 core with both retentive and non-retentive SRAM and a power management unit which controls the active and ultra low power sleep modes. In another work, Hanson et al. [9] presented a new ultra low energy processor with low voltage operations for wireless monitoring systems. They optimized the standby power consumption of the processor with the help of a new low leakage memory, memory size and instruction set adjustments, and power gating. However, the main issue with low-voltage operation is the performance loss, which, for a given processing requirement, can limit the degree of use of voltage-scaling. Parallel computing using multiple cores can alleviate this issue, provided that the algorithms to be executed can be parallelized. To this end, Dreslinski et al. [10] proposed a near threshold computing (NTC), cluster-based multi-processor architecture with a shared cache that operates at a higher supply voltage to be able to serve multiple cores at the same time. In another study, Yu et al. [11] introduced a sub/near threshold co-processor for low energy mobile image processing using architecture level parallelism to compensate the performance loss. Finally, Krimer et al. [12] proposed a massively parallel stream processor operating in NTC to achieve 1 Giga-operations per second with 1 mW of total power consumption.

Unfortunately, even though researchers focused on low energy solutions in both multi-core and single-core approaches individually, the two approaches have not been compared in terms of energy efficiency for the moderate workloads that are typical for biomedical applications. Thus, in this paper we propose as a main contribution a single-core and a multi-core architecture for embedded biomedical signal processing on WBSNs, where algorithms have a limited, yet, at near-threshold voltage, non-negligible complexity and where a significant part of the processing can be done in parallel. We explore the power/performance trade-offs between these proposed architectures for an ECG signal conditioning application while exploiting near threshold computing.

The rest of this paper is organized as follows: First Section 2 presents the multi-lead ECG signal conditioning application. Next, Section 3 introduces the single-core and multi-core processor architecture. Then, Section 4 gives the experimental results and, finally, we summarize the main conclusions of this work in Section 5.

## 2 ECG Signal Conditioning Application

ECG entails the analysis of electrical changes, sensed by electrodes attached to the body, occurring when the heart muscle depolarizes during each heartbeat. In single-lead ECG, the voltage difference between two electrodes placed at both sides of the heart indicates the heart rate (i.e., 60 to 100 beats per minute for an adult with a normal resting heart) and allows to identify weaknesses in different parts of the heart muscle. To obtain a better and more complete picture of the heart muscle activities, up to 12 leads can be used [3]. Each lead shows the activity of the heart from a different point of view.

Unfortunately, raw ECG signals, even when recorded in a controlled environment, contain various types of noise and baseline drifts. ECG signal conditioning is therefore a fundamental application for a sensor node in WBSNs for automated ECG analysis or for signal compression for recording [1]. Hence, our benchmark application is an ECG signal conditioning algorithm based on morphological filtering given in [13]. This algorithm performs baseline correction and noise suppression on ECG signals and operates on multiple leads in parallel and independently. For our case study, we assume 8 leads, which is a typical configuration. The average processing requirements for each lead amount to 681 operations (Ops) per sample, as the algorithm always processes blocks of 1024 samples. To investigate different processing requirements related to the application, we consider ECG sampling rates between $f_s = 125\,\text{Hz}$ and $f_s = 1\,\text{kHz}$ for capturing signals with quality levels from "barely acceptable" to "excellent".

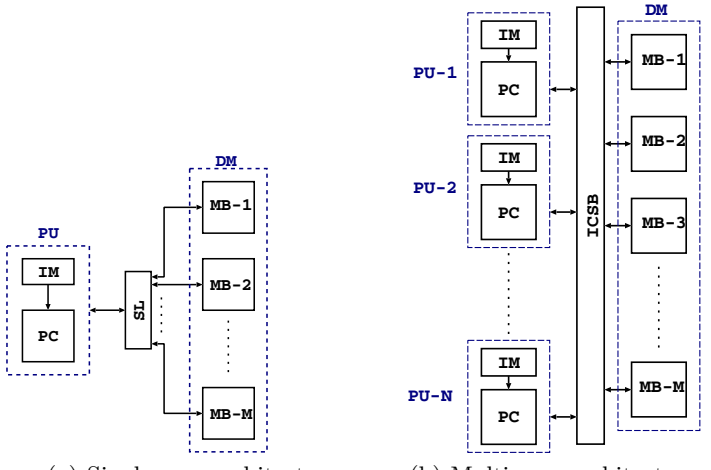
## 3 Processing Platform Architecture

To focus on the comparison between the single-core and multi-core configuration, we build both reference designs using the same processing unit (PU) and a data memory (DM). The designs are implemented in a 90 nm low leakage process technology trading peak performance for significant leakage power reduction, especially in the memories.

*Processing Unit:* A PU comprises a processing core (PC) and a 24-bit wide instruction memory (IM) for 4k instruction words (12 kBytes) which is sufficient for many typical biomedical applications on WBSNs such as delineation and compressed sensing data compression [1], [3]. The PC is a 16-bit Reduced Instruction Set Computer (RISC)-like architecture with sixteen working registers, and a Harvard memory model. The simple two-stage pipeline matches the low

to moderate performance requirements of the application and reduces the number of registers that need to be clocked. In the first pipeline stage, instructions are decoded and read addresses are generated. In the second pipeline stage, the operations are executed and the results are stored in a data memory location or in a working register. Among others, the instruction set comprises arithmetic and logic operations, multi-bit shifts and single-cycle multiplications to support the energy efficient execution of signal processing algorithms. Most instructions occupy only a single (24-bit) instruction-word and are executed in only one clock cycle with a latency of two cycles.

*Data Memory:* The PC can access the DM for reading and writing in the same clock cycle. Therefore the DM requires two separate access ports, one for reading and another one for writing. The 64 kByte of DM, required for 8-lead ECG real time processing, is split into M (i.e, 16) memory banks (MBs) with 2k words per bank. This configuration corresponds to the maximum available from our 2-port memory generator and it allows partial shutdown for leakage power reduction with applications with reduced memory requirements.




(a) Single-core architecture (b) Multi-core architecture

**Fig. 1.** Processing platforms

### 3.1 Single-Core Processor Architecture

The single-core WBSN sensor node reference architecture is shown in Fig. 1(a). A simple selection logic (SL) connects the single PU to the individual MBs and multiplexes the data. The system processes the 8-lead ECG signals sequentially by using 5580 cycles per sample.

When optimized for speed, our single-core design could operate up to 147 MHz at nominal 1.2 V supply voltage, which is much higher than what is required for most biomedical signal processing applications, even for very high sampling rates. Since the cost for this high speed is a reduced energy efficiency, we optimize the

reference design for minimum area instead to lower the active and leakage power consumption. To this end, we request the EDA design tools choose logic gates with weak driving capability. The corresponding circuit is still capable of working up to 50 MHz at nominal voltage, which easily meets the timing requirements of our reference application, given in Section 2.

The second column of Table 1 shows the distribution of the power consumption of the area-optimized design running at 16 MHz clock frequency at 1.2 V. It is interesting to note that the power consumed by the SL and the interconnect network (routing and buffers) between the PU and MBs is almost 15% (0.55 mW) of the total power consumption. A more detailed analysis shows that much of this power is due to glitches on the address and data bus. To alleviate the impact of these glitches we place 48 low-transparent latches at the output ports of the PU (read- and write-addresses, and write-data). The result of this simple measure is a reduction of the overall power consumption of the single-core architecture by 6.7%, as shown in the third column of Table 1.

### 3.2 Multi-core Processor Architecture

The multi-core processor design, shown in Fig. 1(b) consists of N (i.e., 8) PUs with individual IMs. Each PU accesses the 16 shared MBs through a central crossbar interconnect [14] to enable full access to the entire memory space for each PU. This architecture is different from the one proposed by Dreslinski et al. [10] in which several slower cores share a cache that is proportionally faster and thus requires a higher supply voltage. Compared to their single, which relies on a fully shared memory-block configuration, our proposed architecture simplifies the clock-network design[1] and neither requires an additional faster clock, nor level-shifters between the cores and the shared cache. Furthermore, the ability to operate with only a single supply voltage considerably simplifies the overall system design and can result in additional energy savings, because multiple weakly loaded DC/DC-converters can be avoided. The drawback of our approach are the occasional access conflicts when two or more PUs access the same MB on the same port. In this case, the conflicting requests are served one after another based on PU priorities, while the waiting PUs are stalled using clock gating to avoid unnecessary active power consumption.

The multi-core design, which is also optimized for minimum area, is capable of operating up to 48 MHz. For our 8-lead ECG application, all cores are active to process one lead per core in 761 clock cycles per sample. When accounting for the 8x parallel processing, this corresponds to a 12% penalty in terms of timing due to stall-cycles compared to the number of cycles required for a single lead in the single-core architecture. To compensate for this penalty when comparing the two architectures in terms of power consumption, we always adjust the clock frequency of the multi-core design to correspond to the same sampling frequency (throughput) as in the single-core reference architecture. In particular, we provide results at nominal 1.2 V supply voltage for a frequency of 2.3 MHz

[1] As seen in Table 1, the clock tree in the proposed architecture consumes only 5.0% of the whole power consumption.

which ultimately corresponds to the same throughput as the single-core design running at 16 MHz. The corresponding power consumption figures are provided in the two rightmost columns of Table 1. The results show that the overhead of the crossbar in terms of power consumption is insignificant, only 13% of the entire multi-core design. This overhead can be further reduced by applying the same technique for the glitch reduction as in the single-core design. After placing latches in the PUs, the power consumption of the crossbar interconnect is reduced significantly, resulting in the 8.3% power improvement in overall power consumption shown in the rightmost column of Table 1,

**Table 1.** Power distribution of the single-core and the multi-core design with/without latches in the PU at 1.2 V supply voltage and 16 MHz and 2.3 MHz operating frequency, respectively

| | single-core | | multi-core | |
|---|---|---|---|---|
| | w/o latches | with latches | w/o latches | with latches |
| Total | 3.56 mW | 3.32 mW | 3.72 mW | 3.41 mW |
| PUs | 2.53 mW | 2.53 mW | 2.81 mW | 2.81 mW |
| MBs | 0.24 mW | 0.24 mW | 0.24 mW | 0.24 mW |
| SL-ICSB | 0.55 mW | 0.33 mW | 0.48 mW | 0.19 mW |
| Clock Tree | 0.24 mW | 0.22 mW | 0.19 mW | 0.17 mW |
| Reduction | - | 6.7% | - | 8.3% |

The occupied silicon area of the single- and multi-core design is given in Table. 2. As expected, the total area of PUs in the multi-core design is almost 8 times the area of the PU in the single-core design. However, the total area of the multi-core design is only 1.76 times of the total area of the single-core design since the shared MBs are responsible for most of this area.

**Table 2.** Area results for the multi-core and single-core designs (1 GE = 3.136 $\mu m^2$)

| | Single-core | Multi-core |
|---|---|---|
| Topmodule | 644.7 kGE | 1138.1 kGE |
| PUs | 68.0 kGE | 541.4 kGE |
| MBs | 576.7 kGE | 576.7 kGE |
| ICSB | - | 20.0 kGE |

## 4 Experimental Results

The setup we used for the experiments is as follows: 1024 samples of an 8-lead ECG signal are pre-stored in the DM of the single-core and multi-core designs. Each sample occupies 16 bits of memory, which results in 16 kBytes of total storage for the pre-stored ECG samples. The single-core design processes the leads sequentially while in the multi-core design each core processes one lead. The results of each lead are stored individually in the data memory, and the total memory requirement for storing the results is 16 kBytes for each design.

We run our reference application on the two architectures for various workload requirements to explore the power/performance trade-offs between the architectures. A workload requirement in our experiments corresponds to a number of operations per second (Ops/s). This exploration allows us not only to examine the architectures for our reference application, but also to generalize the results and trends to other applications. In addition, we also analyze the architectures with respect to the ECG sampling frequencies corresponding to our application requirement.

We limit the scaling of the operating voltages to the transistor-threshold level (0.5 V) to avoid the performance variability and functional failure issues occurring mainly at sub-threshold regions. Fig. 2 shows the processing capabilities of the two approaches with respect to the supply voltage. At the nominal voltage (1.2 V) the single- and multi-core approaches achieve 50.1 $MOps/s$ and 343 $MOps/s$, respectively. As expected, these processing capabilities decrease with voltage scaling. When the supply voltage of the designs reaches the threshold level, the single-core accomplishes only up to 806.3 $kOps/s$ while the multi-core design still achieves up to 5.58 $MOps/s$.

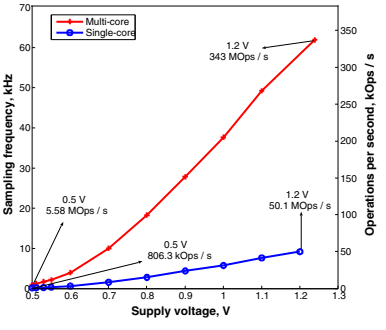
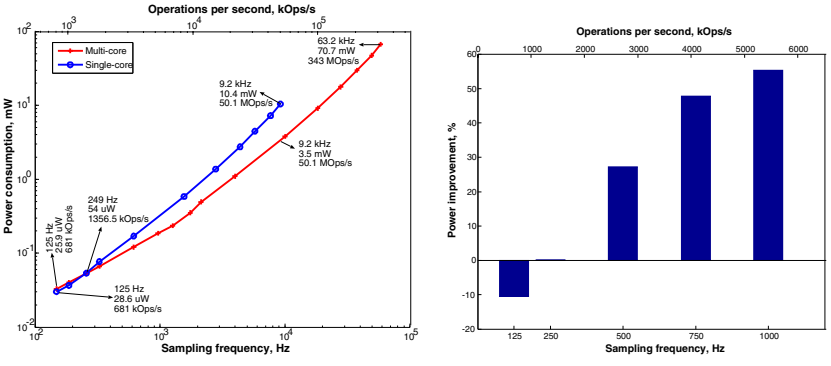



**Fig. 2.** Single-core and multi-core designs: Maximum allowed ECG sampling rate and corresponding number of operations for various supply voltages

Fig. 3(a) shows the total power consumption of the single- and multi-core design for various workload requirements. As can be seen from the figure, the multi-core approach is the only viable solution for workloads between 50.1 $MOps/s$ and 343 $MOps/s$. Moreover, when the workload requirement is between 1356.5 $kOps/s$ and 50.1 $MOps/s$, the multi-core is more energy efficient than the single-core design, because the multi-core design can meet the workload requirements at a lower operating voltage compared to the single-core design. In particular, to meet a high workload requirement (50.1 $MOps/s$) the single-core design operates at 1.2 V and consumes 10.4 mW, whereas the multi-core design operates at 0.7 V and consumes only 3.5 mW. Thus, the multi-core solution consumes almost 66% less power than the single-core design. On the contrary, if the required workload is light (lower than 1356.5 $kOps/s$) the single-core design consumes less power than the multi-core design, because the multi-core design is able to reach the

threshold voltage at 5.58 $MOps/s$ workload while the single-core design reaches at the threshold level at 806.3 $kOps/s$ workload requirement. More precisely, to meet a low workload requirement (681 $kOps/s$), both designs operate at 0.5 V and the single-core design consumes 25.9 $\mu$W while the multi-core design consumes 28.6 $\mu$W. Hence the multi-core design consumes 10.4% more power than the single-core design.

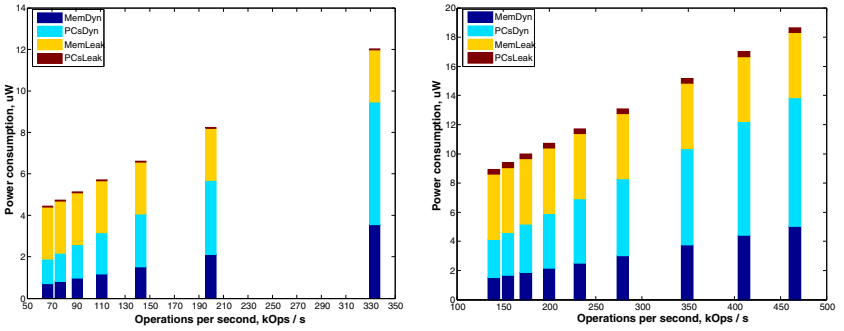



(a) (b)

**Fig. 3.** (a) Total power consumptions for various workloads (b) Power efficiency of multi-core design respect to single-core design for ECG signal conditioning application

The corresponding workload to our application ranges from 681 $kOps/s$ to 5448 $kOps/s$ with an ECG sampling rate ($f_s$) from 125 Hz to 1 kHz. Fig. 3(b) shows the power efficiency of the multi-core design with respect to the single-core design for our application. As the sampling rate increases, the multi-core becomes more and more energy efficient. At the highest sampling rate, $f_s$=1 kHz, the multi-core design is 55% more power efficient. However if the sampling rate is reduced down to 250 Hz, the multi-core design becomes less power efficient. At the lowest ECG sampling rate in our range, $f_s$=125 Hz, the multi-core consumes 10.4% more power than the single core design.

Another interesting point is the comparison between dynamic and leakage power consumptions in the two designs. For our case study, where the lightest workload requirement is 681 $kOps/s$ ($f_s$=125 Hz) the leakage power consumption of the single- and multi-core design is 2.6 $\mu$W and 5 $\mu$W, respectively. Thus the leakage power consumption represents 10% and 17% of the total power consumptions for the single-core and multi-core designs, respectively. Fig. 4(a) and Fig. 4(b) show the dynamic and leakage power consumptions of the PCs and the memories, including both IM and MBs, for various workload requirements for the single-core and multi-core designs. In both figures the dynamic power consumption of the memories is labeled as `MemDyn` while the leakage is labeled as `MemLeak`. Similarly, the dynamic and leakage power consumptions of the PCs are indicated as `PCsDyn` and `PCsLeak`, respectively. As shown in the




(a) Single-core design (b) Multi-core design

**Fig. 4.** Leakage and dynamic power consumption comparison for various workload requirements

figures, MemDyn becomes comparable with MemLeak when the workload is 200 $kOps/s$ and 410 $kOps/s$ for the single-core and the multi-core designs, respectively. As expected, MemLeak in the multi-core design becomes comparable with the MemDyn power at an earlier point, because the total memory leakage power is higher in the multi-core design. Furthermore, the overall leakage and dynamic power consumptions become comparable around at 80 $kOps/s$ for the single-core design while around 140 $kOps/s$ for the multi-core design.

## 5 Conclusion

Embedded biomedical signal processing on WBSNs involves relatively low complex and highly parallel computations on a low-rate physiological data, which creates the opportunity of low voltage operations as well as parallel processing. In this paper we present a single- and a multi-core processor architecture for biomedical signal processing on WBSNs where both energy efficiency and real-time processing are crucial design goals. To address the energy efficiency and data throughput requirements, we explored the power/performance tradeoffs between the two architectures, including near threshold voltage computing, for different workloads using an 8-lead ECG signal conditioning application. Our results show that the multi-core approach consumes 66% less power than the single-core approach for high biosignal computation workloads (i.e., 50.1 $MOps/s$). However, the multi-core architecture becomes more power consuming for relatively lighter workloads and it consequently consumes 10.4% more power (681 $kOps/s$).

## References

1. Mamaghanian, H., et al.: Compressed Sensing for Real-Time Energy-Efficient ECG Compression on Wireless Body Sensor Nodes. IEEE Transactions on Biomedical Engineering 12, 120–129 (2011)

2. Hanson, M.A., et al.: Body Area Sensor Networks: Challenges and Opportunities. IEEE Computer 42, 58–65 (2009)
3. Rincon, F., et al.: Multi-Lead Wavelet-Based ECG Delineation on a Wearable Embedded Sensor Platform. Computers in Cardiology, 289–292 (2010)
4. Hanson, S., et al.: Exploring variability and performance in a sub-200 mV processor. IEEE J. Solid-State Circuits 43, 881–891 (2008)
5. Zhai, B., et al.: A 2.60 pJ/Inst subthreshold sensor processor for optimal energy efficiency. In: Symposium on VLSI Circuits. Digest of Technical Papers, Honolulu (2006)
6. Wang, A., Chandrakasan, A.: A 180 mV FFT processor using sub- threshold circuit techniques. In: IEEE Int. Solid-State Circuits Conference. Dig. Tech. Papers (2004)
7. Dreslinski, R.G., et al.: Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. Proceedings of the IEEE 98, 253–266 (2010)
8. Chen, G., et al.: Millimeter-scale nearly perpetual sensor system with stacked battery and solar cells. In: Solid-State Circuits Conference. Digest of Technical Papers, San Francisco (2010)
9. Hanson, S., et al.: A Low-Voltage Processor for Sensing Applications With Picowatt Standby Mode. IEEE J. Solid-State Circuits 44, 1145–1155 (2009)
10. Dreslinkski, R.G., et al.: An Energy Efficient Parallel Architecture Using Near Threshold Operation. In: 16th International Conference on Parallel Architecture and Compilation Techniques, Brasov, pp. 175–188 (2007)
11. Yu, P., et al.: An Ultra-Low-Energy Multi-Standard JPEG Co-Processor in 65 nm CMOS With Sub/Near Threshold Supply Voltage. IEEE J. Solid-State Circuits 45, 668–680 (2010)
12. Krimer, E., et al.: Synctium: a Near-Threshold Stream Processor for Energy-Constrained Parallel Applications. Computer Architecture Letters 9, 21–24 (2010)
13. Sun, Y., et al.: ECG signal conditioning by morphological filtering. Computers in Biology and Medicine 32(6), 465–479 (2002)
14. Rahimi, A., et al.: A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1–6 (2011)

# Agent-Based Thermal Management Using Real-Time I/O Communication Relocation for 3D Many-Cores

Thomas Ebi[1], Holm Rauchfuss[2], Andreas Herkersdorf[2], and Jörg Henkel[1]

[1] Karlsruhe Institute of Technology
{thomas.ebi,henkel}@kit.edu
[2] Technische Universität München
{holm.rauchfuss,herkersdorf}@tum.de

**Abstract.** A major concern of current and future on-chip systems is the thermal problem i.e. electrical energy is dissipated leading to high chip temperatures. Short term effects may include transient malfunctioning whereas long-term effects may lead to deteriorating functionality (e.g. increased signal travel times) or to irreversible damage due to, for example, electro-migration. The problem worsens with the inception of 3D architectures as the per-surface dissipated electrical energy is larger, e.g. our evaluation shows an increase of 37.5% in peak temperature in an architecture with 2 layers compared to a single layer architecture. Our proposed concept addresses thermal problems in 3D-stacked many-core architectures resulting from high power densities. A hierarchical agent-based thermal management system initiates a proactive task migration onto cooler processing resources while a communication virtualization layer dynamically adapts and protects connectivity between (migrated) tasks and external I/Os.

**Keywords:** Dynamic thermal management, Communication virtualization, 3D-architectures, Many-core systems, MPSoC.

## 1 Introduction

Dependability of embedded on-chip systems becomes an increasing problem as feature sizes continue to shrink. Circuit scalabing has been successfully practiced for several decades but it seemingly reaches its limits. This can be observed by an increased susceptibility of nano-CMOS transistors, i.e. with structure sizes of 45nm and beyond, against errors due to temperature. Though dependability has been a problem since the early days of miniaturization [20], classical methods of fault tolerance like ECC (Error Correction Codes) etc. were sufficient then to address reliability concerns. In recent years, however, it has been recognized that a paradigm shift is necessary [5] and existing solutions are unlikely to scale, and we will need radically new solutions (see [1]). In recent years, various paradigms have been proposed to address the current and upcoming dependability problems. As categorized in [17], in the recent past concurrency and flexibility

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 112–121, 2011.

have been deployed as a means to address the dependability problem and it is envisioned that future approaches go a step further and develop novel means with respect to *resiliency* and *self-adaptability*. Thermal-induced dependability concerns will pose a key problem for future on-chip systems: even though the power dissipation per transistors decreases from technology node to technology node, the power dissipated into heat per area (density) has increased almost exponentially in the past [2].

Moreover, the problem becomes even more severe in the case of future 3D architectures [8] since the relation of surface package/chip area to the number of integrated transistors turns significantly worse (i.e. far smaller). On the positive side, the advantages of 3D architectures are promising and include decreased signal delay between various system components as opposed to 2D architectures etc. [8,13]. Thermal issues have already guided the design and evolution of recent 2D architectures, where thermal management techniques such as clock frequency scaling or dynamic voltage frequency scaling are successfully employed [6]. While stacked 3D architectures have been shown to have increased performance compared to their 2D counterparts [4] – especially in regards to the throughput and latencies of the communication architecture [12] – they are much more prone to excessive heating as the heat needs to dissipate through multiple layers before reaching the heat sink [4].

To address these issues in upcoming 3D architectures, especially for embedded systems with their hard real-time constraints and requirements regarding QoS, latency, etc., new approaches are needed. To allow for faster cooling of hot tiles tasks should be migrated [15] cooler ones. The migration should be triggered by a scalable thermal management. To minimize the downtime (i.e. tasks are stopped) during migration and afterwards guarantee reliable I/O of these tasks with the environment or other tasks a virtualization layer should provide a high-performance abstraction of the communication channels.

The paper is structured as follows: Section 2 provides an overview of related work with respect to thermal management and (I/O) virtualization. Section 3.1 describes the proposed concept for the agent-based, hierarchical thermal management and 3.1 for the communication virtualization, respectively. A preliminary scenario for task migration under real-time constraints is given in Section 4. Our evaluation platform and thermal evaluation is described in Section 5 while Section 6 summarizes the paper.

## 2 Related Work

Dynamic Thermal Management (DTM) approaches rely on reducing the power consumption at thermal hot spots through DVFS and power gating during run-time. Additionally, multi-core architectures have the opportunity to divert the power consumption away from thermal hot spots through run-time task migration. This technique is particularly effective in 3D many-core architectures where a task running on a core has a significant effect on surrounding cores, especially in the vertical direction. A reactive DTM scheme for 3D multi-core architectures

proposed in [9]. identifies task remapping as a key solution to thermal problems in 3D environments, due to the severe performance hits that measures like DVFS or clock gating (stalling) of hot cores incur. This work does not consider scalability for large systems because no distributed controlling scheme is introduced. Additionally, the communication volume and –latency is not taken into account and task remapping cost is not modeled despite the fact that each task remapping is assumed to constantly require 1ms. Zhou et al [22] presented a reactive OS-level task scheduling approach to DTM for 3D systems. Thermal correlation in vertically overlapping cores is considered and their work focuses around tasks as causes for thermal issues. Tasks that cause a large amount of thermal stress to cores are considered hot tasks while computationally inexpensive tasks that cause less thermal stress are considered to be cold tasks. Consequently, hot tasks are assigned less time slices than cold tasks by the scheduling algorithm. Zhu et al [23] consider DTM in 3D architectures not as an emergency measurement whose invocation should be avoided as much as possible, but rather find that because large 3D systems are likely to operate continuously close to the thermal threshold, DTM needs to be invoked continuously. Therefore, DTM techniques that come with a severe performance penalty are considered impractical. When considering several hundreds of cores in future 3D many-core architectures, the scalability of any approach which aims at managing some aspect of these cores becomes a major concern. Any approach where the communication volume and the computational complexity grow too rapidly (e.g. exponentially in respect to the number of cores) will not be suited for a system with a large number of cores.

Regarding virtualization, future many-core systems shall provide a consolidating environment for hundreds or even thousands of tasks. Those tasks have different requirements towards their running environments in terms of real-time, required operating system and performance/throughput. The system needs to be provisioned, scheduled and partitioned into domains for tasks concurrently running. Furthermore, on future dependable systems those tasks are also required to be easily migrate-able between cores and it must be possible to restart them in a transparent way, i.e. they can resume running without reset and their running environment and configuration is recreated. Virtualization of the physical HW by a Hypervisor or Virtual Machine Monitor (VMM) is an established concept for providing such a function. To increase performance the HW may feature extensions supporting virtualization. I/O-devices present a central bottleneck as the cost for I/O-virtualization is particularly high with respect to latency, utilized processing resources and reduced throughput. Future many-core architectures with hundreds or even thousands of domains need to share a limited number of high-speed network-devices. Sharing physical network access between domains may be implemented in HW, SW or in a mixed mode [19]. The generic solution i.e. VMM only, dedicates one virtualization domain as driver domain and exclusively assigns the network card to it [3]. A further improvement to the upper scenario is the employment of multi-queue network cards [16]. The shortcoming for SW-based bridging and multi-queue network cards is that they rely on a driver domain which is interleaved in the network communication path. This

results in an increased latency, (complex) scheduling dependencies and utilization of host cpu and system memory. Multiple-queue network cards are limited in the number of available queue pairs. Domains can also directly access a NIC via virtual network interfaces [21]. Apparently, such approaches require extensions of the NIC i.e. dedicated queues, buffers, interfaces and additional management logic, mostly in form of on-board processing elements running a SW. The concept for direct I/O is restricted to the number of virtual network interfaces implemented in HW. VMM (Virtual Machine Manager) is an established and widely applicable virtualization concept for processing resources. VMM also incorporates centralized and software-based means to manage Inter-Processor and I/O Communication (IPC). However, this centralized and software-based approach is not scalable towards many-core systems with hundreds of processing elements, nor is it adequate for high data throughput (Gb/s), Quality of Service (QoS) and real-time guarantees as this relies on a SW component.

## 3 Concept of Thermal Management for 3D-Stacked Many-Core Architectures

Our approach is divided into two tiers: At the *system level*, thermal management is performed by determining *when* and *where* tasks should be migrated, and in the *architecture level* communication virtualization is utilized in order to keep inter-task communication transparent and meet real-time constraints. These two aspects are detailed in the following.

### 3.1 Thermal Management

In order to provide a light-weight and scalable infrastructure for making task migration decisions we employ the paradigm of agent-based systems. It allows for a self-adaptive and decentralized approach and may cope with global complexity through the local decisions and cooperation with fellow agents – therefore promising scalability [10]. The latter feature is crucial for future many-core systems where hundreds or even thousands of cores are integrated on a 3D architecture. Critical parameters are the structure of such a multi-layer agent system (i.e. hierarchy) and the functional division of individual agents. We have seen in [10] that the most effective configuration of the agent system is a tree-like agent structure consisting of:

- **Low level agents** act on local sensor data. These may be migrated, thus resulting in more favorable thermal sensor readings. Each tile will also include a monitoring infrastructure [11]. However the location of the low level agents – e.g. maybe only 1 per "cylinder" (i.e. a vertical stack of tiles) at first, then more as the temperature rises – and their sampling interval based may be changed depending on the current thermal properties.
- **Cluster level agents** handle most of the decisions in a connected region of adjacent tiles.
- **Global agents** coordinate between cluster agents. There may be more than one global agent to increase reliability through redundancy.

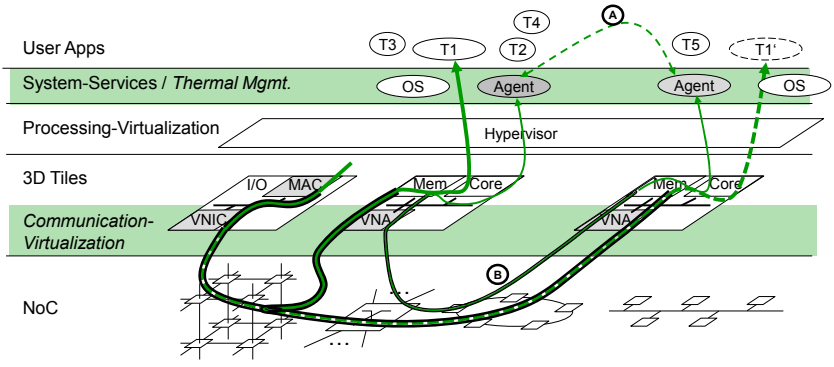



**Fig. 1.** Thermal Management and Communication Virtualization

Furthermore, it is possible to spawn new agents, or a sub-hierarchy of agents, during run-time in case of failure or during proactive actions. The resulting agent system structure is thus dynamic and able to tailer to changing requirements as needed. This is necessary because the overall thermal management system needs to adapt to the frequency and amount of actions associated with mitigating potential thermal hot spots. Logical connections between agents, as seen in Fig. 1 Ⓐ are realized through the virtual physical connections Ⓑ of the communication virtualization outlined in the next section.

### 3.2 Dynamic Communication Virtualization

Communication virtualization provides dependable connection administration (i.e. connection setup, modification and resolution), as well as data transport services with performance and data integrity guarantees for process-to-process, process-to-memory, and process-to-I/O transactions among 3D tiles for computing, memory and I/O. Our focus is on techniques and architectural enhancements in HW to ensure that the intended communication entities become and remain connected in a robust way with specified service guarantees (based on [18]). This implies that communicating user processes may move to different compute tiles during their connection time as a result of task migration. Service guarantees for communication virtualization services primarily focus on aspects relevant to applying 3D many-cores in embedded system environments. Consequently, support for multiple data transport priority classes and compliance to real-time deadline requirements are a strong prerequisite to support real-time tasks in parallel to throughput-orientated tasks on the same 3D many-core architecture.

As shown in Fig. 1, dynamic communication virtualization is realized as a virtualization overlay on top of arbitrary physical NoC interconnects. Specific NoC architectures integrate particular aspects of communication virtualization into the elementary NoC operation, e.g. flit-based wormhole routing in 2D mesh networks with virtual channel buffers for separating logical messages transmitted through an identical physical link. Furthermore, 2D/3D meshes provision

multiple physical routing paths between network entry and exit points representing a virtual connection. However, none of today's NoC concepts check the validity or plausibility of packet addresses, nor does the notion of a connection within a NoC extend up to the user/application process level. The virtualization overlay, when integrated into the network adapter entities of 3D tiles, i.e. *compute* tiles, *memory* tiles and *I/O* tiles, through VNIC (Virtual Network Interconnect) and VNA (Virtual Network Adapter) units, provides dependable communication virtualization services independent of the capabilities of the underlying physical interconnect. Modularizing and distributing virtualization functions and utilizing the agent-based thermal management for triggering and managing only those VNICs and VNAs involved in a to be (re-)configured communication path ensures scalability for 3D many-core architectures.

## 4 Thermal Management and Virtualization under Real-Time Constraints

Dependable 3D many-core computing in embedded system environments frequently requires support for coping with hard real-time constraints. Our approach to DTM allows task migration while ensuring the re-routing of corresponding communication channels under hard (and soft) real-time constraints i.e. real-time tasks can be migrated during run-time without violating their deadline. The interlaced operation of task management with communication virtualization is shown as an example in Fig. 2. Task migration is proactively triggered by the thermal management and it is split into several phases.

Let task T1 be a highly-critical, real-time task running on the left compute tile, with a period $T_P$ and a deadline $T_D$. The agent-based thermal management is then triggered by a thermal event i.e. the tile temperature becomes critical (Ⓐ). Thermal management agents negotiate with each other (Ⓑ) and decide that T1 is migrated to the right compute tile. In order to provision sufficient

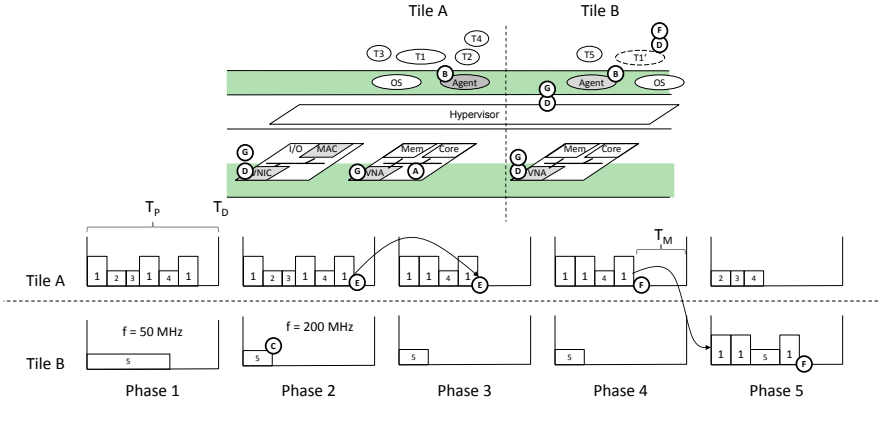



**Fig. 2.** Migrating Task under Real-Time Constraints

processing capacity for task T1, the frequency of the right core is quadrupled (e.g. as it was running in power-safe mode Ⓒ). In our approach, proactive task migration mandates that communication resources (in the Hypervisor, VNA and VNIC) are already pre-configured and a stand-by copy of the task is established on the target core before the actual task migration is initiated (Ⓓ). Thus, time consuming operations are eliminated beforehand out in the critical migration path, but are throttled by spare ressources utilized for this in the involved tiles.

Now, let $T_M$ be the time required for migrating the task state and adapting the communication path between active and stand-by core. $T_M$ obviously depends on the size of the task state and the underlying NoC infrastructure. As $T_M$ may be larger than the time interval between completing T1 in Phase 2 and restarting T1 in Phase 3 (Ⓔ), the task management may have to re-schedule tasks on the original core to maximize this interval (Ⓕ). After finishing T1 in Phase 4, the state of T1 is copied to the shadow task and communication resources are switched between cores such that task T1 can resume operation in Phase 5 on the new target core (Ⓖ) while meeting its deadline.

Since $T_M$ plus the execution latency of a real-time task represent a lower bound for the execution period $T_P$ of a cyclic task, we need to find analytical means to determine an upper bound (worst case) for $T_M$ in dependence of the state size, NoC characteristics and the migration distance between source and target core. The time to logically switch virtualized communication channels between tiles, and thus the techniques in which VNIC and VNA are realized, are critical for obtaining an as small as possible $T_M$. Likewise, efficiency of the hierarchically structured, distributed agent system for task management will affect both the rate of real-time task migrations as well as the absolute number of real-time tasks (with an upper bound of $N_{rt}$) that can be executed by the system.

## 5 Evaluation Platform and Thermal Evaluation

An evaluation platform for prototyping the key parts of the agent-based thermal management and the communication virtualization concepts is presented by means of the prototype shown in Figure 3. To enable development, the prototype

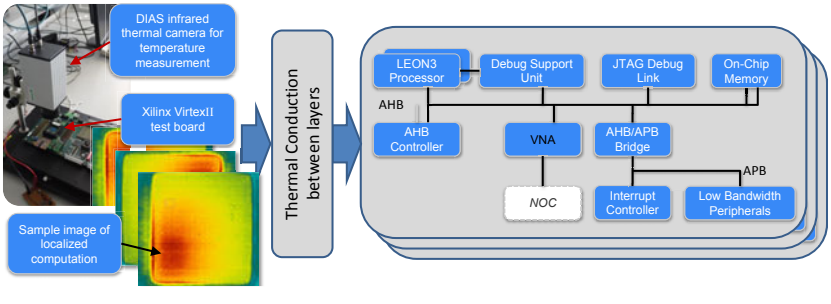



**Fig. 3.** Evaluation Platform for Thermal Management and Communication Virtualization

is realized on an FPGA with a subset of the actual 3D many-core architecture. This is because of the limited capacity of an FPGA and for reducing the overall implementation effort. Only one die layer of a 3D many-core system with a limited number (e.g. eight) of compute tiles – each compute tile contains one RISC core, e.g. Leon3 – can run on the FPGA.

The thermal distribution of each 2D layer is measured separately with an infrared thermal camera. This allows real data to be acquired to determine the thermal state of each layer. On the other hand, current, state-of-the-art approaches rely mainly on thermal models which use high-level power simulation as input. These power simulations have been shown to be inaccurate to a degree of up to 30% [7]. This may lead to thermal behavior which differs significantly from what was predicted in the simulation. Through the use of the thermal camera, however, the inaccuracy induced by the power simulation is avoided. After the temperature distribution inside every layer has been determined, these are combined by calculation of the conducted heat between layers, based on a differential equation solver similar to the one used by the HotSpot thermal simulator [14] and the specific heat capacity $c$ and conductivity $k$ of silicon.

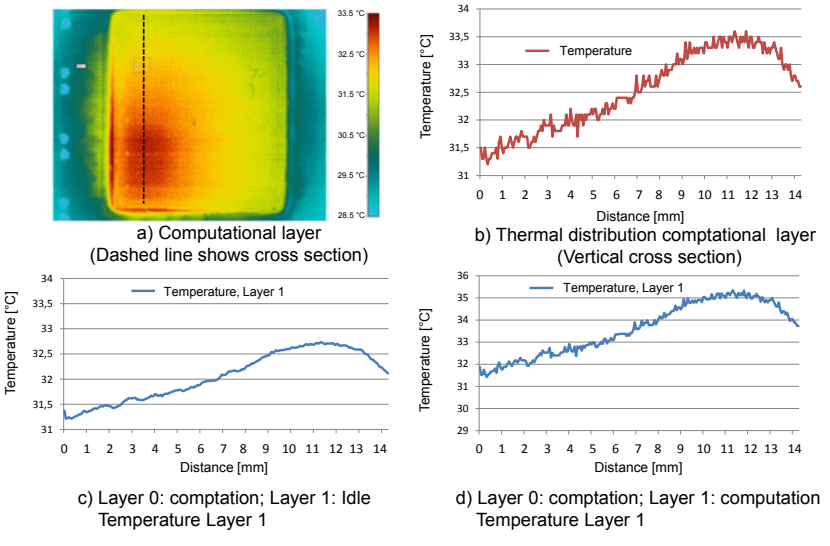



**Fig. 4.** Thermal Behavior of the 3D-Stacked Architecture

For evaluation of the thermal characteristics of the system we have selectively placed 1000 T-flip flops toggling at 50MHz in a rectangular region in one corner of a Xilinx VirtexII FPGA which generates the worst-case switching power in the specified region resulting in high heat generation. Figure 4.a shows the resulting thermal distribution inside of one layer in a one-dimensional cross section shown in Figure 4.b, corresponding to the line shown in Figure 4.a. Assuming an idle layer, i.e. one without switching power consumption, “Layer 1” is placed on top

of a computational layer, "Layer 0", we can calculate the resulting heat dissipation based on the thermal conductivity $k$ and capacity $c$ of silicon. The result is shown in Figure 4.c, where $c$ determines the total amount of heating and $k$ determines how far the heat is spread out on the upper layer, essentially smoothing out the temperature distribution. Similarly, Figure 4.d is the resulting thermal distribution if two layers with the computation of Figure 4.a placed on top of each other. The mutual heating causes a 37,5% increase in peak temperature from the ambient temperature of 31°C.

## 6 Conclusion

We introduced a new thermal management approach for 3D stacked many-core architectures. An initial analysis of the thermal behavior of 3D many-core architectures using real thermal data as opposed to data based on inaccurate power simulations shows the thermal effect of task placement in 3D many-core architectures – with an inefficient placement resulting in a peak temperature increase of 37.5%. Communication virtualization is proposed as a means for making a distributed agent-based management scheme able to meet real-time constraints and keep communication transparent when migrating tasks, and its need is highlighted through an example scenario.

**Acknowledgments.** This work is supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 - spp1500.itec.kit.edu).

## References

1. Austin, T., Bertacco, V., Mahlke, S., Cao, Y.: Reliable systems on unreliable fabrics. IEEE Design & Test 25(4), 322–332 (2008)
2. Banerjee, K., Mehrotra, A., Sangiovanni-Vincentelli, A., Hu, C.: On thermal effects in deep sub-micron vlsi interconnects. In: Proceedings of the 36th Annual Design Automation Conference DAC 1999, New Orleans, Louisiana, United States, pp. 885–891. ACM, New York (1999)
3. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles SOSP 2003, pp. 164–177. ACM, New York (2003)
4. Black, B., Annavaram, M., Brekelbaum, N., DeVale, J., Jiang, L., Loh, G.H., McCaule, D., Morrow, P., Nelson, D.W., Pantuso, D., Reed, P., Rupley, J., Shankar, S., Shen, J., Webb, C.: Die stacking (3d) microarchitecture. In: MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 469–479. IEEE Computer Society, Washington, DC, USA (2006)
5. Borkar, S., Jouppi, N.P., Stenstrom, P.: Microprocessors in the era of terascale integration. In: Proceedings of the Conference on Design, Automation and Test in Europe DATE 2007, pp. 237–242. EDA Consortium, San Jose (2007)
6. Brooks, D., Martonosi, M.: Dynamic Thermal Management for High-Performance Microprocessors. In: Proceedings of the 7th International Symposium on High-Performance Computer Architecture, p. 0171 (2001)

7. Cochran, R., Nowroz, A.N., Reda, S.: Post-silicon power characterization using thermal infrared emissions. In: International Symposium on Low-Power Electronics and Design, pp. 331–336 (2010)
8. Cong, J., Zhang, Y.: Thermal-driven multilevel routing for 3-d ics. In: Proceedings of the Asia and South Pacific Design Automation Conference ASP-DAC 2005, pp. 121–126. ACM, New York (2005)
9. Coskun, A.K., Ayala, J.L., Atienza, D., Rosing, T.S., Leblebici, Y.: Dynamic thermal management in 3d multicore architectures. In: Design, Automation and Test in Europe DATE 2009, pp. 1410–1415 (April 2009)
10. Al Faruque, M.A., Jahn, J., Ebi, T., Henkel, J.: Runtime thermal management using software agents for multi/many-core architectures. IEEE Design & Test 27(6), 58–68 (2010)
11. Al Faruque, M.A., Ebi, T., Henkel, J.: ROAdNoC: Runtime observability for an adaptive network on chip architecture. In: IEEE/ACM International Conference on Computer-Aided Design ICCAD 2008, pp. 543–548 (2008)
12. Feero, B., Pande, P.P.: Performance evaluation for three-dimensional networks-on-chip. In: IEEE Computer Society Annual Symposium on VLSI ISVLSI 2007, pp. 305–310 (March 2007)
13. Haensch, W.: Why should we do 3d integration? In: Proceedings of the 45th Annual Design Automation Conference DAC 2008, pp. 674–675. ACM, New York (2008)
14. Huang, W., Stan, M.R., Skadron, K., Sankaranarayanan, K., Ghosh, S., Velusam, S.: Compact thermal modeling for temperature-aware design. In: DAC 2004, pp. 878–883 (2004)
15. Jahn, J., Al Faruque, M.A., Henkel, J.: Carat: Context-aware runtime adaptive task migration for multi core architectures. In: Design, Automation and Test in Europe, DATE 2011, pp. 1–6 (March 2011)
16. Ram, K.K., Santos, J.R., Turner, Y., Cox, A.L., Rixner, S.: Achieving 10 Gb/s using safe and transparent network interface virtualization. In: Proceedings of the International Conference on Virtual Execution Environments, pp. 61–70. ACM, New York (2009)
17. Rabaey, J.M., Malik, S.: Challenges and solutions for late- and post-silicon design. IEEE Design & Test 25(4), 296–302 (2008)
18. Rauchfuss, H., Wild, T., Herkersdorf, A.: A network interface card architecture for i/o virtualization in embedded systems. In: Workshop on I/O Virtualization (2010)
19. Santos, J.R., Turner, Y., Mudigonda, J.: Taming heterogeneous nic capabilities for i/o virtualization. In: Workshop on I/O Virtualization (2008)
20. von Neumann, J.: Probabilistic logics and synthesis of reliable organisms from unreliable components. In: Automata Studies, pp. 43–98. Princeton University Press, Princeton (1956)
21. Willmann, P., Shafer, J., Carr, D., Menon, A., Rixner, S., Cox, A.L., Zwaenepoel, W.: Concurrent direct network access for virtual machine monitors. In: Proceedings of the 13th International Symposium on High Performance Computer Architecture, pp. 306–317. Citeseer (2007)
22. Zhou, X., Xu, Y., Du, Y., Zhang, Y., Yang, J.: Thermal management for 3d processors via task scheduling. In: 37th International Conference on Parallel Processing ICPP 2008, pp. 115–122 (September 2008)
23. Zhu, C., Gu, Z., Shang, L., Dick, R.P., Joseph, R.: Three-dimensional chip-multiprocessor run-time thermal management. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(8), 1479–1492 (2008)

# Energy Estimator for Weather Forecasts Dynamic Power Management of Wireless Sensor Networks

Nicolas Ferry[1], Sylvain Ducloyer[1], Nathalie Julien[1], and Dominique Jutel[2]

[1] Lab-STICC, Centre de Recherche, Rue Saint Maudé,
56100 Lorient, France
{nicolas.ferry,sylvain.ducloyer,nathalie.julien}@univ-ubs.fr
[2] ERYMA Security System, Z.I. La Montagne du Salut,
56600 Lanester, France
{dominique.jutel}@eryma.com

**Abstract.** Emerging Wireless Sensor Networks (WSN) consist of spatially distributed autonomous sensors. Although an embedded battery has limited autonomy, most WSNs outperform this drawback by harvesting ambient energy from the environment. Nevertheless, this external energy is very variable and mainly depends on weather evolution. Therefore, including weather at design stage and weather forecasts at runtime is essential for autonomy management. This paper presents a Power Estimator to simulate node autonomy for various weather conditions and locations. This work also addresses the integration of Weather Forecasts in the Dynamic Power Management policy (WF-DPM). These two contributions significantly improve the system scaling and the energy availability prediction to help to achieve better node autonomy duration. Experimental results compared various locations to study the weather impact on the system autonomy.

**Keywords:** Wireless Sensor Networks, Weather Forecasts, Dynamic Power Management, Design Tools.

## 1 Introduction

Wireless Sensor Networks (WSNs) are distributed communicating embedded systems. With constant technological progress in thinness and wireless communication, nodes are given increasingly improved features, thus leading to their deployment in more wide-range of environments. As stand-alone equipment, they must be autonomous in energy. Most embedded systems integrate cells, but they keep a limited autonomy. To overcome this drawback, some WSN harvest ambient energy directly or recharge an energy storage system. These WSN have a self-recharge feature that increases the autonomy for a limited amount or for a complete recharge. From the other side, the WSN application scenario depletes the energy resource. Thus, the problem is to estimate the node autonomy as the recharge factor can be highly variable. To handle this point, environment must be take into account; and most criteria like solar, wind, and temperature depend more precisely on weather. Thus, the difficulty is to find relatively

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 122–132, 2011.

good environment estimators to predict this variability. And currently, the best weather estimator we have is given by forecasters.

In this article, our contribution is to propose a Power Estimator design tool able to integrate Weather Forecasts (WF) into a Dynamic Power Management (DPM) to simulate the system autonomy for specific weather environment, scenario, hardware configuration, RF activity, and DPM algorithm. The Power Estimator (PE) goal is to identify at design level the power consumption hot-spots, and thus, to scale accordingly the system in term of harvesters, low power components, battery capacity, and DPM strategy.

This research contribute to the CAPNET project which joins industrial and laboratory partners to equip fire services brigades. In this project, a linear WSN demonstrator must be realize to monitor gas spread and person intrusions in firemen secure area.

This paper is organized as follows: section 2 provides a state-of-the-art on WSN design tools. In section 3, we present available weather forecast models. WF-DPM with QoS strategies is described in section 4, while section 5 focuses on the Power Estimator and the experimental results before concluding.

## 2 States-of-the-Art

Low Power Design [1] advancement has recently moved onto a higher level with algorithmic dedicated implementation, and operating system advancement with power down modes, DVFS, and power-aware scheduling. Thus, power reduction can be applied all along the co-design process [2] by respecting a low power design methodology [3].

The emerging wireless sensor network domain has existed for a few years, however an increasing number of projects have been referenced [4]. Technology integration improves and nodes tend to achieve lower size like Smartdust and the Picocube project [5] which is a 1 cm3 sensor node powered by harvested energy. In this part, we focus mainly on new prototypes for which harvested energy is provided by the weather environment.

Some interesting rechargeable WSN are Heliomote [6] which uses a voluminous solar panel directly connected to a two AA Ni-MH battery through a diode. Prometheus [7] uses a solar panel directly hardwired to a 22F supercapacitor which directly supplies power to the system or charges a secondary Li-Polymer battery. Like Heliomote, it does not harvest energy efficiency with an MPPT. Everlast [8] only has a solar panel and a supercapacitor, but no battery. For optimal power harvesting, it uses a MPPT approach that charges a 100F supercapacitor. However, it requires MCU to perform the MPPT algorithm calculation. PUMA [9] has multiple power sources. It uses a power routing technique to harvest simultaneous multiple power sources. The high energy harvesting is achieved with a combination of MPPT and power defragmentation. However, it requires MCU control based on input from light and wind sensors. AmbiMax [10] is a multiple harvester system with solar and wind experiments. It includes MPPT tracking using an hysteresis comparator coupled with an additional sensor to hardware calculated MPPT. The harvested power is sent to a boost regulator before charging a Li-Polymer battery.

We have noted that these systems embed energy harvesters, but none of them, except Duranode with its current solar policy, has sought to predict the potential energy harvested for energy management.

In the WSN domain, CMOS circuit power consumption repartition evolves so that clock and leakage are no longer the main drawbacks against RF transceivers and heavy industry sensors. Thus, concentrating the analysis only on the MCU or the operating systems is not a complete approach. It is recommended that the tool supports a power estimation of the whole system.

In that domain, power estimation tools work at specific levels. At physical level, SPICE is a general-purposes circuit estimator. It is well defined for transistor and PCB simulations, but requires an enormous amount of work to model the whole system. At instruction levels, there are many MCU simulators: Simbed for the Motorola M-CORE, Skyeye for the MIPS platform, eSimu for the ARM9 and EMSIM for the strongARM platform. SoftExplorer is a CPU Power Estimation tool which estimates power consumption of C or ASM languages for a modeled CPU. Specific FPGA cards rely directly on Xilinxs, Altera or Intel manufacturers, but all of these simulators are accurate for the particular specific chip for which it is intended to work. Nevertheless in WSN, MCU is definitely not the most consuming resource. This is particularly true, that in our project MCU consumes no more than 2% of the whole system. At the architectural level, prototyping work includes PowerOpt from ChipVision, which is a low-power synthesis C/SystemC to verilogRTL tool. Or Interconnect Explorer which studies bus interconnexion power consumption. These tools are useful for design optimization at their level, but here again a global vision with less precision could be more pertinent for our needs. At the OS level, Softwatt analyses power dissipation for Embra, Mipsy, and MXS platforms. One interesting tool is powerTOSSIM which is a TinyOS Simulator mainly for the Mica2 node. Finally, new research projects are trying to model the WSN in the whole by using macromodeling, or components modeling like CAT which is the nearest whole system simulator. However, they do not model the external environment which impacts weather and rechargeable energy.

Thus, we underline the lack of a global view simulator, which may be less accurate than dedicated separate tools but which can integrate the entire components and the weather environment. So, what models can be used to predict weather environment?

## 3 Weather Forecasts

Meteorology is the science which studies atmospheric phenomena mainly in the first 31 km of the atmosphere. The meteorologist Lorenz discovered the chaos theory [11] showing that the atmosphere cannot be entirely determinist. Small variations in the initial states can result in huge consequences in the final results: this phenomenon is known as the "Butterfly Effect". Therefore, the final interpretation is left to human interpretation to formulate the weather forecasts. A new predictive approach consists in evaluating a set of different runs. Each run slightly modifies an initial parameter corresponding to a well-chosen particular error/perturbation. Therefore, the standard deviation can be exploited to give a trusted indicator for the predictions.

Multiple Meteorology centers use home or different meteorology models. We could cite many: WRF, GFS, ECMWF, UKMO, JMA, ARPEGE, and AROME... Among these different models, the Global Forecast System (GFS) is an American numerical weather forecast model. This model divides the Earth into a mid-meshing grid of squares of 30 km and the atmosphere into 64 altitude layers. A model

computation is called "a run" and predicts up to 16 days forward. The main advantage of this model is the output which is freely available and in real-time after generation. We reuse information given by the Meteociel website [12] which digests the GFS information in a vector. Each piece of interesting information is collected to supply the simulated harvested external energy potential. However, solar information is only given as a variable icon representation. To translate this information, we rely on another website to convert solar potential into a sunset irradiance in W/m2.

The European JRC - Photovoltaic Geographical Information System (PVGIS) is a website [13] for photovoltaic technology. For Europe and Africa, the system can give a photovoltaic estimation of the solar potential during an entire typical day of a specific month. PVGIS gives three sunset indicators: IR global clear-sky sunset, IR global real-sky sunset, IR diffuse real-sky sunset as a time function. IR global clear-sky is the maximal sunset irradiance expected with no nebulosity. IR global real-sky is the standard sunset irradiance for the average cloud-covered sky in a month. IR diffuse real-sky is the expected sunset irradiance by diffuse luminosity which can be translated as the minimum irradiance that can be harvested even by a heavily clouded sky. All these information must be merged to provide the weather coefficients.

Since meteorology data are now widely available through multiple sources, they can be extracted directly from Internet websites. Weather coefficients can be combined together to produce an energy function prediction over time. In the Power Estimator, the scenario builder works as follows for the meteorology part. It can connect to specific meteorology Internet websites, from which the weather data is downloaded in HTML string syntax. The next step consists in parsing and extracting the weather data from the page. Of course, this module is generic and is implementation depends on the target website. Weather Data are presented in a vector structure where each column represents a weather component (Sunset irradiance, Wind speed, Temperature...) and the row increments in the time dimension. As most websites do not give the sunset irradiance, we must combine various website data.

Depending on your world position (Latitude, Longitude) on the globe, and the month in which the simulation must run, PVGIS gives you sunset measurements: IR clear-sky sunset, IR med sunset, IR diffuse sunset. These values are extracted for each month once at start up. The time granularity is given every 15 minutes.

Weather data are interpolated according to a Lagrange Law, while PVGIS data are interpolated as well, and finally they are combined into a two dimensional vector of the simulation resolution. The result is a data file that can be exploited in the simulator to reproduce the weather parameters. However, we tested the Runge interpolation error that forced us to fallback to a Spline or Hermite cubic interpolation. Since energy is scavenged from the environment, the node model must include a model for each power source type. Currently, CAPNET focuses only on solar, wind, and thermal power sources. The meteorology context is constructed with the Weather-Builder integrated into the simulator. The context can be created manually, scanned from a website, or extracted from a meteorology file recorded day after day.

Then, when the DPM replays a scenario, the weather state is updated and provides the weather data for the simulation. Finally, the simulator converts them into power depending on the physical installed hardware.

However, we must keep in mind that despite WF models continuously evolved and offered better accuracy; WFs suffer from interpretation errors and/or inaccuracies.

Currently, most WF models can predict 1-day up to 3-days forward with relatively good accuracy but a complete week with more uncertainty. The WF theory limit is announced to a maximal of a two-week forward window.

## 4 Dynamic Power Management

Dynamic Power Management (DPM) policy controls the different power states enabled for the system. This model is not directly the microprocessor program, but rather a subset of the global power states view. The DPM has been modeled as a Finite State Machine (FSM) composed of a set of states $\{S_i\}$ and transitions $\{T_i\}$. Each state represents a particular system power consumption state $\{Pt_i\}$, and has a variable duration $\{Tt_i\}$. The DPM commands the behaviors of the system, entering special power modes or monitoring power supply. A Sequence is a set of one to many states linked together and form a path to produce a service.

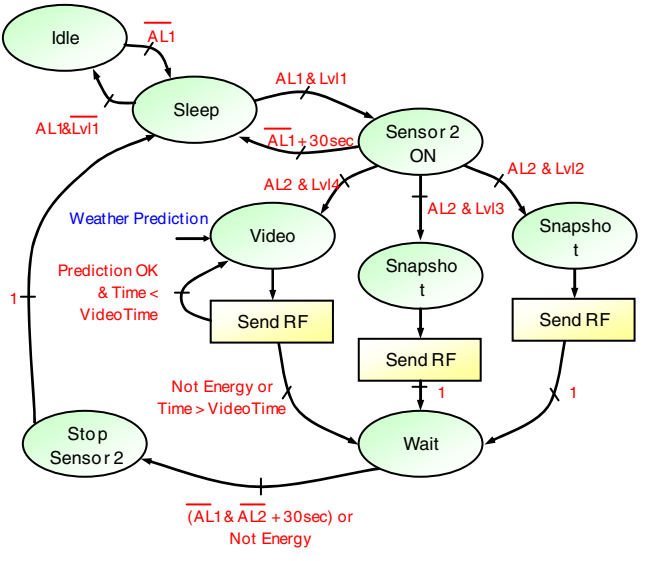



**Fig. 1.** A WF-DPM example using weather forecasts results to make low power decisions

The WF-DPM model introduces the capacity to exploit the meteorological WF coefficients. WFs can be used for various applications like power switch policies to determine which power harvesting source will probably have the highest energy availability. For CAPNET application, WFs is computed on the supervisor which transfer only the indicator to stay or not in waiting states.

In CAPNET, the intrusion detection service is defined by the DPM shown in figure 1. The system uses two sensors for detection except when there is not enough energy; the node directly sends an alarm and enters an IDLE waiting state. When the sensor 1 detection flag is raised, Sensor 2 is activated to verify the intrusion. If both are raised, the node sends an alarm to the supervisor. With lot of energy, the system sends a video for a variable duration. Or it sends a snapshot in high or low resolution depending on the

energy level. Then, the DPM waits for the two sensor releases before stopping sensor 2 and returning to Sleep State. Video time duration depends on a maximal timing and the energy prediction function.

We define an energy function based on a real energy state of charge (SOC) and the WFs. With WFs, we have prediction data for each ambient microsource: the sunset irradiance potential in W/m2, the wind speed potential in Km/h, the thermal temperature potential in °C. Due to specific models built for each microsource, we can determine the equivalent power produced. This gives us a power representation in Watt and, for a one-second resolution, the energy in Joules. Each microsource power potential is cumulated to provide the Potential Ambient Energy Value (PAEV), which is a total amount of external energy expected in the near future. We can quantify the harvested energy by expressing the PAEV as:

$$PAEV = \sum_{i=1}^{n} E_i . \mathrm{Re}\, s \tag{1}$$

Where i is the index in seconds in the WF forward moving windows, n its length size, Ei the cumulated energy of each harvester and Res is the resolution in seconds. Thus, if we define the battery State Of Charge in Joules, we can express the battery charging duration as:

$$SOC + \sum_{i=1}^{n} E_i . \mathrm{Re}\, s \le SOC_{Full} \tag{2}$$

The DPM has several states, each having its own level of power consumption. A sequence is a set of states realized to perform a typical task, but the calculation can apply to multiple sequences. If we know the duration of each state, we can calculate the Required Energy value for a sequence.

We can express the total Required Energy (RE) for a sequence as:

$$RE = \sum_{j=1}^{nbT} Pt_j . Tt_j \tag{3}$$

Thus, if we define the battery State Of Charge in Joules, we express the maximal sequence duration as:

$$SOC - \sum_{j=1}^{nbT} Pt_j . Tt_j \ge SOC_{LowLevel} \tag{4}$$

We have to validate that the conditional expression (5) is always true for all values:

$$SOC + \sum_{i=1}^{n} E_i \cdot \mathrm{Re}\, s - \sum_{j=1}^{nbT} Pt_j . Tt_j \ge SOC_{LowLevel} \tag{5}$$

The Energy function result can be one of the two cases: if true the sequence has potentially enough energy to be executed, otherwise the sequence will probably fail due to a lack of energy.

As batteries have special behaviors [14], modeling battery charging with only a sum/minus operator is a simplification. In fact, we have built a battery model that supports both discharging and charging in order to handle these behaviors. Such a model has been developed based on previous work done by [15]. The model enables us to estimate efficiently the battery capacity to handle service. From this point, we can explore multiple solutions to adapt the quality of service.

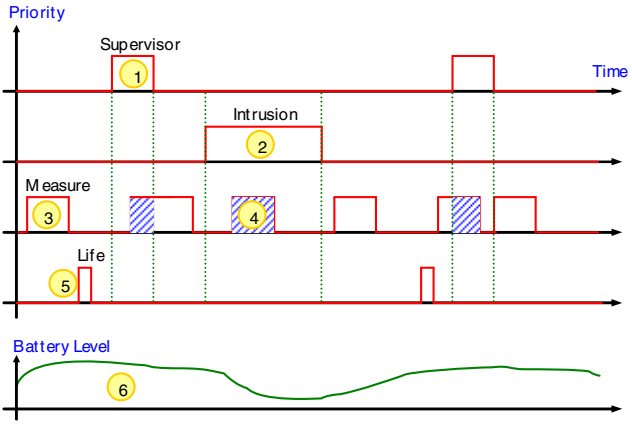



**Fig. 2.** Service scheduling ordered by priority levels. (1) The supervisor request has the highest priority; it masks all other low priority services. (2) When an intrusion has occurred, it runs the DPM shown in figure 1. (3) Task Measures acquire gas and temperature data. (4) Measure task can be masked by a higher priority task. (5) Life indicator is sent periodically. (6) The battery SOC can be used to redefine task priority.

By using a WF-DPM, we want to explore QoS techniques to reduce power consumption dynamically. We reference various techniques: service choice, service variable quality rating, service priority, service delay, and service delegation.

- Service Choice: at runtime, the DPM dynamically chosen which service is active or not. This can be used to select conditionally a service between two or more. A service choice example is the three branches to select video quality in the figure 1.
- Service Variable Quality Rating: this technique defines a variable criterion on a device or on a feature. For multimedia streams one can adjust the bandwidth and the calculation to decrease power consumption. In figure 1, the video mode adapts the bandwidth depending on the battery level.
- Service Priority: in such systems, each active service has a priority which defines the task ordering during power modes. Higher priority levels mask lower services. It can be combined with the service delay technique.

Figure 2 gives an example of service priority handling defined by the energy level.

- Service Delay: when not enough energy is available for a service, it is delayed in time until having the requested energy. In figure 2, a life packet is sent periodically, but it cannot be emitted while in low battery state.
- Service Delegation: when the service does not have enough energy, it can be delegated to another node which can accept or refuse to handle the job. This can typically occur when a device jams.

These techniques can be implemented in the DPM or a DVFS scheduler. Nevertheless, we underline the need to utilize these QoS techniques at a more global view, not only at CPU level. We have not quantified here the potential gain for each technique, but the result seems to be strongly application dependant.

## 5 Experimental Results

We have built a Power Estimator (PE) to simulate the whole node power/energy consumption. The PE is built with Labview from National Instruments. The complete PE description and the simulation process are detailed in [16]. Each component model is characterized from physical measures by applying the FLPA methodology [17].

For this experiment, we measure the relevance of a prototype deployment, by estimating its autonomy in various towns. We use a nominal scenario from the CAPNET project. The scenario is called "Fire Services - Container Boilover". It integrates toxic gas spread detection and human intrusion in the secure perimeter.

The scenario hypotheses for a 7 day duration are:

- Intrusion (both) Sensors: 20 activation each day, duration: 120 seconds, 140 total.
- Gas Sensor OLTC 50 activity: every 15 minutes, 15 second duration, 2016 total.
- Gas Sensor OLTC 80 activity: every 15 minutes, 15 second duration, 2016 total.
- WFs enhancements disabled (static weather – no predictive DPM).
- Weather data in winter from February 09 2011 10:00 GMT to February 17 2011 01:00 GMT for nine cities of France: {Brest, Lorient, Rennes, Dunkerque, Lille, Metz, Strasbourg, Aurillac, and Nice}.

The PE is setup with components listed in Table 1, and the local weather. Then, the simulation is run. Table 1 shows the energy and power consumption results independently from recharge. Results are grouped by components. The average total power consumption is 716 mW, battery duration is near 1.18 days for the 2Ah battery.

**Table 1.** CAPNET Node Power Consumption

| Components | Type | Min (mW) | Max (mW) | Avg. (mW) | % |
|---|---|---|---|---|---|
| Concertina | Sensor 1 | 355.66 | 393.38 | 368.84 | 51.54 |
| Miwi 1 | Radio 1 | 115.5 | 181.5 | 115.92 | 16.2 |
| OLTC 50 | Gas sensor 1 | 0 | 1074.6 | 72.458 | 10.13 |
| OLTC 80 | Gas sensor 2 | 0 | 884.51 | 58.216 | 8.135 |
| MT48T35AV | RAM | 0 | 150 | 26.326 | 3.679 |
| LM3100 | DCDC | 25.84 | 55.03 | 25.928 | 3.623 |
| Mygale | Sensor 2 | 0 | 583.49 | 19.268 | 2.693 |
| MAX618 | DCDC | 0 | 93.446 | 10.822 | 1.512 |
| PIC24F | CPU | 0 | 52.8 | 9.2664 | 1.295 |
| LM3100 | DCDC | 0 | 84.023 | 6.0276 | 0.8423 |
| LM3100 | DCDC | 0 | 12.627 | 2.2161 | 0.3097 |
| µCAM | Video | 0 | 305 | 0.21 | 0.029 |
| Miwi 2 | Radio 2 | 0 | 181.5 | 0.0834 | 0.0116 |
| Total Sensors | | | | 519 | 72.58 |
| Total Radios | | | | 116 | 16.23 |
| Total DCDC | | | | 45 | 6.3 |
| Total RAM | | | | 26 | 3.63 |
| Total CPU | | | | 9 | 1.26 |

By studying Table 1, we can find the blocking elements. In this WSN application, the first surprise comes from the PIC which should not impact the consumption with **1.26%**. We have also noticed that the RAM should have a low **3.63%** consumption due to their

idle mode. The DC/DC converters have a high efficiency ratio so their impact depends more on the connected devices. They are proportionally constant with **6.3%**. The Radio1 is not negligible here: it affects **16.23%** of the consumption due to permanent listening. The radio analysis part is left to a partner laboratory, thus we do not go into detail here. The highest consuming element (i) is the professional Sensor 1 with **369mW**, besides all sensors cumulated reach **72.58%** of the total consumption. This is explained by analysis of the DPM whose Sensor1 runtime is permanently on. This limitation cannot be addressed easily because the monitoring requires constant listening and depends more on manufacturer technology. Currently, some manufacturers do not provide low power equipment; however it would be of interest to encourage most of them to follow this direction.

The next simulation runs are executed sequentially for each city. In Table 2, we report the battery autonomy results in days.

**Table 2.** Deployment in various localizations

| Localization | No Recharge Reference | Autonomy with Solar energy | Autonomy with Wind energy | Autonomy Solar + wind |
|---|---|---|---|---|
| Brest | 1.18 | 7 | 7 | 7 |
| Lorient | 1.18 | 7 | 7 | 7 |
| Rennes | 1.18 | 6.71 | 7 | 7 |
| Dunkerque | 1.18 | 5.78 | 7 | 7 |
| Lille | 1.18 | 5.5 | 6.8 | 7 |
| Metz | 1.18 | 4.74 | 4.05 | 7 |
| Strasbourg | 1.18 | 7 | 1.18 | 7 |
| Aurillac | 1.18 | 6.65 | 1.18 | 6.65 |
| Nice | 1.18 | 7 | 1.24 | 7 |

By examining the results, we remarked that with the nominal 2Ah battery, the node autonomy is 1.18 days without any recharge (ii). With the Solar-10W harvester, we expect sunset exposure which nearly depends on latitudes like those given by PVGIS. However here, Strasbourg is the exception which proves the rule. Finally, the solar harvesting approach even without too much irradiance greatly extends the autonomy beyond the worst reference case (iii).

The results of the Rutland 504 wind harvester particularly correlate with the average dominant wind speed expected in these regions. Detailed battery curves clearly show that recharge is very fast. This is particularly true because of the Rutland is over-scaled for the needs (iii) and really suffers from the minimal speed start point.

When cumulating both solar and wind harvesters, all the cities, except Aurillac, recharge for the required 7 days of simulation. Thus, finally the PE certifies the battery autonomy of 2Ah will be sufficient for the deployment location (iv).

Finally for the CAPNET project, the PE has helped to identify power limitations:

(i) the highest power consumption components.
(ii) the worst-case autonomy duration.
(iii) the harvesters scale for the energy demand.
(iv) the battery autonomy for a given scenario and location.

Thus, the PE has provided clues to validate the battery scaling depending on weather condition and location.

## 6 Conclusion

In this paper, we have underlined the lack of a design tool which provides a global power consumption view during the design stage. It is essential for validating WSN designs to study the weather availability impact in the deployed location. To do that, we have proposed a simulator able to integrate Weather Forecasts DPM in order to provide better energy management.

Experimental results show that the Power Estimator has been able to identify the most power consuming hot-spots, and it helps to calibrate the WSN to fit its requirements in terms of system scaling and autonomy. This contribution improves WSN design and can be reuse for other WSN. We conclude in the interest to focus at design dynamically over the blocking element. For example MCU power consumption still not required to be more optimized compared to some industrial sensors.

As future work, we will want to experiment more deeply the WF-DPM to evaluate the predictive WF energy gain.

**Acknowledgments.** We would like to thank the CAPNET project partners. Fire Service partners: SDIS29 (Service Départemental d'Incendie et de Secours du Finistère). Industrial partners: ERYMA Security Systems (Leader), DeltaDore, AtlanticRF. Academics partners: I.E.T.R Rennes, Lab-STICC Lorient. Financial pole: "Pôle Image et Réseaux", "Conseil Général du Morbihan", "Région Bretagne", and "Ministère Français de l'Economie de l'Industrie et de l'Emploi". Proofreader: Joanna Ropers.

## References

1. Chandrakasan, A.P., Brodersen, R.W.: Low Power Digital CMOS Design. Kluwer Academic Publishers, Dordrecht (1995) ISBN, 0-7923-9576-X
2. Schmitz, M.T., Al-Hashimi, B.M., Eles, P.: System-Level Design Techniques for Energy-Efficient Embedded Systems. Kluwer Academic Publishers, Dordrecht (2004) ISBN, 1-4020-7750-5
3. Pedram, M., Rabaey, J.: Power-Aware Design Methodologies. Kluwer Academic Publishers, Dordrecht (2002) ISBN, 1-4020-7152-3
4. http://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes
5. Chee, Y., Koplow, M., Mark, M., Pletcher, N., Seeman, M., Burghardt, F., Steingart, D., Rabaey, J., Wright, P., Sanders, S.: PicoCube: A 1cm3 Sensor Node Powered by Harvested Energy. In: DAC (2008)
6. Raghunathan, V., Kansal, A., Hsu, J., Friedman, J., Srivastava, M.: Design Considerations for Solar Energy Harvesting Wireless. In: IPSN (2005)
7. Jiang, X., Polastre, J., Culler, D.: Perpetual Environmentally Powered Sensor Networks. In: IPSN (2005)
8. Simjee, F.I., Chou, P.H.: Everlast: Longlife Supercapacitor-operated wireless Sensor Node. In: ISLPED (2006)
9. Chen, M., Rincón-Mora, G.A.: Single Inductor, Multiple Input, Multiple Output (SIMIMO) Power Mixer-Charger-Supply System. In: ISLPED (2007)

10. Park, C., Chou, P.H.: AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Node. In: IEEE SECON (2006)
11. http://www.meteociel.fr/ (last visited May 25, 2011)
12. http://re.jrc.ec.europa.eu/pvgis/ (last visited May 25, 2011)
13. Lorentz, E.: The essence of chaos. The Jessie and John Danz Lecture Series. University of Washington Press (1993) ISBN 0-295 97514-8
14. Benini, L., Bruni, D., Macii, A., Macii, E., Poncino, M.: Discharge Current Steering for Battery Lifetime Optimization. IEEE Transactions on Computers 52(8) (2003)
15. Erdinc, O., Vural, B., Uzunoglu, M.: A dynamic lithium-ion battery model considering the effects of temperature and capacity fading. In: International Conference on Clean Electrical Power, p. 383 (2009)
16. Ferry, N., Ducloyer, S., Julien, N., Jutel, D.: Power/Energy Estimator for Designing WSN Nodes with Ambient Energy Harvesting Feature. EURASIP Journal on Embedded Systems 2011 (2011)
17. Laurent, J., Senn, E., Julien, N., Martin, E.: Functional Level Power Analysis: An Efficient Approach for Modeling the Power Consumption of Complex Processors. In: DATE (2004)

# Self-reference Scrubber for TMR Systems Based on Xilinx Virtex FPGAs

Ignacio Herrera-Alzu and Marisa López-Vallejo*

Department of Electronics Engineering
E.T.S.I. Telecomunicación
Universidad Politécnica de Madrid, Madrid, Spain
{iherrera,marisa}@die.upm.es

**Abstract.** SRAM-based FPGAs are sensitive to radiation effects. *Soft errors* can appear and accumulate, potentially defeating mitigation strategies deployed at the Application Layer. Therefore, Configuration Memory scrubbing is required to improve radiation tolerance of such FPGAs in space applications. Virtex FPGAs allow runtime scrubbing by means of dynamic partial reconfiguration. Even with scrubbing, intra-FPGA TMR systems are subjected to common-mode errors affecting more than one design domain. This is solved in inter-FPGA TMR systems at the expense of a higher cost, power and mass. In this context, a self-reference *scrubber* for device-level TMR system based on Xilinx Virtex FPGAs is presented. This *scrubber* allows for a fast SEU/MBU detection and correction by peer frame comparison without needing to access a *golden* configuration memory.

**Keywords:** SRAM-based, FPGA, EDAC, SEU, MBU, SEFI, TMR, Scrubber, Scrubbing, Readback, Reconfiguration, Configuration Memory, Reliability.

## 1 Introduction

When used in space applications, electronic devices are exposed to an ionizing radiation that is orders of magnitude higher than the one received at sea level. This affects the behaviour of semiconductors, inducing both a long term cumulative degradation as well as instantaneous damage or Single Event Effects (SEE). SEE can be reversible (*soft errors*) or destructive (*hard errors*).

Xilinx Virtex FPGAs are also sensitive to these effects. Tolerance to both long term degradation and destructive effects has been improved in Virtex-4QV and Virtex-5QV series by hardening the technology. For example in 4QV series, Total Ionization Dose (TID) of 300 krad(Si) and Single Event Latchup (SEL) immunity has been reported [1]. However, their SRAM-based Configuration Layer is still susceptible to several types of *soft errors*, namely Single Event Upsets (SEU),

* This work has been funded by CICYT Project TEC2009-08589.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 133–142, 2011.

Multiple Bit Upsets (MBU) and Single Event Functional Interrupts (SEFI). 4QV series characterization for SEU and SEFI has also been reported in [1], but an equivalent report for 5QV series has not been published yet.

When high-energy particles hit the Virtex FPGAs, SEUs can appear either in Application Layer (memory elements driven by users application) or in the underlying Configuration Layer (logic and routing resources set by configuration SRAM). When in Application Layer, SEU effects can be persistent or transient depending on whether the affected logic has memory or is memoryless. When in Configuration Layer, SEUs can affect logic or routing resources, and their effects are persistent until a reconfiguration restores their initial state. A particular case are SEFIs affecting the configuration logic, which cannot be repaired without re-initializing the FPGA.

A SEU in Configuration Layer may propagate as a functional failure in the Application Layer, independently of the user functionality implemented. Indeed a SEU in a routing resource may induce single or multiple errors in the Application Layer. This situation is dramatic considering that about 90% of the configuration bits are linked to routing resources [2], and that configuration memory density keeps on growing in every new FPGA generation, thus increasing the MBU cross-section [3]. Complementary SEU tolerance techniques have been incorporated to enable the use of SRAM-based FPGAs in harsh radiation environments: Error Detection And Correction (EDAC, such as TMR and majority voting) at Application Layer, and memory scrubbing at Configuration Layer. The reliability improvement of TMR systems when scrubbing is applied has been deeply analyzed in [4].

In this paper we propose a simple *scrubber* architecture for an inter-FPGA TMR system. As opposed to techniques based on embedded Error Correction Codes (ECC) [5], multiple errors per *frame* can be detected and corrected. The faulty *frame* is fully restored by taking advange of the redundant configuration data. And, as opposed to previously reported systems, it does this without nominally needing to access a *golden* configuration memory or a reference EDAC code memory [6]. The proposed *scrubber* also allows a fast SEFI detection by periodically monitoring the already known SEFI symptoms. Because radiation environment is variable, it allows dynamic scrub rate adjustment, thus reducing power consumption to the minimum necessary. Finally, the proposed *scrubber* concept is universal for Xilinx FPGAs from Virtex-4 on, only some control logic specificities need to be tailored for each FPGA series.

The paper is organized as follows. Section 2 discusses the scrubbing techniques for Xilinx Virtex FPGAs. Section 3 introduces the proposed self-reference *scrubber* for a TMR system. Section 4 presents the simulation results for the proposed *scrubber*. Finally, Section 5 presents the conclusions of this paper and outlines the future work.

## 2 Scrubbing Xilinx Virtex FPGAs

Xilinx Virtex FPGAs can be dynamically reconfigured after the initial power-on configuration practically an unlimited number of times. Dynamic global

reconfiguration affects the whole programmable fabric, and application execution is interrupted. Dynamic partial reconfiguration only affects part of the fabric while the rest operates without interruption. The dynamic partial reconfigurability is a unique feature of Xilinx Virtex FPGAs and can be exploited in different ways. In space applications it is particularly useful as it allows, on one hand, Configuration Memory scrubbing for SEU mitigation and, on the other, Application Layer in-flight reconfigurability [7].

### 2.1 Internal versus External Scrubbing

Configuration Memory scrubbing is driven by a Configuration Manager (informally known as *scrubber*), typically implemented in a radiation-hardened device external to the FPGA, either HW or SW driven. *Scrubbers* internal to the FPGA may also be implemented by using the Virtex primitives ICAP and FRAME_ECC [5],[8]. These primitives are theirselves implemented in the programmable fabric, so they are also susceptible to radiation. Although fault-tolerant implementations of internal *scrubbers* have been proposed [9], external scrubbers are typically more reliable [10]. In this work, only external *scrubbers* will be considered.

All Virtex FPGA configuration ports support external scrubbing. Among them, the 8-bit parallel SelectMAP (SMAP) interface is typically used as a good trade-off between speed and pinout penalty. For example, in Virtex-4QV series this interface can be clocked up to 80MHz. At this rate it is possible to fully configure the largest 4QV (XQR4VLX200) in about 70ms, excluding block RAM contents.

### 2.2 Configuration Memory Structure

For Virtex-4 and subsequent FPGA families, the minimum configurable element in the Configuration Memory is a *frame*, which is a set of 41 words of 32 bit each. The *frames* do not cover the whole programmable fabric height, but only the height of a memory *row*. Each FPGA has an specific number of *rows* that cover the whole fabric area. The *rows* are grouped in two halves, so called *top* and *bottom*. In addition, *frames* in each *row* are grouped into *columns* with variable sizes which depend on the *column type*. With such memory structure it is possible to define multiple bidimensional reconfigurable regions. The number of *frames*, *rows*, *columns* and *column types* are specific for each FPGA of the Virtex series. Fig. 1 depicts the generic Configuration Memory structure described above.

In addition to the programmable fabric, the Virtex Configuration Layer also includes configuration control logic and associated registers for monitoring and control. Among these 32-bit registers, the Frame Address Register (FAR) allows addressing any individual *frame*. The FAR is composed of several fields, with different organization depending on the Virtex FPGA generation. The FAR resets to the left-most *frame* of the *top-row 0*, and auto-increments right-up first. After the top right-most *frame*, the FAR goes back to the left-most *frame* of the *bottom-row 0* and auto-increments right-down.

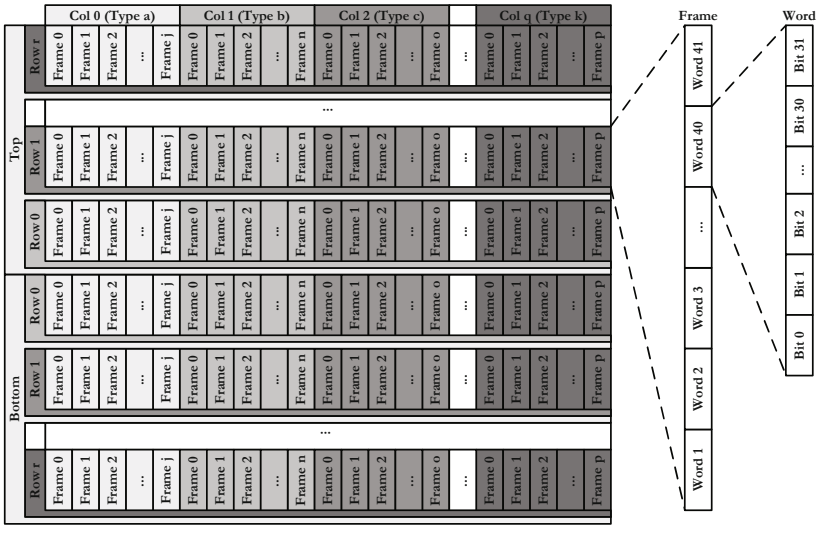



**Fig. 1.** Configuration Memory Sctructure for Xilinx FPGAs from Virtex-4 on

### 2.3 Readback and Scrubbing Configuration Memory

Two main strategies exist for SEU/MBU mitigation: open-loop and closed-loop scrubbing. Open-loop scrubbing (or *blind scrubbing*) consists on a preventive and cyclic SEU/MBU correction (i.e. without prior detection) through dynamic partial reconfiguration of the full Configuration Memory. The configuration data is retrieved from a *golden* data source, typically a radiation-hardened non-volatile memory, and written through the configuration port. As long as scrub rate is faster than the expected SEU rate, SEUs will not accumulate. An example of an open-loop *scrubber* for Virtex-4 is [11].

Closed-loop scrubbing instead is a two step process. First the configuration *frames* are cyclically read back for SEU/MBU detection. Only if an error is detected, the correction process is initiated. There are different possibilities for SEU/MBU detection and correction. The simplest one is the comparison of readback *frames* against their *golden* counterparts. In case they differ, the *golden frame* is used to scrub the faulty *frame* through the configuration port. Similarly to the open-loop scrubbing, this solution implies the permanent availability of the *golden* configuration memory, with the corresponding power consumption. Another possibility, which is available in Virtex-5 FPGAs, is to check the CRC computed during the readback process by means of an embedded circuitry. In case error is detected, the *frame* Error-Correcting Code (ECC) bits are used to locate the error. Once located, the faulty *frame* is scrubbed with a corrected content through the configuration port. However in this last case, the process is relatively complex and only single errors per *frame* can be corrected.

In Virtex FPGAs, Configuration Memory readback is done by reading *frame* 32-bit words on a byte basis (in case of using 8-bit wide SMAP) from the Frame

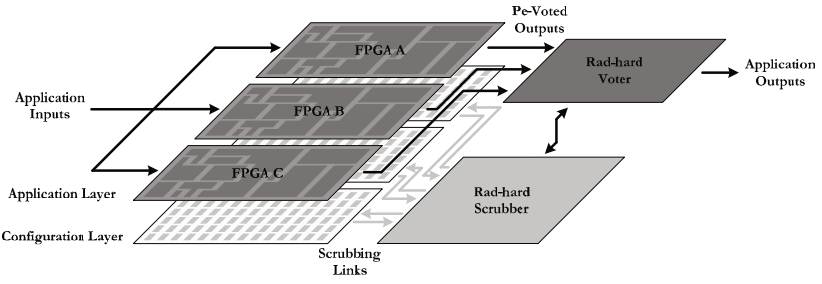



**Fig. 2.** Inter-FPGA TMR System

Data Register Output (FDRO). Likewise, scrubbing is done by writing *frame* 32-bit words on a byte basis to the Frame Data Register Input (FDRI). After a complete *frame* is read back or scrubbed, the FAR auto-increments as described in Section 2.2. This exempts the *scrubber* from continuously updating FAR before each access, thus reducing the processing overhead.

## 3 Self-Reference *Scrubber* for Inter-FPGA TMR Systems

### 3.1 Inter-FPGA TMR Systems

Inter-FPGA TMR, or device-level TMR, implies the use of independent FPGAs for each of the three design domains. A diagram of such system is depicted in Fig. 2. A three-way external *scrubber* is represented, interfacing the three SMAP ports at the Configuration Layer. A three-way external voter is also represented, voting the three application data streams at the Application Layer. These two elements are critical and must be implemented in radiation-hardened devices, either jointly or separated. Strategies for inter-operation between both elements will be covered in a future work, but are out of the scope of this paper.

This architecture is used for high reliability and availability systems where common-mode errors affecting more than one design domain must be avoided. In intra-FPGA TMR systems, and as highlighted in Section 1, this is more likely to happen as configuration memory densities increase [3]. In addition, inter-FPGA TMR is required for those applications that cannot afford a potential outage due to SEFI.

### 3.2 *Scrubber* General Description

In this work we propose a closed-loop *scrubber* for TMR systems, whose diagram is shown in Fig. 3. The *scrubber* is governed by the FSM, which in turn manages the Readback and Scrubbing Controllers. The SMAP interfaces handle the bidirectional links with the FPGAs. The *Frame* Buffers are 41 words each, i.e. 1,312 bits each. The SEFI Detection block is monitoring some specific FPGA pins and registers and asynchronously alerts the FSM in case SEFI is found. Finally, the *Golden* Configuration Memory is mainly used for power-on configuration and SEFI recovery.

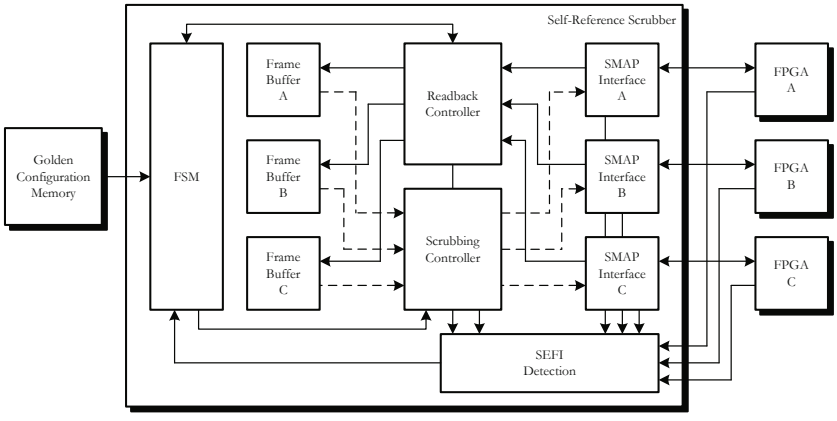



**Fig. 3.** Self-Reference *Scrubber* Diagram

**FSM and *Scrubber* Operation.** A simplified state diagram for the *scrubber* FSM is shown in Fig. 4. In a nominal operation, three identical configuration *frames* are read back from the FPGAs and stored in their corresponding *Frame* Buffers. At the same time, *frame* triplets are compared on a byte basis as they are read through SelectMAP interface in order to detect SEU/MBUs. At the end of each *frame* readback, a flag indicates if any mismatch was found in each particular domain. This implies that the faulty element at system level is the Frame In Error (FIE). If *frames* are found identical, readback resumes and new *frames* are read back. If one FIE flag is raised, the *dirty frame* is scrubbed with a correct *frame* from one of the other two *Frame* Buffers. After that, the FIE flag is cleared and the readback for the three FPGAs is resumed.

There is one scenario in which SEU/MBU will go undetected due to a common-mode error: when three particles hit each FPGA in exactly the same *frame*, the same bits within the *frame*, and within the same scrub cycle. If instead of three particles is two, the two *dirty frame* will be taken as good and the error will be propagated to the third one. However, the probability of such remote events is extremely low. The also remote possibility of having simultaneously three different SEU/MBU locations in the three *frames* under comparison can be recovered by scrubbing the three *frames* with data from the *golden* memory. Apart from this case and for power-on configuration and SEFI recovery, the *golden* memory can be almost permanently unpowered.

In case a persistent FIE is found (i.e. the same *frame* address is found *dirty* in consecutive readbacks), the *dirty* FPGA is passivated and the sytem turns to DMR mode. In this mode, FIE detection is performed by comparing bytes from two domains. Again, if the *frames* are identical the readback resumes and new *frames* are read back. If they are not, the two *frames* are considered *dirty* and are scrubbed with data from the *golden* memory at the expense of higher power and slower operation. After that, readback for the two FPGAs is resumed.

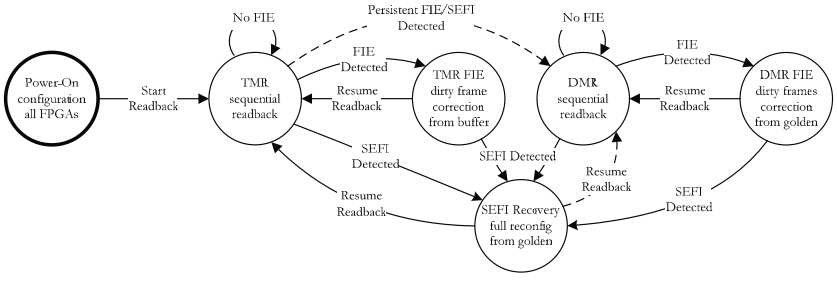



**Fig. 4.** Self-Reference *Scrubber* FSM States

As highlighted in Section 1, one of the main advantages with respect to other embedded SEU/MBU correction techniques proposed, is that multiple errors in a single *frame* can be detected and corrected. This is because the full *dirty frame* is scrubbed regardless of how many bits in error it may have.

**Readback and Scrubbing Controllers.** These blocks translate high level commands from the FSM into atomic commands for the SMAP Interfaces. In addition, the Readback Controller compares each byte triplet for the incoming readback *frames*. Scrub rate can be dynamically adjusted by FSM in order to adapt to the changing radiation conditions and to limit *scrubber* power consumption. Previous publications state that scrub rate should be at least ten times the expected SEU rate [13],[14]. In any case, no more than one FIE should be detected across all three FPGAs within a single scrub cycle. Note that this does not guarantee that TMR protection at Application Layer is not defeated, as SEUs in routing resources may induce multiple errors. If more than one FIE is detected within a scrub cycle, or FIEs are detected in consecutive scrub cycles at any given time, one should consider increasing the scrub rate. A shorter scrub cycle improves the Mean Time To Failure (MTTF) figure [4]. Therefore, the trade-off between reliability and power consumption must be managed.

**SEFI Detection and Recovery.** Concurrent SEFI detection is achieved by several means in accordance with [1] and [12]. Configuration pins are continuously monitored for Power-On-Reset (POR) and SMAP SEFIs. Configuration control registers are cyclically polled to detect SEFI symptoms, such as abnormal values or flags indicating that the control logic has reached an invalid state.

In case SEFI is detected while in TMR mode, the failing FPGA is fully reconfigured from *golden* memory while the other two are still operating. No power-cycle is needed, according to [12]. During this time, the readback controller seeks for mismatches between the two available *frames* (DMR mode). Once the failed FPGA is available again, its FAR is synchronized with the other two to resume readback in lockstep in TMR mode. In case of persistent FIE/SEFI on the same FPGA (i.e. present after consecutive reconfiguration cycles), the failing FPGA is passivated and the sytem turns to DMR mode. Finally, in case SEFI is detected

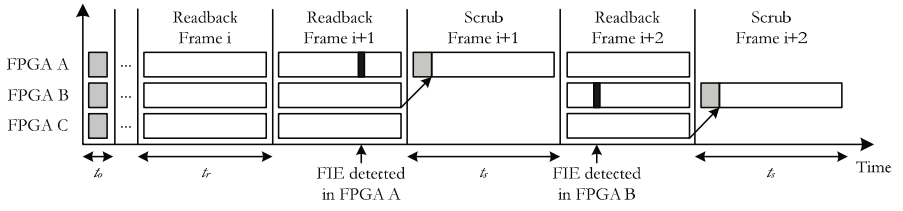



**Fig. 5.** Self-Reference *Scrubber* Simulation Timeline

while in DMR mode, the failing FPGA is in turn fully reconfigured. During this time, FIE detection is suspended as it works in purely simplex mode. Once the failed FPGA is available again, the system returns to DMR mode.

## 4 Simulation Results and Discussion

A VHDL description for the system shown in Fig. 3 was simulated for the purpose of validating the self-reference *scrubber* concept. For the FPGAs a cycle-accurate simulation model from previous work was used [15]. The model represents functionally the Configuration Layer of a XQR4VLX200 FPGA, implementing both the 39,120 configuration *frames* and the associated control logic. Such model allows for fault injection at random configuration bits of any of the three Configuration Memories and at random instants within the scrub cycle. Some of the known SEFI effects have also been modeled.

The simulation timeline for the basic readback-scrubbing operation in TMR mode is shown in Fig. 5. The *frame* readback time is $t_r$ and the *frame* scrub time is $t_s$. The overhead time required to poll some of the configuration registers to detect SEFIs and to initalize some other registers before each scrub cycle is $t_o$. An overhead time is also included in $t_s$, as a single *frame* scrubbing also requires initialization of some control registers. The simulated SelectMAP clock frequency is 20MHz as an initial trade-off between reliability and power.

The main *scrubber* simulated performance is summarized in Table 1. Besides Simulated Time, CPU Time for the full readback and configuration cycle simulations, running on an processor at 3.2GHz, is shown for reference.

CPU Time is more than reasonable taking into account that three Configuration Memories with more than 51Mbit each are simulated, and that SelectMAP interface is simulated with clock cycle accuracy. The table shows that $t_r$ and $t_s$ are in the same order of magnitude, as they are mainly driven by the SelectMAP clock cycle. The main difference is due to the overhead time included in $t_s$. If no FIE is found within a complete scan, the full readback cycle takes $t_{r,f} \approx t_o + (33{,}720 \times t_r)$, where 33,720 is the number of *frames* to be read back (after excluding the BRAM areas). If SEFI is found, the full reconfiguration cycle takes, for one or more FPGAs running concurrently, $t_{s,f} \approx 33{,}720 \times t_s$ provided that the interface with the *golden* memory is at least as fast as the SelectMAP

**Table 1.** Self-Reference *Scrubber* Simulation Results

| *Scrubber* Performance | $t_o$ | $t_r$ | $t_s$ | $t_{r,f}$ | $t_{s,f}$ |
|---|---|---|---|---|---|
| Simulated Time | 8.0 $\mu$s | 8.2 $\mu$s | 12.1 $\mu$s | 276.5 ms | 276.5 ms |
| CPU Time | - | - | - | 49 s | 52 s |

*Note*: SelectMAP Clock Cycle = 50ns

interface. The simulation results for $t_{r,f}$ and $t_{s,f}$ confirm the validity of the simulation model for the purpose of evaluating the *scrubber* concept, and provides a better insight of the *scrubber* performance.

## 5 Conclusions and Future Work

A self-reference *scrubber* for inter-FPGA TMR systems based on Xilinx Virtex FPGAs has been presented in this paper. This *scrubber* allows for a fast SEU/MBU detection by peer *frame* comparison, and correction by scrubbing the *dirty frame* with one of the other two. Scrubbing relies on dynamic partial reconfiguration. SEFI can be detected by polling internal registers or FPGA pins, and recovered by means of full reconfiguration. A *golden* configuration memory is not needed for SEU/MBU mitigation, but it is for SEFI recovery.

With the proposed *scrubber* the system redundancy (and reliability by extend) may vary over time. From initial TMR mode, persistent errors will lead the system to a DMR mode with reduced reliability. In DMR mode, *golden* memory is required for SEU/MBU mitigation. In addition, the scrub rate can also adapt to dynamic radiation environment. The *scrubber* concept has been validated with a simulation model for the FPGA Configuration Layer from a previous work, and some simulation results have been presented. The *scrubber* performance during readback and scrub cycles are in line with expectations.

As future work it is foreseen to prototype the *scrubber* in hardware. This will allow to characterize *scrubber* performance and power consumption in different redundancy and scrub rate configurations. Hardware-based fault injection via SelectMAP is planned for these experiments, and convergence with simulation-based results will also be analyzed. Future work is also planned to develop strategies for inter-operation of *scrubber* and voter modules, as well as for user data recovery after a full FPGA reconfiguration.

## References

1. Allen, G., Swift, G., Carmichael, C.: Virtex-4 VQ static SEU characterization summary. Jet Propulsion Laboratory, Pasadena, CA. National Aeronautics and Space Administration (2008)
2. Sonza Reorda, M., Sterpone, L., Violante, M.: Multiple errors produced by single upsets in FPGA configuration memory: a possible solution. In: European Test Symposium, pp. 136–141. IEEE Computer Society, Los Alamitos (2005)

3. Quinn, H., Morgan, K., Graham, P., Krone, J., Caffrey, M., Lundgreen, K.: Domain crossing errors: Limitations on single device triple-modular redundancy circuits in Xilinx FPGAs. IEEE Transactions on Nuclear Science 54(6), 2037–2043 (2007)
4. Ostler, P.S., Caffrey, M.P., Gibelyou, D.S., Graham, P.S., Morgan, K.S., Pratt, B.H., Quinn, H.M., Wirthlin, M.J.: SRAM FPGA reliability analysis for harsh radiation environments. IEEE Transactions on Nuclear Science 56(6), 3519–3526 (2009)
5. Xilinx: SEU Strategies for Virtex-5 Device, http://www.xilinx.com/support/documentation/application_notes/xapp864.pdf
6. Lanuzza, M., Zicari, P., Frustaci, F., Perri, S., Corsonello, P.: A self-hosting configuration management system to mitigate the impact of Radiation-Induced Multi-Bit Upsets in SRAM-based FPGAs. In: IEEE International Symposium on Industrial Electronics, pp. 1989–1994. IEEE, Los Alamitos (2010)
7. Osterloh, B., Michalik, H., Habinc, S.A., Fiethe, B.: Dynamic partial reconfiguration in space applications. In: NASA/ESA Conference on Adaptive Hardware and Systems, pp. 336–343. IEEE, Los Alamitos (2009)
8. Xilinx: Correcting Single-Event Upsets with a Self-Hosting Configuration Management Core, http://www.xilinx.com/support/documentation/application_notes/xapp989.pdf
9. Heiner, J., Collins, N., Wirthlin, M.: Fault tolerant ICAP controller for high-reliable internal scrubbing. In: IEEE Aerospace Conference, pp. 1–10. IEEE, Los Alamitos (2008)
10. Berg, M., Poivey, C., Petrick, D., Espinosa, D., Lesea, A., LaBel, K.A., Friendlich, M., Kim, H., Phan, A.: Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis. IEEE Transactions on Nuclear Science 55(4), 2259–2266 (2008)
11. Berg, M.: The NASA Goddard space flight center radiation effects and analysis group Virtex 4 scrubber. In: Xilinx Radiation Test Consortium (XRTC) Meeting (2007)
12. Xilinx: Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory, http://www.xilinx.com/support/documentation/application_notes/xapp1088.pdf
13. Quinn, H., Morgan, K., Graham, P., Krone, J., Caffrey, M.: A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5. In: 9th European Conference on Radiation and Its Effects on Components and Systems, pp. 1–8. IEEE, Los Alamitos (2007)
14. Adell, P., Allen, G.: Assessing and mitigating radiation effects in Xilinx FPGAs. Jet Propulsion Laboratory, Pasadena, CA. California Institute of Technology (2008)
15. Herrera-Alzu, I., López-Vallejo, M.: Cycle-Accurate Configuration Layer Model for Xilinx Virtex FPGAs. In: 13th European Conference on Radiation and Its Effects on Components and Systems. IEEE, Los Alamitos (2011)

# Cell-Based Leakage Power Reduction Priority (CBLPRP) Optimization Methodology for Designing SOC Applications Using MTCMOS Technique

Henry X.F. Huang, Steven R.S. Shen, and James B. Kuo*

Circuits, Devices and Systems Laboratory
School of Computer & Information Engineering
Peking University, Shenzhen Graduate School, Shenzhen 518055, China

**Abstract.** This paper describes a straightforward cell-based leakage power reduction priority (CBLPRP) optimization methodology for designing high-speed low-power SOC applications using MTCMOS technique. The CBLPRP methodology is based on the cell swapping priority depending on the total leakage power reduction for a cell changing from LVT type to HVT type. Experimental results show that by employing CBLPRP Methodology on the ISCAS benchmark circuits, a 10-20% reduction in the leakage power consumption could be achieved as compared to the one using the GDSPOM technology.

**Keywords:** CBLPRP, MTCMOS, GDSPOM, BFS, Dual-threshold CMOS, Power Optimization.

## 1 Introduction

For today's SOC applications with the growing use of portable electronic systems, reduction in power consumption has become more and more important. Especially for CMOS integrated circuits using technology in the nanometer regime, leakage power becomes a significant component of the total power consumption. For achieving low static power consumption, multi-threshold CMOS (MTCMOS) technology [1]-[8] has been proven to be an effective approach.

In the past, optimization strategies for reducing the static power consumption of an SOC system have been reported [4]-[6]. Until now, gate-level dual-threshold static power optimization methodology (GDSPOM) offers an efficient way of achieving the goal. As shown in Fig. 1 [4], in the GDSPOM procedure, a register transfer language (RTL) design is first synthesized into a gate-level netlist of cells using CMOS devices with a high threshold voltage (HVT). Then static timing analysis (STA) as described in Ref. [4] is performed to report the cells with the most timing violated paths-the so-called bottleneck cells as marked in circle in the figure. Then the bottleneck cells are

* J.B. Kuo was on sabbtical leave from NTUEE.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 143–151, 2011.

targeted for swapping the device type from HVT to low threshold voltage (LVT). After swapping the bottleneck cells to LVT type, STA is performed again to report new bottleneck cells. This STA procedure continues until all the timing paths meet the required timing constraints.

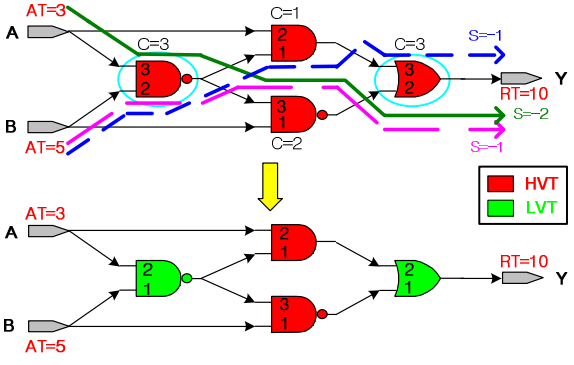



**Fig. 1.** A GDSPOM cell swapping example [4]

Although GDSPOM is an effective approach for reducing the static power consumption of an SOC system application, it is still too complicated. In this paper, a straightforward cell-based leakage power reduction priority (CBLPRP) optimization methodology is described for optimizing the static power consumption of an SOC system application using MTCMOS technique. Based on the CBLPRP optimization methodology with the MTCMOS technique, for the ISCAS benchmark circuits[9], it shows about a 10-20% reduction in leakage power consumption achieved when compared to the one using the GDSPOM [4] technology. In the following sections, the CBLPRP procedure is described first, followed by the performance of the ISCAS benchmark circuits using the CBLPRP methodology, discussion and conclusion.

## 2 CBLPRP Optimization Methodology

Fig. 2 shows the flow chart of the cell-based leakage power reduction priority (CBLPRP) optimization methodology used for designing high-speed low-power SOC applications using MTCMOS technology. As shown in the figure, an RTL design is synthesized into a gate-level netlist of cells using CMOS low threshold voltage devices with an LVT library. Then, by swapping each cell from LVT type to HVT, the cells are sorted in terms of the total leakage power consumption. Depending on the saving on the leakage power consumption, the cell with the most saving is swapped first for reaching the utmost power saving. Then, STA is performed to estimate whether the cell that is swapped from LVT type to HVT type meets the timing constraints. If it meets the constraint, then the cell is placed in the cell list; if not, then the original cell type is kept. These procedures continue until all cells are evaluated. Finally, a cell-swapping script is executed to create the netlist built with dual-threshold HVT/LVT cells.

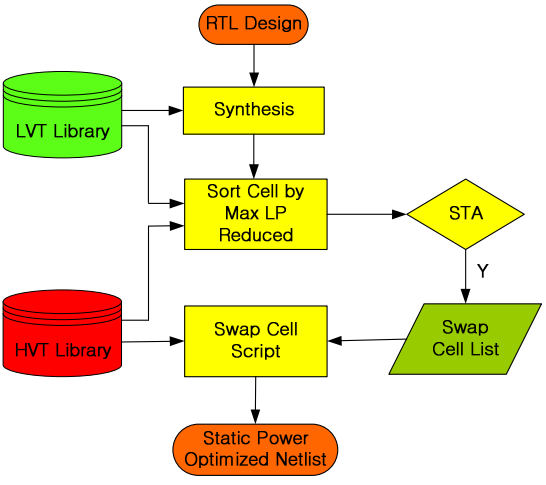



**Fig. 2.** Flow chart of the CBLPRP optimization methodology

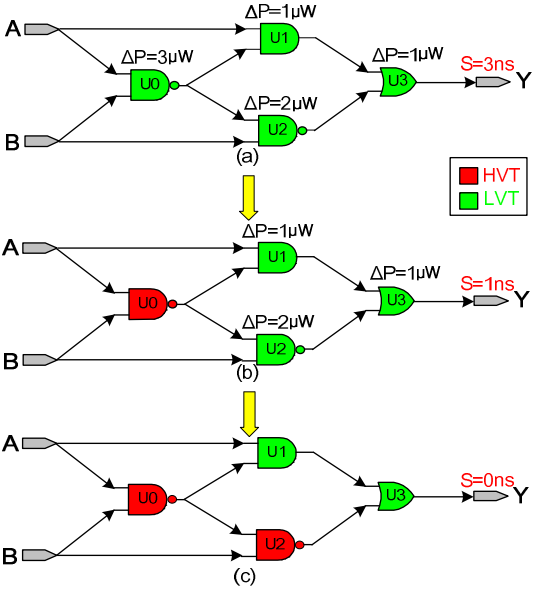



**Fig. 3.** An example showing the cell swapping for the CBLPRP optimization methodology

Fig. 3 illustrates an example showing the cell swapping of the CBLPRP optimization strategy. As shown in the figure, next to each cell, ΔP is defined as the amount of total leakage power saving when swapping the cell from LVT type to HVT. For instance, as shown in Fig. 3(a), with the slack value of 3ns, the cell U0 has the ΔP of 3μW, which is the largest in power consumption saving among all cells, hence it is targeted for cell type change. (Note that the slack value of the circuit[4] is defined as the difference between the arrival time and the required time. When the arrival time is shorter or equal to the

required time, it meets the timing constraint- the slack value is positive. On the other hand, when the arrival time is longer than the required time, it does not satisfy the timing constraint and thus it has set up a timing violation- the slack value is negative.) After changing the cell to HVT type for the cell U0, STA is performed to evaluate the timing constraint to see if it meets it. If yes, the new cell type is recorded in the cell list. If not, the original cell type is kept. As shown in Fig. 3(b), after the swapping of the type of the cell U0 to HVT, the slack value is reduced to 1ns. The CBLPRP optimization procedure will continue until all cells have been evaluated. Thus the optimized netlist is shown in Fig. 3(c) with the slack value of 0ns.

Fig. 4 shows the algorithms of (a) finding the cell with the maximum reduction in leakage power consumption and (b) cell swapping, used in the CBLPRP optimization methodology. The visit attribute, which defines whether this cell is evaluated, is set to 1 if the cell is evaluated; if not, stays 0. As shown in Fig. 4(b), from the selected cell as specified by the algorithm depicted in Fig. 4(a), if the HVT type cell can meet the required timing constraint, the cell type is changed. Otherwise, the original LVT cell type is maintained. This procedure continues until the visit attribute of each cell becomes 1.

```
procedure get_max_leakage_reduced_cell ($input_Netlist){
   @cell_array = all cells in $input_Netlist with visit=0
   $max_cell = NULL
   $max_leakage_reduced = 0
   foreach $cell (@cell_array) {
                $HVT_Leakage = the total leakage power when $cell in HVT
                $LVT_Leakage = the total leakage power when $cell in LVT
                $Leakage_reduced = $LVT_Leakage - $HVT_Leakage
                if ($Leakage_reduced > $max_leakage_reduced)    {
                    $max_leakage_reduced = $Leakage_reduced
                    $max_cell = $cell
       }
   }
   return $max_cell
}
```

(a)

```
procedure get_optimized_netlist($input_Netlist) {
   $Max_Leakage_Reduced_cell =
          &get_max_leakage_reduced_cell($input_Netlist)
   while ($Max_Leakage_Reduced_cell != NULL) {
       swap $Max_Leakage_Reduced_cell to HVT cell
       set visit attribute of the cell to 1
       Static Timing Analysis
       if(timing is not meeting){
            swap $Max_Leakage_Reduced_cell to LVT cell
       }
       $Max_Leakage_Reduced_cell =
              &get_max_leakage_reduced_cell($input_Netlist)
   }
}
```

(b)

**Fig. 4.** Algorithms of (a) finding the cell with the maximum reduction in leakage power consumption and (b) cell swapping, used in the CBLPRP optimization methodology

## 3 Performance Evaluation

The CBLPRP optimization methodology has been implemented in TCL using the Prime Time/ Prime Power programs from Synopsys [9][10]. In order to assess the performance of the CBLPRP optimization methodology for designing low-power high-speed SOC applications using a 65nm CMOS foundry technology with MTCMOS technique, a 16-bit multiplier with an array architecture has been tested. The 16-bit multiplier is generated from an RTL source code from the ISCAS benchmark circuit C6288 [11]. Under a clock cycle of 0.8ns, the CBLPRP optimization procedure reassigns 1155 cells out of 3199 in the multiplier circuit to swap from LVT to HVT. Fig. 5 shows the block diagram of the 16-bit dual-threshold multiplier design optimized by the CBLPRP optimization methodology. As shown in the figure, the HVT cells are in red and the LVT ones are in green.

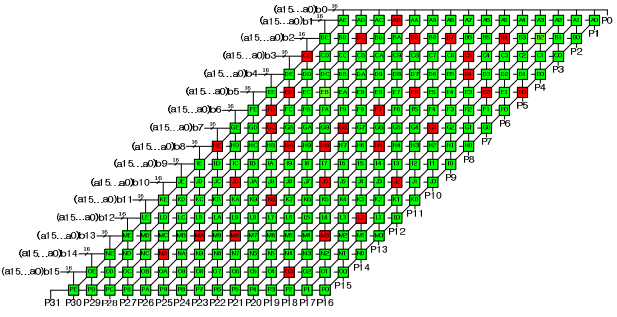


**Fig. 5.** Block diagram of the 16-bit multiplier design optimized by the CBLPRP optimization procedure

Fig. 6 shows the signal path from the input In290 (the 2nd multiplier bit) to the output Out6287(the 32nd product bit) using the CBLPRP approach. This signal path is randomly selected to show how the swapping of the cell types for lowering the leakage power consumption is accomplished. As shown in the figure, before the CBLPRP procedure, the cells in this path are all in LVT-type, having an arrival time of 0.74159ns, which meets the 0.8ns operating clock cycle constraint. After performing the CBLPRP procedure, 10 cells have been swapped from LVT to HVT with the data arrival time of 0.77ns, which also meets the operating clock cycle requirement.

The result of the 16-bit multiplier optimized by the CBLPRP procedure has been compared with that implemented by all HVT cells, LVT cells and by BFS[5]/ GDSPOM algorithms[4]. The one with all HVT cells, has the least leakage power consumption of 125.1μW, but not satisfying the timing requirement. All-LVT multiplier has the highest leakage power of 1293μW. The power consumption of the one adopting the CBLPRP optimization procedure is 837.5μW, which is 35% less than the all-LVT one at 0.8ns, meeting the timing requirement. Table 1 shows the static power consumption and the number of HVT cells of the 16-bit multiplier using (a) breadth first search (BFS) [5], (b) GDSPOM [4], and (c) CBLPRP procedures. Fig. 7 shows the static power consumption versus operating clock cycle of the 16-bit multiplier using (a) BFS, (b) GDSPOM, and (c) CBLPRP procedures. Also shown in the figure are the results for the one using all LVT cells. Among all, the multiplier using the CBLPRP dissipates the least leakage power in comparison with the GDSPOM and BFS ones. At the operating clock cycle of

0.88ns, the static power consumption of the CBLPRP one is 59.86% less than the all-LVT one, while GDSPOM is 47.56% less.

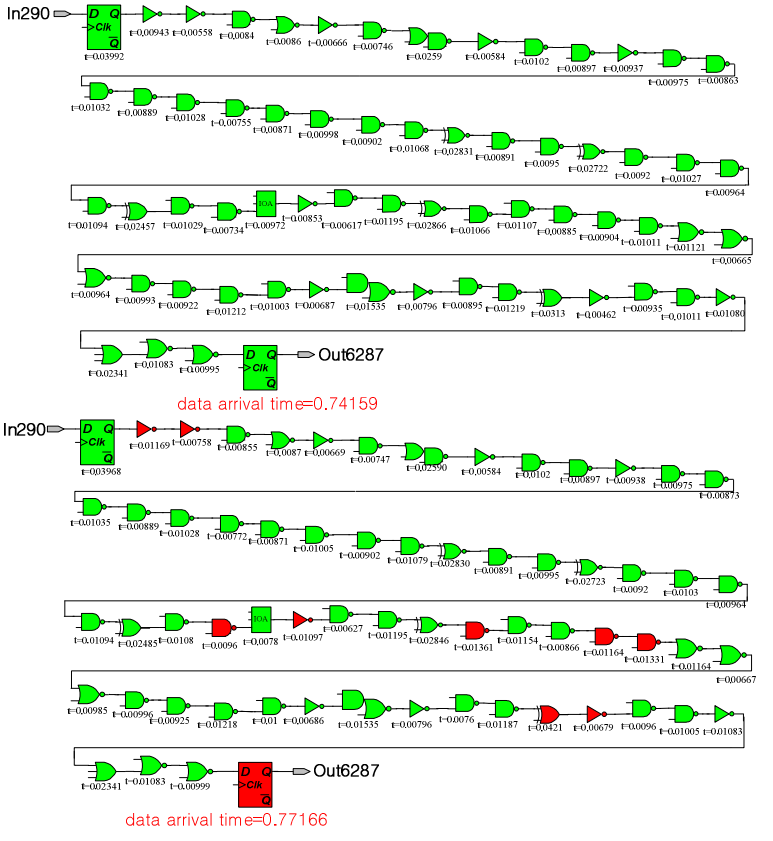



**Fig. 6.** Critical path from In290 to Out6287 in the 16-bit multiplier before and after the CBLPRP procedure. Red indicates the cells in HVT and green is for the cells in LVT.

**Table. 1.** Static power consumption and the number of HVT cells of the 16-bit multiplier using (a) BFS, (b) GDSPOM, and (c) CBLPRP procedures

| Period (ns) | BFS | | | GDSPOM | | | CBLPRP | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pstdby (μW) | Reduction (%) | High Vt Cells | Pstdby (μW) | Reduction (%) | High Vt Cells | Pstdby (μW) | Reduction (%) | High Vt Cells |
| 0.76 | 1248 | 3.48 | 89 | 1114 | 13.84 | 796 | 1016 | 21.42 | 876 |
| 0.8 | 1228 | 5.03 | 137 | 989.6 | 23.46 | 1124 | 837.5 | 35.23 | 1155 |
| 0.84 | 1213 | 6.19 | 170 | 845 | 34.65 | 1504 | 681.9 | 37.81 | 1421 |
| 0.88 | 1192 | 7.81 | 207 | 678 | 47.56 | 1914 | 519 | 59.86 | 1748 |
| 0.92 | 1161 | 10.21 | 273 | 534 | 58.7 | 2294 | 368.3 | 71.52 | 2144 |
| 0.96 | 1143 | 11.6 | 302 | 376.6 | 70.87 | 2644 | 243.7 | 81.15 | 2552 |
| 1 | 1119 | 13.46 | 342 | 214.5 | 83.41 | 3004 | 147.3 | 86.08 | 3039 |

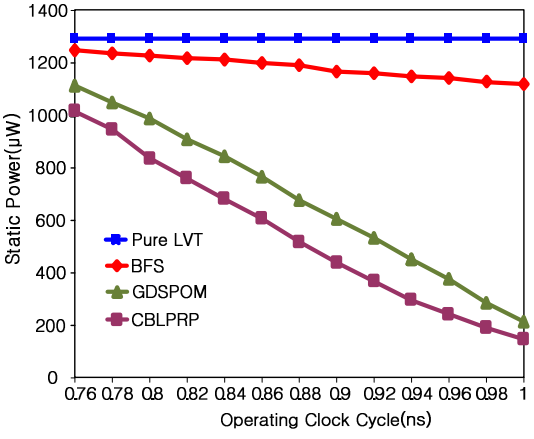



**Fig. 7.** Static power consumption of the 16-bit multiplier using (a) BFS, (b) GDSPOM and (c) CBLPRP procedures

## 4 Discussion

In addition to the multiplier circuit, the performance of the CBLPRP optimization methodology has also been evaluated using the other ISCAS benchmark circuits[11]. Table 2 and Fig. 8 show the static power consumption of the circuits using (a) BFS, (b) GDSPOM, and (c) CBLPRP procedures. As shown in the table and the figure, about 10% less static power consumption is for the CBLPRP one as compared to the GDSPOM one, while the leakage power of BFS one is highest. Therefore, the CBLPRP optimization methodology, which adopts a straightforward optimization algorithm, is effective for optimizing the static power consumption of a VLSI system design using MTCMOS technique.

**Table 2.** Static power consumption of the ISCAS benchmark circuits using (a) BFS, (b) GDSPOM, and (c) CBLPRP procedures

| Circuit Chosen | period (ns) | Gate # | Pstdby (μW) HVT | Pstdby (μW) LVT | Pstdby (μW) MVT | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | BFS | Reduction (%) | GDSPOM | Reduction (%) | CBLPRP | Reduction (%) |
| C432 | 0.6 | 113 | 6.05 | 51.27 | 21.79 | 57.5 | 14.96 | 70.82 | 7.94 | 84.51 |
| C499 | 0.45 | 197 | 12.48 | 127.2 | 62.82 | 50.61 | 78.95 | 37.93 | 34.67 | 72.74 |
| C1908 | 0.6 | 219 | 11.32 | 105.1 | 56.64 | 46.11 | 39.66 | 62.26 | 20.21 | 80.77 |
| C2670 | 0.45 | 562 | 41.53 | 377.8 | 310.6 | 17.79 | 51.35 | 86.41 | 46.87 | 87.6 |
| C3540 | 0.65 | 481 | 17.57 | 150 | 91.9 | 38.73 | 41.73 | 72.18 | 28.19 | 81.21 |
| C5315 | 0.4 | 800 | 44.6 | 409.1 | 340.1 | 16.87 | 141.6 | 65.39 | 76.34 | 81.34 |
| C6288 | 0.85 | 3199 | 125.1 | 1293 | 1203 | 6.96 | 808.7 | 37.46 | 638.3 | 50.63 |
| C7552 | 0.28 | 1479 | 82.13 | 838.6 | 655.2 | 21.87 | 371.9 | 55.65 | 265.1 | 68.39 |

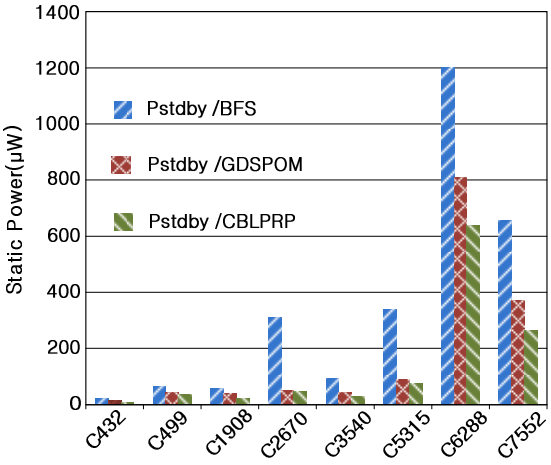



**Fig. 8.** Static power consumption of the ISCAS benchmark circuits using (a) BFS, (b) GDSPOM, and (c) CBLPRP procedures

## 6 Conclusion

In this paper, a straightforward cell-based leakage power reduction priority (CBLPRP) optimization methodology for designing high-speed low-power SOC applications using MTCMOS technique is described. The CBLPRP methodology is based on the cell swapping priority depending on the total leakage power reduction for a cell changing from LVT type to HVT type. Experimental results show that by employing CBLPRP methodology on the ISCAS benchmark circuits, a 10-20% reduction in the leakage power consumption could be achieved as compared to the one using the GDSPOM technology.

## References

1. Kuo, J.B.: Low-Voltage SOI CMOS Devices and Circuits. Wiley, New York (2004)
2. Usami, K., Kawabe, N., Koizuki, M., Seta, K., Furusawa, T.: Automated Selective Multi-Threshold Design for Ultra-Low Standby Applications. In: Low Power Electronics and Design Conf. Proc., pp. 202–206 (2002)
3. Kao, J., Narendra, S., Chandrakasan, A.: MTCNMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Pattern. In: Design Automation Conf. Proc., pp. 495–500 (1998)
4. Chung, B., Kuo, J.B.: Gate-Level Dual-Threshold Static Power Optimization Methodology (GDSPOM) Using Path-Based Static Timing Analysis (STA) Technique for SOC Application. The VLSI J. Integration, 9–16 (2008)
5. Wei, L., Chen, Z., Johnson, M., Roy, V., De, V.: Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits. In: Design Automation Conf. Proc., pp. 489–494 (1998)

6. Wei, L., Chen, Z., Roy, K., Johnson, M., Ye, Y., De, V.: Design and Optimization of Dual-Threshold Circuits for Low-Voltage Low-Power Applications. IEEE Trans. VLSI Systems, 16–24 (1999)
7. Samanta, D., Pal, A.: Optimal Dual-VT Assignment for Low-voltage Energy-Constrained CMOS Circuits. In: International Conf. on VLSI Design, pp. 193–198 (2002)
8. Wang, Q., Vrudhula, S.: Algorithms for Minimizing Standby Power in Deep Submicrometer, Dual-Vt CMOS Circuits. IEEE Trans. CAD of IC and Systems, 306–318 (2002)
9. Synopsys, Prime Time User Guide, v2010.06
10. Synopsys, Prime Power User Guide, v2006.06
11. ISCAS High-Level Models, http://www.eecs.umich.edu/~jhayes/iscas/

# NBTI Mitigation by Giving Random Scan-in Vectors during Standby Mode

Toshihiro Kameda[1,3,⋆], Hiroaki Konoura[1,3], Yukio Mitsuyama[2,3], Masanori Hashimoto[1,3], and Takao Onoye[1,3]

[1] Dept. Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University
{kameda.toshihiro,hashimoto,onoye}@ist.osaka-u.ac.jp
[2] School of Systems Engineering, Kochi University of Technology
mitsuyama.yukio@kochi-tech.ac.jp
[3] JST CREST

**Abstract.** Negative Bias Temperature Instability (NBTI) is one of the serious concerns for circuit performance degradation. NBTI degrades PMOS transistors under negative bias, whereas without negative bias they recovers. In this paper, we propose a mitigation method for NBTI-induced performance degradation that exploits the recovery property by shifting random input sequence through scan paths. With this method, we prevent consecutive stress that causes large degradation. Experimental results reveal that random scan-in vectors successfully mitigate NBTI and the path delay degradation is reduced by 71% in a test case when standby mode occupies 10% of total time.

**Keywords:** NBTI, NBTI mitigation, Performance degradation, Scan path, Aging, Reliability.

## 1 Introduction

In nanoscale integrated circuits design, performance degradation due to aging effects is becoming a major concern. Conventionally, to cope with the performance degradation due to aging, a design margin is given so that even an aged circuit will satisfy the performance requirement. However, as the necessary margin increases with technology scaling, the performance loss due to the margin is becoming less acceptable. Therefore, mitigating aging effects is highly demanded.

Among various aging effects, negative bias temperature instability (NBTI) is nowadays one of the most influential effects. NBTI causes a gradual increase in $|\Delta \mathrm{Vth}|$ while a negative bias is applied to PMOS, i.e. PMOS is ON. Here, this condition is defined as stress phase of NBTI. The $|\Delta \mathrm{Vth}|$ increase reduces drain current and consequently lengthens path delay, which leads to timing failures. On the other hand, while PMOS is OFF, $|\Delta \mathrm{Vth}|$ gradually decreases to its initial value before the stress injection. This condition is defined as recovery phase of

⋆ This work is in part supported by NEDO.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 152–161, 2011.

NBTI. The PMOS degradation by NBTI gradually progresses through circuit operations that repeat the stress and recovery phases. Besides, reference [1, 2] pointed out that the ratio of stress and recovery phases, which is hereafter called stress probability, is a key parameter that determines the amount of degradation in a long-term perspective. As the stress probability gets close to 1, the degradation increases and approaches to the worst case that DC stress is given. This NBTI property of recovery can be exploited for NBTI mitigation at circuit level by controlling the stress probability.

A possible way to control the stress probability is input vector control (IVC) [4–6]. IVC is well known as a leakage reduction technique that exploits the dependency of leakage current on the input vector [3]. For NBTI mitigation, a pre-determined input vector is given to a combinational circuit to minimize performance degradation during a standby mode. In [4], Y. Wang et. al. evaluated IVC technique for NBTI mitigation taking into account the temperature that affects NBTI-induced performance degradation. The same authors also derived Pareto sets that simultaneously optimize NBTI mitigation and leakage currents in [5]. D. R. Bild et. al. introduced Internal Node Control (INC) to IVC in order to improve the controllability of internal nodes [6]. INC adds some devices to set the internal node to 0 or 1, which involves increase both in area and path delay.

However, only using a single input vector, a considerable number of PMOSs are in stress phase and their performance degrades in the standby mode, because CMOS gates are basically inverting logic and all the logic values cannot be 1 simultaneously. This problem becomes significant as the portion of standby mode increases, because the degradation during the standby mode becomes comparable, or rather larger than that during the active mode. To overcome this problem, S. Jin et. al. presented a multiple input vector approach [7]. They derived a set of input vectors and allocated standby time to each of them to maximize the NBIT mitigation effect [7]. On the other hand, the number of input vectors is thought to increase as the target circuit becomes larger, which means larger memory is necessary to store input vectors. A possible way to give multiple input vectors is to use scan paths.

We then propose a random input vector approach that shifts random values through scan paths. This approach gives different input vectors for each scan-shift operation, which means the proposed random input vector approach can be regarded as one of multiple input vector approaches, though it does not prepare input vectors beforehand. First, we experimentally show the difference in performance degradation with different scan-in vectors, and reveal that random scan-in vectors attain a good mitigation effect. We then evaluate how the quality of randomness affects the mitigation effect with various linear feedback shift register (LFSR) configurations.

It should be noted that another circuit level approach is power gating for NBTI mitigation [8]. As for circuits that adopt power gating for leakage current reduction during standby mode, NBTI is mitigated as a byproduct. On the other hand, it is difficult to apply power gating to high-performance design whose permissible speed degradation is highly limited, because the sleep transistor

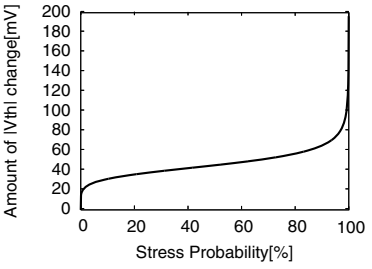



**Fig. 1.** Threshold voltage degradation vs. stress probability(10 years later)

degrades the performance and increase uncertainty in performance. In addition, aging of sleep transistor could be a serious problem in some cases. The proposed NBTI mitimigation is applicable to even such high-performance designs.

## 2 NBTI Model

To characterize NBTI, a number of measurement results and closed-form models have been proposed [1, 11, 12]. Among them, long-term degradation and its dependency on stress probability are well modeled in [1]. We therefore assume the model below in this work, though other models can be used as long as the dependency on stress probability is well characterized,

$$|\Delta V_{th,t}| = \left( \frac{\sqrt{K_v^2 \cdot T_{clk} \cdot \alpha}}{1 - \beta_t^{1/2n}} \right)^{2n}, \tag{1}$$

where $K_v$ is a parameter dependent on supply voltage and temperature. $T_{clk}$ is a clock period, and $\alpha$ is the stress probability of PMOS. $\beta_t$ is a parameter that has a dependence on temperature, $T_{clk}$, $\alpha$, and $t$. Moreover, $n$ is equal to 1/6 in a hydrogen molecule diffusion based model [13].

Here, in Eq. (1), when $\alpha$ approaches to 1, $|\Delta V_{th,t}|$ reaches an infinite value and is not appropriate. In such a situation, as its upper limit, we use Eq. (2) which models only stress phase of NBTI[10].

$$\Delta V_{th,t} = \left(K_v^2 t\right)^n \tag{2}$$

Figure 1 exemplifies the amount of threshold voltage degradation after a lapse of ten years in the condition that supply voltage is 1.1V and temperature is 125°C. The initial threshold voltage is -180mV. Here, other model parameters derived for transistors in a 65nm process [1] were used . The degradation is quite large when the stress probability is near 100%. This result suggests that for PMOSs with near 100% stress probability, even a small reduction in the stress probability is helpful to mitigate $|\Delta V_{th}|$ increase.

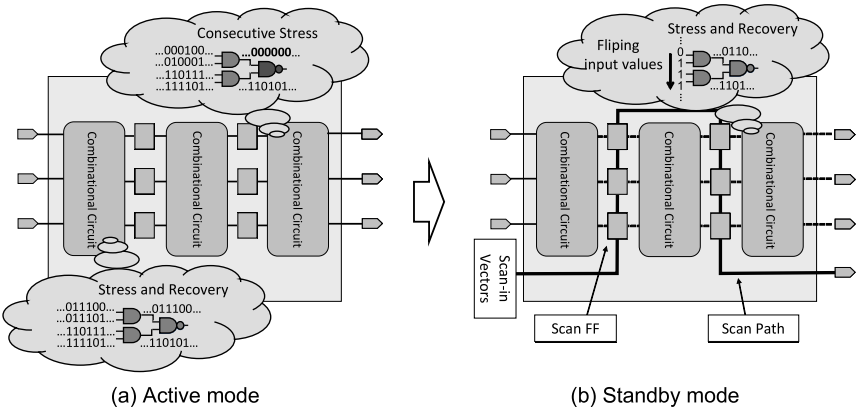



**Fig. 2.** Conceptual diagram of our mitigation technique using scan path

## 3 NBTI Mitigation by Giving Scan-in Vectors

To mitigate NBTI, we give a set of input vectors to combinational circuits through scan paths and switch PMOSs in the circuit aiming to make PMOSs in consecutive stress experience recovery phase. By using a set of input vectors, the multiple input vector approach settles the problem of single IVC that some of PMOSs are fixed in stress phase during the standby mode. Figure 2 illustrates the conceptual diagram of the NBTI mitigation. To give a set of input vectors to combinational circuits, we use a scan path which is often embedded for design for testability (DFT). In active mode, there are PMOSs that are hardly in recovery phase (a PMOS in the upper NAND gate in Fig. 2(a)). The multiple input vector approach gives a chance to such PMOSs to be in recovery phase during standby mode (Fig. 2(b)).

To apply the multiple input vector approach to a circuit, we need to determine what set of input vectors should be used as scan-in vectors. A possible way is to gather the input vectors derived by IVC techniques, which is proposed in [7]. On the other hand, because an additional memory that stores input vectors is necessary, the number of input vectors should be small, while targeted PMOSs should have a chance to be in recover phase. In addition, during scan-in and scan-out operations, the input vectors given to combinational circuits are different from those derived by the technique of [7]. This means that even if a good set of input vectors are available, they cannot be efficiently enforced to the combinational circuit in time when the scan paths are long and/or the standby time is short, because it takes a long time to change the input pattern.

Another approach is to give random scan-in vectors supposing that almost all PMOSs have a chance to be in recovery phase, and this is the approach proposed in this paper. In this approach, every set of input values given to a combinational circuit, which is generated each time the scan path is one-bit shifted with a

random scan-in value, is regarded as an input vector in multiple input vector approach. An advantage of this approach is that an additional memory to store the input patterns is unnecessary and the number of input vectors can be easily increased by lengthening the cycle of random numbers. As weak points, this cannot guarantee that targeted PMOSs have a chance of recovery phase and additional mechanism to generate random patterns is necessary.

In the following, we focus on the approach of random scan-in vectors, and experimentally evaluate how much mitigation of performance degradation can be obtained through a case study.

## 4 Evaluation of NBTI Mitigation

This section shows a case study of NBTI mitigation.

### 4.1 Experimental Setup

We selected 32bit-ALU, which operates addition, subtraction and logical operations, and 32bit-MDU (Multiply Divide Unit), which operates multiplication and division, in MIPS R3000 processor as target circuits for evaluation. These circuits were synthesized with Synopsys DFT Compiler using an industrial 65nm standard cell library. 32bit-ALU and 32bit-MDU have 68 and 198 FFs, for which the Synopsys DFT Compiler construct 7 and 20 scan paths with an ordinary option, respectively. Their circuit sizes are 3149 and 5005 NAND2-equivalent gates, and their increase rates from the non-scan design circuits are 3.93% and 7.44%.

Besides, the performance degradation depends on circuit operations during active mode because stress probabilities of PMOS transistors change depending on input vectors. For taking into account different situations of circuit operation, the following three situations are considered for 32bit-ALU.

**General:** *Both operations and operands are randomly selected and executed.*
**16bit-add:** *Higher 16bit of operands are fixed to 0. Only addition is executed.*
**Const-add:** *One operand is fixed to* $00000001_{(16)}$*, and the other is randomly selected. Only addition is executed.*

Operations for 32bit-MDU are listed below.

**General:** *Both operations and operands are randomly selected and executed.*
**32bit-mult:** *Operands are randomly selected. Only multiplication is executed.*
**32bit-div:** *Operands are randomly selected. Only division is executed.*
**16bit-mult:** *Higher 16bit of operands are fixed to 0. Only multiplication is executed.*
**16bit-div:** *Higher 16bit of operands are fixed to 0. Only division is executed.*

We gave four types of scan-in vectors ("all 1", "all 0", "0101" and "random") during the standby mode. "all 0" applies 0's to scan inputs, and "all 1" applies

1's. In these two inputs which are special cases of single-IVC, DC stress is given during the standby mode. On the other hand, "0101" and "random" change input vectors to combinational circuits during the standby mode. "0101" injects 1 and 0 alternately to scan inputs, and "random" inputs 1 and 0 randomly. In any cases, primary inputs are fixed to the last inputs of the previous active mode.

We evaluated the initial circuit delay and the delay after 10 years assuming supply voltage is 1.1V, temperature is 125°C and initial threshold voltage is -180mV. We adopted the procedure of NBTI-aware timing analysis [9] in this paper. We first calculated stress probabilities of all PMOSs with logic simulation taking into account both active and standby modes, next annotated $|\Delta \mathrm{Vth}|$ for each PMOS according to the stress probability, and carried out transistor-level static timing analysis with Synopsys Nanotime. To estimate the upper bound of NBTI mitigation effect, we also computed the circuit delay supposing a practically-infeasible situation that all PMOSs were always in recovery phase during the standby mode. We call this case "all recovery". In this work, an operation to store and restore FF values is not considered, assuming ALU and MDU are flushed in standby mode.

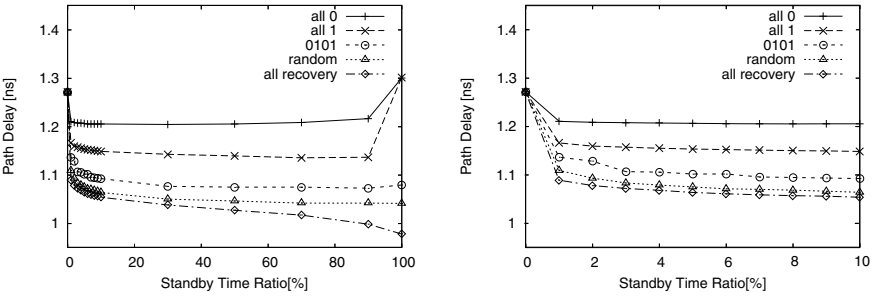
### 4.2 Evaluation Results

**Evaluation on Standby Time Ratio**

Figures 3 (a)(b) show the dependency of critical path delay on the ratio of standby mode in the case of *16bit-mult*. Here, the ratio is defined as (time_of_standby_mode) / (time_of_active_mode + time_of_standby_mode). Then, 0% means that the circuit is in active mode all the time, and 100% corresponds to the case that the circuit is always in standby mode. The initial critical path delay of 32-bit MDU is 0.978 ns.

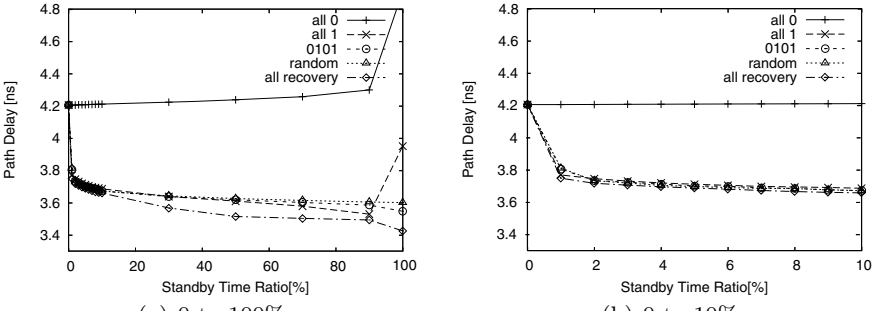
We can see from Fig. 3 (a)(b) that the critical path delays of "all 0" and "all 1" are larger than those of "0101" and "random". A single input vector during the standby mode significantly degrades the circuit delay, though other single-IVC input vectors may give better results. It should be noted here that even such single-IVC input vectors are expected to increase the circuit delay as the ratio of standby time approaches to 100%, because almost DC stress is given to the circuit. On the other hand, "random" attained the minimum delay increase, which indicates that giving a set of input vectors to combinational circuits is helpful to mitigate NBTI. Another important observation is that even a short ratio of standby time, such as 2%, greatly contributed to NBTI mitigation (Fig. 3 (b)), which suggests that our approach is applicable for busy circuits operating with little idle time.

Figures 4 (a)(b) show the result of *16bit-add*, where the initial critical path delay of 32bit ALU is 3.424ns. Below 90% of the ratio, the critical path delays are almost the same for "all 1", "0101" and "random", but above 90%, "all 0" and "all 1" induced a significant delay increase, whereas "random" mitigated NBTI further. This is because the degradation of PMOS in DC stress during the standby mode becomes quite large along with the increase of the standby time ratio. Similarly to Fig. 3, a small standby ratio reduced the delay increase.




(a) 0 to 100% (b) 0 to 10%

**Fig. 3.** Critical path delay vs. standby time ratio (16bit-mult)




(a) 0 to 100% (b) 0 to 10%

**Fig. 4.** Critical path delay vs. standby time ratio (16bit-add)

**Table 1.** Critical path delay with and without NBTI mitigation (32bit MDU). The initial delay is 0.978ns. Numbers in parentheses mean the ratios of delay increase from the initial circuit.

| Operation | No standby time | All recovery | w/ mitigation | | | |
|---|---|---|---|---|---|---|
| | | | all0 | all1 | 0101 | random |
| 16bit_mult | 1.271 (30.0%) | 1.055 (7.87%) | 1.205 (23.2%) | 1.148 (17.4%) | 1.092 (11.7%) | 1.064 (8.79%) |
| 32bit_mult | 1.131 (15.6%) | 1.040 (6.33%) | 1.116 (14.1%) | 1.109 (13.4%) | 1.083 (10.7%) | 1.051 (7.46%) |
| 16bit_div | 1.262 (29.0%) | 1.059 (8.28%) | 1.198 (22.5%) | 1.097 (12.2%) | 1.081 (10.5%) | 1.070 (9.41%) |
| 32bit_div | 1.077 (10.1%) | 1.043 (6.64%) | 1.077 (10.1%) | 1.069 (9.30%) | 1.060 (8.38%) | 1.052 (7.57%) |
| general | 1.049 (7.26%) | 1.037 (6.03%) | 1.049 (7.26%) | 1.048 (7.16%) | 1.049 (7.26%) | 1.047 (7.06%) |

**Table 2.** Critical path delay with and without NBTI mitigation (32bit ALU). The initial delay is 3.424ns. Numbers in parentheses mean the ratios of delay increase from the initial circuit.

| Operation | No standby time | All recovery | w/ mitigation | | | |
|---|---|---|---|---|---|---|
| | | | all0 | all1 | 0101 | random |
| 16bit_add | 4.206 (22.8%) | 3.659 (6.86%) | 4.211 (23.0%) | 3.687 (7.68%) | 3.673 (7.27%) | 3.671 (7.21%) |
| const_add | 4.565 (33.3%) | 3.74 (9.22%) | 4.579 (33.7%) | 3.735 (9.08%) | 3.739 (9.20%) | 3.744 (9.35%) |
| general | 3.599 (5.11%) | 3.581 (4.59%) | 3.611 (5.46%) | 3.590 (4.85%) | 3.599 (5.11%) | 3.599 (5.11%) |

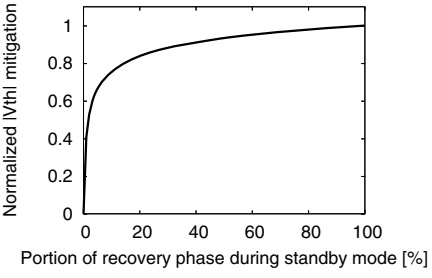



**Fig. 5.** $|V_{th}|$ mitigation at DC-stressed PMOSs

**Evaluation on Various Operations**

Table 1 lists the delay increases of 32bit MDU in the case of 10% of standby ratio. In the situations of *16bit-mult* and *16bit-div*, the amounts of mitigation with "random" are large, and especially in *16bit-mult* situation, "random" reduced the delay increase by 71% ((30.0%-8.79%)/30.0%).

Table 2 lists the evaluation results of 32bit ALU. Similarly to 32bit MDU, except for *general*, "random" attained large NBTI mitigation effects, though "0101" and "all 1" also mitigated NBTI significantly. Here, it is surmised that "all 1" achieves large mitigation because operands in which higher bits are fixed to 0s experience 1s during the standby mode.

We can also see that the circuit delay mitigated by "random" is close to that of "all recovery" in all situations. This result indicates that random scan-in vectors attained the mitigation effect close to the upper limit. Please recall that this upper limit cannot be obtained actually because no input vectors to make all PMOSs be in recovery phase simultaneously.

Let us investigate the reason why the NBTI mitigation by "random" is close to that of "all recovery". In this evaluation, we assumed standby time occupies 10% of the total time, which means stress probabilities can be reduced by 10% with "all recovery". On the other hand, the variation of stress probabilities due to "random" is less than 10%. Figure 5 shows $|V_{th}|$ mitigation for a PMOS whose

**Table 3.** Critical path delay with different LFSR configurations (10 yeas later)

| Configuration | 32bit ALU | 32bit MDU |
|---|---|---|
| random | 3.671 ns | 1.064 ns |
| 4bit LFSR | 3.676 ns | 1.104 ns |
| 8bit LFSR | 3.672 ns | 1.064 ns |
| 12bit LFSR | 3.672 ns | 1.064 ns |
| 16bit LFSR | 3.676 ns | 1.064 ns |

stress probability is 100% as a function of the portion of recovery phase during the standby mode. Here, $|V_{th}|$ mitigation is normalized by that of "all recovery". This figure indicates that if we could allocate 3% of standby time to recovery phase, 60% of mitigation effect can be obtained. When 15% of standby time can be given to recovery phase, we can have 80% of mitigation effect. We think this is the reason why "random" attained a good NBTI mitigation effect.

### 4.3 Random Number Generation

The previous section showed that random scan-in vectors are useful to mitigate NBTI. On the other hand, when applying this scheme to a circuit on a chip, we need an on-chip random number generator. We here focus on LFSR, which is a popular circuit to generate pseudo random numbers, and evaluate the relation between the quality of random number and NBTI mitigation effect. As the number of registers in LFSR increases, the cycle of random number becomes longer and the quality of random number improves, while the area needed for implementation becomes large.

We here evaluate critical path delays when LFSRs with different bits are used. The results of 4bit, 8bit, 12bit, and 16bit LFSRs are shown in Table 3. When 8bit or larger-bit LFSRs are used, there is little difference from "random" in mitigation efficiency. For 32bit ALU and MDU in MIPS R3000 processor, 8bit LFSR can generate a sufficiently random scan-in vectors for NBTI mitigation. This means that we do not have to shift the scan-in vectors fast and slow shifting is sufficient, because the number of LFSR cycles is not large, for example 256 in case of 8bit LFSR. This property is desirable in terms of power dissipation because fast shifting of scan-in vectors involves large power dissipation.

## 5 Conclusion

In this paper, we proposed a NBTI mitigation method that inject random patterns to scan paths, and evaluated the NBTI mitigation effect through a case study. The experimental results with 32bit ALU and MDU in MIPS R3000 processor showed that the proposed method well mitigated NBTI by 71% in a test case and the reduction of performance degradation is comparable to the upper limit that assumed all PMOS are in recovery phase. We also confirmed that 8bit

LFSR is capable of random number generation for NBTI mitigation. As a future work, we will evaluate the effectiveness including both PBTI and NBTI, since the proposed method works for them simultaneously.

## References


1. Wang, W., Reddy, V., Krishnan, A.T., Vattikonda, R., Krishnan, S., Cao, Y.: Compact modeling and simulation of circuit reliability for 65nm CMOS technology. IEEE Transactions on Device and Material Reliability 7, 509–517 (2007)
2. Ramey, S., Prasad, C., Agostinelli, M., Pae, S., Walstra, S., Gupta, S., Hicks, J.: Frequency and Recovery Effects in High- $k$ BTI Degradation. In: International Reliability Physics Symposium, pp. 1023–1027 (2009)
3. Abdollahi, A., Fallah, F., Pedram, M.: Leakage current reduction in CMOS VLSI circuits by input vector control. IEEE Transactions on Very Large Scale Integration Systems 12(2), 140–154 (2004)
4. Wang, Y., Luo, H., He, K., Luo, R., Yang, H., Xie, Y.: Temperature-aware NBTI modeling and the impact of input vector control on performance degradation. In: Design, Automation, and Test in Europe conference, pp. 546–551 (2007)
5. Wang, Y., Chen, X., Wang, W., Balakrishnan, V., Cao, Y., Xie, Y., Yang, H.: On the efficacy of input vector control to mitigate NBTI effects and leakage power. In: International Symposium on Quality Electronic Design, pp. 19–26 (2009)
6. Bild, D.R., Bok, G., Dick, R.P.: Minimization of NBTI performance degradation using internal node control. In: Design, Automation, and Test in Europe Conference, pp. 148–153 (2009)
7. Jin, S., Zhang, L., Li, H., Li, X., Yan, G.: M-IVC: Using Multiple Input Vectors to Minimize Aging-induced Delay. In: Asian Test Symposium, pp. 437–442 (2009)
8. Calimera, A., Macii, E., Poncino, M.: NBTI-aware sleep transistor design for reliable power-gating. In: Great Lakes Symposium on VLSI, pp. 333–338 (2009)
9. Konoura, H., Mitsuyama, Y., Hashimoto, M., Onoye, T.: Comparative study on delay degrading estimation due to NBTI with circuit/instance/transistor-level stress probability consideration. In: International Symposium on Quality Electronic Design, pp. 646–651 (2010)
10. Cao, Y.: Reliability mechanisms and the impact on IC designs. In: Asia and South Pacific Design Automation Conference, Tutorial, vol. 4, pp. 342–372 (2009)
11. Alam, M.A., Mahapatra, S.: A comprehensive model of PMOS NBTI degradation. Microelectronics Reliability 45, 71–81 (2005)
12. Lee, J.H., Wu, W.H., Islam, A.E., Alam, M.A., Oates, A.S.: Separation method of hole trapping and interface trap generation and their roles in NBTI reaction-diffusion model. In: International Reliability Physics Symposium, pp. 745–746 (2008)
13. Krishnan, A.T., Chansellor, C., Chakravarthi, S., Nicollian, P.E., Reddy, V., Varghese, A., Khamankar, R.B., Krishnan, S.: Material dependence of hydrogen diffusion: implications for NBTI degradation. In: IEEE International Electron Devices Meeting, pp. 688–691 (2005)

# An On-Chip All-Digital PV-Monitoring Architecture for Digital IPs

Hossein Karimiyan, Andrea Calimera, Alberto Macii,
Enrico Macii, and Massimo Poncino

Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, Italy
{hossein.karimiyan,andrea.calimera,alberto.macii,
enrico.macii,massimo.poncino}@polito.it

**Abstract.** This paper presents an on-chip all-digital sensor architecture to capture process variation information. The proposed solution is based on the concept of *variation amplification* and uses the propagation delay measurement in a chain composed of series connected pass-transistors. The proposed sensor circuit is able to capture the local variation of nMOS and pMOS transistor individually. A sensor block is proposed, which contains N-type and P-type sensor circuit along with scan, control, and measurement circuitry. An array of sensor blocks with scan chain connection gathers process variation information all across the die. Detailed SPICE level simulations conducted for an industrial 45nm CMOS technology indicates its feasibility in sensing, and on-chip all-digital measurement of process variation effect.

**Keywords:** Process variation, sensor.

## 1 Introduction and Previous Works

From almost early days of integrated circuits, un-similarity of circuit elements after fabrication was a concern for circuit designers. While for the analog domain the un-similarity of electronic devices (known as *device mismatching*) has been always considered a major source of circuit-failure, digital applications started to face the same problems only recently [17].

For old CMOS technologies (>100nm), where the dimensions of the devices were macroscopic units with respect to the tolerance of the fabrication processes, un-similarity of devices was a second-order effect. But as the devices shrunk to nanometer lengths, the effects of process-variation become non-negligible anymore.

Process-variation manifests in the form of large spread in the electrical parameters of circuit devices, resulting in sensible variations of different circuit metrics, like operating frequency and power consumption. Such variations may range from few percentages to several orders of magnitude, also depending on the spatial-scale they reach: wafer-to-wafer (W2W), die-to-die (D2D), and with-in-die variations (WID) [1]. For instance, 20X variation in leakage power and a 1.5X variation in delay between fast and slow dies have been reported [18]. Given the power/performance

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 162–172, 2011.

metrics, those figures translate into an increase of dies that must be discarded because either too slow or too much power consuming, which decreases the fabrication yield.

While the binning method [15] has been successfully adopted for tackling W2W and D2D variations, WID variations, which are more localized, require new dedicated solutions, not just during the fabrication, but also in the earlier phases of the design flow [16]. Therefore, developing new design techniques that can address the problem of variability has become the main challenge for designers and CAD developers [2][4].

Several design methodologies have been proposed in the last years, from those based on *Design for Manufacturability* approaches (like litho-friendly and Restricted Design Rules layout design), where variability of a given design is either mitigated or amortized, to *Adaptive* strategies, which attempt to solve the variability issues by "sensing and correcting" the desired parameters using various knobs that affect them. These schemes are also called *Monitor&Control* (M&C) strategies, to emphasize their analogy with closed-loop control systems.

Although they have been proven to be extremely flexible, the applicability of M&C strategies poses some critical design issues. First, the choice of the most appropriate monitoring approach, which depends on the desired metric to be controlled, and the second, the design of proper monitoring units that can probe the system, in real-time. When considering performance and thus *timing yield*, several options are available. Drain current measurement between identical neighboring devices is conventionally used to characterize local random mismatch [8]. However, measurement of small currents through minimum sized transistors requires sophisticated analog measurement techniques and off-chip equipments. The threshold voltage of a MOSFET, which is expression of different device parameter (like *gate dimension*, *channel doping, etc.*), is another measure of the process variation [7][12], but its direct measurement necessitates current or voltage references and sophisticated analog blocks, which are not feasible at low supply voltages [15].

A more accurate, yet effective expression of process variation in digital circuits is given by the propagation delay, which could be measured using direct and indirect methods. Direct delay measurement requires either on-chip or off-chip time-to-digital converter (TDC) circuits [17]. The test circuits are either replica of main circuit or some standard testbench circuits constructed by nMOS and pMOS devices; hence the measured delay value is an average of the specifications and so an average of the variation effect.

The use of ring-oscillator as indirect delay measurement is proposed, where the shift in the oscillating frequency is used as indicator of the process variation [3][5][6][12]. Even if implementable with a compact layout, the deployment of ring oscillators needs external time-domain or frequency-domain processing equipments. Moreover, the measured delay values represent an average of nMOS and pMOS specifications.

Contribution of this paper is twofold. First, it presents a new sensor circuit, which exploits the concept of PV-amplification through a chain of pass transistors. The idea is applied to a chain of pass transistors constructed by the nMOS or pMOS devices only. Since the process variation affects nMOS and pMOS devices in different manner, individual process measurement opens the filed for possible different control or tuning, like body biasing, which could be applied on nMOS and pMOS sections

separately. This allow build a sensor circuit that can be adopted for measuring, but most important, to support M&C strategies. Second, we propose a monitoring solution that leverages the sensor array to collect process variation information across the chip. We propose the use of the scan-chain as main communication pipe between sensors, thus avoiding the use of any external equipment.

The remainder of this paper is organized as follows. Section 2 and 3 present the basic idea of the proposed method, introduces the sensor block, and the on-chip all-digital sensor array integration. Design, simulation, and analysis results are presented in Section 4, and Section 5 provides summary and conclusion remarks.

## 2 Process Variation Sensor Structure

The main goal in the proposed N-Sensor and P-Sensor circuits is to capture the variation information of nMOS and pMOS devices individually. Fig. 1 shows the basic structure of the N-Sensor circuit constructed by nMOS devices. The P-Sensor has the same structure, but uses pMOS devices in the chain. Each N-Sensor (P-Sensor) consists of a chain of nMOS (pMOS) devices in the pass transistor connection form. One end of the chain is driven by a driver (I1), while the other end of the chain (O1) is connected to a dynamic gate.

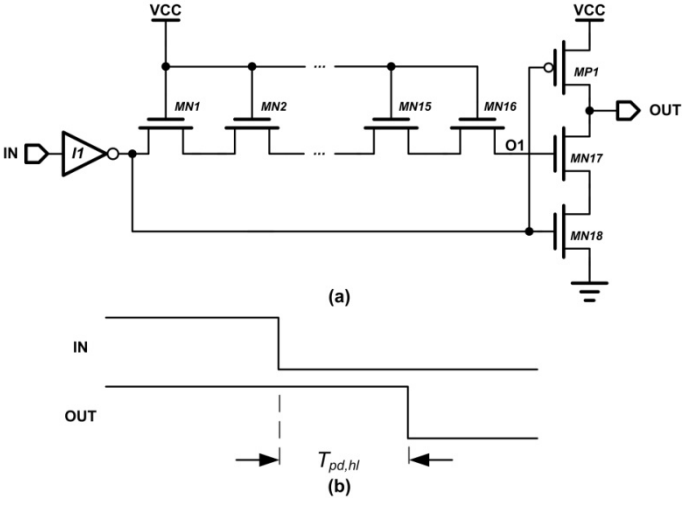



**Fig. 1.** Proposed process variation sensor, (a) N-Sensor constructed using nMOS devices, and (b) propagation delay

The proposed N-Sensor in Fig. 1 operates as follows. When the input (IN) is set to the HIGH-logic value, it forces the dynamic gate to operate in the pre-charge mode and charges the output node (OUT) to the HIGH logic value. Changing the IN input to the LOW logic value, the dynamic gate starts the evaluation phase. The rising edge at the output of the inverter I1 starts to propagate toward the end of line. The rising

edge reaches the dynamic gate after passing through the series connected transistors and experiencing a delay. Due to the threshold voltage drop effect [15], the output node of the pass transistor chain has reduced swing between GND and VCC-$V_{th,n}$ , where $V_{th,n}$ is nMOS threshold voltage. This value is sufficient for the dynamic gate to operate correctly and recover the rail to rail swing, which is also helpful to avoid possible leakage current in the next stages.

The falling edge propagation delay of the N-Sensor circuit in Fig. 1 is composed of the rising edge delay of the input inverter, rising edge delay of the pass transistors, and falling edge delay of the dynamic gate:

$$t_{pd,hl} = t_{inv,lh} + t_{tg,lh} + t_{dyn,hl} \tag{1}$$

Where *hl* indicates HIGH-to-LOW (or falling edge) propagation delay, and *lh* indicates LOW-to-HIGH (or rising edge) propagation delay. It is worth mentioning that, except the rising edge delay of the input inverter, in which pMOS is involved, the remaining components in propagation delay of the N-Sensor circuit are just due to the nMOS devices.

The propagation delay value of pass transistors is proportional to the number of pass transistors in the chain, their physical dimensions and orientation, and how much they are affected by local process variation. According to the Elmore RC delay model [15], the propagation delay of pass transistor connection is estimated by the following equation:

$$t_{tg,lh} \approx 0.69 \sum_{k=0}^{n} C_{eq,N} R_{eq,N} k = 0.69 C_{eq,N} R_{eq,N} \frac{n(n+1)}{2} \tag{2}$$

Where $C_{eq,N}$ and $R_{eq,N}$ are equivalent capacitance and channel resistance of each nMOS transistor, respectively, and $n$ is the number of stages. At the device level, there are four main sources of process variation: Line edge roughness (LER), (which affects gate dimensions, length and width), oxide thickness roughness (OTR), and random dopant fluctuation (RDF) in the channel. Variations on those physical parameters or circuit parameters manifest as fluctuations in threshold voltage of MOSFET devices in the circuit. Any fluctuations on the threshold voltage of transistors, changes the equivalent channel resistance and hence changes the propagation delay through the RC network.

Using high threshold device on the input inverter, which is helpful to reduce process variation effect ( $\Delta t_{inv,lh} \approx 0$ ), and applying (2) in (1), the variation of the propagation delay can be approximated by the following equation:

$$\Delta t_{pd,hl} \approx \Delta t_{inv,lh} (\approx 0) + 0.69 C_{eq,N} \Delta R_{eq,N} \frac{n(n+1)}{2} + \Delta t_{dyn,hl} \tag{3}$$

According to (3) any change on device parameter affects the total propagation delay by a multiply of '*n(n+1)/2*'. In other words, process variation effect on the delay of a single nMOS device is amplified by a high gain value. The process variation amplification (PVA) spreads the propagation delay and makes it suitable for on-chip digital measurement.

Similarly, the main part of the propagation delay in the P-Sensor circuit is mostly due to pMOS device. Deploying the PVA, P-Sensor is used to measure the process variation effect on pMOS device.

## 3 Sensor Integration

In order to capture process variation information across the die area, an array of sensor blocks is suggested. Fig. 2 shows the array and the internal design of the proposed sensor block. Each block contains an N-Sensor and a P-Sensor circuit. In addition to sensors, each block has a scan chain interface, trigger & control logic, and the TDC circuitries.

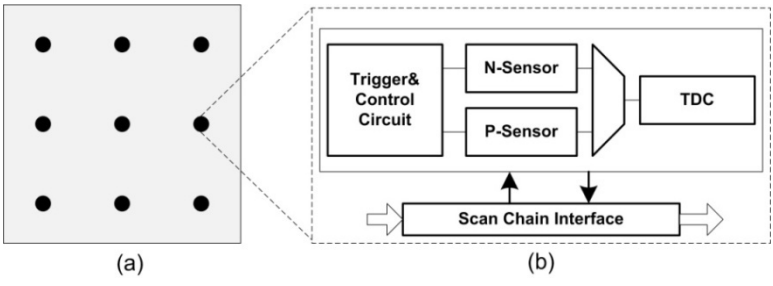



**Fig. 2.** (a) Sensor array embedded on the chip, (b) internal structure of each sensor

Each measurement session starts with getting access to the specific element in the array by addressing its Scan Chain Interface. Then the Trigger & Control Circuit sends rising and falling edges to N-Sensor and P-Sensor, respectively. The TDC circuit measures the delay value and stores it in the flip-flops (FFs), which are locally accessible through the Scan Chain Interface. Fig. 3 shows the internal design and delay measurement circuit of each sensor block circuit.

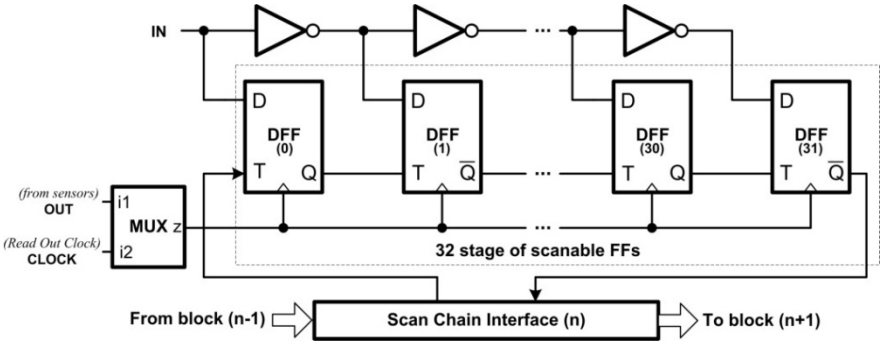



**Fig. 3.** The delay measurement setup using linear delay line with locally scanable FFs

The Sensor's output and a local Clock are available in Fig. 3 as a clock input for the FFs. The former is used during measurement phase and the later in the read out phase. During the measurement phase the FFs sample the IN at the rising edge of the OUT signal, providing a binary word which contains a number of 1s proportional to the delay introduced by the sensor; during the read-out phase the FFs are controlled and synchronized with the scan-chain by the main clock. The measured delay value in the delay line method is in the thermometric code form and conversion logic is usually needed to convert it to the binary form [17]. However, the local scan chain is helpful to remove the conversion logic and post-process information at upper layer using software applications, which result in more compact and low-power sensor circuit.

## 4 Design, Simulation and Analysis Results

The proposed N- and P-Sensor circuits have been designed and simulated using an industrial 45nm CMOS technology provided by ST Microelectronics. The simulation results were obtained at room temperature (27°C) and 1.1V nominal supply voltage using SPICE simulator.

Smaller devices with lower threshold voltage have been proven to be more sensitive to process variation. This suggested us to map the chain to minimum feature size and low threshold voltage, thus making it more prone to process variation.

Table 1 compares the propagation delay and its distribution in sensors with varying stage numbers. Result obtained through the Monte Carlo simulation using large number of samples with 3-sigma distribution in device parameters including LER, RDF, and threshold voltage of both pMOS and nMOS devices.

**Table 1.** Delay comparison of sensors with different stage numbers

| Type | No. of Stages | Mean (ps) | Sigma (ps) |
|---|---|---|---|
| P-Sensor | 4 | 148.4 | 34.1 |
| | 8 | 307.0 | 70.1 |
| | 16 | 697.3 | 154.4 |
| N-Sensor | 4 | 78.2 | 22 |
| | 8 | 160.1 | 44.39 |
| | 16 | 365.7 | 93.4 |

Fig. 4 shows the input and few samples of output waveform and propagation delay distribution of N-Sensor circuit with the 16 stages. The PVA effect of the chain manifests in large delay variations at the output of the sensor, i.e., large variance on the delay distribution. Having larger delay spread is synonym of larger sensibility to process variation, which in turn, allows easier and less expensive measurement and digital conversions. This is proven by Fig. 4(c) which shows the delay distribution has a mean value of around 350ps and spread up to 750ps.

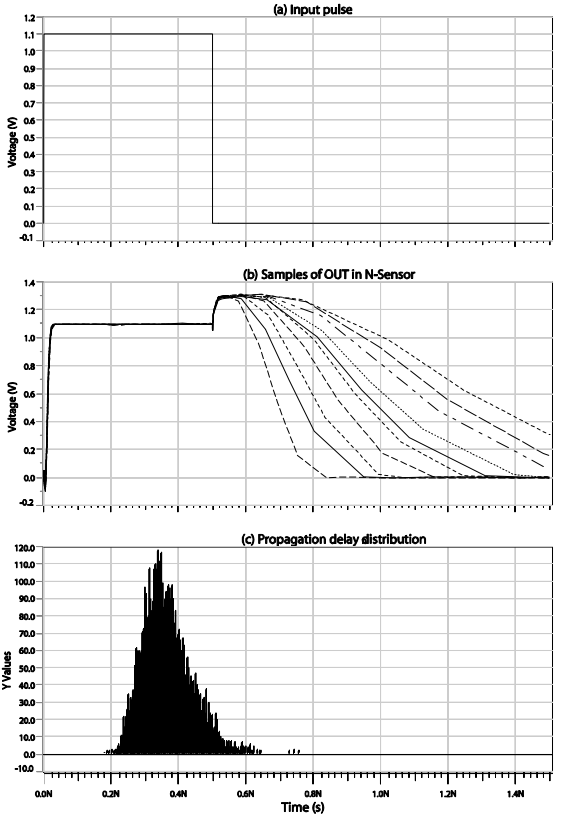



**Fig. 4.** Simulation result of the N-Sensor with 16 stages obtained at fixed supply and room temperature with 3-sigma distribution in device parameters, (a) input pulse, (b) samples of output pulse, and (c) propagation delay distribution

Fig. 5 shows the input and few samples of output waveform of P-Sensor circuit and its delay distribution. Since pMOS compared to nMOS has lower mobility and higher channel resistance, number of stages for P-Sensor is set to 8. Due to the PVA, the delay distribution in Fig. 5(c) has a mean value of 300ps and spread up to 600ps.

Realizing on-chip process variation sensor through direct propagation delay measurement requires TDC circuit. Among different TDC methods, the linear delay line is a simple and compact method [14][17]. It has the minimum resolution of one gate delay, which with respect to the current technologies, seems too rough to capture the small delay variation in a single device. However, deploying the PVA effect, the propagation delay in Fig. 4 and Fig. 5 has wider spread, which makes the basic inverter delay (≈20ps) enough resolution; hence the linear delay line method is sufficient for delay measurement in this design. The optimal number of stages in TDC circuit is chosen based on delay spread, i.e. 150ps~750ps, and the basic gate delay, which with a safe bound 32 stages is used.

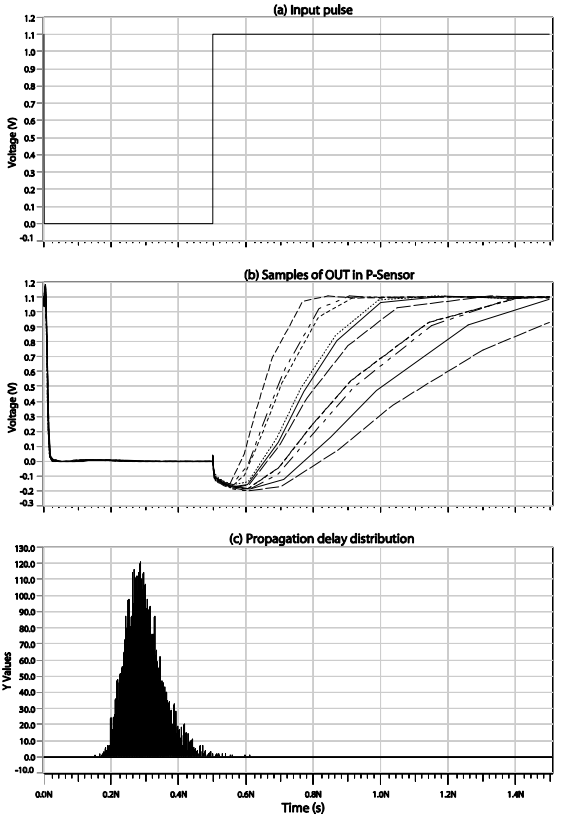



**Fig. 5.** Simulation result of the P-Sensor with 16 stages obtained at fixed supply and room temperature with 3-sigma distribution in device parameters, (a) input pulse, (b) samples of output pulse, and (c) propagation delay distribution

Fig. 6 shows the long run simulation and quantization result, and indicated the number of times that propagation delay is reached to each FF in the chain of TDC and gets captured. Despite the quantization effect, comparing the transistor level simulation result of Fig. 4 and Fig. 5, with digitized values in Fig. 6 shows the similar distribution and indicates the circuit's capability to capture the variation effect in digital form without using any external measurement equipment.

Based on the fact that process and even environmental variation effect is not fast changing, each block could be utilized at lower speeds to reduce the dynamic power. During each process variation measurement session, other blocks are not needed and they made idle using power gating (PG) switch to eliminate static power. Table 2 compares different variation sensor designs in their sensing method, circuit parameters, and objective. Most published circuits are intended to characterize process rather than circuit tuning. This work is targeted for circuit tuning, and deploy

a simple sensor circuit array to measure variation effect and makes the resulted digital signature available on-chip for any possible circuit enhancement in run-time.

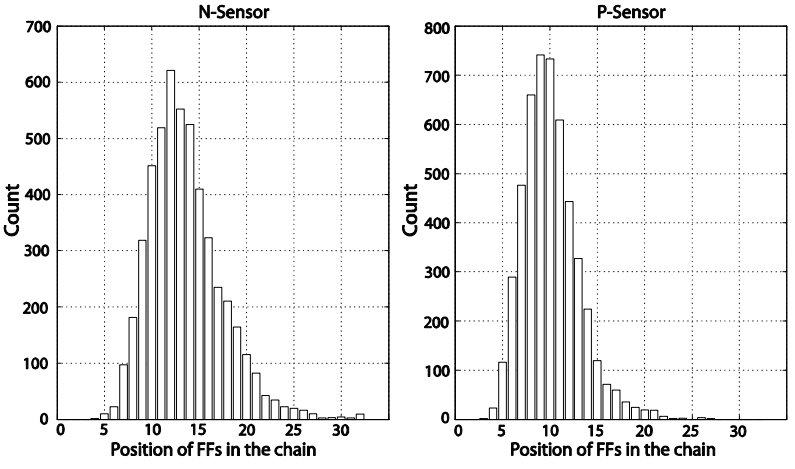



**Fig. 6.** Quantized delay as it captured on the TDC's FFs for N-Sensor and P-Sensor

**Table 2.** Several sensor design comparison

| *Design* | *Sense Variable* | *On-Chip* | *Objective* |
|---|---|---|---|
| [7] | Threshold Voltage | No | Process Characterization |
| [8] | Drain Current | No | Process Characterization |
| [9] | Delay | Yes | Process Characterization |
| [10] | Threshold Voltage | Yes | Process Characterization |
| [5] | Delay | No | Circuit Tuning |
| This work | Delay | Yes | Circuit Tuning |

## 5 Conclusion

The significance of process variation effect in recent technologies has increased the need for accurate characterization and measurement of this effect for any possible circuit enhancement or tuning. In this work, an on-chip all-digital variation sensor circuit to detect local variations of MOSFET parameters was proposed. Using the delay line, the proposed sensing method was aimed to capture variation information of the nMOS and pMOS devices individually. The basic concept, its practical realization, and sensitivity to variation have been described. The array of proposed sensor block was able to characterize variation information all across the die in a digital signature. Using the scan chain connection between sensors blocks, processing unit was able to capture the variation information without requiring any analog or complex external measurements equipments. The feasibility and accuracy of the test structure is demonstrated in an industrial 45nm CMOS technology in both nMOS and

pMOS versions. The results indicate that the sensor can measure random mismatch in nMOS and pMOS device parameters individually with minimal time and complexity, and thus enable run-time optimization and yield improvement.

## References


1. Sylvester, D., Agarwal, K., Shah, S.: Variability in nanometer CMOS: Impact, analysis, and minimization. The VLSI Journal Integration 41(3), 319–339 (2008)
2. Meijer, M., Liu, B., van Veen, R., Pinade de Gyvez, J.: Post-silicon tuning capabilities of 45nm low-power CMOS digital circuits. In: 2009 Symposium on VLSI Circuits, pp. 110–111 (2009)
3. Bhushan, M., Gattiker, A., Ketchen, M.B., Das, K.K.: Ring oscillators for CMOS process tuning and variability control. IEEE Trans. on Semiconductor Manufacturing 19(1), 10–18 (2006)
4. Kim, C.H., Hsu, S., Krishnamurthy, R., Borkar, S., Roy, K.: Self calibrating circuit design for variation tolerant VLSI systems. In: 11th IEEE International On-Line Testing Symposium, IOLTS 2005, pp. 100–105 (2005)
5. Nourani, M., Radhakrishnan, A.: Testing On-Die Process Variation in Nanometer VLSI. IEEE Design & Test of Computers 23(6), 438–451 (2006)
6. Johguchi, K., Kaya, A., Mattausch, H.J., Koide, T., Izumi, S., Sadachika, N.: Measurement-Based Ring Oscillator Variation Analysis. IEEE Design & Test of Computers 27(5), 6–13 (2010)
7. Agarwal, K., Hayes, J., Nassif, S.: Fast Characterization of Threshold Voltage Fluctuation in MOS Devices. IEEE Transactions on Semiconductor Manufacturing 21(4), 526–533 (2008)
8. Agarwal, K., Liu, F., McDowell, C., Nassif, S., Nowka, K., Palmer, M., Acharyya, D., Plusquellic, J.: A Test Structure for Characterizing Local Device Mismatches. In: 2006 Symposium on VLSI Circuits, pp. 67–68. Digest of Technical Papers (2006)
9. Drego, N., Chandrakasan, A., Boning, D.: All-Digital Circuits for Measurement of Spatial Variation in Digital Circuits. IEEE Journal of Solid-State Circuits 45(3), 640–651 (2010)
10. Rao, R., Jenkins, K.A., Kim, J.-J.: A Local Random Variability Detector with Complete Digital On-Chip Measurement Circuitry. IEEE Journal of Solid-State Circuits 44(9), 2616–2623 (2009)
11. Liang, X., Wei, G.-Y., Brooks, D.: Revival: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency. IEEE Micro 29(1), 127–138 (2009)
12. Agarwal, K.: On-die sensors for measuring process and environmental variations in integrated circuits. In: Proc. of the 20th Great Lakes Symposium on VLSI 2010, pp. 147–150 (2010)
13. Keshavarzi, A., Schrom, G., Tang, S., Ma, S., Bowman, K., Tyagi, S., Zhang, K., Linton, T., Hakim, N., Duvall, S., Brews, J., De, V.: Measurements and modeling of intrinsic fluctuations in MOSFET threshold voltage. In: Proc. of the 2005 International Symposium on Low Power Electronics and Design (ISLPED 2005), pp. 26–29 (2005)
14. Restle, P.J., Franch, R.L., James, N.K., Huott, W.V., Skergan, T.M., Wilson, S.C., Schwartz, N.S., Clabes, J.G.: Timing uncertainty measurements on the Power5 microprocessor. In: IEEE International Solid-State Circuits Conference, vol. 1, pp. 354–355. Digest of Technical Papers (2004)
15. Weste, N., Harris, D.: CMOS VLSI Design: A Circuits and Systems Perspective, 4th edn. Addison Wesley, Reading (2010)

16. Dadgour, H.F., Banerjee, K.: A Novel Variation-Tolerant Keeper Architecture for High-Performance Low-Power Wide Fan-In Dynamic or Gates. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18(11), 1567–1577 (2010)
17. Henzler, S.: Time-to-digital converters, 1st edn. Springer, Heidelberg (2010)
18. Zhao, W., Liu, F., Agarwal, K., Acharyya, D., Nassif, S.R., Nowka, K., Cao, Y.: Rigorous extraction of process variations for 65nm CMOS design. IEEE Transactions on Semiconductor Manufacturing 22(1), 196–203 (2009)

# Chip Level Statistical Leakage Power Estimation Using Generalized Extreme Value Distribution

Alireza Khosropour, Hossein Aghababa, Ali Afzali-Kusha, and Behjat Forouzandeh

Nanoelectronics Center of Excellence,
School of Electrical and Computer Engineering
University of Tehran
Tehran, Iran
{a.khosropour,h.aghababa}@ece.ut.ac.ir

**Abstract.** Previous works for full-chip leakage power estimation are all based on Wilkinson's approach which approximates sum of lognormal random variables as another lognormal by matching the first and second moments. In this paper we will show that natural logarithm of leakage deviates from normal distribution by scaling transistor sizes, as a result distribution of leakage power cannot be described by lognormal distribution anymore. We will introduce generalized extreme value distribution as the best candidate for full-chip leakage power estimation and we will prove its superiority over lognormal approximation through simulation results in 45nm technology.

## 1 Introduction

Variations in process parameters play an important role to shift deterministic design paradigm of CMOS circuits to a new statistical paradigm. leakage power has become the most stringent constraint on the design of sub 90 nm CMOS circuits where it shows 20x variations compared to 30% variations in delay [1]. This intensive amount of variability in leakage will translate in terms of yield loss if it is not taken into account at different stages of a design.

There have been many published approaches in order to estimate leakage distribution of a chip under process variations [2, 3, 4, 5, 6, 7]. Some of these efforts are based on Wilkinson's [2] method which approximates sum of lognormal random variables as another lognormal by matching the first two moments [3, 4, 5, 6]. Some others try to approximate $log(P_{chip})$ by a Gaussian distribution and finally they find the corresponding lognormal distribution of $P_{chip}$ using a nonlinear transform [7, 8] . The only difference amongst these published papers is their method in estimation of final lognormal distribution. We will show through Spice simulations that $log(P_{chip})$ is not Gaussian for sub 45nm technologies, therefore lognormal approximation will not be accurate enough for estimation of leakage distribution. We will introduce a framework for accurate approximation of PDF and CDF of full-chip leakage using Generalized Extreme Value Distribution (GEV).

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 173–179, 2011.

This paper is organized as follows. Section 2 describes the traditional Wilkinson's method and explains why this method is not accurate enough. In section 3 we will develop our method and verify its accuracy through simulations. Finally in section 4 we conclude this paper.

## 2 Wilkinson's Method

We know that leakage power distribution for a single gate is lognormal and can be modeled by an exponential function of variation sources as $P_{gate} = e^{\mathrm{v_{nom}}+\sum \alpha_i \cdot \Delta X_i}$ , where $\mathrm{v_{nom}}$ is natural logarithm of leakage in the absence of any variations in parameters, $\Delta X_i$'s are variations in process parameters from their nominal value and $\alpha_i$'s are coefficients of the model which can be calculated through curve fitting. In addition we know that full-chip leakage is sum of leakages of individual gates in the circuit that can be expressed as $P_{chip} = \sum_i P_i$ , where $P_\mathrm{i}$ is the leakage of the $i^{\text{th}}$ gate. The main problem in estimation of full chip leakage distribution is that there is no closed form expression for sum of lognormal distributions, so we are inevitable of using approximations. The first and most popular approximation to lognormal sums which is known as Wilkinson's method, is widely used in leakage power estimation.

In Wilkinson's approach, first the mean and variance of full chip leakage power are calculated as following:

$$m = \mu_{chip} = E\left[P_{chip}\right] = \sum_i E[P_i] = \sum_i \mu_i \tag{1}$$

$$v = {\sigma^2}_{chip} = E\left[(P_{chip} - \mu_{chip})^2\right] = \sum_i {\sigma_i}^2 + \sum_{i,j,i\neq j} 2cov(P_i, P_j) \tag{2}$$

Then a lognormal distribution is fitted to leakage by matching its first two moments with $m$ and $v$. PDF of a lognormal random variable is given in equation (3). Equation (4) shows how to find parameters of a lognormal random variable by moment matching.

$$f(x|\mu,\sigma) = \frac{1}{x\,\sigma\sqrt{2\pi}} \exp\left(\frac{-(\ln x-\mu)^2}{2\,\sigma^2}\right) \tag{3}$$

$$\mu = ln\left(m^2/\sqrt{v+m^2}\right) \quad , \quad \sigma = \sqrt{log\left(\frac{v}{m^2}+1\right)} \tag{4}$$

Since parameters of lognormal distribution are calculated by matching the first and second moments, the skewness and $Y\%$ percentile point $X$ of the lognormal distribution are forced to be:

$$skewness = \left(e^{\sigma^2}+2\right)\sqrt{e^{\sigma^2}-1} \tag{5}$$

$$X = \exp\left(\sqrt{2}\sigma erf^{-1}(2Y-1)+\mu\right) \tag{6}$$

Monte Carlo (MC) simulations on ISCAS85 benchmark circuits reveal that natural logarithm of leakage deviates from Gaussian distribution for scaled technologies. Therefore lognormal approximation (as suggested by Wilkinson's approach) will not be accurate enough to estimate PDF of leakage. Fig.1 depicts natural logarithm of leakage for c1355 circuit at different technology nodes where it can be readily seen that skewness increases by scaling to the point that at 45nm technology leakage distribution will not be lognormal anymore as shown in Fig.2. Fig.2 compares histogram of leakage (obtained

from MC simulations) with PDF obtained from Wilkinson's approximation for c1355 at 45nm technology node. As shown in this figure, lognormal distribution doesn't follow leakage power at certain areas of the distribution specifically at both tails where it is of great interest. In all of our experiments variation sources are assumed to be $vth_{nmos}$, $vth_{pmos}$, $L_{eff}$, $t_{ox}$. We have assumed Gaussian distribution and $3\sigma$ value for inter/intra-die variations respectively 8% and 6% of the nominal value for each of these variation sources. Table 1 compares the mean, variance, 99% percentile point and skewness of leakage powers for ISCAS85 benchmark circuits obtained from MC simulations and Wilkinson's approach for 45nm technology.

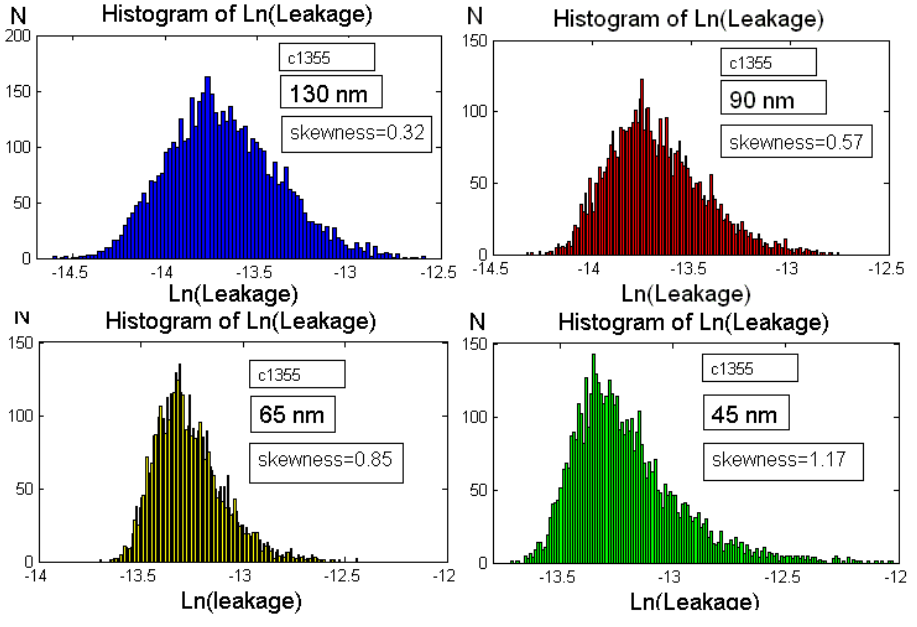



**Fig. 1.** Comparison between skewness of Ln(Leakage) for c1355 at different technology nodes

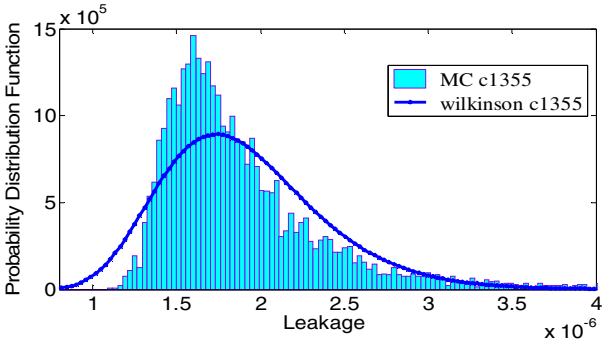



**Fig. 2.** Comparison between leakage distributions obtained from Monte Carlo and Wilkinson's approach for c1355 circuit at 45nm technology node

**Table 1.** Mean , variance , skewness , 99% prcentile point of ISCAS85 benchmark circuits calculated using Wilkinson's approach compared to Monte Carlo simulation results (45nm)

| Bench Mark | Wilkinson | | | | Monte Carlo | | | |
|---|---|---|---|---|---|---|---|---|
| | μ(uW) | σ(uW) | skewness | 99%(uW) | μ(uW) | σ(uW) | skewness | 99%(uW) |
| C432 | 0.67(-1.4%) | 0.23(0.0%) | 1.05(-54.9%) | 1.38(-7.4%) | 0.68 | 0.23 | 2.33 | 1.49 |
| C499 | 0.80(-1.2%) | 0.35(-5.4%) | 1.49(-36.6%) | 2.05(-4.6%) | 0.81 | 0.37 | 2.35 | 2.17 |
| C880 | 1.34(0.0%) | 0.53(-1.8%) | 1.26(-48.7%) | 3.07(-6.9%) | 1.34 | 0.54 | 2.46 | 3.30 |
| C1355 | 1.93(1.0%) | 0.63(-1.5%) | 1.04(-55.5%) | 3.88(-7.1%) | 1.91 | 0.64 | 2.31 | 4.18 |
| Ave | -0.4% | -2.1% | -48.8% | -6.5% | - | - | - | - |

It is obvious from Table 1 that skewness obtained from Wilkinson's approach has -48.8% error compared to MC results. Also 99% percentile point error for Wilkinson's approach is about 6.5%.

## 3 Generalized Extreme Value Distribution

In section 2 we proved that natural logarithm of leakage is not normally distributed, therefore leakage will not be lognormal as depicted in fig.2 where it is readily seen that lognormal approximation underestimates skewness of leakage. In order to find the best distribution which characterizes statistics of full-chip leakage power, we have considered more than 60 types of distributions. These distributions were fitted to MC data and they were sorted based on different goodness of fit (GOF) tests such as Chi-Squared, Kolmogorov-Smirnov , Anderson-Darling. In all of our observations there were about 13 families of distributions which fitted to leakage power better than lognormal distribution, so they were candidates for full-chip leakage power distribution. Here we sort these 13 family of distributions for c1355 circuit based on chi-squared test: 1.Generalized Extreme Value 2.Frechet(3P) 3.Pearson5(3P) 4.Burr 5.Log-Logistic(3P) 6.Dagum(4P) 7.Lognormal(3P) 8.Fatigue Life(3P) 9.Generalized Gamma(4P) 10.Dagum 11.Inverse Gaussian(3P) 12. Pearson5 13.Log-logistic 14.Lognormal , … (3P means 3 parameter distribution).

It should be mentioned that the first 11 distributions in the above list have more than 3 parameters .The only 2 parameter distributions are Pearson5 and Log-Logistic which are not far better than Lognormal according to our observations. This means that we should choose between one of the 3 parameter distributions if we want to obtain better fits. We decided to choose generalized extreme value distribution (GEV) for full-chip leakage power estimation not for the sole reason that it ranked 1st in the above list. The more important reason behind this decision is that GEV's parameters can be easily estimated given its mean, variance and skewness.
PDF and CDF of GEV are as follows:

$$f(x|\mu,\sigma,k)=\frac{1}{\sigma}\left[1+k\left(\frac{x-\mu}{\sigma}\right)\right]^{\left(-\frac{1}{k}-1\right)}\exp\left(-\left(1+k\left(\frac{x-\mu}{\sigma}\right)\right)^{-\frac{1}{k}}\right) \tag{7}$$

$$F(x|\mu,\sigma,k)=exp\left(-\left(1+k\left(\frac{x-\mu}{\sigma}\right)\right)^{-\frac{1}{k}}\right) \tag{8}$$

Mean, variance and skewness of GEV are given as follows:

$$Mean = m = \begin{cases} \boldsymbol{\mu + \sigma \frac{\Gamma(1-k)-1}{k}} & \boldsymbol{if \quad k \neq 0, k < 1} \\ \mu + \sigma\gamma & if \quad k = 0 \\ \infty & if \quad k \geq 1 \end{cases} \tag{9}$$

$$Variance = v = \begin{cases} \boldsymbol{\sigma^2 \frac{(g_2 - {g_1}^2)}{k^2}} & \boldsymbol{if \quad k \neq 0, k < 0.5} \\ \sigma^2 \frac{\pi^2}{6} & if \quad k = 0 \\ \infty & if \quad k \geq 0.5 \end{cases} \tag{10}$$

$$Skewness = s = \begin{cases} \boldsymbol{\frac{g_3 - 3g_1g_2 + 2g_1^3}{(g_2 - g_1^2)^{3/2}}} & \boldsymbol{if\ k \neq 0} \\ \frac{12\sqrt{6}\zeta(3)}{\pi^3} & if\ k = 0 \end{cases} \tag{11}$$

In the above equations $\gamma$ is Euler's constant, $\Gamma(t)$ is gamma function, $\zeta(t)$ is Riemann zeta function and $g_n = \Gamma(1 - (n.k))$ . In Equation 11 one can observe that skewness of GEV distribution is only a function of parameter k.

By calculating the skewness of full-chip leakage power, the parameter k can be easily estimated using numerical methods. After obtaining k, the other parameter of distribution $\sigma$ can be calculated as:

$$\sigma = \sqrt{\frac{k^2\, v}{g_2 - g_1^2}} \tag{12}$$

The remaining parameter μ can be easily calculated by substituting k and $\sigma$ in equation (9) and obtaining:

$$\mu = m - \sigma \frac{\Gamma(1-k)-1}{k} \tag{13}$$

By calculating 1st, 2nd and 3rd moments of full-chip leakage power, one can easily obtain parameters of corresponding GEV distribution from equations 11-13. 1st and 2nd moments of full-chip leakage power were given in equations 1 and 2. On the other hand the 3rd moment or equivalently the skewness of full-chip leakage can be calculated as follows:

$$skewness_{P_{chip}} = \frac{E\left[\left(P_{chip} - \mu_{chip}\right)^3\right]}{\sigma_{chip}^3} = \tag{14}$$

$$\frac{1}{\sigma_{chip}^3}\left\{\sum_i E[(P_i - \mu_i)^3] + \sum_{i,j,i\neq j} 3\, E\left[(P_i - \mu_i)^2 . \left(P_j - \mu_j\right)\right] + \sum_{i,j,k,i\neq j\neq k} 3! \times E\left[(P_i - \mu_i)\left(P_j - \mu_j\right)(P_k - \mu_k)\right]\right\}$$

Fig.3 compares PDF and CDF of corresponding lognormal and GEV distributions with MC results for c1355 circuit. Process variation sources are assumed to be the same as those in section 2. It is obvious that GEV distribution fits to MC data far

better than lognormal distribution. Table 2 illustrates the improvement in estimation of 99% percentile point and skewness obtained by GEV compared to traditional lognormal approximation (Wilkinson).

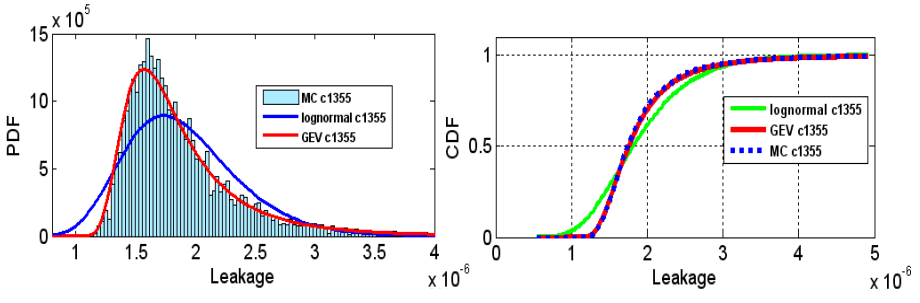



**Fig. 3.** Comparison between PDF and CDF of lognormal and GEV distributions with MC results in 45 nm technology for c1355 circuit

**Table 2.** Improvement in Skewness and 99% Percentile Point Errors obtained by GEV versus Wilkinsons Approaximation

| Benchmark | Our method | | Wilkinson | |
|---|---|---|---|---|
| | Skewness Error | 99% Percentile Point Error | Skewness Error | 99% Percentile Point Error |
| C432 | -8.4% | 2.6% | -54.9% | -7.4% |
| C499 | -8.7% | 1.8% | -36.6% | -4.6% |
| C880 | -11.5% | 1.5% | -48.7% | -6.9% |
| C1355 | -7.0% | 2.1% | -55.5% | -7.1% |
| Ave | -8.9% | 2.0% | -48.8% | -6.5% |

## 4 Conclusion

We proved that lognormal distribution cannot model full-chip leakage power in certain areas of the distribution for scaled technologies. We mentioned that it is due to the inherent skewness in the nature of leakage power which is not taken into account in Wilkinson's method. Then we proposed the idea of using 3rd moment of leakage power in conjunction with Generalized Extreme Value Distribution to obtain better approximations. Superiority of our model in estimation of PDF and CDF of leakage were illustrated in Fig.3. Our model reduces the error in estimation of 99% percentile point from 6.5% to 2% and error in estimattion of skewness reduces from 48.8% to 8.9%.

We are going to improve this model by developing a new method for calculation of skewness of full-chip leakage with less computational complexity. In this paper we only assumed a straightforward method for calculation of skewness using equation 14. However using additive models as suggested by [6] will reduce the computational complexity for calculation of first and second moments. We will further develop the idea of using additive models for calculation of 3rd moment in our future works.

## References

1. Borkar, S., Karnik, T., Narendra, S., Tschanz, J., Keshavarzi, A., De, V.: Parameter variations and impact on circuits and microarchitecture. In: IEEE DAC, pp. 338–342 (2003)
2. Abu-Dayya, A.A., Beaulieu, N.C.: Comparison of methods of computing correlated lognormal sum distributions and outages for digital wireless applications. In: IEEE 44th Vehicular Technology Conference, vol. 1, pp. 175–179 (1994)
3. Chang, H., Sapatnekar, S.S.: Full-chip analysis of leakage power under process variations, including spatial correlations. In: Proc. DAC, pp. 523–528 (June 2005)
4. Rao, R., Srivastava, A., Blaauw, D., Sylvester, D.: Statistical Estimation of Leakage Current Considering Inter- and Intra-Die Process variation. In: International Symposium on Low Power Electronics and Design, pp. 64–67 (2002)
5. Rao, R., Devgan, A., Blaauw, D., Sylvester, D.: Parametric Yield Estimation Considering Leakage Variability. In: Design Automation Conference, pp. 442–447 (2003)
6. Cheng, L., Gupta, P., He, L.: Efficient Additive Statistical Leakage Estimation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28(11) (November 2009)
7. Li, T., Yu, Z.: Statistical analysis of full-chip leakage power considering junction tunneling leakage. In: Proc. DAC, pp. 99–102 (June 2007)
8. Li, X., Le, J., Pileggi, L.: Projection based statistical analysis of full chip leakage power with non-log-normal distributions. In: Proc. DAC, pp. 103–108 (June 2006)

# Using Silent Writes in Low-Power Traffic-Aware ECC

Mostafa Kishani[1], Amirali Baniasadi[2], and Hossein Pedram[1]

[1] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Iran
[2] Electrical & Computer Engineering, University of Victoria, Canada
{mkishani,pedram}@aut.ac.ir, amirali@ece.uvic.ca

**Abstract.** Using Error Detection Code (EDC) and Error Correction Code (ECC) is a noteworthy way to increase cache memories robustness against soft errors. EDC enables detecting errors in cache memory while ECC is used to correct erroneous cache blocks.

ECCs are often costly as they impose considerable area and energy overhead on cache memory. Reducing this overhead has been the subject of many studies. In particular, a previous study has suggested mapping ECC to the main memory at the expense of high cache traffic and energy. A major source of this excessive traffic and energy is the high frequency of cache writes. In this work, we show that a significant portion of cache writes are silent, i.e., they write the same data already existing. We build on this observation and introduce Traffic-aware ECC (or simply TCC). TCC detects silent writes by an efficient mechanism. Once such writes are detected updating their ECC is avoided effectively reducing L2 cache traffic and access frequency.

Using our solution, we reduce L2 cache access frequency by 8% while maintaining performance. We reduce L2 cache dynamic and overall cache energy by up to 32% and 8%, respectively. Furthermore, TCC reduces L2 cache miss rate by 3%.

## 1 Introduction

Memory reliability impacts the overall processor reliability significantly. This is due to the fact that an error in memory can easily propagate to other system elements and influence both data results and control flow. Soft errors, i.e. errors which are transient and change one or more bit values transiently (unlike permanent errors which are steady forever), are the main cause of errors in memory [12, 13].

Cache memories deserve special attention as they are the closest memory layer to the CPU, effectively having the most data exchange. Consequently, any error occurring in the cache memories is likely to propagate into the CPU structures. Moreover, cache memories use SRAM cells and are therefore very susceptible to errors (unlike main memory which uses DRAM cells [14]). Soft error probability has remained the same for SRAM cells [15, 16, 17, 18]. Meantime, cache sizes continue to grow increasing the overall likelihood of soft error in cache memories.

In order to protect the system against such errors, error checking and correcting schemes are used to prevent errors from propagating to other parts of system.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 180–192, 2011.

Exploiting *Errors Detection Codes* (EDCs) and *Error Correction Codes* (ECCs) is one way to achieve error protection. EDCs are often low cost codes such as parity bits capable of detecting single or multiple bit flips in a cache block. Conventionally, EDCs are used to check data integrity on each cache read operation. Compared to EDCs, ECCs (e.g., hamming code) come with higher complexity (and hence overhead) as they require error correction capabilities. Previous work has suggested many solutions to reduce ECC overhead [1, 2, 3, 4, 5, 6]. In particular, a previous study [1], has suggested mapping ECC to the main memory as regular data, referred to also as MMECC. MMECC reduces area and leakage energy overhead at the cost of low performance degradation. On the negative side, MMECC increases both cache traffic and dynamic energy as it increases the frequency of cache write operations (as it writes the ECC value using an extra cache access).

In this paper, we extend previous work and use the observation that a considerable portion of cache writes write the data already existing. We refer to this group of writes as *silent writes*. Conventionally, cache blocks written by a silent write are treated as dirty blocks. This imposes additional unnecessary overhead calculating, updating and rewriting ECC values. As we show in this work detecting and avoiding these redundant computations can save energy and traffic in cache memory.

To this end, we introduce Low-Power Traffic-Aware ECC (TCC). By detecting and skipping conventional steps for silent writes, TCC reduces both cache traffic and energy consumption.

TCC relies on an efficient and effective mechanism to detect silent writes. This is done by comparing low cost signatures associated with each block to find out if a write is in fact silent. To minimize signature cost, we use the already existing parity code as block signature and show that 98% of the non-silent writes can be detected by using parity. In particular we make the following contributions:

- We show that a significant portion of cache writes to the L2 cache are silent writes. We show that in the case of silent write, writing the cache block and updating the associated ECC can be avoided.
- We introduce an efficient mechanism to detect silent writes by using the already existing parity code as block signature. We show that 98% of non-silent writes can be detected by using parity code as signature.
- By skipping ECC calculation and update for silent writes, we reduce cache access frequency (max: 32%), dynamic energy (max: 32%) and miss rate (max: 3%).

The rest of this paper is organized as follows: Section 2 explains related work. Section 3 describes background information including decoupled EDC and ECC. We review our motivating observations in Section 4. In Section 5 we present TCC in more details. We report methodology and results in Section 6 and 7 respectively. Finally in Section 8, we offer concluding remarks.

## 2 Related Work

Li [5] proposed ECC power gating for clean lines to reduce leakage power. Sadler and Sorin [4] used punctured error codes instead of conventional hamming codes for error detection and correction. Punctured code can be separated to EDC and ECC parts. In

order to save cache read latency, they proposed decoupling EDC from ECC and suggested using a dedicated cache called Punctured ECC Recovery Cache (PERC) to hold the ECC part of the code. Meantime, EDC is held in the data cache. In order to reduce area and power, Kim [6] suggested using up to one dirty line per cache set, storing ECC only for this single dirty line.

Yoon and Erez [1] suggested mapping ECC to the memory system (MMECC), instead of storing it at the end of the cache line. MMECC saves cache space by writing ECC in the memory space. This does not impact memory traffic significantly as ECC is only updated when necessary. Meantime, MMECE comes with an additional space overhead in the last cache level. Furthermore, MMECC increases the traffic of the last level cache as a result of regular ECC updates. These consequences could potentially have a negative impact on both power and performance. In this work we show that taking application behavior into account can reduce the cache traffic overhead associated with MMECC.

Lepak and Lipasti [7] introduced the concept of silent stores as store instructions writing the same value already stored.

Zhang [2] proposed In-Cache Replication (ICR) for L1 data caches. ICR uses existing cache space to hold replicas of cache blocks. Kim and Somani [3] proposed parity caching, shadow checking and selective checking.

Protecting the most error prone blocks (as [5] and [6] do) increases cache reliability. This level of protection, however, may not be sufficient for highly reliable and critical applications. TCC does not compromise error correction capability as it stores ECC for all dirty cache blocks uniformly while imposing little area, power and bandwidth overhead. We reduce both the area and the leakage energy overhead associated with ECC by storing the required information in the memory (similar to [1]) rather than a dedicated cache (as [4], [5] and [6] do). Meantime, we improve MMECC [1] by reducing the associated traffic overhead by limiting the L2 cache writes to the writes that are not silent.

## 3 Decoupled EDC and ECC Background

On a cache read operation, EDC is read and calculated to check block integrity to assure correctness. Cache read operations are on the critical path and occur frequently. Therefore, implementing EDC operations efficiently would enhance both the overall performance and energy.

While hamming code has proven to be a viable solution to detect and correct errors in caches, it suffers from significant overhead. Consequently designers have preferred fast and low demanding error detection mechanisms like parity.

It is important to note that ECC is not needed in write-through caches, as there is always an intact copy of the data block available in the upper memory level. In the case of a write-back cache, however, the upper level only includes a copy of the clean data, making dirty blocks that are not yet evicted, vulnerable to errors. To protect write-back caches against errors we use ECC if EDC shows an error. Since correction needs a more powerful code, write-back caches exploit two different codes, a low-power and low-latency code (like parity) for error detection, and a powerful ECC for correction (like hamming). EDC and ECC are calculated and stored on a dirty write-back into the cache.

When EDC detects an error, if the cache line is clean the correct block is read from the higher memory level. Otherwise, ECC is used to produce the correct cache block.

One way to store EDC and ECC is to place them at the end of the cache line. This, while easy for EDC, is not affordable for ECC as ECC comes with significant memory overhead. For example, a cache using a single error correction hamming code requires eight bytes of overhead for a 64-byte block. The associated increase in the cache line size increases the cache area and energy (both dynamic and leakage).

## 4 Motivation

We are motivated by the fact that a significant portion of consecutive writes on the L2 and L1 caches are silent, i.e., they rewrite the same block written previously, consuming energy and bandwidth without contributing to performance. As presented, on average, silent writes account for 37% of L2 cache writes.

In **Fig. 1** we report the share of silent writes (see Section 6 for methodology).

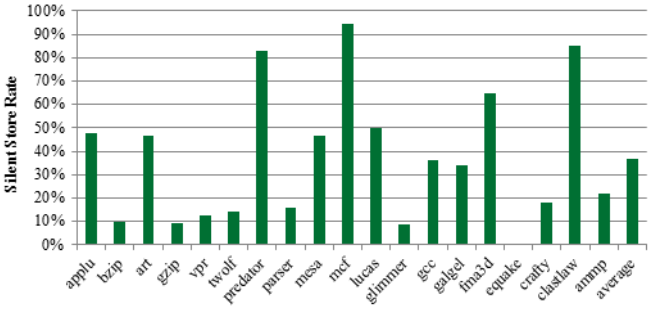



**Fig. 1.** Silent write frequency for L2 cache

In addition, and in a system using MMECC, silent writes result in calculating and rewriting already stored parity and hamming bits hence furthermore wasting activity.

Identifying and avoiding this redundant activity improves energy efficiency in two ways.

- Since previous and new ECCs are equal, calculating the hamming code can be eliminated for silent writes. Note that the XOR operations required to calculate the hamming code are four times more than the XOR operations needed for block comparison.
- In methods like Memory Mapped ECC [1], ECC should be written to memory or data cache on each cache write operation. This could be avoided for silent writes, reducing cache access activity.

## 5 TCC

MMECC requires frequent cache write operations as it writes the ECC associated with each cache block either to the memory or to the cache upon each data cache write. This is

unnecessary as cache writes rewrite the same data frequently as presented in Section 4. We take advantage of this observation and suggest Traffic-Aware ECC, or simply TCC. TCC uses application behavior to reduce cache/memory traffic and eliminate a significant share of the hamming calculations and cache updates performed by conventional ECC solutions.

TCC is applied to a memory system already decoupling error detection from error correction [4] and utilizing memory mapping [1] (requiring treating ECC as a regular data). TCC uses interleaved parity as EDC. ECC is computed and written upon a cache write-back from L1 to L2. Parity comes with one byte overhead and is stored at the end of each cache line in TCC. Hamming, on the other hand, has an eight byte overhead per 64-byte size cache line and is stored in the memory.

TCC allocates an address location to the ECCs and maps the ECC of each cache block to a memory address.

To accommodate ECC, and similar to MMECC, TCC stores 8 bytes of ECC as the block-ECC for the 64-byte cache block used in this study. We group eight block-ECCs to form a memory block of 64 bytes. We refer to the cache blocks associated with each of the block-ECCs stored in one memory block as adjacent blocks. In an 8-way set associative cache, cache lines in the eight consecutive cache sets with the same way number form the adjacent blocks. For instance, cache lines in the way 0 of set0 to set7, are adjacent blocks.

In the case of cache writes, if the ECC line is cached, the new ECC should be stored using an extra cache write. Otherwise, a cache block is assigned to the associated ECC. In case one or more adjacent blocks are dirty, their block-ECC has already been saved in the memory (as we just save ECC for dirty cache blocks). Here we read the already stored ECC block from memory.

### 5.1 Block Comparison

Detecting silent writes needs comparing each ready-to-write block to the old block. This requires reading the old block dissipating power comparable to that of a cache write. For TCC to achieve its goal, it is important to perform this comparison efficiently as suggested in the next section.

**Detecting Silent Writes.** To reduce the overhead associated with silent write detection, we exploit low cost small signatures (explained in the next section) associated with each block rather than comparing the entire block addresses.

To this end we use a small dedicated cache, referred to as the signature cache. The number of signature cache lines is the same as the L1 cache and the line size is one byte. When a data block is written from L2 to L1, its signature is calculated and saved in the signature cache.

At the time of a write back from L1 to L2, the signature of the ready-to-write block is calculated and compared to the old signature saved in the signature cache. If different, the write is not a silent write (i.e. the new and old data blocks are not the same), hence the ready-to-write block should be written to the L2 cache, and the ECC should be computed and rewritten. If the signatures are equal, there is still a chance that the two blocks are unequal. Therefore, TCC takes an extra step comparing the old block (which is read from the L2 cache) to the ready-to-write block. If equal, the write operation is a silent write and no further action is required, as both blocks and their

associated ECCs are equal. Otherwise, the write is not silent. Therefore, the new data block should be written to the L2 cache, and the ECC must be computed and written following the conventional approach. The hardware overhead of comparing the old block to the ready-to-write block is a 64-bit comparator. We take the power overhead associated with this comparator in the energy results presented in Section 7.3. In the next section we present how the signature is calculated.

**Parity as Signature.** The interleaved parity code, which is used for error detection, should be calculated and saved for each cache block. We use the already existing parity bits as signature bits. Using parity as signature has two benefits: a) we no longer need extra effort to calculate signature b) we no longer need extra storage for signature as parity is already stored in the cache block. Fig. 2 shows the steps TCC takes during L2 cache read and writes.

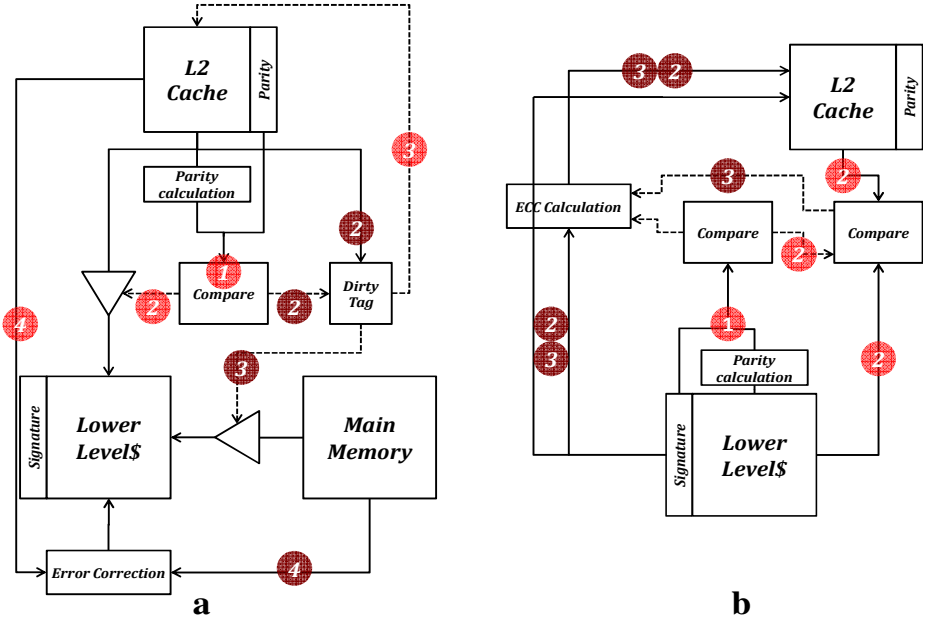



**Fig. 2.** L2 cache read and write mechanisms used in a TCC-enhanced system. (Note that labels with the same number and color happen in parallel. The dashed lines show the control signal) **a:** L2 cache read operation 1) read the data block and the associated parity/signature. Calculate the data block parity independently and compare to the fetched parity. 2) (Lighter label) in the case of a parity match, the block is error-free. Write the block to the lower level cache. 2) (Darker labels) parity mismatch. Check if the block is dirty or not. 3) (Darker label) block is not dirty. Read the correct data from the main memory. 3) (Lighter label) block is dirty. Check the cache for the ECC. 4) (Lighter label) ECC is cached. Read the ECC from cache and correct the data block. 4) (Darker label) ECC is not cached. Read the ECC from the main memory and correct the block. **b:** L2 cache write operation. 1) Read the data block and the associated parity/signature. Calculate its parity and compare to the signature 2) (Darker labels) signature mismatch. So this is a non-silent write, i.e. the ready-to-write block is not equal the previously stored block. Calculate the ECC; write the block and ECC to the cache. 2) (Lighter labels) signature match. Compare current L2 block to the ready-to-write block. If they match, this is a silent write and the write operation is done. 3) (Darker labels) block mismatch. So this is a non-silent write. Calculate the block ECC; write the block and ECC to the cache.

Our study shows that about 98% of the time, unequal data blocks come with unequal associated signatures (i.e., 98% of the time comparing signatures is enough to find out that a write is not silent). Meantime, in 2% of the cases signatures are equal while their associated blocks are different.

In this work we propose using parity as signature assuming that the cache architecture holds parity as EDC for each block while using SEC-DED hamming code as ECC. Meantime, our solution can be used for any ECC type which uses parity as EDC. In the event when parity is not used as EDC, other appropriate data representations could be used as signature instead of parity.

## 6 Methodology

We use Simplescalar 2 [8] to evaluate our solutions. We execute 500M representative instructions from SPEC2000 [9] using SimPoint [11]. In order to estimate cache dynamic and leakage energy, we use CACTI 6 [10] tool.

**Table 1.** CPU Configuration

| Processor component | Value |
|---|---|
| Integer Functional Unit | 4 ALU, 1 Multiplier/Divider |
| Floating Point Functional Unit | 4 ALU, 1 Multiplier/Divider |
| Instruction Fetch Queue/LSQ/RUU size | 4/32 / 64 Instructions |
| Decode/Issue/Commit Width | 4 / 4 / 4 instructions |
| Memory Latency | First Chunk 512 cycle/Inter Chunk 128 cycle |
| Memory System Ports (to CPU) | 2 |
| Branch Predictor | Comb: 1024 meta size,<br>Bimodal: 2048,<br>2level: 8 bits history and 1024 arra<br>BTB: 512, 4-way |
| Mis-prediction Latency | 3 cycle |
| Level 1 Instruction Cache | 32KB / 3cycles access latency / 4 ways / 64 bytes per block / LRU |
| Level 1 Data Cache | 64KB/3cycles access latency / 4 ways / 64 bytes per block / LRU |
| Level 2 Instruction Cache | 256KB / 12cycles access latency / 4 ways / 64 bytes per block / LRU |
| Level 2 Data Cache | 1MB/12cycles access latency / 8 ways / 64 bytes per block / LRU |

Table 1 shows the system configuration used in this study. We assume that both L2 and L1 caches have same block size [20, 21, 22]. Note that we take into account all extra tasks contributing to energy consumption in TCC and MMECC when comparing to a conventional processor. These tasks include:

- Signature read and write
- Signature comparison
- Data block read and write
- Data block comparison

In the conventional method, ECC is stored along the data block, so the length of the data block is more than TCC. Moreover, we assume that parity is stored at the end of the cache block and is read by each cache read operation.

We calculated the cost of both data and signature read and update using CACTI tool. Meantime, we use synopsys HSPICE tool [19] for 45nm technology to estimate block and signature comparison cost.

## 7 Results

In this section we report experimental results. We report performance in Section 7.1. We present TCC impact on cache access frequency in Section 7.2. In Section 7.3 we report energy consumption. Finally, we present how TCC impacts cache miss rate in 7.4. To provide better understanding we compare TCC to MMECC [1] and a conventional system.

### 7.1 Performance

In **Fig. 3** we report performance for TCC and MMECC compared to a conventional system storing ECC entirely in a dedicated cache array space.

As presented in **Fig. 3**, TCC shows competitive performance compared to MMECC. While both methods show slight performance loss compared to the conventional system, TCC seems to better maintain performance. On average, TCC and MMECC show 0.07% and 0.06% performance loss compared to the conventional system.

This performance loss has two reasons. First, unlike the conventional system that stores both ECC and the cache block in the same cache access, MMECC and TCC require an additional cache access to store ECC. Both MMECC and TCC store the data, compute the ECC and then store the ECC in the cache space. This results in extra traffic which can impact performance. Second, we assume that MMECC and TCC use cache space to store ECC while the conventional system stores ECC in a separate array. Consequently, TCC and MMECC increase cache miss rate slightly (see Section 7.4) which has a negative impact on performance.

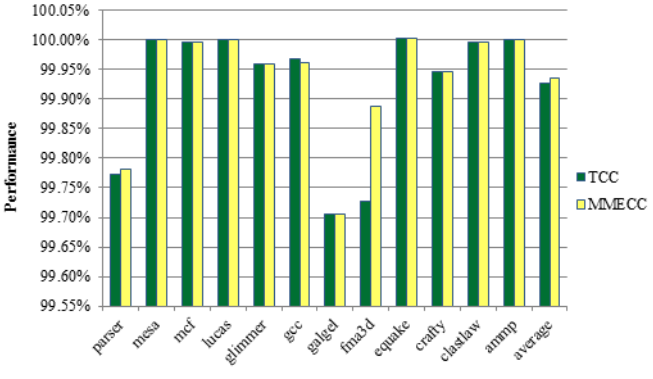



**Fig. 3.** Relative Performance

### 7.2 Cache Access

In **Fig. 4** we report the number of L2 data cache accesses for TCC and MMECC compared to the conventional system. On average, TCC and MMECC show 21% and 32% cache access frequency increase compared to the conventional system respectively. As TCC updates ECC by an extra L2 cache access (except when the new block is equal to previous one), its L2 access is higher than conventional system. TCC shows less L2 access frequency compared to MMECC as we avoid writing the data block and the associated ECC in the case of a silent write. On average, TCC shows 8% (up to 32%) L2 access reduction compared to MMECC. The amount of access frequency difference between TCC and MMECC depends on how often silent writes occur. For example in equake, bzip and glimmer, TCC and MMECC accesses are almost equal. This is consistent with **Fig. 1** where these three benchmarks show a low rate of silent writes.

On the other hand, TCC accesses in clastlaw and mcf is much less than MMECC as silent write frequency is high for these two benchmarks.

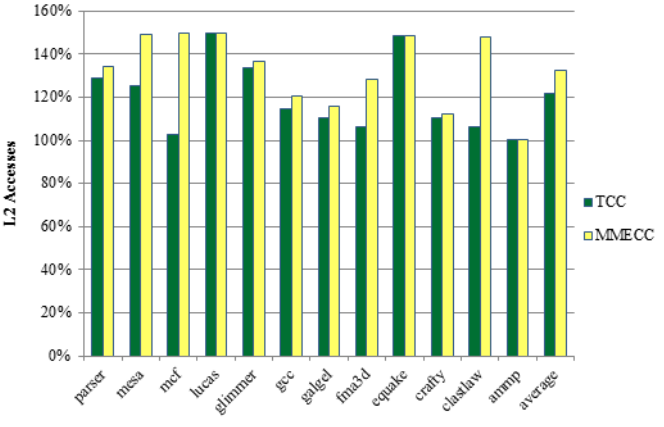



**Fig. 4.** L2 cache accesses frequency

### 7.3 Energy

In this section we report energy consumption for L2 data cache for TCC and MMECC compared to the conventional system. We report dynamic and total energy in two following sections respectively.

**Dynamic Energy.** In Fig. 5 we report L2 dynamic energy consumption for TCC and MMECE compared to the conventional system. MMECC shows an average increase of 8% (max: 22%) in energy consumption. Meantime, TCC shows an average reduction of 1% (max: 19%) and 9% (max: 32%) compared to the conventional and MMECC respectively. This is explained as follows. As shown in Section 7.2 MMECC shows higher cache access compared to the conventional system increasing cache dynamic energy. For TCC, L2 access frequency is higher than the conventional

method. However, the conventional method stores ECC at the end of each cache block increasing cache energy consumption per access. On the other hand, TCC's cache access frequency is lower than MMECC. Therefore, TCC consumes less dynamic energy compared to MMECC.

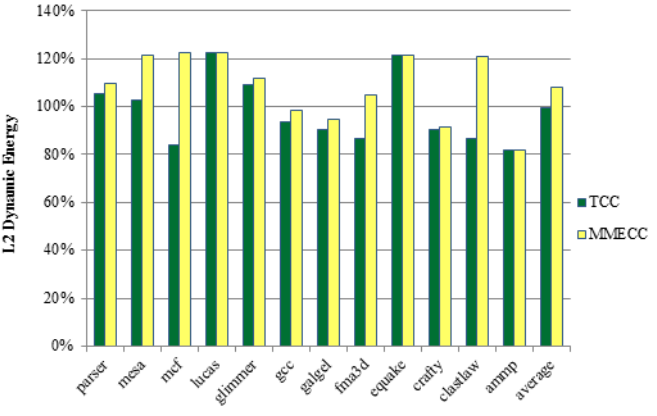



**Fig. 5.** Relative L2 data cache dynamic energy

**Total Energy.** In Fig. 6 we show L2 data cache total energy consumption for TCC and MMECC compared to the conventional system. We measure total energy as the summation of leakage and dynamic energy. As TCC and MMECC do not use a dedicated cache space to store ECC, they show 13% less leakage energy consumption compared to the conventional system. Meantime, TCC shows only 0.06% higher leakage energy compared to MMECC as TCC uses extra space to store signatures. Since TCC has low dynamic energy consumption (see Dynamic Energy Section), it shows 1% (up to 8%) and 12% (up to 13%) reduction in L2 data cache total energy compared to MMECC and conventional system respectively.

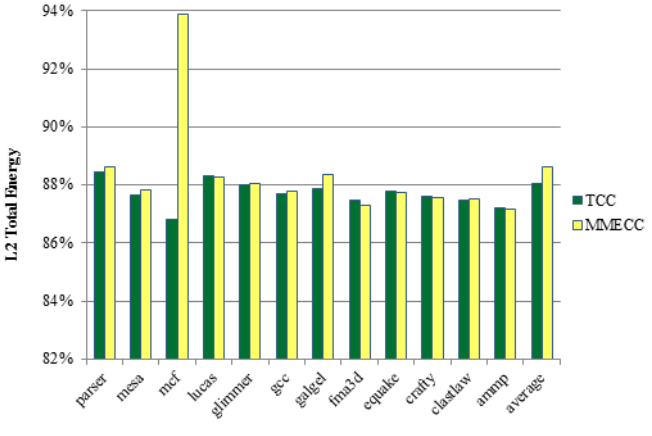



**Fig. 6.** Relative L2 data cache total energy consumption

### 7.4 Miss Rate

In **Fig. 7** we report L2 data cache miss rate for TCC and MMECC compared to the conventional system. TCC and MMECC show higher miss rate compared to the conventional method as they use some cache space to store ECC rather than using a dedicated cache. TCC shows 0.15% (up to 3%) less cache miss rate compared to MMECC. This reduction could be explained as follows. Both TCC and MMECC store ECC only for dirty cache lines. This results in an increase in the cache space occupied by ECC as the number of dirty cache lines increases. Meantime, TCC treats silent writes as clean writes as these writes do not change the value of the cache line. As TCC does not save ECC for these cache lines, ECC occupies less cache space for TCC compared to MMECC. TCC's miss rate is 0.7% (up to 3%) higher than the conventional system. MMECC's miss rate is 0.85% (up to 6%) more than conventional system. Note that the conventional system uses a dedicated cache array to store ECC.

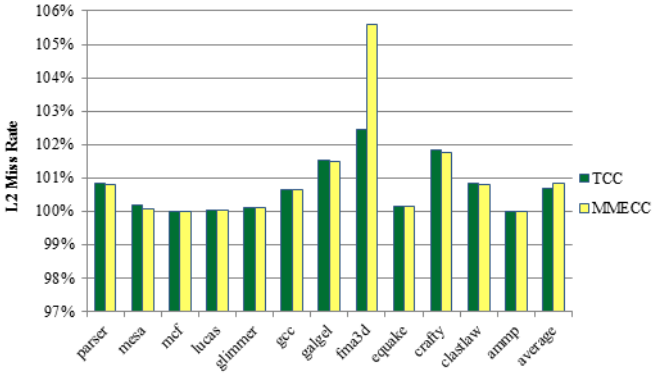



**Fig. 7.** L2 data cache miss rate

### 7.5 L1 Cache Size and Performance

In **Fig. 8** we show how variations in L1 data cache size impacts performance for TCC and MMECC compared to the conventional system. TCC shows better performance compared to MMECC for both 32KB and 128KB L1 data cache sizes.

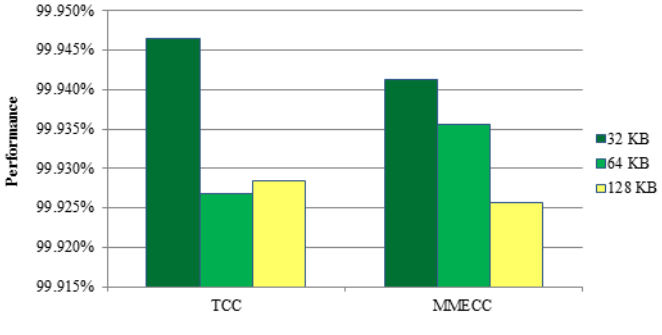



**Fig. 8.** Performance sensitivity to L1 data cache size

### 7.6 L1 Cache Size and Total Energy

In **Fig. 9** we report how variations in L1 data cache size impacts total energy for TCC and MMECC compared to the conventional system. For all L1 data cache sizes, TCC shows lower total energy compared to both MMECC and conventional system.

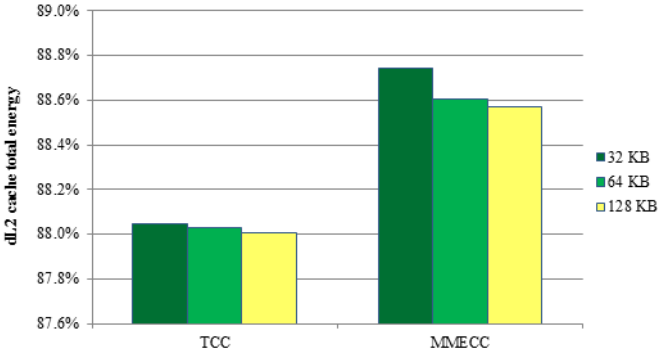



**Fig. 9.** Total energy sensitivity to L1 data cache size

## 8 Conclusion

We showed that a significant portion of cache writes are silent, i.e., they write the data already existing in the cache. Based on this observation, we proposed Low-Power Traffic-Aware ECC to reduce cache traffic and access frequency. To detect silent writes, we use parity bits as signature bits and propose an efficient detection method. TCC avoids cache block write as well as ECC calculation and ECC write for silent writes. We consider the overhead of silent write detection in our experimental results and show that by taking silent writes into account, we improve both cache traffic and energy while maintaining performance.

## References

[1] Yoon, D.H., Erez, M.: Memory mapped ECC: low-cost error protection for last level caches. In: International Symposium on Computer Architecture (ISCA), pp. 116–127 (2009)

[2] Zhang, W., Gurumurthi, S., Kandemir, M., Sivasubramaniam, A.: ICR: In-Cache Replication for Enhancing Data Cache Reliability. In: International Conference on Dependable Systems and Networks, DSN (June 2003)

[3] Kim, S., Somani, A.K.: Area Efficient Architectures for Information Integrity in Cache Memories. In: International Symposium on Computer Architecture, ISCA (May 1999)

[4] Sadler, N.N., Sorin, D.J.: Choosing an Error Protection Scheme for a Microprocessor's L1 Data Cache. In: International Conference on Computer Design, ICCD (October 2006)

[5] Li, L., Degalahal, V.S., Vijaykrishnan, N., Kaandemir, M., Irwin, M.J.: Soft Error and Energy Consumption Interactions: A Data Cache Perspective. In: International Symposium on Low Power Electronics and Design, ISLPED (August 2004)

[6] Kim, S.: Area-Efficient Error Protection for Caches. In: Conference on Design Automation and Test in Europe, DATE (March 2006)
[7] Lepak, K.M., Lipasti, M.H.: Silent stores for free. In: ACM/IEEE International Symposium on Microarchitecture, pp. 22–31 (December 2000)
[8] Burger, D., Austin, T.M.: The SimpleScalar tool set, version 2.0. ACM SIGARCH Computer Architecture News 25(3), 13–25 (1997)
[9] http://www.spec.org/cpu2000/
[10] Muralimanohar, N., Balasubramonian, R., Jouppi, N.P.: Architecting Efficient Interconnects for Large Caches with CACTI 6.0. MICRO 28(1), 69–79 (2008)
[11] Perelman, E., Hamerly, G., Biesbrouck, M.V., Sherwood, T., Calder, B.: Using SimPoint for Accurate and Efficient Simulation. In: ACM SIGMETRICS the International Conference on Measurement and Modeling of Computer Systems (June 2003)
[12] Karlsson, J., Ledan, P., Dahlgren, P., Johansson, R.: Using heavy-ion radiation to validate fault handling mechanisms. MICRO, 8–23 (February 1994)
[13] Sosnowski, J.: Transient fault tolerance in digital systems. MICRO, 24–35 (February 1994)
[14] Scheick, L.Z., Guertin, S.M., Swift, G.M.: Analysis of Radiation Effects on Individual DRAM Cells. IEEE Transact. Nucl. Sci. 47(6), 2534–2545 (2000)
[15] Standards, J.: JESD89 Measurement and Reporting of Alpha Particles and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, JESD89-1 System Soft Error Rate (SSER) Method and JESD89-2 Test Method for Alpha Source Accelerated Soft Error Rate (2001)
[16] Gordon, M.S., Rodbell, K.P., Heidel, D.F., Cabral Jr., C., Cannon, E.H., Reinhardt, D.D.: Single-event-upset and alpha-particle emission rate measurement techniques. IBM Journal of Research and Development 52(3) (May 2008)
[17] Maiz, J., Hareland, S., Zhang, K., Armstrong, P.: Characterization of Multi-Bit Soft Error Events in Advanced SRAMs. In: IEEE International Electron Devices Meeting, IEDM (December 2003)
[18] Slayman, C.W.: Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations. IEEE Trans. on Reliability 5(3), 397–404 (2005)
[19] http://www.synopsys.com/tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx
[20] Kalla, R., Balaram, S., Tendler, J.M.: IBM Power5 chip: a dual-core multithreaded processor. MICRO, 40–47 (March-April 2004)
[21] http://www.amd.com/us/products/desktop/processors/athlon/Pages/AMD-athlon-processor-for-desktop.aspx
[22] http://software.intel.com/en-us/articles/recap-virtual-memory-and-cache/

# SWAT: Simulator for Waveform-Accurate Timing Including Parameter Variations and Transistor Aging

Christoph Knoth, Carsten Uphoff, Sebastian Kiesel, and Ulf Schlichtmann

Institute for Electronic Design Automation, Technische Universität München
http://www.eda.ei.tum.de/

**Abstract.** This paper presents SWAT, a highly optimised statistical timing analyser for digital circuits that combines transistor-level analysis accuracy with gate-level analysis performance. It is based upon a current source model (CSM) for logic cells which considers transistor aging and process variation. Static timing analysis is performed using a very accurate waveform model. SWAT employs waveform truncation and dedicated solvers to significantly improve analysis performance without noticeable loss of accuracy. Parameter variations and aging can be handled both by Monte Carlo simulations and by a special sensitivity propagation mode, which expresses arrival times as a function of local and global parameter variations. Simulation times for ISCAS85 circuits are less then 2s for nominal and less than 28s for sensitivity mode.

## 1 Introduction

Timing analysis for sign off is a crucial and time consuming step in the design flow. It requires very accurate models for logic cells to represent nonlinearities of input capacitance and cell delay. Detailed models must be used for metal interconnect. While the nonlinear delay model (NLDM) provides sufficient accuracy in earlier design stages with rough interconnect estimations, it has problems in accurately predicting cell delays in the presence of these resistive, strongly coupled interconnects. New current source models (CSMs) such as CCS and ECSM are proposed by EDA vendors to replace NLDM as they account more naturally for the current source behaviour of transistors than voltage sources do for driver modelling [11, 2]. They produce precise output voltage waveforms from current waveforms which are stored in lookup tables indexed by signal slew and output load. However, this requires to reduce a precise waveform to a simple ramp signal at inputs of cells, resulting in severe timing errors [7]. CSMs proposed by academia are different. They provide a cell's port current as a function of its port voltages. Therefore, these CSMs support arbitrary input waveforms and arbitrary loads. Moreover, they are applicable to multi driver situations like timing analysis of clock meshes [13]. While many different CSMs have been proposed, there are hardly any reports about these models actually being used for designing ICs. One reason might be insufficient accuracy for a wider range of logic

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 193–203, 2011.

cells or weak simulation performance due to complex CSMs. Another reason are complex and time consuming characterisation schemes that are required for most approaches. This problem is magnified by the need for parametric CSMs to model the impact of parameter variations on cell delay. [13] showed a CSM simulator for clock mesh simulation. Therefore, only buffers and inverters were used. [5] presented the integration of CSMs in commercial tools like Spectre and UltraSim. [15] and [12] addressed statistical analysis. [15] used a very simple CSM with constant capacitors to derive an analytical expression for the variational waveform. [3] used a more complex CSM. Timepoints of crossing voltage levels are modelled as linear or quadratic functions of process parameters. [12] calculated variation waveforms based on simplified transistor models. All three showed only small examples consisting of one or a few logic cells. [9] presented a fast statistical timing simulator using simplified transistor models. The key component is an efficient threading algorithm, allowing effective parallelising.

For advanced technology nodes temporal performance degradation due to aging becomes more severe. Most work for these aging effects is done at transistor level [4]. Some research is done for NLDM but no approach has yet been presented for CSMs.

This paper has two contributions. First, to the best of our knowledge, this is the first time a CSM including aging effects is presented. Second, a very efficient statistical waveform simulator with integration to commercial tools is presented.

## 2 Transistor Aging

Quicker scaling of feature sizes than supply voltage led to higher field strength which causes transistor performances to degrade over lifetime [4]. Currently the dominating aging effect is negative bias temperature instability (NBTI). Its extent depends on various stress conditions like switching activity, temperature, or negative biasing. At transistor level, NBTI is modelled as an increase in threshold voltages of PMOS transistors (Fig. 1). Its value is calculated using vendor specific degradation equations such as (1). The required values for operating conditions (OC) and workload (WL) over time are obtained from an inhouse aging tool.

$$\Delta v_{th} = f(OC, t, WL) \tag{1}$$

Aging aware timing models capture the impact of this drift on cell delay. For timing analysis with CSMs, this transistor degradation must be propagated to the CSM model components. It then influences every point of the waveform resulting in a more precise delay estimation.

## 3 Aging Aware Current Source Model

Many CSM approaches lack in efficient characterisation strategies that allow to model transistor level phenomena. Most often, a plethora of time consuming transient simulations is needed. The approach described in [6] circumvents

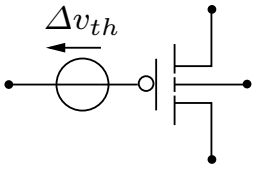



**Fig. 1.** $\Delta v_{th}$ modelling for NBTI

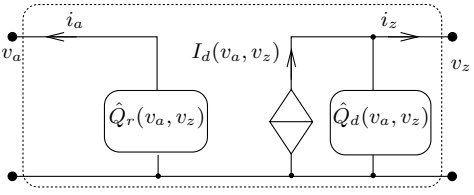
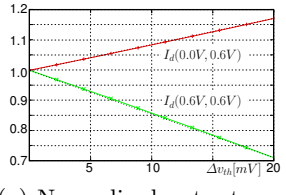



**Fig. 2.** CSM for one inverting stage

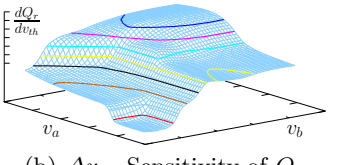
this burden and relies on quick DC simulation only. The CSM components of Fig. 2 are based upon their physical sources in the original transistor netlist. That means, the port charges $Q_{r,d}$ and port current $I_d$ explicitly capture the contributions of every circuit element to currents flowing into the ports of the logic cell. Logic cells like buffers with multiple inverting stages are modelled by abutted CSMs of Fig. 2. As presented in [5], this CSM also captures the influence of parameter variation as well as static Vdd drop. Each CSM component is equipped with a nominal lookup table (LUT) and LUTs containing linear sensitivities.

In this work, this CSM has been taken as a basis for a new method to support timing analysis of degraded circuits. Analogous to [5], the degradation of model components is derived from the component degradation in the original logic cell.

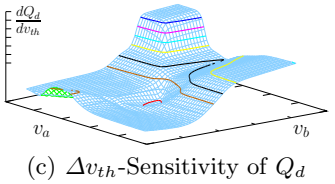
The aging analysis of Sec. 2 provides a drift in threshold voltage for PMOS transistors in the inverting stage. The actual value of $\Delta v_{th}$ is unknown during CSM generation. Hence, a parametric CSM has to be generated. Consistent to modelling parameter variation, another LUT is characterised capturing the



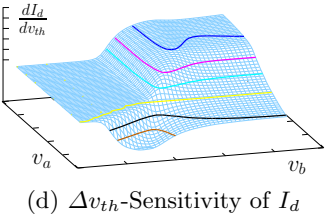

(a) Normalised output current for different NBTI drifts and operating points




(b) $\Delta v_{th}$-Sensitivity of $Q_r$




(c) $\Delta v_{th}$-Sensitivity of $Q_d$




(d) $\Delta v_{th}$-Sensitivity of $I_d$

**Fig. 3.** Sensitivities of CSM components w.r.t. $\Delta v_{th}$

sensitivity of model components $I_d$ and $Q_{r,d}$ w.r.t. $\Delta v_{th}$. This is achieved by the following sequence. After running the nominal SPICE simulations to obtain the LUTs for $I_d$ and $Q_{r,d}$, the model parameters of all transistors are modified automatically. Their threshold voltages are deflected by some reasonable amount $v_{th} = v_{th} + \Delta v_{th}$. The precise value of $\Delta v_{th}$ is not crucial since degradation is almost linear. Figure 3(a) shows $I_d$ as a function of $\Delta v_{th}$ for different operating points. With this modification made to the original cell description, CSMs characterisation is re-run to obtain linear sensitivities of model components w.r.t. $\Delta v_{th}$. These are stored in LUTs as shown in Figs. 3(b)-3(d). Since the modelling approach requires only DC simulations, this procedure is not time consuming.

## 4 CSM Simulator 'SWAT'

A special timing simulator employing this variation and aging aware CSM has been developed. Written in C++, it is equipped with shell and scripting interface and accesses the designs through Cadence OpenAccess design database. The simulator exploits the CSM properties to provide various kinds of analyses. At heart, it performs a block-based static timing analysis (STA) which models waveforms as voltage-time-pairs and uses CSMs for driver and receiver modelling (Fig. 4). It allows to incorporate interconnect models at different levels of abstraction. Parameter variation, local supply voltage drop, and aging degradation are supported.

Calculating the voltage waveforms for each net of the design is done by solving modified nodal analysis (MNA) equations, which labels the simulator as a SPICE-like tool. However, to cope with the large dimensions of digital circuits, three measures are applied: using simplified cell models, exploiting temporal and spatial latency, and using optimised solvers.

The CSM is explained in Sec. 3. It reduces simulation time by avoiding expensive transistor model evaluations. It also reduces the number of nodes in the equation system. The second measure is to exploit latency in digital circuits, where signals traverse from the inputs to the outputs. Similar to block-based STA, each cell is evaluated separately once all its input waveforms are available.

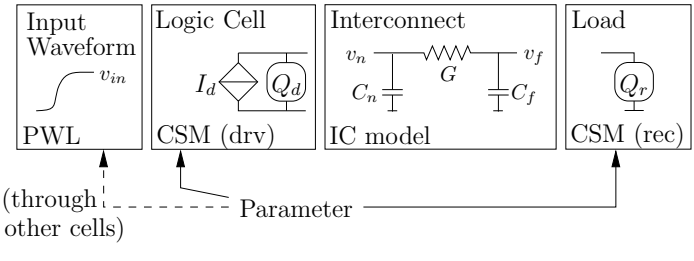



**Fig. 4.** Typical simulation scenario

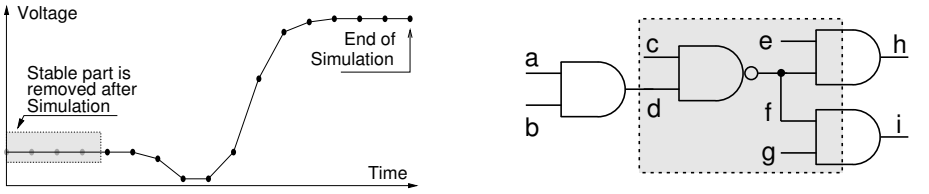



**Fig. 5.** Waveform truncation and spatial latency

The output waveforms are calculated for every cell input and the one with latest/earliest arrival time is propagated to the next cell. To obtain the cell delay, only the shape of signal transition matters. Consequently, there is no need to simulate the entire voltage waveform for every net. Instead, the waveform is truncated by eliminating points with constant voltage prior to the signal transition and the corresponding time interval is added to the arrival time. Similarly, simulation is stopped when the output signal is stable. These two measures, visualised in Fig. 5, decompose a very large simulation task into a large number of small and short transient simulations. It is therefore crucial to have efficient methods to calculate the output waveform of a single cell. This issue is addressed by the third measure, the optimised solvers.

Normal SPICE simulators construct an equation system based on the incidence matrix of circuit elements. Every equation enforces the sum of currents flowing into a node to be zero. Static and dynamic checks are performed to avoid numerical problems and instability [14,8]. The generality of such general purpose simulators comes at the cost of additional CPU time for calling general matrix solvers and dynamic time step control. In the presented approach of SWAT using CSMs, valuable circuit information is provided and exploited to reduce simulation time significantly. Due to CSM modelling and cell-based analysis, the circuit structure is generally known and shown in Fig. 4. A cell, consisting of several stages, is excited by a PWL voltage source which models the output waveform of the preceding cell. Depending on the interconnect model, the output load to drive will be one of the following cases:

*A*) Pure grounded capacitance
*B*) CRC-Pi Model with near and far capacitance
*C*) CRC-Pi Model with near and far capacitance and coupling capacitors to other nets
*D*) Dense matrix from Model Order Reduction
*E*) RC mesh including coupling to other nets.
*F*) An interconnect model as in *A*)-*E*) and a separate nonlinear receiver model for the logic cells.

When a load model like *E*) is used there is no information about the circuit until the simulation is conducted. Hence, general algorithms to construct and solve circuit equations must be used. For the other cases, circuit topology and matrix structure are known prior to the simulation. This information is used

in SWAT to reduce the overhead of general SPICE simulators. It has also been observed that the vast majority of cell models consists of one or two stages.

Consequently, the simulator provides optimised solvers for those cases to achieve optimal performance. They are hard coded C functions that implement the solution for the resulting circuit equations. All remaining gates and complex interconnect structures are dealt with a general matrix solver [1].

Simulating a single-stage cell (e.g. inverter) with a Pi wire model as in Fig. 4 results in the 3x3 nonlinear differential algebraic equation system of (2).

$$\boldsymbol{f}(\boldsymbol{v}(t+1,\boldsymbol{p}),\boldsymbol{p},t) = \begin{bmatrix} v_{in} - v_{prev} \\ I_d(v_{in},v_n) + G\cdot(v_n - v_f) + \frac{d}{dt}\left(C_n v_n + Q_d(v_n,v_{in})\right) \\ G\cdot(v_f - v_n) + \frac{d}{dt}\left(C_f v_f + Q_r(v_f,v_{next})\right) \end{bmatrix} \tag{2}$$

$v_{next}$ is the constant voltage of the node following the receivers. The value is known since every CSM models an inverting stage. $v_{in}$ is identical to the output waveform of the previous cell but must be part of the equation system to model the dynamic influence of the input on the output. The nonlinear algebraic equations are solved for near and far output node, $v_n$ and $v_f$, iteratively by using Newton-Raphson method (3-4).

$$\boldsymbol{J}\cdot\Delta\boldsymbol{v} = \boldsymbol{J}\cdot\begin{bmatrix}\Delta v_{in}\\ \Delta v_n\\ \Delta v_f\end{bmatrix} = -\boldsymbol{f}(\boldsymbol{v}) \quad \text{and} \quad v_i = v_i + \Delta v_i \tag{3}$$

with

$$\boldsymbol{J} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{dI_d}{dv_{in}} + \frac{d}{dt}\frac{dQd}{v_{in}} & \frac{dI_d}{dv_n} + G + \frac{d}{dt}\left(C_n + \frac{dQd}{dv_n}\right) & -G \\ 0 & -G & G + \frac{d}{dt}\left(C_f + \frac{dQ_r}{dv_f}\right) \end{bmatrix} \tag{4}$$

Solving (3) for the Newton correction vector $\Delta\boldsymbol{v}$ is required more than once for every timepoint of the waveform. Its solution of (5) therefore exploits structural constants of (4) and (2) to obtain utmost efficient C-Code.

$$\Delta\boldsymbol{v} = \begin{bmatrix} -r_1 \\ (J_{33}\cdot J_{21}\cdot r_1 - J_{33}\cdot r_2 + r_3\cdot J_{23})/(-J_{32}\cdot J_{23} + J_{22}\cdot J_{33}) \\ -(J_{32}\cdot J_{21}\cdot r_1 - J_{32}\cdot r_2 + J_{22}\cdot r_3)/(-J_{32}\cdot J_{23} + J_{22}\cdot J_{33}) \end{bmatrix} \tag{5}$$

Within the code, the amount of calculations is small and hence efficient memory access is crucial. Usually general purpose solvers are optimised for larger matrices, resulting in large overhead for this tiny system. Instead, employing the circuit information allows to use very simple and fast data structures. Table 1 compares the time needed for updating the numerical values in the linear system and solving. Using the tailored solver speeds up this crucial part for the overall performance by a factor of 26.

The accuracy of SWAT is controlled by setting the fixed timestep and tolerances for system variables. If latter are violated, the timestep is reduced for the entire waveform.

A common approach to reduce the computational cost of SPICE-like simulations is to use simplified or constant Jacobians [10]. This skips the computation of derivatives at the expense of more iterations to converge. This method is not employed in SWAT for two reasons. First, fixed timesteps and the small dimensions result in quick convergence. Second, for analyses, which are discussed later in this section, the exact derivatives are needed.

**Table 1.** Time needed to update and solve the linear system

| | |
|---|---|
| Optimised Solver | 0.56 s |
| General Solver | 14.69 s |

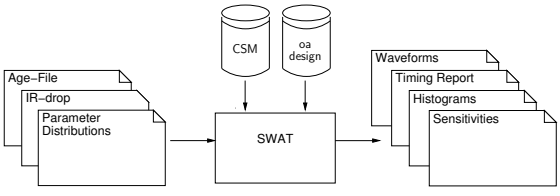



**Fig. 6.** Simulator use cases

Additionally to STA, SWAT is capable of analysing the impact of parameter variation. As shown in Fig. 6, information on local IR drop, local temperatures, $v_{th}$-drifts from the aging analysis, and distributions of local and global process parameters can be supplied to the tool. It then generates delay histograms from Monte Carlo simulations and/or timing reports of the aged circuit.

As an alternative to this re-running of STA with varied parameters, SWAT also provides a sensitivity mode. Here, the influence of parameter variations is determined within a single simulation by solving the sensitivity network. For example, this allows to quantify the impact of local vdd drops or aged cells on the path delays. In this mode each arrival time is a triplet of a nominal value and vectors containing its linear sensitivities w.r.t. parameter variations.

$$T_{arr} = (T^n_{arr}, \boldsymbol{\gamma}, \boldsymbol{\lambda}) \tag{6}$$

$\boldsymbol{\gamma}$ contains the sensitivities w.r.t. the global parameters. $\boldsymbol{\lambda}$ accumulates sensitivities w.r.t. parameters affecting each cell individually. Obviously its size increases with every cell in the path, requiring efficient data structures and algorithms. The sensitivities of arrival times in (6) are derived quantities. Primarily, a variation of parameter $p$ results in a voltage change for each timepoint. Hence, at first the voltage sensitivities $\nu_p(t) = {}^{dV}/_{dp}$ are calculated. The sensitivity of the arrival time, $\tau_p$, is calculated according to (7) once the output waveform is completed.

$$\tau_p = \frac{dT_{arr}}{dp} = -\frac{dV}{dp} \cdot \frac{dt}{dV}\bigg|_{V=0.5V_{dd}} \tag{7}$$

$\tau_p$ is used to shift the entire waveform. Hence, the sensitivity of every waveform entry is modelled as a shift in time ($\tau_p$) and a voltage change ${}^{dV}/_{dp}$. Since $\tau_p$

already expresses some sensitivity of each timepoint, the calculated voltage sensitivity for each waveform entry is modified in (8) to obtain the final voltage sensitivity $\nu_p$.

$$\nu_p(t) = \frac{dV}{dp} - \left(-\frac{dV}{dt} \cdot \tau_p\right) \tag{8}$$

During simulation, the voltage and time sensitivities are calculated and propagated. The variation of the arrival time finally results from the sum of the products of $\boldsymbol{\gamma}$ and $\boldsymbol{\lambda}$ with their parameter distributions.

To compute the output waveform sensitivities, direct sensitivity computation is used [8]. It is more efficient than the adjoint network method since the number of measurements (i.e. points in output waveform) is large compared to the number of parameters. The sensitivity network is constructed by differentiating the circuit equations (2) w.r.t. the parameter. Original and sensitivity network use the same Jacobian matrix (4) and only the right hand side of (3) must be updated. This avoids a large amount of re-computation and memory accesses. Most importantly, the optimised code for solving the linear equations can be reused. The new vector contains the sensitivities of CSM components w.r.t. the parameter which are interpolated from the LUTs. As Fig. 4 shows, at each cell new waveform variation is introduced and variation of the input waveform is propagated. Calculating the sensitivity of the output waveform w.r.t. voltage changes of the input waveform employs the same algorithm. Since the current cell is not affected by local parameters of its predecessors, first the sensitivity w.r.t. the input waveform, $^{dV_{out}}/_{dV_{in}}$, is calculated. Thereafter, the sensitivity of the input waveform, $^{dV_{in}}/_{dp}$, is propagated to the output by

$$\frac{dV_{out}}{dp} = \frac{dV_{out}}{dV_{in}} \cdot \frac{dV_{in}}{dp} \tag{9}$$

## 5 Results

Before analysing the simulator, the correctness of the aged CSM is verified by testing each gate individually against SPICE transistor simulation. Figure 7 plots the normalised delays of various types of logic cells for different slope-load combinations. Each time, the simulation is done for the new and the aged cell. All simulations are repeated using CSMs and led to an average delay error of 1%.

Table 2 compares arrival times of an inverter chain simulated in SWAT against the transistor netlist in SPICE. For this, and for all other simulation, SWAT uses a time step of 1ps. The nominal delay values have errors less than 1%. The errors of sensitivities w.r.t. to channel length, $L$, and oxide thickness, $TOX$, mainly result from the CSM LUT-approximation, not from the sensitivity calculation.

Table 3 analyses the performance gains of waveform truncation and optimised linear solvers. Column 2 lists the simulation times using the general solver and no waveform truncation. By only simulating the waveform transitions, simulation times are reduced by 5X. This value depends on the number of skipped stable points. Hence, the longer the path, the higher the speedup. Using the optimised

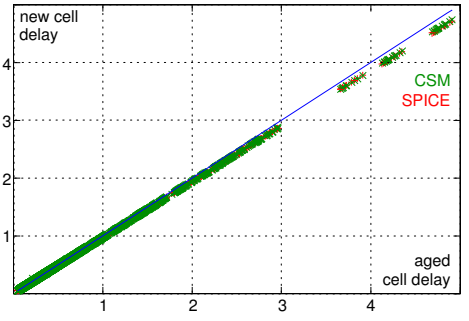



**Fig. 7.** Cell delay prediction for new and aged cells using CSMs or SPICE

**Table 2.** Comparison of SWAT and SPICE arrival times

| | $T_{arr}$ [1e-10 s] | | | $^{dT}/_{dL}$ [1e-11] | | | $^{dT}/_{dTOX}$ [1e-12] | | |
|---|---|---|---|---|---|---|---|---|---|
| Net | SPICE | SWAT | Err [%] | SPICE | SWAT | Err [%] | SPICE | SWAT | Err [%] |
| 5 | 4.01 | 3.98 | 0.70 | 1.98 | 1.94 | −1.82 | 3.42 | 3.36 | −1.87 |
| 4 | 3.18 | 3.15 | 0.76 | 1.64 | 1.57 | −4.06 | 2.60 | 2.59 | −0.52 |
| 3 | 2.49 | 2.47 | 0.72 | 1.17 | 1.12 | −3.76 | 2.14 | 2.11 | −1.57 |
| 2 | 1.59 | 1.58 | 0.79 | 0.75 | 0.72 | −4.27 | 0.13 | 1.24 | −0.92 |
| 1 | 0.90 | 0.90 | 0.22 | 0.30 | 0.29 | −2.95 | 0.80 | 0.78 | −2.26 |

**Table 3.** Simulation speedups due to waveform truncation (WT) and optimised solvers (OS) compared to the general solver (GS)

| | Simulation time [s] | | | | Speedup | | |
|---|---|---|---|---|---|---|---|
| Ciruit | GS, no WT | GS, WT | OS, no WT | OS, WT | WT | OS | OS, WT |
| c880 | 6.7 | 1.6 | 1.4 | 0.4 | 4.1 | 4.9 | 18.0 |
| c1355 | 9.2 | 2.0 | 1.6 | 0.4 | 4.7 | 5.9 | 24.1 |
| c1908 | 14.7 | 2.7 | 2.7 | 0.6 | 5.5 | 5.4 | 24.1 |
| c3540 | 29.0 | 5.5 | 6.2 | 1.3 | 5.3 | 4.7 | 22.5 |
| c6288 | 52.0 | 8.3 | 8.7 | 1.5 | 6.3 | 6.0 | 34.0 |

solvers accelerates solving one timestep. It further reduces simulation times by about 5X. Together, both methods lead to performance gains of 18-34X without sacrificing accuracy.

Table 4 shows simulation times for sensitivity mode and for 200 Monte Carlo (MC) simulations for different circuits. They are influenced by the number of output pins and the lengths of corresponding paths. The longer the path, the more computations are necessary for the sensitivity propagation of local parameters. However, even for c6288 with 16 paths longer than 97 stages the simulation completed within 28 seconds. The table lists the relative errors of standard deviations for path delays. The average error is 4.1% and the sensitivity w.r.t. channel width is least accurate. The accumulated errors for other parameters do not exceed 10% even for very long paths.

**Table 4.** Simulation results for sensitivity mode and for 200 MC runs (1ps resolution)

| Circuit | | Paths | | | | Runtime [s] | | Rel. Error of std. dev. for arrival times $T$ [%] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | TOX | | VTH | | L | | W | | RDSW | |
| Name | Size | # | Max. | Avg. | Median | Sens-Mode | MC | Avg | Max | Avg | Max | Avg | Max | Avg | Max | Avg | Max |
| c880 | 435 | 26 | 30 | 13 | 7 | 3.8 | 103.1 | 2.9 | 6.1 | 2.8 | 3.9 | 1.3 | 3.1 | 4.5 | 10.4 | 4.0 | 5.7 |
| c1355 | 590 | 32 | 29 | 29 | 29 | 6.6 | 134.7 | 0.2 | 0.2 | 2.0 | 2.0 | 3.3 | 3.3 | 11.4 | 11.4 | 5.3 | 5.3 |
| c1908 | 1057 | 25 | 50 | 40 | 37 | 9.6 | 213.8 | 1.2 | 1.7 | 1.4 | 1.9 | 2.5 | 2.8 | 9.1 | 10.2 | 5.0 | 5.4 |
| c3540 | 1983 | 22 | 57 | 32 | 39 | 13.5 | 368.7 | 1.2 | 3.0 | 2.6 | 3.4 | 0.4 | 0.4 | 7.5 | 11.8 | 3.7 | 4.9 |
| c6288 | 2416 | 32 | 125 | 79 | 97 | 27.5 | 563.4 | 1.0 | 9.0 | 4.5 | 4.7 | 5.2 | 5.9 | 12.3 | 13.4 | 7.2 | 7.8 |

## 6 Conclusion

This paper presents a waveform accurate block-based STA (SWAT). It is based upon a new current source model (CSM) which captures the influence of parameter variation and transistor aging. SWAT supports Monte Carlo simulations and timing analysis of aged circuits. It further provides the propagation of sensitivity waveforms to obtain arrival time distributions without MC simulations. It employs waveform truncation and dedicated solvers to achieve runtimes of a few seconds for ISCAS85 benchmark circuits.

**Acknowledgement.** This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89).

## References

1. Anderson, E.C., Dongarra, J.: Performance of LAPACK: A Portable Library of Numerical Linear Algebra Routines. Proceedings of the IEEE 81(8), 1094–1102 (1993)
2. Cadence: ECSM - Effective Current Source Model (2007), `http://www.cadence.com/Alliances/languages/Pages/ecsm.aspx`
3. Goel, A., Vrudhula, S.: Statistical waveform and current source based standard cell models for accurate timing analysis. In: ACM/IEEE Design Automation Conference (DAC), pp. 227–230 (June 2008)
4. Huard, V., Parthasarathy, C., Bravaix, A., Guerin, C., Pion, E.: CMOS device design-in reliability approach in advanced nodes. In: IEEE International Reliability Physics Symposium (IRPS), pp. 624–633 (2009)
5. Knoth, C., Eichwald, I., Nordholz, P., Schlichtmann, U.: White-Box Current Source Modeling Including Parameter Variation and Its Application in Timing Simulation. In: van Leuken, R., Sicard, G. (eds.) PATMOS 2010. LNCS, vol. 6448, pp. 200–210. Springer, Heidelberg (2011)
6. Knoth, C., Kleeberger, V.B., Nordholz, P., Schlichtmann, U.: Characterization and Implementation of Nonlinear Logic Cell Models for Analog Circuit Simulation. In: International Symposium on Integrated Circuits (ISIC), pp. 97–100 (December 2009)
7. Ling, D.D., Visweswariah, C., Feldmann, P., Abbaspour, S.: A moment-based effective characterization waveform for static timing analysis. In: ACM/IEEE Design Automation Conference (DAC), pp. 19–24 (2009)

8. Pillage, L.T., Rohrer, R.A., Visweswariah, C.: Electronic Circuit and System Simulation Methods. McGraw-Hill, Inc., New York (1995)
9. Raja, S., Varadi, F., Becer, M., Geada, J.: Transistor Level Gate Modeling for Accurate and Fast Timing, Noise, and Power Analysis. In: ACM/IEEE Design Automation Conference (DAC), pp. 456–461 (June 2008)
10. Stewart, G.W.: Afternotes on Numerical Analysis. SIAM, Philadelphia (1995)
11. Synopsys. Composite Current Source (2006), `http://www.synopsys.com/products/solutions/galaxy/ccs/cc_source.html`
12. Tang, Q., Zjajo, A., Berkelaar, M., van der Meijs, N.: RDE-Based Transistor-Level Gate Simulation for Statistical Static Timing Analysis. In: ACM/IEEE Design Automation Conference (DAC), pp. 787–792 (June 2010)
13. Venkataraman, G., Feng, Z., Hu, J., Li, P.: Combinatorial algorithms for fast clock mesh optimization. IEEE Transactions on VLSI Systems 18(1), 131–141 (2010)
14. Vlach, J., Singhal, K.: Computer Methods for Circuit Analysis and Design, 2nd edn. Van Nostrand Reinhold, 115 Fifth Avenue, New York (1994)
15. Zolotov, V., Xiong, J., Abbaspour, S., Hathaway, D.J., Visweswariah, C.: Compact modeling of variational waveforms. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 705–712 (2007)

# Parsimonious Circuits for Error-Tolerant Applications through *Probabilistic Logic Minimization*

Avinash Lingamneni[1,2], Christian Enz[2], Krishna Palem[1], and Christian Piguet[2]

[1] NTU-Rice Institute for Sustainable and Applied Infodynamics, Department of ECE, Rice University, Houston, USA
[2] Integrated and Wireless Systems Division, Centre Suisse d'Electronique et de Microtechnique SA, Neuchatel, Switzerland

**Abstract.** Contrary to the existing techniques to realize *inexact* circuits that relied mostly on scaling of supply voltage or pruning of "least-significant" components in conventional correct circuits to achieve cost (energy, delay and/or area) and accuracy tradeoffs, we propose a novel technique called *Probabilistic Logic Minimization* which relies on synthesizing an inexact circuit in the first place resulting in *zero hardware overhead*. Extensive simulations of the datapath elements designed using the proposed technique demonstrate that normalized gains as high as 2X-9.5X in the Energy-Delay-Area product can be obtained when compared to the corresponding correct designs, with a relative error magnitude percentage as low as 0.001% upto 1%.

**Keywords:** Low Power Design, Error-Tolerant Applications, Probabilistic Logic Minimization, Inexact Electronic Design, Adders, Multipliers.

## 1 Introduction and Related Work

With the growing desire for ultra low energy systems, *inexact* circuits, or circuits in which the accuracy of the output can be traded for substantial cost (energy, area and/or delay) savings, have been receiving increasing attention. In these *inexact* systems, error (either caused probabilistically due to inherent variations/perturbations or introduced deterministically) can be viewed as a commodity that can be traded for significant gains in energy and/or delay, as conceptualized in [11] and widely adopted in [7], [14], [9], [10], [5], [12], [4], [8]. However, most of the (physical) implementations of this principle previously involved scaling the circuit-level parameters (such as supply voltage ($V_{dd}$)) of the conventional hardware taking advantage of error tolerance of the application and had significant drawbacks such as huge associated overhead to ensure an accurate supply-voltage fine-tuning by addition of level shifters, routing of multiple supply voltage planes and/or metastability-tolerant flip flops necessitated by the possibility of exceeding the critical voltage scaling point [10] that might

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 204–213, 2011.

trigger a massive failure owing to the inherent variations present in the supply voltage routing [3].

As a result of these drawbacks, alternate architecture-level approaches that achieve *zero hardware overhead* were proposed recently such as *probabilistic pruning* [8] that attempts to "prune" or delete non-significant portions of the circuits during the design phase, thereby achieving gains in all 3 dimensions – energy, delay and area. While this technique can be classified as a "bottom-up" approach to designing inexact systems (given that it begins with an already synthesized circuit and attempts to prune it), in this paper, we propose an alternate "top-down" architecture-level approach called *probabilistic logic minimization* which performs a parsimonious synthesis of inexact circuits driven by the overarching philosophy of *value of information guided design* and is shown to achieve higher savings for a lesser error than the probabilistic pruning technique. The proposed algorithm takes advantage of the notion of introducing *bit-flips* in the minterms of boolean functions that was proposed in [6] in the context of fast error detection and subsequently used in [13] for enhancing circuit yield and for designing low-power inexact circuits in [4]. However, this utilization of bit-flip based technique in the synthesis of inexact circuits has mostly been ad-hoc with limited insights to general circuits (especially datapath circuits that dominate the energy consumption in inexact systems such as motion estimation block in video encoders [14]) and lacks a general guiding algorithm to attune circuits to specific inexact applications to glean further cost (energy, delay and area) gains and we hope to address these issues in our paper. The major contributions of this paper are summarized below:

- We propose a novel architecture-level approach called *Probabilistic Logic Minimization* to realize parsimonious inexact circuits driven by the philosophy of *value of information guided designs* that results in minimal cost (energy, area and/or delay) circuits attuned for the end application.
- We propose a methodology for seamless integration of the proposed probabilistic logic minimization technique into a conventional design flow to realize inexact circuits at a faster rate than a custom design methodology needed by some previous approaches in literature ([7], [5], [12]).
- We apply the proposed probabilistic logic minimization algorithm on critical datapath elements such as adders and multipliers and show through extensive simulations with different application benchmarks that significantly more savings across all 3 dimensions – energy, delay and area, can be obtained by our technique with *zero hardware overhead*

## 2 Probabilistic Logic Minimization for a Cost vs Accuracy Tradeoff

*Probabilistic Logic Minimization* is an architecture-level technique with *zero hardware overhead*, wherein we systematically minimize circuit components' (or nodes') boolean functions guided by the significance and the input combination probabilities of those nodes while staying within the error boundaries dictated by the application.

### 2.1 Logic Minimization through Bit-Flips in Boolean Function Minterms

The notion of introducing bit flips in the minterms of boolean functions to create approximate functions was proposed in [6] and this principle can be harnessed to glean cost gains (energy/area/delay) through literal reduction in the context of inexact circuits while causing an error due to such bit flip(s). However, not all bit flips of minterms would result in expanding the prime implicant (PI) cubes and some of them might result in negative gains. Hence, it is important to identify the "favorable" bit-flips (or the bit-flips which further minimize the function) and discard the non-favorable ones. To illustrate through an example, Figure 1(a) shows a function (Carry logic) that is widely prevalent in most datapath elements. Assuming that the application would only be able to tolerate at most one bit-flip at this logic function (probability of error = 1/8), Figures 1(b) and 1(c) give an example of favorable 0 to 1 and 1 to 0 bit flips respectively as they minimize the logic function whereas Figure 1(d) shows an unfavorable bit flip leading to an increased logic function complexity. Hence, we can conclude that introduction of favorable bit-flips would lead to further minimization of a logic function owing to the expansion of PI cubes, thereby achieving cost (energy, area and delay) gains at the expense of error which is proportional to the number of such bit flips introduced.

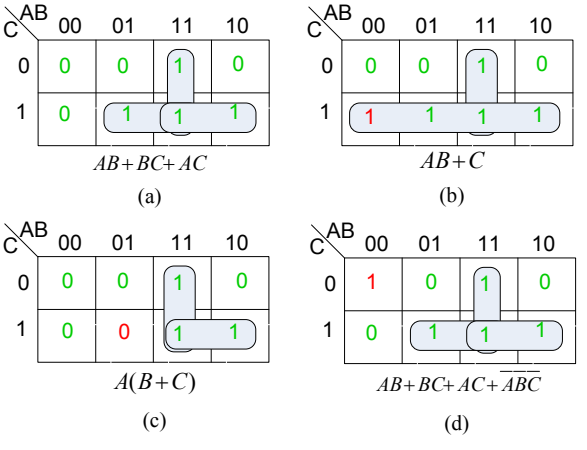



**Fig. 1.** Example of K-Maps of the (a) initial Correct Function (Carry Logic of a Full Adder) (b) function with a favorable 0 to 1 bit flip (c) function with a favorable 1 to 0 bit flip (d) function with a non-favorable 0 to 1 bit flip

While the benefit of this imprecise "bit-flip" minimization cannot be denied, it gives rise to another interesting question not answered before: *Given a circuit node with many favorable bit flip possibilities, each with similar cost gains, how do we select the right minimization for the node?* In other words, is the error introduced by each of the bit flips equal? While conventional wisdom calls for an

assumption of uniform input combination probabilities, it is never the case with most applications, more so with multimedia applications where inputs are highly correlated and also, circuit-topology dependent. Hence, our proposed technique takes advantage of such correlation to guide the minimization algorithm and glean further savings.

In general, given a circuit node with $n$ inputs, there are $2^n$ possible minterms, of which we could flip the bits at atmost $k$ minterms (constrained by the application's error tolerance), to derive the minimum cost function. We propose a probabilistic extension to the minimization scheme wherein all the favorable bit flips are ranked based on their input combination probabilities and the bit flip(s) having the least corresponding input combination probabilities are performed. For example, in Figure 1(b) and (c), the minimized functions have the same gains (3 ORs and 2 ANDs reduced to 1 OR and 1 AND). But if the probability of input to the logic function being '001' is higher than the input being '011', then a bit flip at '011' would likely cause an error with a lesser probability. Hence, *a bit flip occurring at the least likely input combination would result in lesser error for the same amount of savings.*

### 2.2 Application of Probabilistic Logic Minimization to Datapath Circuits

We explore the foundations to achieve optimal logic minimizations in datapath elements for error tolerant applications through an example of a widely used 3-input one-bit full adder. From Table 1, we observe that not all full adders used to construct the datapath elements such as ripple carry adders and array multipliers have similar input transition characteristics. For the array multiplier, the full adders receiving the inputs directly from the partial products (AND-ed inputs) are denoted as 'Ext.', the full adders present inside the partial product reduction matrix are denoted as 'Int.' and finally, the full adders present in the final carry propagate stage are denoted as CPA. Hence, *the probability of an input combination occurring at a node is generally either (a) only dependent on the input test vectors (such as full adders in ripple carry adder (RCA) and external full adder in array multiplier) (b) only dependent on the circuit topology (such as internal full adder of array multiplier) (c) a combination of both (such as CPA full adder in array multiplier).*

Another key observation from Table 1 is that there is a strong correlation between the input vector combination(which in turn depend on the application) and the amount of logic minimization (or the number of bit flips) that can be performed on a node. We could potentially group all the input combinations with values less than one or two standard deviations from the mean of the group and then, favorable bit flips can be done within this group to obtain the most amount of minimization. For example, in the audio benchmark, input combinations {'001', '011', '101', '110'} for the external full adder of the array multiplier can be grouped together and favorable bit flips identified among them.

**Table 1.** Input Combination Probabilities of various Full Adders in Datapath Elements

| Test Vectors | Component | Probability of Various Input Combinations | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| Uniform | RCA | 0.129 | 0.121 | 0.125 | 0.121 | 0.125 | 0.126 | 0.13 | 0.124 |
| | ArrMul (Ext.) | 0.542 | 0.024 | 0.167 | 0.017 | 0.145 | 0.04 | 0.025 | 0.041 |
| | ArrMul (Int.) | 0.349 | 0.047 | 0.079 | 0.048 | 0.314 | 0.051 | 0.056 | 0.055 |
| | ArrMul (CPA) | 0.388 | 0.088 | 0.082 | 0.01 | 0.218 | 0.092 | 0.091 | 0.03 |
| Audio [2] | RCA | 0.258 | 0.02 | 0.133 | 0.12 | 0.129 | 0.121 | 0.014 | 0.205 |
| | ArrMul (Ext.) | 0.5207 | 0.0004 | 0.2209 | 0.0002 | 0.2556 | 0.0005 | 0.0003 | 0.0014 |
| | ArrMul (Int.) | 0.394 | 0.029 | 0.041 | 0.048 | 0.309 | 0.032 | 0.038 | 0.109 |
| | ArrMul (CPA) | 0.272 | 0.073 | 0.148 | 0.001 | 0.291 | 0.126 | 0.085 | 0.004 |
| Image [1] | RCA | 0.355 | 0 | 0.382 | 0 | 0.148 | 0 | 0.115 | 0 |
| | ArrMul (Ext.) | 0.846 | 0 | 0.132 | 0 | 0.024 | 0 | 0 | 0 |
| | ArrMul (Int.) | 0.38 | 0.027 | 0.192 | 0.015 | 0.298 | 0.033 | 0.037 | 0.019 |
| | ArrMul (CPA) | 0.867 | 0.03 | 0.013 | 0 | 0.081 | 0.007 | 0.003 | 0 |

### 2.3 A General Algorithm for Probabilistic Logic Minimization

A circuit can be represented as a *directed acyclic graph* with nodes representing components such as gates (or even bigger blocks like full adders), input or outputs and with edges representing interconnects. Let a graph $\mathcal{G}$ represent a circuit with N nodes and W edges. For any given node $i$ in the graph, we have

- node.function($i$) – denotes that function computed by node $i$,
- node.significance($i$) – denotes the significance of node $i$,
- node.fanin($i$) – denotes the fanin of node $i$,
- node.inputprobability($i$)($j$) – denotes the transition probability of input combination $j$ occurring at node $i$. The range of values of $j$ are 0 to $2^{fanin} - 1$
- node.functionmin(i) – denotes the minimized function of the node
- node.valuemin(i) – denotes the "value" or normalized cost gains of the minimized function

While function, significance and fanin values of all nodes are derived from the graph structure and user input, the inputprobability, functionmin and costmin are computed during the execution of the algorithm. The notion of assigning significance to a circuit node is one of the guiding principles for achieving an optimal inexact circuit. While this assignment of significance can be user defined, there are various algorithms in literature which can used to assign significance to circuit nodes depending on the amount of error they can cause at the circuit outputs assuming the rest of the circuit nodes operate correctly. The algorithm for the proposed technique is described in Algorithm 1.

**Algorithm 1.** Pseudo-code for the *Probabilistic Logic Minimization* (PLM) algorithm on a Circuit or Graph $\mathcal{G}$

```
1: //Main Function in the Algorithm
2: function PLM(MaxError)
3:     //Compute the probability of each input transition at all nodes
4:     Benchmark( );
5:     //Compute the most cost-effective minimization at each node
6:     for all i ← 1 to N do
7:         ComputeMinimization(node(i));
8:     end for
9:     //Iteratively minimize each node based on their "value" until the error bound
10:    while Error ≤ MaxError do
11:        NodetoMinimize = FindMinimum (node.significance × node.costmin);
12:        MinimizeNode(NodetoMinimize);
13:    end while
14: end function
15:
16: function BENCHMARK
17:    RunBenchmark();
18:    for all i ← 1 to N do
19:        for all j ← 1 to 2^fanin − 1 do
20:            node.inputprobability(i)(j) = ComputeInputProbability;
21:        end for
22:    end for
23: end function
24:
25: function COMPUTEMINIMIZATION(node)
26:    for all j ← 1 to 2^fanin − 1 do
27:        //Estimate the gains obtained by the bit flip at the input sequence j in the
       K-Map through synthesis tools
28:        costgain(j) = EstimateCostGain(bitflip(j));
29:        if costgains(j) > 0 then
30:            ValueofBitFlip(j) = costgain(j) / inputprobability(j);
31:        else
32:            ValueofBitFlip(j) = 0;
33:        end if
34:    end for
35:    //Nested loops can be used above to felicitate more than 1 bit-flip per node
36:    MaxValue = FindMaximum(ValueofBitFlip(j)):
37:    functionmin ← ComputeFunction(MaxValue);
38:    costmin ← MaxValue;
39: end function
40:
41: function MINIMIZENODE(node)
42:    function ← functionmin;
43: end function
```

Given the dominance of applications with varying output significance, we use the Relative Error Magnitude metric for quantifying the error in the inexact circuits, used widely in [5], [8], computed as

$$\text{Relative Error Magnitude} = \frac{1}{\mathcal{V}} \sum_{k=1}^{\mathcal{V}} \frac{|\mathcal{O}_k - \mathcal{O}'_k|}{\mathcal{O}_k}$$

where $\mathcal{V}$ is the total number of simulation cycles or test vectors given to the circuit, $\mathcal{O}_k$ is the expected correct output vector and $\mathcal{O}'_k$ is the obtained erroneous output vector for the $k^{th}$ input vector.

## 3 Experimental Results and Analysis

### 3.1 Proposed Logic Synthesis Based CAD Flow

The proposed design flow for realizing inexact circuits through *Probabilistic Logic Minimization* is shown in Figure 2. The main object of interest in the proposed CAD flow is the *Probabilistic Logic Minimizer*, which interfaces with a functional simulator and an error estimator to perform the probabilistic logic minimization governed by the application's error margins. All the designs are implemented using TSMC 65nm standard cell library. We have used 3 different sets of test vectors in this paper- uniform random distribution (from Matlab), test vectors from image data (from [1]) and test vectors from audio data(from [2]).

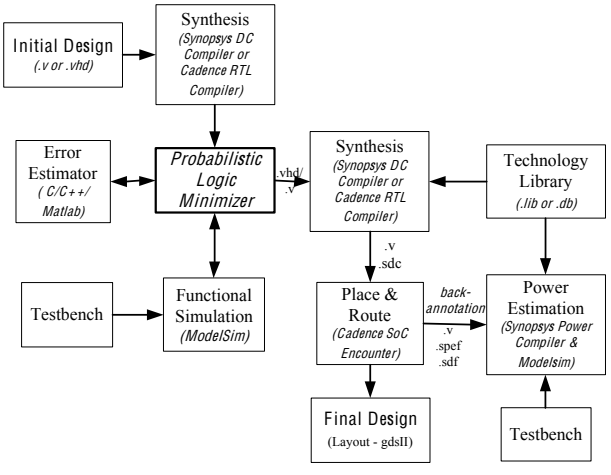



**Fig. 2.** A logic synthesis based flow for designing *Probabilistic Logic Minimized* circuits

### 3.2 Results and Analysis

The main results of this paper, namely the normalized gains (Conventional / Proposed) values for different metrics – Area, Delay, Energy, Energy-Delay Product (EDP) and Energy-Delay-Area product (EDAP) – obtained by applying the probabilistic logic minimizations for a 16-bit ripple carry adder and a 16-bit array multiplier for different application benchmarks are given in Figure 3 and 4 respectively. However, we have also implemented other types of adders such as Carry-Select, Kogge-Stone and Sklansky and multipliers such as Wallace-tree and Dadda multipliers, with varying bit-widths (upto 64-bits), with varying synthesis contraints and for different application benchmarks, and have obtained similar gains (not shown here due to space constraints).

As evident from the results, the probabilistic logic minimization approach results in highly energy, delay and area efficient datapath elements. For the uniform test vectors, in the case of ripple carry adders (with highest frequency synthesis constraint), probabilistic minimization yields savings upto 8X with an relative error magnitude of less than 1% compared their conventional correct counterparts while in the case of array multiplier(with lowest power synthesis constraint), it resulted in savings of about 7X with a relative error magnitude of less than 6.5%. It can be seen that using application specific test vectors (like audio and image), the savings have increased (upto 9.5X in the case of ripple carry adders and upto 8.25X in the case of array multipliers) with comparable error values.

On further observation, apart from the synthesis constraints, we can notice that the number of nodes in the circuit that can be minimized also dictate the gains across various metrics. For example, the delay gains in the ripple carry adders dominate the energy gains as there are only a limited number of nodes ($O(n)$) to minimize and almost all the node minimizations lead to critical path delay reduction. Whereas, in the case of an array multiplier, the energy gains dominate the delay gains due to the presence of a larger number of nodes ($O(n^2)$), most of which not on the critical-path, that can be minimized.

Compared to the gains obtained by the probabilistic pruning technique [8] (about 2X-7.5X savings in energy-delay-area product for relative error magnitude percentages upto 8% in the context of arithmetic adders), we observe that the gains obtained by the proposed technique are higher with even lesser error values (about 2-9.5X savings in energy-delay-area product for relative error magnitude percentages less than 1%). This can be attributed to the fact that the probabilistic pruning technique is limited by the circuit nodes and their corresponding paths (a node is either pruned or not), while the proposed approach has an inherent ability to "finetune" the node through logic minimization incase it can not completely delete it, thereby gleaning more savings.

To summarize, one of the key inferences from the simulation results is that : *the significant gains achieved in a circuit through the proposed probabilistic logic minimization technique are technology-independent, bit-width independent and only proportional to the amount of circuit nodes minimized.*

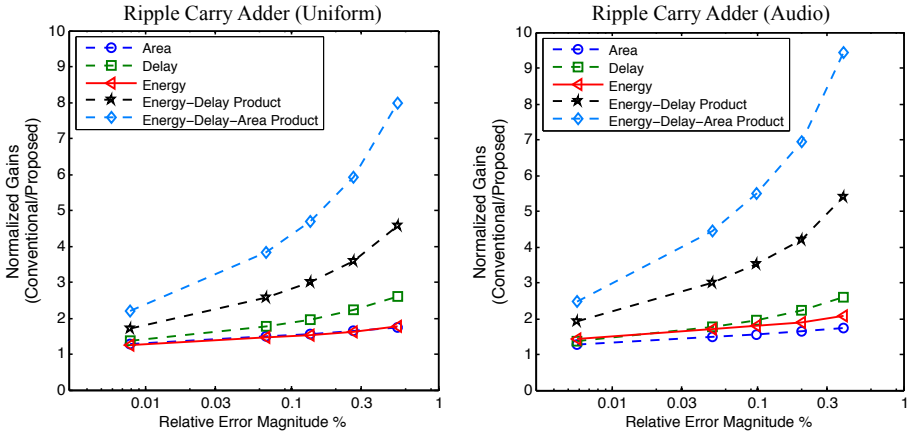



**Fig. 3.** Normalized gains Vs Relative Error percentage of minimized ripple carry adders for different benchmarks

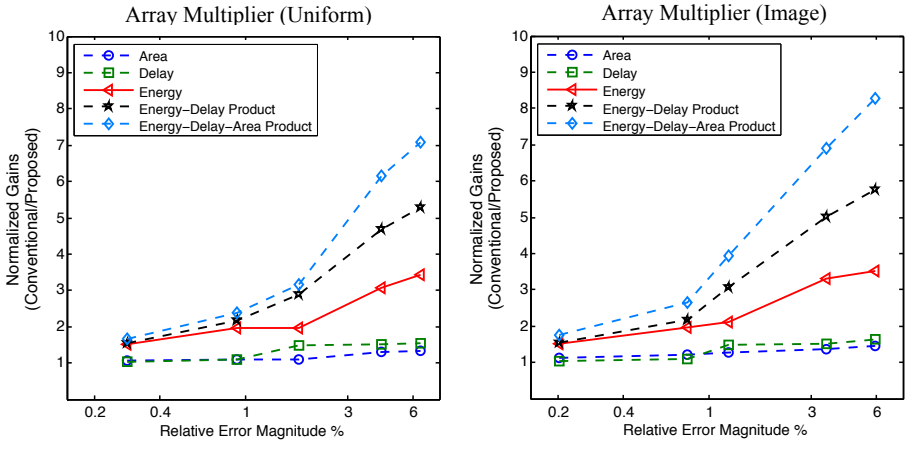



**Fig. 4.** Normalized gains Vs Relative Error percentage of minimized array multipliers for different benchmarks

## 4 Conclusion and Future Directions

In this paper, we proposed a novel architecture-level approach, called *probabilistic logic minimization*, to realize inexact circuits in a "top-down" fashion and believe to have convincingly shown through extensive simulations that our technique achieves significantly better savings along all three dimensions – energy, delay and area – with *zero* overhead and more savings than previously made possible by "bottom-up" architecture-level techniques such as *probabilistic pruning*. We have described in detail the general algorithm to apply the proposed technique

on any circuit that can be represented as a network of nodes and a CAD flow to achieve faster design time and fabrication for inexact circuits.

The future directions of our work include formulating an elaborate mathematical and optimization model to realize optimum inexact circuits through the proposed technique and extending the proposed technique to implement large-scale error-tolerant systems such as video encoder/decoder and hearing aids.

## References


1. Mediabench, `http://euler.slu.edu/~fritts/mediabench`
2. NCH Software, `http://www.nch.com.au/acm/index.html`
3. Alioto, M., Palumbo, G.: Impact of supply voltage variations on full adder delay: analysis and comparison. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 14(12), 1322–1335 (2006)
4. Bharghava, R., Abinesh, R., Purini, S., Govindatajulu, R.: Design of low power systems using inexact logic circuits. The Journal of Low Power Electronics 6(3), 401–414 (2010)
5. Chakrapani, L.N.B., Muntimadugu, K.K., Lingamneni, A., George, J., Palem, K.V.: Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation. In: The Proc. of CASES, pp. 187–196 (2008)
6. Choudhury, M., Mohanram, K.: Approximate logic circuits for low overhead, non-intrusive concurrent error detection. In: The Proc. of Design, Automation and Test in Europe, pp. 903–908 (March 2008)
7. George, J., Marr, B., Akgul, B.E.S., Palem, K.: Probabilistic arithmetic and energy efficient embedded signal processing. In: Proc. of the IEEE/ACM Intl. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems, pp. 158–168 (2006)
8. Lingamneni, A., Enz, C., Nagel, J.L., Palem, K., Piguet, C.: Energy parsimonious circuit design through probabilistic pruning. In: The Proc. of the 14th Design, Automation and Test in Europe, pp. 764–769 (March 2011)
9. Mohapatra, D., Karakonstantis, G., Roy, K.: Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator. In: The Proc. of the ISLPED, pp. 195–200 (2009)
10. Narayanan, S., Sartori, J., Kumar, R., Jones, D.: Scalable stochastic processors. In: The Proc. of the Design, Automation and Test in Europe, pp. 335–338 (March 2010)
11. Palem, K.V.: Energy aware algorithm design via probabilistic computing: From algorithms and models to Moore's law and novel (semiconductor) devices. In: The Proc. of the CASES, pp. 113–117 (2003)
12. Palem, K.V., Chakrapani, L.N., Kedem, Z.M., Lingamneni, A., Muntimadugu, K.K.: Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects. In: Proc. of the CASES, pp. 1–10 (2009)
13. Shin, D., Gupta, S.: Approximate logic synthesis for error tolerant applications. In: The Proc. of Design, Automation and Test in Europe Conference (DATE), pp. 957–960 (March 2010)
14. Varatkar, G.V., Shanbhag, N.R.: Energy-efficient motion estimation using error-tolerance. In: The Proc. of the ISLPED, pp. 113–118 (2006)

# Sub-Row Sleep Transistor Insertion for Concurrent Clock-Gating and Power-Gating

Karthikeyan Lingasubramanian, Andrea Calimera,
Alberto Macii, Enrico Macii, and Massimo Poncino

Dipartimento di Automatica e Informatica
Politecnico di Torino, 10129 Torino, Italy

**Abstract.** Concurrent *clock gating* (CG) and *power gating* (PG) can help to tackle both static and dynamic power simultaneously, thereby enabling the design of low-power and energy efficient applications. Unfortunately the automatic integration of the two techniques in standard design flows is limited by several technical impediments. Among them, physical constraints during the Sleep Transistor Insertion (STI) imposed by row-based layout rules are certainly the most critical. Although determining the feasibility of the whole clock-gating and power-gating (CG-PG) integration, the adopted STI methodology may have drastic effects on several circuit metrics, like operating frequency, throughput and power savings. In this paper we introduce a layout-friendly STI approach for fine-grained CG-PG inclusion. The proposed method, that is aware of the timing-driven strategies adopted by most of the commercial placer tools, allows sub-row insertion of independent sleep-transistor cells, therefore enabling finer resolution in the CG-PG integration, along with minimal cell displacement and negligible layout disruption. This enables a larger number of cells to be power-gated (i.e., larger potential power-savings w.r.t. state-of-the-art fine-grained STI strategies), without delay overhead. Experimental results, conducted on a set of circuit benchmarks mapped onto an industrial 65nm technology, indicate that more than 50% of the total number of cells can be clock- and power-gated simultaneously, without any speed degradation.

## 1 Introduction

Out of the prevalent low power design strategies [3, 12], Clock Gating (CG) and Power Gating (PG) are the prominent methods used to curtail dynamic and static power respectively. Apart from the substantial power savings they guarantee, their popular appeal is motivated by the high compatibility with standard CMOS technologies and their scaling, by the flexibility to automatic applications and by the easy integration with commercial design frameworks provided by the major EDA vendors [1, 2, 4, 10, 14].

Unfortunately, their implementation does not come for free and timing/area issues need to be considered very carefully [8, 10]. In addition, both require dedicated power-managed units, which are in charge of identifying the idle conditions

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 214–225, 2011.

and generate the control signals accordingly. Such control units, which are commonly designed as independent yet separate entities, may represent a primary source of area and timing overheads and could make the verification phases of the design more difficult. It is therefore intuitive to understand that, although very desirable from the point of view of the achievable energy savings, merging clock-gating and power-gating under a unique control domain may represent a valuable solution to mitigate the implementation costs.

Since they share the same idle conditions of the circuit, CG and PG can be ideally managed by the same control signal and, in particular, the same signal used to control the clock-gating registers can be used to drive the sleep transistors. Using the CG conditions as master control is desirable: CG is a well stable technique, known for quite a long time, and well supplied by all the automatic logic synthesis tools; moreover, the CG technique works at the lowest level of granularity (idle conditions extracted at the gate-level, with a time resolution of a clock-period) and it can therefore provide the finest power control ever possible.

Although the feasibility of the CG-PG approach has been proven in [11], its automation is far for being trivial. While several works proposed automatic design flows [9, 15], their implementation is limited by the need of customized standard gates, which include dedicated self-contained sleep transistors. Unfortunately, most of the CMOS libraries provided by silicon vendors do not provide designers with such feature. To overcome such issue, [5, 6] introduced a CG-PG design strategy where the granularity of the sleep transistor insertion is a single layout row instead of a single cell. The idea is to constrain the placement of the cells belonging to the same CG-cluster, namely the cells controlled by the same clock-gating signal, in a single row (or adjacent rows) and then insert dedicated sleep transistors for each row (or group of rows). Although feasible, such solution results in high timing overhead due to excessive cells displacing and layout disruption (more than 30% in the best case). The main source of this behavior relies on the fact that state-of-the-art timing-driven placement algorithms attempt to minimize the wire delays privileging square-like topographical cell distributions, that is, cells that belong to the same timing path, and thus to the same CG-cluster, are distributed on multiple rows. This does contrast with row-based STI.

The main contribution of this work is to propose a STI methodology, aware of these physical issues, that allows an efficient integration of clock gating and power gating. Compliance with timing driven placements is achieved by means of a layout-friendly STI methodology in which groups of cells can be clock gated and power gated simultaneously in smaller atomic unit, namely the *sub-rows*. This guarantees finer granularity of the sleep transistor insertion without sacrificing the integrity of the layout. Experimental results, conducted on a set of benchmarks mapped onto an industrial 65nm technology provided by STMicroelectronics, show that more than 50% of the total number of the cells can be clock gated and power gated simultaneously with zero-delay overhead.

## 2 Understanding Concurrent CG and PG

Clock-Gating [4] is the most representative technique for dynamic power-reduction. It is based on the idea of disabling (i.e., *gating*) the clock signal to a specific register (or bank of registers) for which the input signal does not change, or that feeds a portion of combinational logic that is not performing useful computations during some clock cycles. An auxiliary circuitry, usually called the *activation function*, is used for intercepting the idle conditions and generate the activation signal ($F_a$) for the gating of the clock. These conditions, extracted by sensing the primary and state inputs of the circuit, might be derived from a state-based description of the netlist of the circuit and can be purely topological or include functional information.

Power-Gating[10, 12] is a stand-by leakage reduction technique that is based on the insertion of switch transistors on the pull-up and/or pull-down of the logic circuit. Such transistors, well known as *sleep transistors*, detach the circuit from the power/ground rails during the idle periods. They are driven by dedicated control signals (the *sleep* signals), that, like the $F_a$ for clock-gating, are in charge of identifying the periods under which the circuit is not performing any useful computation.

CG and PG may look as unrelated techniques, but a more careful analysis reveals that they are two different ways of exploiting the same property of a design, namely its *idleness*. While CG stops the clock to a logic block during idle cycles, PG disconnects a logic block from the ground line during idle periods. Their fundamental difference lies in the way idleness is determined. CG extracts (structurally or functionally) the cycle-by-cycle idle conditions and suppresses the clock in those cycles; conversely, PG is activated by an external signal, which identifies the idleness of the entire circuit. In principle, nothing prevents us from using the CG conditions extracted from the circuit and implemented by $F_a$ also for controlling PG, so that both dynamic and static power is saved during the idle intervals. Although guaranteeing the finest timing granularity in detecting the idle conditions, using the same control unit to manage CG and PG simultaneously minimizes the design overheads. The conceptual integration of the two techniques is shown in Figure 1. The CG conditions (represented by the logic denoted by $F_a$) are used as additional PG conditions (logically OR-ed) to

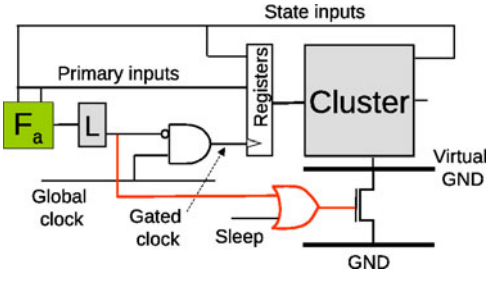



**Fig. 1.** Conceptual Integration of CG and PG [5]

those externally provided as sleep signals. The activation signal is filtered by a latch that is transparent when the global clock is low. The purpose of the latch is to filter potential glitches of the activation signal that should not propagate when the global clock is high.

### 2.1 CG-PG: Limiting Factors and Design Issues

The integration of CG and PG under a single control domain raises important design issues [5], namely, *clustering* and *physical STI*. Both are direct consequences of the natural clustering defined by the clock-gating, but they have different implications on the resulting CG-PG circuit.

A group of cells that are clock-gated by the same register automatically defines the CG-cluster. Each CG-cluster can ideally be power-gated by a dedicated sleep transistor driven by an independent sleep signal. As long as the clusters do not overlap, the scheme of Figure 1 is applicable as is. Unfortunately, due to logical interdependencies of the gates, the clustering may not be mutually exclusive, and some cells could be shared among several clusters. While having intersecting clusters is totally tolerated for the application of CG alone, when considering also the effects of PG, this is not true anymore. This is due to the fact that different clusters may work in different power-modes (active or idle) independently. However, when idle, all the cells of a cluster are detached from the ground rail and they are not able to feed stable logic-values to other interlaced clusters (see Figure 2(a)). We refer to this problem as the *clustering* problem. A conservative approach to solve the clustering problem, is to identify those cells, which belong to multiple clusters and leave them as non power-gated, Figure 2(b). Previous works [6] suggested to create dedicated sleep signal obtained by the intersection of the control signals of multiple clusters. In this work we opted for the conservative approach.

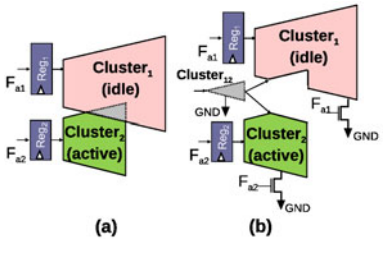



**Fig. 2.** CG-PG Issues in Row-Based Layouts

The second main issue encompasses the STI and the fact that STs cannot be placed anywhere in the layout. Unless one can use MTCMOS cells with internal sleep transistors [9, 15], the implicit CG clustering may require that, cells physically close in the layout (e.g., placed in the same layout row) must be grouped in separated CG-clusters. This limits the applicability of the state-of-the-art STI approaches and requires the adoption of placement-aware clustering schemes,

which, unfortunately, may induce excessive layout disruption and difficulties in reaching the timing compliance [5, 6].

As we describe in Section 3, we tackle these problems from another side and we propose a STI methodology that can help to perform efficient identification of groups of cells that are gated by a unique control signal while respecting the timing-driven placement.

## 3 Proposed Methodology

### 3.1 Problem Statement and Methodology Overview

Let us consider a benchmark circuit, $b17$, from the ITC99 suite. After performing physical synthesis with the CG feature enabled, the circuit shows three distinct CG clusters $C_1$, $C_2$, $C_3$, each one driven by its dedicated activation function $F_a$, and one non-CG cluster $NC$ that contains all those cells that are not clock-gated. Let us represent the total number of cells in the circuit as a global set $G$, where $\{C_1 \cup C_2 \cup C_3 \cup NC\} = G$. The sets $C_1$, $C_2$ and $C_3$ are not necessarily unique or mutually exclusive, i.e., $\{C_1 \cap C_2\} \neq \emptyset$, $\{C_2 \cap C_3\} \neq \emptyset$, $\{C_1 \cap C_3\} \neq \emptyset$, $\{C_1 \cap C_2 \cap C_3\} \neq \emptyset$. Meanwhile, the sets $C_1$, $C_2$ and $C_3$ are mutually exclusive to set $NC$, $\{NC\} \notin \{C_1 \cup C_2 \cup C_3\}$. The cells that can be concurrently power-gated and clock-gated are represented by $CGPG = \{CGPG_1 \cup CGPG_2 \cup CGPG_3\}$ where the three $CGPG$ sets are defined as:

$$CGPG_1 = \{C_1\} \notin \{\{C_1 \cap C_2\} \cup \{C_2 \cap C_3\} \cup \{C_1 \cap C_3\} \cup \{C_1 \cap C_2 \cap C_3\}\} \quad (1)$$

$$CGPG_2 = \{C_2\} \notin \{\{C_1 \cap C_2\} \cup \{C_2 \cap C_3\} \cup \{C_1 \cap C_3\} \cup \{C_1 \cap C_2 \cap C_3\}\} \quad (2)$$

$$CGPG_3 = \{C_3\} \notin \{\{C_1 \cap C_2\} \cup \{C_2 \cap C_3\} \cup \{C_1 \cap C_3\} \cup \{C_1 \cap C_2 \cap C_3\}\} \quad (3)$$

While respecting the above conditions is sufficient to address the clustering problem, i.e., each CGPG cluster can be independently power-gated without affecting the functionality of the remaining clusters, the problem of physical STI still holds. Figure 3(a), gives the layout of $b17$ after timing driven placement. The different colors represent different sets, $CGPG_1$, $CGPG_2$, $CGPG_3$ and $NC$. The picture clearly highlights that none of the rows in the layout contains cells belonging to a unique cluster; since each cluster needs its dedicated virtual-ground rail, adopting a row-based STI scheme would imply the routing of multiple metal line through each and every row. Apart from introducing drastic layout disruption and cell displacing, this approach may prove infeasible when the number of clusters starts growing (i.e., number of clusters per row is larger than 2).

We propose to address this issue adopting a finer STI scheme in which the atomic unit of power-gating is scaled to portion of rows rather than entire rows. As shown in Figure 3(b), the layout is divided in uniform sub-rows using regular columns of white-spaces. Such white-space will host the sleep transistor

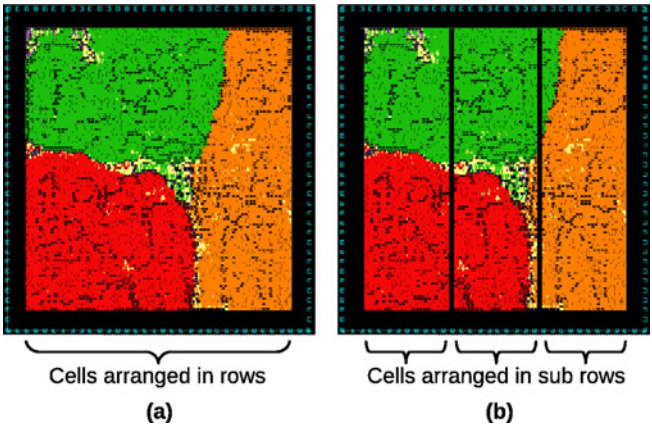



**Fig. 3.** Layout of $b17$ after Row-Based Placement (a), and after column splitting(b)

cells. Exploiting the spatial-locality of cells belonging to the same cluster, and the fact that placement algorithms privilege square-like placement policies, the probability that cells placed in the same sub-row belonging to the same cluster can increase substantially. Such a probability gets higher with larger number of columns.

### 3.2 Sub-Row STI

In order to support the proposed methodology a new automatic STI scheme has been developed and integrated into standard physical design flow. Figure 4 shows a detailed comparison between a state-of-the-art row-based STI and the proposed sub-row STI. In the row-based approach (Figure 4(a)) the cells in each row $i$ can be connected to a dedicated sleep transistor $ST_i$ placed at the boundary of the rows[1] through a dedicated virtual-ground rail $Vgnd_i$ [1]. As described in the previous section, this approach fails when the rows contain cells belonging to more than one cluster. Using our method (Figure 4(b)), each sub-row of coordinate $i, j$ can be connected to a sleep transistor $ST_{ij}$ through a dedicated virtual-ground rail $Vgnd_{ij}$ (Figure 4(c)).

In a given sub-row $SR_{ij}$, if all the cells in the row are clock gated by a single unique CG cell, then the clock enable signal of that CG cell can be used to control the sleep signal for $ST_{ij}$. If $SR_{ij}$ contains cells that are clock gated by more than one CG cell, or contains cells that are not clock gated, then the corresponding $ST_{ij}$ is driven by the external sleep signal only.

Apart from the above mentioned design methodology, some important aspects of our design has to be noted. In order to guarantee a proper interface between cells belonging to different power-gating domains (i.e., cells driven by different

[1] Sleep transistor can be also placed in dedicated rows (called ST-rows) as described in [14].

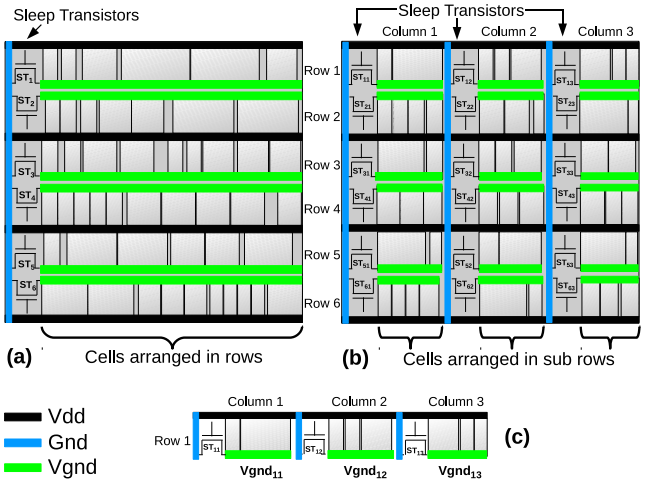



**Fig. 4.** (a) Row-Based ST Insertion. (b) Sub-Row Based ST Insertion with Three Columns of STs (ST rows = 6, No. of STs = 18). (c) Dedicated $V_{gnd}$ Rails Used for each ST.

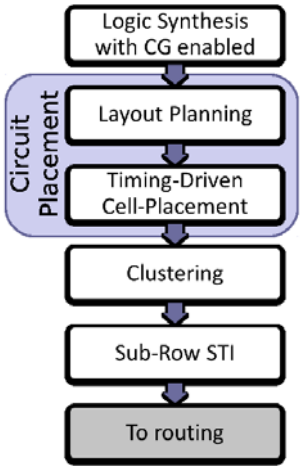
CG conditions), we adopted the virtual-ground pull-up strategy proposed in [7], thereby preventing the usage of isolation cells. It is also worth mentioning that in our methodology the sub-rows containing registers or single flip-flops are not power-gated. This prevents the usage of state-retention flops.

### 3.3 Proposed Design Flow

The flow described in Figure 5 has been implemented. The first step performs the RTL synthesis of the circuit with the clock-gating feature enabled and using the constraints defined by the user (i.e., area, timing and power).

The obtained gate-level netlist is then fed to the second stage, the *circuit placement*. As first sub-step, we perform the *Layout Planning* during which, taking as reference the area estimation obtained from the synthesis, an initial floorplan of the layout is drawn. The latter is then split into uniform columns. Once the skeleton of the layout has been obtained, the cells are placed through the layout using a commercial *Timing driven cell placement*. The physical constraints obtained from the planning phases avoid the tool to place standard cells in the white-space columns that, instead, will be used for the placement of the STs.

The third step encompasses the *Clustering* of the placed CG-design. Dedicated algorithms parse the topological structure of the circuit detecting the clock-tree, the clock-gating registers, their activation functions and the cones of logic associated with them, namely, the natural CG-clusters $C_i$. All these data, combined with the physical information of the placed design, are used to identify the CGPG-clusters, i.e., sub-rows of the layout containing CG-clusters subtracted




**INPUT:** $cell_{ij}^{k}$, $k$-th cell in a sub-row $SR_{ij}$ from row $i$ and column $j$
**INPUT:** $C_i$, $i$-th CG cluster
**INPUT:** $nCG$, No. of CG clusters

$unique = 0$
**for** $m = 1 \rightarrow nCG$ **do**
  **if** $cell_{ij}^{k} \in C_m$ **then**
    $unique++$
    $c\_CG = m$
  **end if**
**end for**
**if** $unique = 1$ **then**
  $cell_{ij}^{k} \in C_{c\_CG}$ only
**else**
  $cell_{ij}^{k} \in NC$
**end if**

**OUTPUT:** Cell coloring

**Fig. 5.** Proposed CG-PG flow and pseudo-code for the coloring of a cell in a sub-row

off the intersections. This crucial part of the flow lies on the identification of the cells in each sub-row that belong to unique CG-clusters. For a given cell, we check whether it is clock gated or not. If yes, then it has to be checked whether it is clock gated by one CG-cluster or multiple CG cells. Figure 5 provides a simple pseudo-code to perform this identification, which has to be recursively run for all the cells $C^k$ in each sub-row $(i, j)$. Once the cell coloring has been completed, an immediate check to find what are the sub-rows that can be power-gated, is performed, along with the identification of corresponding enable signal. It has to be noted that the run time of these identification steps are considerably low and does not affect the design flow.

The flow concludes with the insertion of the ST cells [7] in the sleep-transistor cavities. The size of each ST is defined based on the size of the sub-row and on the amount of current it injects in the virtual-ground during the active mode [13].

## 4 Experimental Results

The methodology described in Section 3 has been implemented and integrated into an industrial DesignKit for a 65nm CMOS technology provided by STMicrolectronics. EDA tools belonging to a commercial design platform, the *Galaxy Implementation Platform* of Synopsys, has been used for the synthesis and the placement, Synopsys *Design Compiler* and *ICCompiler* respectively. In-house code written in TCL has been then used to drive and synchronize the entire CG-PG flow. As benchmarks we used a sub-set of circuits extracted from the ITC'99 suite (only the largest ones) and a unit of a network-on-chip (NoC) design, namely, a 10x10 switch[2].

[2] Main function of the switch unit is to analyze the incoming data packets, determine the source and the destination device of the packets, and forward them appropriately.

Results collected in Table 1 prove the superiority of our approach w.r.t. state-of-the-art solutions. The design metric used for the comparison is given by the number of cells that can be simultaneously clock- and power-gated avoiding timing degradation, while maintaining compatibility with the physical layout rules. Obviously, the larger the number of cells one can gate, the larger the power-savings. In Table 1, the label *CG+PG cells* indicates absolute values, while the label *CG+PG cells* represents the corresponding percentage out of the total number of cells in the design. The comparison is done using the following testers: a cell-by-cell approach[15], self-gated CMOS gates are used, (label *single-cell* in Table 1); a row-based scheme [5], where each layout row is supplied with a single virtual ground (label *single-row* in Table 1); our sub-row solution (label *sub-row* in Table 1). To notice that the single-cell is just an emulation of the ideal case where each and every cell can be ideally driven by an independent signal; as a matter of fact its implementation is impracticable in our technology which does not provide such feature. Nevertheless, it gives an asymptotic measure of the CG-PG potentials.

**Table 1.** Results for the integration of clock gating and power gating with 15 ST columns

| Circuit | CG+PG cells | | | CG+PG % | | |
|---|---|---|---|---|---|---|
| | Single-Cell | Sub-Row | Single-Row | Single-Cell | Sub-Row | Single-Row |
| b17 | 19558 | 16238 | 0 | 96.09 | 79.78 | 0 |
| b18 | 6248 | 5010 | 0 | 73.13 | 58.64 | 0 |
| b19 | 12734 | 9075 | 0 | 70.84 | 50.49 | 0 |
| switch | 14702 | 6591 | 0 | 72.47 | 32.49 | 0 |
| b20 | 8112 | 449 | 0 | 87.66 | 4.85 | 0 |
| b21 | 8313 | 508 | 0 | 88.30 | 5.40 | 0 |
| b22 | 12141 | 555 | 0 | 86.75 | 3.97 | 0 |

A first observation relies on the results obtained through the single-row approach. Since the timing-driven placement algorithms do not work in terms of entire rows, using a row-based scheme none of the rows in any circuit contain cells belonging to a unique cluster, and then, none of the rows can be subjected into concurrent CG and PG. A viable solution is to route multiple virtual ground rails for each and every row (one per cluster), but, as discussed in [5], this would imply substantial timing overhead (mote than 25%); while here the target is zero-delay overhead. Such a goal is achieved by our sub-row scheme. Let us consider the first four benchmarks: *b*17, *b*18, *b*19 and *switch*. On average, the sub-row method shows 55% of cells that can be gated (79.79% for the best case, the *b*17 32.49% for the worst case, the *switch*). These numbers prove the effective efficiency of the proposed STI by themselves, and gain even more importance when considering that the single-row strategies do not allow any gating.

It is worth mentioning that the success of the integration procedure mainly depends on the topological structure of the circuit. Considering the benchmarks

$b20$, $b21$ and $b22$, for instance, the yield falls to a mere 4.74%. Even though they have considerable amount of cells that satisfy the conditions for CGPG integration (see column *single-cell*), the resulting CG clusters result to be heavy interlaced, namely, they show large intersections. Resulting effect is that most of the cells are left un-gated. In this case, a potential solution is to make the intersections as autonomous clusters driven by ad-hoc control signals generated by intersecting the enable signals [6]. But this would open a complete new optimization scenario that is subject of on-going research.

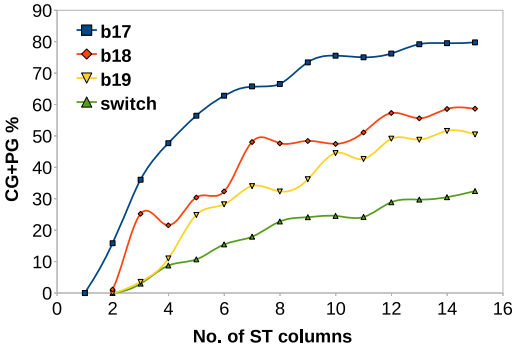



**Fig. 6.** Change in CG+PG % for increasing number of ST columns

To conclude this evaluation section, we would like to highlight that previous results have been obtained splitting the layout in a fixed number of different columns, i.e., 15. The latter is the best case for the concurrent CG-PG integration, but also the worst case in terms of area (larger silicon area blocked by the ST columns). Adjusting this parameter may therefore help to reach the optimal tradeoff. Fig. 6 gives the change in the amount of concurrency of CG and PG (CG+PG %) for increasing number of columns. As general trend, CG+PG % increases almost linearly at initial stages, later it starts to saturate. Considering the benchmark $b17$ (topmost curve in the plot), for instance, one can observe that increasing the number of rows from 10 to 15 helps to increase the CG-PG yield only marginally (from 74.3% to 79.78%). Therefore, using only ten rows could help to make the layout more compact still preserving the power-savings. Future releases of the tools will help to automatically identify the optimal number of rows.

## 5 Conclusions and Future Work

We have presented a sub-row STI methodology for effective integration of CG and PG. The proposed approach, although being fully compliant with the layout rules imposed by industrial semi-custom row-based design styles, this solution is aware of the algorithms adopted by state-of-the-art placement algorithms. This helps to minimize layout disruption and cell displacing while achieving zero delay

overhead. Experimental results show that, the amount of concurrency in CG and PG can reach more than 79% in the best case, thereby indicating the effectiveness and efficiency of the method.

The future development of this work will concentrate on further augmentation of the amount of CG and PG concurrency by considering the groups of cells belonging to the intersections of the clusters and isolating them as independent CGPG-clusters [6] distributed through the sub-rows of the layout. The control signals for the new formed clusters should be the combination of the enable signals used for the corresponding CG cells. Also the reduction of the layout segmentation by identifying the adjacent sub-rows that contain cells belonging to the same clusters, and merging them so to share the same virtual-ground rails, can provide more compact structure.

## References

1. Babighian, P., Benini, L., Macii, A., Macii, E.: Post-layout leakage power minimization based on distributed sleep transistor insertion. In: Proceedings of the 2004 International Symposium on Low Power Electronics and Design, ISLPED 2004, pp. 138–143 (2004)
2. Benini, L., De Micheli, G.: Automatic synthesis of low-power gated-clock finite-state machines. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 15(6), 630–643 (1996)
3. Benini, L., De Micheli, G., Macii, E.: Designing low-power circuits: practical recipes. IEEE Circuits and Systems Magazine 1(1), 6–25 (quarter 2001)
4. Benini, L., Siegel, P., De Micheli, G.: Saving power by synthesizing gated clocks for sequential circuits. IEEE Design Test of Computers 11(4), 32–41 (winter 1994)
5. Bolzani, L., Calimera, A., Macii, A., Macii, E., Poncino, M.: Enabling concurrent clock and power gating in an industrial design flow. In: Design, Automation Test in Europe Conference Exhibition, DATE 2009, pp. 334–339 (April 2009)
6. Bolzani, L., Calimera, A., Macii, A., Macii, E., Poncino, M.: Placement-aware clustering for integrated clock and power gating. In: IEEE International Symposium on Circuits and Systems, ISCAS 2009, pp. 1723–1726 (May 2009)
7. Calimera, A., Benini, L., Macii, A., Macii, E., Poncino, M.: Design of a flexible reactivation cell for safe power-mode transition in power-gated circuits. Circuits and Systems I: Regular Papers, IEEE Transactions on 56(9), 1979 –1993 (2009)
8. Garrett, D., Stan, M., Dean, A.: Challenges in clock gating for a low power asic methodology. In: Proceedings of International Symposium on Low Power Electronics and Design, pp. 176–181 (1999)
9. Ikebuchi, D., Seki, N., Kojima, Y., Kamata, M., Zhao, L., Amano, H., Shirai, T., Koyama, S., Hashida, T., Umahashi, Y., Masuda, H., Usami, K., Takeda, S., Nakamura, H., Namiki, M., Kondo, M.: Geyser-1: A mips r3000 cpu core with fine-grained run-time power gating. In: 15th Asia and South Pacific, Design Automation Conference (ASP-DAC), p. 369–370 (January 2010)
10. Keating, M., Flymm, S., Aitken, R., Gibbon, A., Shi, K.: Low Power Methodology Manual for System-on-Chip Design. Springer, Heidelberg (2007)
11. Macii, E., Bolzani, L., Calimera, A., Macii, A., Poncino, M.: Integrating clock gating and power gating for combined dynamic and leakage power optimization in digital cmos circuits. In: 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, DSD 2008, p. 298–303 (September 2008)

12. Roy, K., Mukhopadhyay, S., Mahmoodi-Meimand, H.: Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits. Proceedings of the IEEE 91(2), 305–327 (2003)
13. Sathanur, A., Benini, L., Macii, A., Macii, E., Poncino, M.: Fast computation of discharge current upper bounds for clustered power gating. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 19(1), 146–151 (2011)
14. Sathanur, A., Pullini, A., Benini, L., Macii, A., Macii, E., Poncino, M.: Timing-driven row-based power gating. In: Proceedings of the 2007 International Symposium on Low Power Electronics and Design, ISLPED 2007, pp. 104–109 (2007)
15. Usami, K., Ohkubo, N.: A design approach for fine-grained run-time power gating using locally extracted sleep signals. In: International Conference on Computer Design, ICCD 2006, pp. 155–161 (October 2006)

# A Methodology for Power-Aware Transaction-Level Models of Systems-on-Chip Using UPF Standard Concepts*

Ons Mbarek, Alain Pegatoquet, and Michel Auguin

LEAT, University of Nice-Sophia Antipolis-CNRS,
250-Rue Albert Einstein, Bâtiment-4, 06560-Valbonne, France
{mbarek,pegatoquet,auguin}@unice.fr

**Abstract.** Building efficient and correct system power management strategies relies on efficient power architecture decision-making as well as respecting structural dependencies induced by such architecture. Transaction Level Modeling allows a rapid exploration, verification and evaluation of alternative power management architectures and strategies. This paper introduces an efficient methodology for making system power decisions at Transaction-Level (TL) by adding and verifying power intent and management capabilities into TL-models. A generic framework that abstracts relevant concepts of the IEEE 1801 (UPF) standard and implements assertion-based contracts is used throughout the methodology. A TL-model example is considered to validate the methodology.

**Keywords:** Transaction Level Modeling (TLM), IEEE 1801 (UPF) standard, power-aware design and verification, power intent specification, assertion-based contracts, power domain.

## 1 Introduction

As systems-on-chip (SoCs) grow in embedded functionalities and complexity, the importance of power management increases as well. Approaches like power gating and adaptive voltage scaling are widely used to reduce power in SoCs. The most basic form of these approaches is to partition the chip logic into multiple voltage regions or power domains, each with its own power supply and power control unit. This leads to introduce additional power elements mediating interfaces between power domains [1]. These elements contribute in defining the per-domain power states. So, the different operating power modes of a SoC can be seen as the different combinations of these power domains' states. They are controlled by a power management block to implement an appropriate power management strategy. A good strategy would be to efficiently enable functional resources based on the required application load (e.g. reading an email on a smartphone or capturing pictures). In such case, each operating

* This work is supported by the French National Research Agency (ANR) project HELP bearing reference ANR-09-SEGI-006.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 226–236, 2011.

power mode corresponds to a different system use case. This requires a good understanding of both hardware and software parts of the final system, as well as their interactions with power management. Indeed, placement and behavior of each element in power architecture can create dependencies between power domains which constraints the legal operating power modes. For example, a wrong placement of power elements or selection of irrelevant behavior for them can alter the intended system functionality. Therefore, verification becomes compulsory and constitutes a complex task. The recent UPF (Unified Power Format) standardization [2] enables defining and verifying the power intent that represents the power management architecture (set of power domains, power switches, supply networks…) and strategy (legal system power modes and power transitions) specifications, throughout a RTL to GDSII design flow [3]. However, the earlier power intent is added to the system, the higher the power reduction and the easier the verification would be. Transaction-Level (*TL*) models [4] provide faster simulation times than Register-Transfer Level (*RTL*) ones and are mainly dedicated to verify the whole system including the embedded software. So, specifying power intent and the corresponding power management block with above mentioned capabilities at Transaction-Level is a promising solution to early validate a power management structure and to handle complex verification issues. Besides, this allows a rapid exploration of different power design alternatives and an early decision-making of the most energy-efficient one before starting design at RTL. Unfortunately, taking advantage from UPF standard capabilities at Transaction-Level is not possible since there is still no power-aware TL-simulator understanding power intent and constraints as captured by UPF.

In this paper, we propose a "*PwARCH*" generic framework that abstracts relevant power concepts specified by the IEEE 1801 (UPF) standard. By following a four-stage methodology, this framework allows adding high-level power architecture to a TL-model and building a power management strategy upon it. A verification process has been also built in order to check different types of contracts which express properties between power and functional architecture and are implemented using assume and guarantee assertion types. Moreover, our methodology enables exploring different power design alternatives and choosing the most energy-efficient one.

The paper is organized as follows. Section 2 gives an overview of related works, and highlights our contributions. Section 3 describes the "*PwARCH*" framework. In section 4, our methodology flow is explained. The case study in section 5 applies our proposal to a TL-model example. Finally, section 6 summarizes our paper.

## 2 Related Work

Many ad-hoc approaches have addressed power modeling and estimation at different transaction levels. Since there is no standard way to create TL-power models for a system or IP cores, each of these approaches support different criteria to explore power profiles in a TLM context. Most of the proposed approaches focus on the instrumentation of existing TL simulation platforms to apply such profiles. Usually, power profiles are fed with power consumption metrics coming typically from IP datasheets or low level simulations. Instrumentation-based solutions mainly range from transaction-based power modeling solutions to component-centric ones. Authors

in [5] propose a method to build transaction-based models which resume all types and granularities of transfers between the different blocks of an existing TL-platform as well as the different relationships among them. However, [6] and [7] focus on power modeling of each hardware component in a TL-platform. These two solutions are not generic enough since they suppose that a cycle-accurate simulation platform already exists, which is not always the case. Similarly, authors in [8] adopt a state-based power profiling technique applied to each component in a TL-platform. Nevertheless, this method assumes that DPM (Dynamic Power Management) and DVFS (Dynamic Voltage and Frequency Scaling) power architectures of each considered IP core are available. Contrary to this work, we consider in this paper a strong relationship between specific low-power architecture, a system power management strategy controlling it and the resulting power savings. Our approach proposes a way to investigate this relationship starting from TL-models. Compared to previously mentioned works, it is also an instrumentation-based approach. Particularly, it is generic enough to be applied to any TL-model. Furthermore, our power models are neither component-based ones, nor transaction-based. Instead, we use state-based models relying on power-domain reasoning. Such reasoning has been involved throughout the proposed methodology by abstracting relevant UPF concepts to adapt them to a TL modeling usage. Some works [9], [10] have considered IEEE 1801 (UPF) specifications and simulation semantics to achieve simulation-based or formal power-aware verification. But, as far as we know, none of them have used UPF semantics at Transaction-Level. The key difference with mentioned state of the art works is that we do not only perform early TL power estimation, but we also deal with power-aware design, management and verification issues at this modeling level.

## 3 Overview of the *PwARCH* Framework

Our main objective is to augment an existing SystemC/TLM model with power intent and control capabilities including power-aware verification. The "*PwARCH*" framework has been developed as a static software library to help achieving that goal in an instrumentation-based manner. Fig. 1 depicts its generic set of C++ classes where each group serves a specific purpose. The main features of the framework are explained in the following sections:

**Abstracting UPF Concepts.** The UPF standard [2] offers semantics for implementation and verification of low power design intent. It describes a HDL (Hardware Description Language) functionality subset of a power distribution and the behavior of its power elements in a side file. This file serves the entire RTL to GDSII design flow and can be modified throughout this flow in an incremental process (Fig. 2). Fig. 3 gives an example [12] of the main power elements used by UPF semantics to specify the power design intent explained as follows: each power domain (e.g. *CRC_GEN*) is supplied by a power net (e.g. *VDD_HIGH*) and overlays a functional block (e.g. *Checker*). Power switches are in charge of shutting down or powering up the power domains depending on the value changes of their control signals (e.g. *crc_sd* signal). Retention registers (*RR*) are used to save the internal state of crucial modules when they are switched off. Level shifters (*LS/ELS*) are used for communication between domains with different supply voltages.

Isolation elements (*ISO*) are used to avoid undefined signal values at the output of a power-gated domain. A power controller must be defined as a HDL functional block controlling all these power elements through specific control signals.

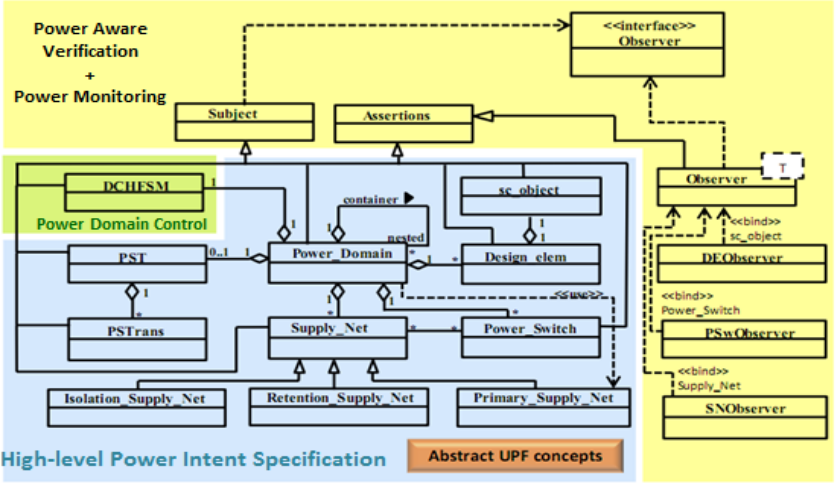



**Fig. 1.** *PwARCH* framework structure: UML class diagram

As far as we know, TL-simulators that capture UPF semantics are not available yet. Moreover, we can observe that only a subset of power components from UPF semantics can be used to express TL power intent, as their behavior is compatible with the TLM system view. Despite this compatibility, some attributes and simulation behavior of UPF components considered at TL must be even abstracted. For instance, power switch controls have to be included transparently in a TLM simulation. The "*PwARCH*" framework helps to build abstracted UPF specifications according to power gating and multi-voltage power requirements. By starting power design intent at TL, we aim at generating a UPF constraints file describing the most energy-efficient system power architecture, and being the golden low power reference to RTL design team (Fig. 2). As shown on Fig. 1 (abstract UPF concepts part), composition hierarchy between abstract UPF components in "*PwARCH*" is the same as in UPF-defined semantics. For instance, supply nets are instantiated in the context of an existing power domain. Unlike UPF, we added other power components (such as Design_elem (*DE*) objects) and composition constraints (e.g. Power State Table (*PST*) objects are only instantiated in the context of a composite Power Domain). This facilitates managing hierarchical structure

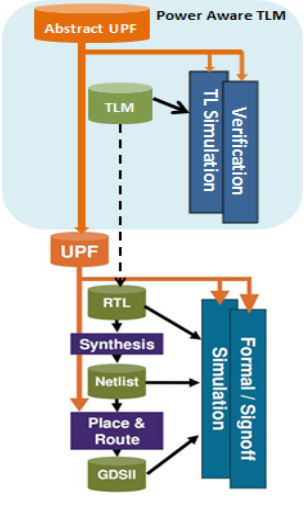



**Fig. 2.** Starting UPF flow from Transaction-Level

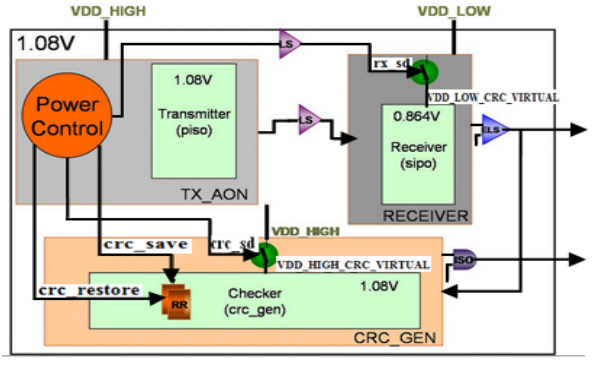



**Fig. 3.** Example of UPF concepts

and control, and mapping the functional design to the power-aware one. Basic features of each power component that are captured in our framework are highlighted hereinafter using illustrations concerning our case-study in section 5:

Each functional SystemC module is attached to a design element (***DE***) object. The key element in our approach is a power domain (***PD***) object which consists in a collection of *DE* objects that share the same primary supply nets. Each *PD* can be controlled individually. A hierarchical instantiation of *PD* objects is allowed to facilitate the control of their states and their attached power components. For that, we defined a *PD* of type *container* that includes at least another *PD* instantiated in its context (e.g. *PD_top_level* in Fig. 8 (b)), and a *PD* of type *nested (*e.g. in Fig. 8 (b), *PD_SRAM* is *nested* in *PD_AO).* A *nested PD* can also be a *container* for other *PD* objects (e.g. *PD_Periph* is *nested* in *PD_AO*, and is a *container* for *PD_GPIO* (Fig. 8 (b)). Fig. 7 (b) shows an example of such constructed hierarchy corresponding to the power design in Fig. 8 (b). To apply power gating technique, abstract power switch (***PSw***) objects must be added at the boundary of the switched *PDs*. Each one is characterized by at least one input supply net and only one output supply net. Controlling the state of a *PSw* is equivalent to changing the state of its output supply net. Supply net (***SN***) objects can be either of type *primary* (so of type power or ground net), or *retention*, or *isolation*. In Fig. 8 (b), *VDDSoC* is a *primary SN* for *PD_AO*, whereas *ret* is a *retention SN* for *PD_SRAM* and *PD_GPIO*. Each *SN* is characterized by a (state,voltage) pair. Particularly, the output *SN* of a *PSw* is of type *switched* which is in addition characterized by a (state,input net) pair used to control states of *PSw* objects. A *Switched* supply net must be specified as the *primary* power net of a power-gated *PD.* (e.g. in Fig. 8 (b), the output *SN* of the power switch $S_2$ named *VDDPeriph_SW,* is the *primary* power net of the *PD_Periph*). Note that in this work, the concept of voltage domain is merged with the concept of power domain. Hence, depending only on its attached *primary* power net, a *PD* can be *power-gated* if its power net is of type *switched* (e.g. PD_SRAM in Fig. 8 (b)). In this case, it can be entirely powered-down. Moreover, a *PD* can be of type *voltage-scaled* if its attached *primary* power net has more than one state (e.g. *PD_CPU* in Fig. 8 (b) has a *VDDCPU* power net with two possible voltage values ($V_H$ /$V_L$)). Otherwise, a *PD* can be *non-scaled* if it has a unique *primary* power net with a single state and not of a *switched* type (e.g. *PD_AO* in Fig. 8 (b) is a *non-scaled PD* having *VDDSoC* as a *primary* power net).

A power state table (***PST***) object is a two-dimensional static table that captures the global power states of a *container PD.* An example of a *PST* is given in Fig. 7 (a) according to the power design of Fig. 8 (b). Columns of a *PST* represent *PD*s local power states in terms of their power net states. We denote each of them by ***LPS***

(*Local Power State).* Lines of a *PST* represent global power states of the including *container PD*. Each of them is denoted by ***GPS*** *(Global Power State)* and consists in one legal combination of *LPS*s of the *nested PD*s. Contrary to a *LPS*, a *GPS* refers to a functional tasks set matching a specific system use case. According to a specified *PST*, legal transitions between *GPS*s must be specified using ***PSTrans*** objects. Fig. 7 (c) illustrates example of legal transitions specified according to the *PST* of Fig. 7 (a).

**Building Power-aware Verification.** Our verification solution is not only an UPF-like one that mainly focuses on verifying functional failure and wrong placement of power components. It emphasizes also on verifying additional interfaces resulting from our methodology. Depending on the types of communicating interfaces, possible errors have been classified into well-defined categories of contracts as explained later. According to our approach, each interface can be characterized by a set of assume/guarantee properties [11] that form the component's contract. As we target a simulation-based verification, these contracts consist in executable specifications that are monitored at runtime. In "*PwARCH*", assume and guarantee properties are seen as two types of assertions that form a contract and raise an exception when a property is violated during simulation. A generic class *Assertions* has been defined (Fig. 1) with corresponding *Assume* and *Guarantee* methods. A *Satisfy* method can be called as well by *Assume* or *Guarantee* methods to verify that a specific condition is satisfied.

**Power Analysis and Estimation.** A power monitor is attached to each *PD* object and is automatically updates its power values using its power structure and its *LPS*. Each power monitor is triggered when its *PD* receives a *power event (PwE)*. When such an event occurs, it provokes a change in at least a non-switched *SN* state or a *PSw* state. To capture such events, an observer like a *SNObserver* object is attached to each non-switched *SN* and a *PSwObserver* object is attached to each *PSw* (Fig. 1). Due to the hierarchical *PD*s construction, a *PD* state change will automatically and recursively update power values of *PD*s in higher levels of hierarchy.

## 4 A Power Domain Based Methodology

We propose a power-domain-based methodology to add power management capabilities to the different functional parts of a SoC described in SystemC/TLM. Fig. 4 illustrates the overall flow of the proposed methodology. It is mainly composed of three stages processed sequentially as indicated in Fig. 4. A fourth stage is dedicated for verification and occupies an orthogonal position regarding to previous stages. The methodology is iterative allowing the designer to explore different power design alternatives and retain the most energy-efficient one. Note also that the different stages are complementary and dependent of "*PwARCH"* library. In the following, the main purposes of each stage are explained.

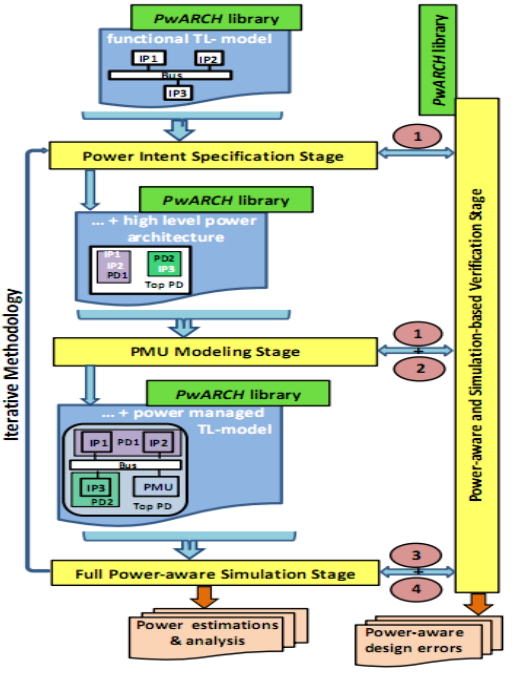



**Fig. 4.** TL power-aware methodology flow

**Power Intent Specification Stage.** Here the designer configures the system power intent by instantiating adequate objects from "*PwARCH*". In a first phase, transaction traces resulting from a TL functional simulation of the embedded software inform about possible correlations between HW blocks in order to identify power reduction opportunities (e.g. gathering for instance the strongly correlated blocks in a same *PD)*. Furthermore, the existing HW architecture is mapped to a power one by attaching *DE* objects to functional modules. Technology data and available low-level power values are attached to *DE* objects used to update power consumption values. In a second phase, according to power domains boundaries and transaction traces, a system power management strategy is established by specifying a *PST* and its *PSTrans* set. Here, each possible combination of *PDs* states is bound to an appropriate part of the SW flow, with respect to inter-power objects dependencies.

**Power Management Unit Modeling Stage.** To control local and global power states, a TL power management unit (***PMU***) is modeled as an additional SystemC module communicating through bus transactions with other modules of the platform. Since it uses a *PST*, a *PMU* activity only impacts the global power state of the *container PD* of this *PST*. For each *GPS* in a *PST* will correspond a transaction in the functional SystemC code that must be preceded by a power control (***PwCTr***) one. A *PwCTr* transaction writes to a *PMU* control register requesting hence to set the *container PD* in a specific *GPS* (Fig. 5). The *DE* issuing such transaction must be blocked as long as the *PMU* performs required transitions. We consider that each *DE* holds a particular event and remains waiting for it whenever it issues a *PwCTr*. When the *PMU* finishes its activity, it notifies this event so that the suspended *DE* resumes its activity. Our *PMU* generic model (Fig. 5) is mainly composed of a power manager (***PM***) and a set of domain power controllers (***DPCs***). Each *DPC* module changes the *LPS* of a *power-gated* domain between sleep and wake-up states (e.g.

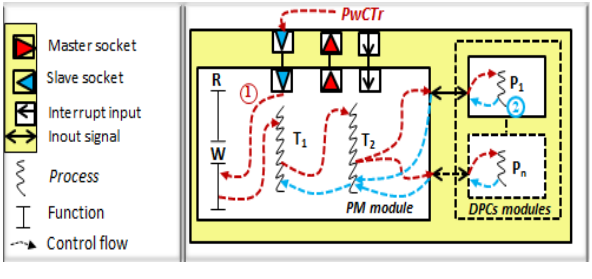



**Fig. 5.** Generic PMU TL-module

process $P_1$ in Fig. 5). The *PM* module only changes the *LPS* of non-switched *PD*s ($T_1$ in Fig. 5) and requests adequate *DPCs* to change their domains states ($T_2$ in Fig. 5).

**Power-aware Verification Stage.** A three-step verification process is used to check after each stage that interactions among some components obey to specific contracts (Fig. 4). Three types of components are handled throughout the methodology. *Pw components* represent UPF-based concepts. *Functional components* consist in hardware TL-modules. *Mixed components* (*PMU* modules and their *PM* and *DPCs* submodules) are *functional components* using *pw components* to apply a system power management strategy. For lack of space, contracts which are supported by our approach are only briefly described in the following. Contracts of type 1 specify interfaces among *pw components* so as to ensure the respect of hierarchy of composition and structural dependencies among them (e.g. a *PSw* attached to a non-valid *PD*). They are already included in "*PwARCH*". Contracts of type 2 verify interactions among *mixed components*, and also between *mixed* and *pw components*, so as to ensure the *PMU* correct functionality (e.g. a transition to a *GPS* is required during simulation, but it is set as an illegal *PSTrans*). Contracts of type 3 specify interfaces between *functional* and *pw components* to verify that a TL-model enters correctly a *GPS* (e.g. noting an activity in a TL-module (*DE*) belonging to a powered-down *PD*). Contracts of type 4 specify interfaces between *functional* and *mixed components* to check that the added power features do not alter the intended functionality (e.g. issuing a transaction to a module which is undergoing a power state transition). Actually, contracts 2, 3, and 4 are manually added.

**Full Power-aware Simulation Stage.** At this stage, we simulate the resulting TL power-managed behavior. It is processed in parallel with the verification one. During simulation, functional coherence between the augmented TL-model and the power design needs to be verified. The system power-aware behavior is proved coherent if no verification properties are violated during simulation. Then, power logs are generated and plotted. They are used to compare different power management solutions and select the most energy-efficient power design.

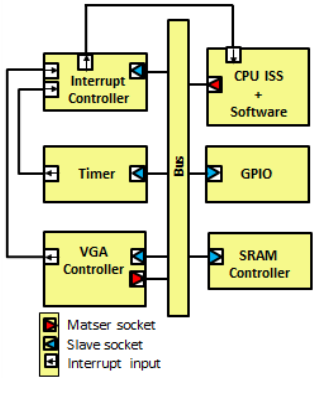


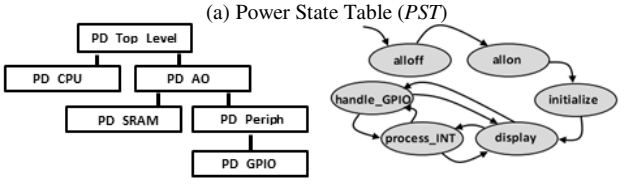

**Fig. 6.** Case-study platform

## 5 Application To a Case Study

To demonstrate our methodology, we consider an existing Approximately-Timed (AT) [4] TL-platform (Fig. 6) with no power management features. The embedded application implements Conway's game of life. First, a software flow analysis is performed in order to determine possible system use cases. In the following, we give scenarios that match with *GPS*s of the *PST* shown on Fig. 7 (a), and according to the power design of Fig. 7 (b). The CPU computes a first

| GPS \ LPS | VDDSoC | VDDCPU | VDDSRAM_SW | VDDPeriph_SW | VDDGPIO_SW |
|---|---|---|---|---|---|
| alloff | OFF | OFF | RAM_OFF | PER_OFF | GPIO_OFF |
| allon | ON | ON_H | RAM_ON | PER_ON | GPIO_ON |
| initialize | ON | ON_H | RAM_ON | PER_OFF | GPIO_OFF |
| display | ON | ON_L | RAM_ON | PER_ON | GPIO_OFF |
| process _INT | ON | ON_H | RAM_ON | PER_ON | GPIO_OFF |
| handle_GPIO | ON | ON_H | - | PER_ON | GPIO_ON |

(a) Power State Table (*PST*)




(b) Power domains hierarchy (c) Set of *PSTrans*

**Fig. 7.** Application of the power intent specification stage

image by reading and writing from/to the SRAM (*initialize* scenario). Then, peripherals are initialized (*allon* scenario). The VGA controller uses a double-buffer to avoid visual glitches when the image changes. First, it reads the image from the memory (first buffer) and displays it (*display* scenario). Games of life iterations are cadenced by the timer. Hence, an interrupt which is raised by the timer, is driven to the interrupt controller which drives it to the CPU. Then, the CPU handles this interrupt by computing a new image in a second buffer while communicating again with the SRAM (*process_INT* scenario). Henceforth, the VGA Controller is informed by the CPU about the new image address, and will display this new image after the display reaches the end of the screen (*display* scenario again). A button mapped as a GPIO is checked periodically (*handle_GPIO* scenario). This SW flow is periodically repeated. As shown on the Fig. 7, different power architecture alternatives have been elaborated and evaluated while taking into account this SW flow. Fig. 7 shows an application of the power intent specification stage of our methodology according to alternative (b) of Fig. 8. HW components of this TL-model have been implemented on a Virtex-4 FPGA device. The Xilinx Power Estimator tool has been then used to

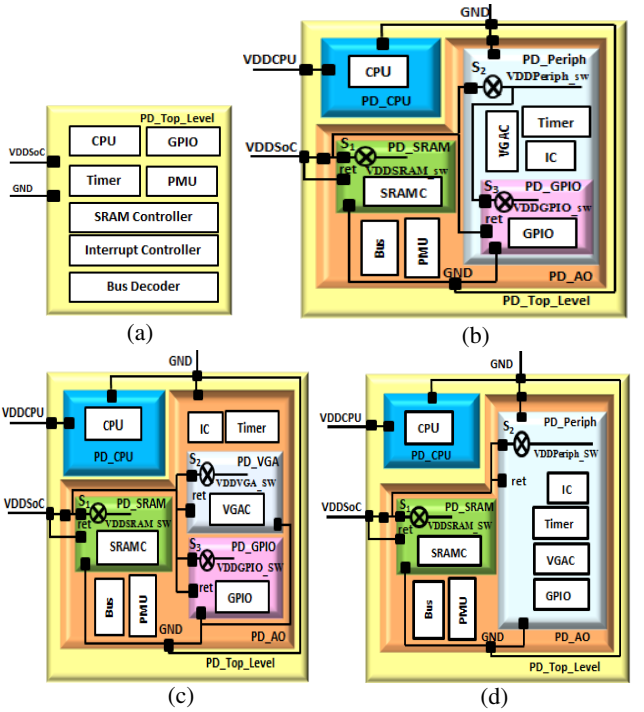



**Fig. 8.** Power-aware architecture alternatives

get technology-dependent power characteristics (such as leakage current and load capacitance) which are used to feed power models of each *DE*. Results show that (b), (c) and (d) alternatives in Fig. 8 provide at least 90% of energy savings compared to a unique power domain design ((a) alternative). The (b) alternative represents the most energy-efficient power domains partitioning since about 58% of energy savings is observed compared to (d) alternative and 7.3% compared to (c). Furthermore, the obtained power-aware simulation speed remains similar compared to the non-instrumented version. For instance, simulation time for alternative (b) is only 0.03% slower than alternative (a).

## 6 Conclusion and Future Work

We have presented a novel, efficient and generic methodology to augment a TL-model with power including verification capabilities. This methodology mainly aims at an early decision-making of the most energy-efficient and correct power management design alternative. It allows also mapping TL hardware architecture to a power one using UPF-based concepts in a generic "*PwRACH*" framework. Future works will focus on automating the insertion of remaining types of contracts and the power intent construction of power design alternatives.

## References

1. Keating, M., Flynn, D., Aitken, R., Gibbons, A., Shi, K.: Low Power Methodology Manual: for System-on-Chip Design (integrated circuits and systems). Springer, Heidelberg (2007)
2. Unified Power Format (UPF 2.0) Standard: IEEE standard for design and verification of low power integrated circuits. IEEE 1801™ (March 27, 2009)
3. Bembaron, F., Kakkar, S., Mukherjee, R., Srivastava, A.: Low Power Verification Methodology Using UPF. In: Proc. of Design & Verification Conference & Exhibition (DVCon), San Jose, CA, pp. 228–233 (2009)
4. Open SystemC initiative. SystemC Transaction Level Modeling Library 2.1.0 (2009), `http://www.systemc.org`
5. Dhanwada, N., Lin, I.-C., Narayanan, V.: A Power Estimation Methodology for SystemC Transaction Level Models. In: 3rd IEEE/ACM/IFIP Conference on Hardaware/Software Codesign and System Synthesis, pp. 142–147 (2005)
6. Lee, I., Kim, H., Yang, P., Yoo, S., Chung, E.-Y., Choi, K.-M., Kong, J.-T., Eo, S.-K.: PowerViP: Soc Power Estimation Framework at Transaction Level. In: 11th Asia and South Pacific Design Automation Conference (ASP-DAC), Japan, pp. 551–558 (2006)
7. Ben Atitallah, R., Niar, S., Dekeyser, J.L.: MPSOC Power Estimation Framework at Transaction Level Modeling. In: 19th International Conference on Microelectronics (ICM), Egypt, pp. 245–248 (2007)
8. Lebreton, H., Vivet, P.: Power Modeling in SystemC at Transaction Level, Application to a DVFS Architecture. In: Proc. of the 2008 IEEE Computer Society Annual Symposium on VLSI, France, pp. 463–466 (2008)
9. Hazra, A.S., Mitra, A., Dasgupta, P., Pal, A., Bagchi, D., Guha, K.: Leveraging UPF-Extracted Assertions for Modeling and Formal Verification of Architectural Power Intent. In: 47th Design Automation Conference (DAC), Anaheim, CA, pp. 773–776 (2010)

10. Trummer, C., Kirchsteiger, C.M., Weiss, R., Dalton, D., Pistaur, M.: Simulation-based Verification of Power Aware System-on-Chip Designs Using UPF IEEE 1801. In: 27th NORCHIP Conference, Trondheim, Norway, pp. 1–4 (2009)
11. Meyer, B.: Applying "design by contract". IEEE Computer 25, 40–51 (1992)
12. Magic Blue Smoke blog, http://synopsysoc.org/magicbluesmoke/2008/05

# Unified Gated Flip-Flops for Reducing the Clocking Power in Register Circuits

Takumi Okuhira[1] and Tohru Ishihara[2]

[1] Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395 Japan
[2] Kyoto University, Yoshidahonmachi, Sakyo-ku, Kyoto 606-8501 Japan

**Abstract.** Since the clocking power consumption in today's processors is considerably large, reducing the clocking power consumption contributes to the reduction of the total power consumption in the processors. Recently, a gated flip-flop is proposed for reducing the clocking power consumption of flip-flop circuits. The gated flip-flop employs a clock-gating circuit which cuts off an internal clock signal if the data stored in the flip-flop does not need to be updated. Although this reduces the clocking power consumption, the power dissipated in the clock-gating circuit is still large. For reducing the power dissipated in the clock-gating circuit, this paper proposes a technique for unifying the multiple clock-gating circuits, which reduces the overhead of the clock-gating circuit. Post-layout simulation results obtained using a commercial embedded processor which employs our unified gated flip-flop demonstrate that our technique reduces the power consumption of a core part of the processor by 25% on average and 33% at the best case compared to the same processor with the conventional gated flip-flop.

## 1 Introduction

Ever increasing performance of microprocessors results in an explosion of the power consumption in the microprocessors. In a typical microprocessor, the power dissipated in register circuits is dominant [1]. Figure 1 shows a breakdown of the power contributions in a commercial RISC-type microprocessor which is synthesized using a commercial 65nm process technology. As can be seen from the Figure, the power consumption of register circuits is around 40% of the total power consumption of the microprocessor. This means that the reduction of the power dissipated in the register circuits largely contributes to the reduction of the total power consumption of the microprocessor. Another important observation is that more than 80% of the power consumption in the register circuits is dissipated due to clock-signal transitions in flip-flop circuits. This is mainly because a probability of signal transitions in a clock port of a flip-flop is much higher than that in a data port of the flip-flop. Therefore, reducing the clocking power in the flip-flops contributes to the power reduction of microprocessors.

Clock-gating is widely used for reducing the power dissipated in the clock ports of the register circuits [2]. This reduces the probability of the clock-signal transitions in the flip-flop circuits. For example, SYNOPSYS Power_Compiler

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 237–246, 2011.

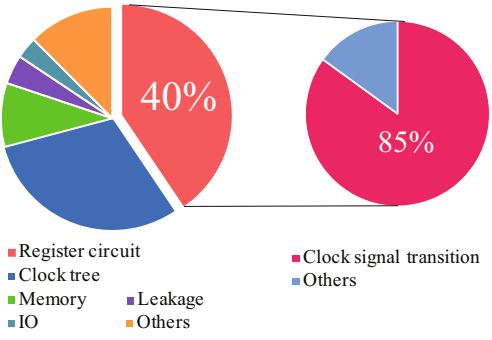



**Fig. 1.** Power Breakdown of Microprocessor

exploits an enable signal connected to a register circuit for cutting off the clock signal to the flip-flops [3]. The register here is defined as a set of flip-flops which share a common enable signal. If the enable signal is false, the clock signal is gated and any flip-flop in the register does not consume the clocking power. However, there still exists a room for reducing the clocking power, since this approach cannot separately stop the clock supply to a subset of the flip-flops in the register circuit which is controlled by a common enable signal. For making finer grained clock gating possible, a technique of gated flip-flop is proposed [4]. This technique makes it possible to perform bitwise clock gating. The gated flip-flop employs an internal clock-gating circuit which stops the clock supply to the flip-flop if an input value of the flip-flop is equal to a value stored in the flip-flop. The main drawback of the gated flip-flop is its large area and power overheads in the internal clock-gating circuit. This paper proposes a technique for unifying the multiple clock-gating circuits, which reduces the power and area overheads of the clock-gating circuit.

The lest of the paper is organized in the following way. Section 2 summarizes related work. Our idea for unifying multiple gated flip-flops is presented in Section 3. In Section 4, experiments and results using a commercial microprocessor are shown. Section 5 concludes this paper.

## 2 Related Work

### 2.1 Fine-Grained Clock Gating

Recent microprocessors used in many computer systems have a 32-bit or 64-bit data path. Although those processors normally perform 32-bit or 64-bit operations, narrow-width operations are still frequent. Brooks et al. [6] show that roughly 50% of the integer instructions in SPECint95 benchmark programs require less than or equal to a 16-bit precision, even though the underlying processor has a 64-bit data path. Since conventional processors always assign the full-bit width of a register entry even for the narrow-width operand, the upper

bits of the register entry are useless and dissipate wasteful clocking power in corresponding flip-flops. A technique presented in [6] detects these useless upper bits dynamically and cuts off the clock supply to these bits for reducing the wasteful clocking power consumption. The narrow-width operand is exploited in a concept called partially guarded computation [7]. The idea is to divide a functional unit into two parts, MSP (Most Significant Part) and LSP (Least Significant Part), and to perform only the LSP computation if the range of output data can be covered by LSP only. MSP computation is disabled dynamically by using clock-gating for saving the power consumption of the MSP. Similar idea is applied to register files [8]. The idea is to partition a register file into two subword banks. If an MSP of each register file entry is used for sign extension only, the clock supply to the MSP is gated for saving the power consumption. The techniques presented above drastically reduce the power dissipated for clocking as well as the power dissipated in functional units. However, the problem is that they involve a considerable overhead for detecting opportunities of clock gating. For example, all of above presented techniques need to detect all-zeros or all-ones in an input of the MSP, which involves considerable delay, area and power overheads. Another problem is that those techniques can be applied to registers on a data path only. Unlike those techniques, our technique aims at both reducing the overhead to dynamically detect the opportunities for clock gating and reducing the switching activities of clock ports in every registers including pipeline registers, a register file, status register and so on.

### 2.2 Low Power Flip-Flops

In the past, several low power designs for the master-slave flip-flop have been proposed, such as data look-ahead flip-flop [9], clock-on-demand flip-flop [10], and gated flip-flop [4]. The common idea among the above techniques is to insert conditional circuitry into their clock path to cut off the clock signal in a case that the inputs will produce no change in the outputs. Strollo et al. [4] have proposed two versions of the gated flip-flop. In the double-gated flip-flop, each of the master and slave latches has its own clock-gating circuit. The circuit consists of a comparator and a signal keeper. The comparator checks whether or not the current input value and the value stored in the flip-flop are different from each other. The signal keeper keeps an internal clock signal unchanged if the current input value and the value stored in the flip-flop are the same from each other. In the sequential-gated flip-flop, the master and the slave share the same clock-gating circuit instead of having their own individual circuit as used in the double-gated flip-flop. This drastically reduces the clocking power consumption. However, one of the most critical issues in the gated flip-flops is its power overhead in the clock-gating circuit. As the frequency of state transitions in the flip-flop increases, the power overhead by the clock-gating circuit increases. Therefore, if the value stored in the flip-flop needs to be updated frequently, the gated flip-flop consumes more power than a normal flip-flop. Another serious issue is its area overhead. This area overhead leads to an increase of a chip area and may consume more power. Our technique aims at reducing these overheads

by sharing the clock-gating circuit among multiple flip-flops. In [5], multiple flip-flops are combined together and a single internal clocking circuit is shared among the multiple flip-flops. This largely reduces the clocking power of the flip-flops. However, it does not aim at reducing the power consumption of the clock-gating circuit. Our technique reduces the power consumption of both the clock-gating circuit and the clocking circuit, which drastically reduces the power consumption of entire register circuits.

### 2.3 Preliminary Analysis

The left of Figure 2 shows an example of the conventional gated flip-flop. The gated flip-flop consists of a normal flip-flop and a clock-gating circuit for cutting off the clock supply to the flip-flop part if values of input D and output Q of the flip-flop are the same from each other. This reduces the clocking power if the value stored in the flip-flop does not need to be updated. However, if the state transition probability of the flip-flop is higher than a specific value, the gated flip-flop dissipates more power than a normal flip-flop. It is important to use the gated flip-flop to register bit whose state transition probability is less than the specific value for reducing the power consumption of the register circuits.

The right of Figure 2 shows the power consumptions of a normal master-slave flip-flop and a gated flip-flop for different state transition probabilities in flip-flops. Horizontal axis shows the state transition probability of the flip-flops. Vertical axis shows the power consumption of the flip-flops. The state transition probability represents the average number of state translations in a flip-flop per a clock cycle. Since the state transition in a flip-flop occurs at most once in a clock cycle, the state transition probability is no more than 1. As can be seen from the Figure, the gated flip-flop consumes less power than that of the normal flip-flop if the state transition probability is lower than 23%. This is because the power consumption in the clock gating circuit is less than the reduction of the clocking power in the flip-flop part of the gated flip-flop if the state transition probability is lower than 23%.

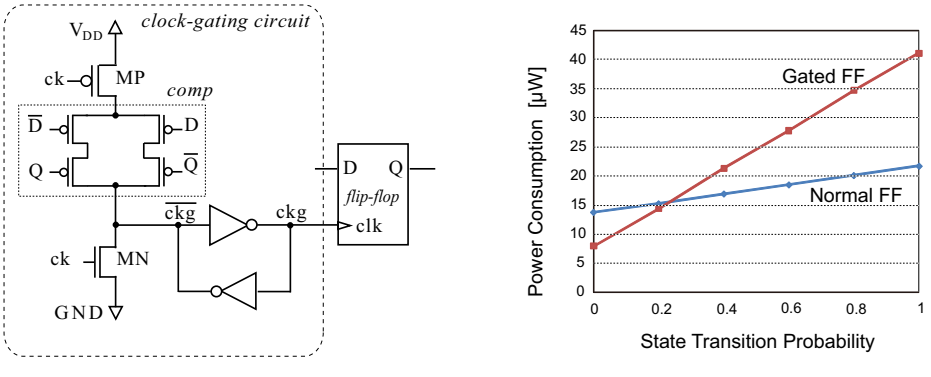



**Fig. 2.** Gated Flip-Flop [4] and its power consumption

# 3 Unified Gated Flip-Flop

## 3.1 Unification of Multiple Clock-Gating Circuits

Our idea for reducing the power overhead in the clock-gating circuit is to share a single clock-gating circuit among multiple flip-flops. An example of a 2-bit unified gated flip-flop is shown in Figure 3. For reducing the load capacitance of the clock input port per bit, multiple flip-flops share a *ck* pin. The opportunities for cutting off the clock signal are detected by a circuit composed of multiple $comp_i$s connected in parallel. The number of $comp_i$ circuits is equal to the number of flip-flops unified together. An internal clock signal *ckg* connected to flip-flops is activated if a value of $D_i$ is different from a value of $Q_i$. Therefore, if a value of a D port even in a single flip-flop differs from a value of a Q port in the same flip-flop, the clock is provided to all flip-flops. This implies that the switching probability in *ckg* port increases as the number of flip-flops unified increases. This may increase the clocking power consumption even though the load capacitance of *ck* port per bit is reduced by unifying multiple flip-flops.

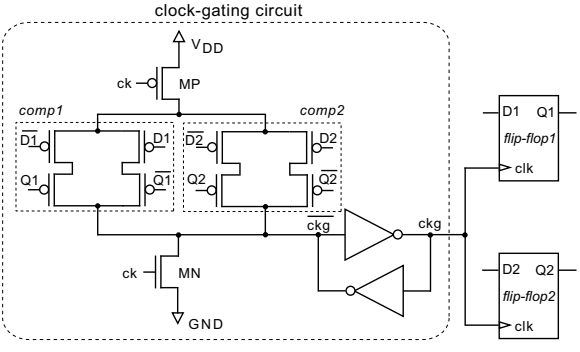



**Fig. 3.** Schematic of 2-bit Unified Gated Flip-Flops

## 3.2 Delay Results

Table1 shows setup time, hold time and C-Q delay of five different types of flip-flops in picosecond. All results are obtained through circuit simulation using HSPICE of SYNOPSYS. SPICE netlists used in the simulation are extracted from the physical layout designed by ourselves. The target process technology is $0.18\mu m$ CMOS process and a supply voltage used is a 1.8V. Normal FF, CONV GFF, 2-BIT UGFF, 3-BIT UGFF, and 4-BIT UGFF represent normal master-slave flip-flop, conventional gated flip-flop [4], 2-bit unified, 3-bit unified, and 4-bit unified gated flip-flops, respectively. The C-Q delay represents a clock to Q output signal delay. As can be seen from the Table, a width of latch window and C-Q delay of gated flip-flops are much larger than those of the normal flip-flop. Therefore, it is very important to apply the unified gated flip-flops to non-critical paths and normal flip-flops to critical paths so that the power consumption can be reduced without degrading the performance of the target circuits.

**Table 1.** Timing parameters

| | setup [ps] | | hold [ps] | | C-Q delay [ps] | |
|---|---|---|---|---|---|---|
| | rise | fall | rise | fall | rise | fall |
| Normal FF | 97.4 | 48.5 | −97.4 | −48.5 | 195 | 184 |
| CONV GFF | 293 | 366 | 97.5 | 97.5 | 315 | 291 |
| 2-BIT UGFF | 341 | 488 | 97.5 | 97.5 | 345 | 324 |
| 3-BIT UGFF | 379 | 488 | 146 | 97.5 | 389 | 358 |
| 4-BIT UGFF | 488 | 537 | 146 | 146 | 416 | 389 |

### 3.3 Layout and Area Overhead

The left of Figure 4 shows a layout image of a 2-bit unified gated flip-flop designed using a commercial $0.18\mu m$ process technology. The right of Figure 4 shows layout areas required for the unified gated flip-flops. Horizontal axis shows the number of flip-flops unified together. Vertical axis shows the ratio of the layout area required for a $n$-bit unified gated flip-flop to that required for the $n$-bit normal flip-flops. For example, the layout area of a 1-bit gated flip-flop is double of the area of the 1-bit normal flip-flop. Our unified gated flip-flop shares the clock-gating circuit among several flip-flops. Therefore, as the number of flip-flops unified together increases, the area of the unified gated flip-flop per bit decreases. For example, area overhead per bit involved in a 4-bit unified gated flip-flop is 40% smaller than that of the conventional 1-bit gated flip-flop.

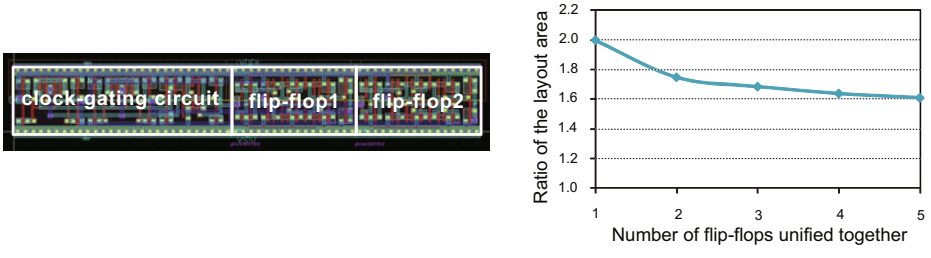



**Fig. 4.** Layout of 2-bit Gated Flip-Flop

### 3.4 Overview of the Test Chip

The left of Figure 5 shows a layout image of a test chip designed with $0.18\mu m$ CMOS process. A block specified as *power measurement circuit* includes a circuits under test which consists of normal flip-flops, conventional gated flip-flops, 2-bit, 3-bit, 4-bit, and 5-bit unified gated flip-flops. Each test block has 60 flip-flops concatenated together so as to amplify the power consumption of a flip-flop under test. The right of Figure 5 shows the timing diagram of the 2-bit unified gated flip-flop. As can be seen from the diagram, the 2-bit unified gated flip-flop functions correctly. Similarly, the test chip demonstrates that 3-bit, 4-bit, and 5-bit unified gated flip-flops work correctly as well.

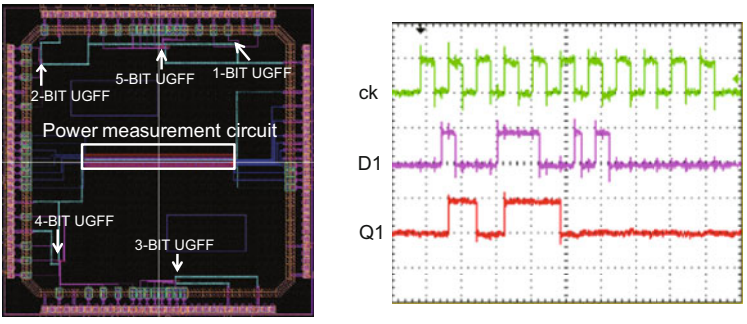



**Fig. 5.** Layout Image and Timing Diagram of the Test Chip

### 3.5 Power Measurement Results

Figure 6 shows the power measurement results. Horizontal vertical axes represent the state transition probability and the power consumption of the flip-flops. Since our previous work [12] demonstrates that an average state transition probability of registers used in a commercial microprocessor is around 0.1, our unified gated flip-flop greatly reduces the power consumption of register circuits.

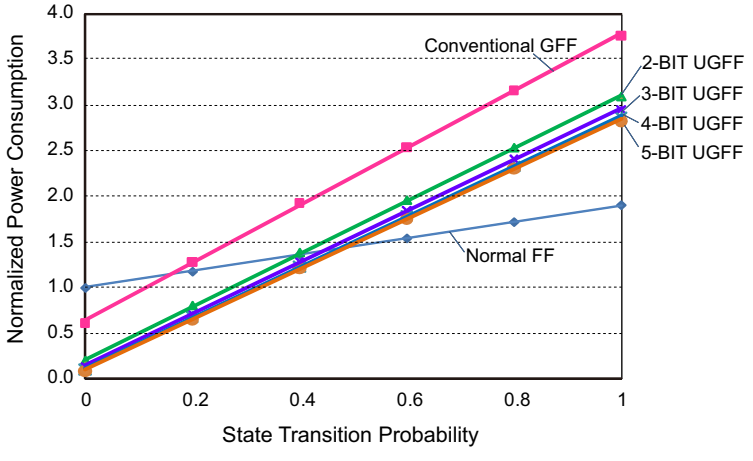



**Fig. 6.** Power Measurement Result

## 4 Evaluation with a Commercial Microprocessor

### 4.1 Experimental Setup

We use Media embedded Processor (MeP) [11], which is originally developed by Toshiba in this experiment. It is a RISC type processor with five pipeline stages and 16 general purpose registers.

First, we characterize gated flip-flops using Liberty$^{TM}$ NCX, a cell-library characterization tool of SYNOPSYS. Then, the processor core is synthesized and place-and-routed using Design_Compiler and Astro of SYNOPSYS, respectively. The processor core represents a core logic part of the processor which excludes SRAM blocks like a cache memory and a scratchpad memory. The number of gates in the processor core is around 46,000. Note that the conventional clock gating is applied to the processor core in advance as a base of the evaluation. Next, we run benchmark programs presented in the bottom of Figure 8 on an post-layout model of the processor to obtain toggle information of every gates. Verilog-XL of Cadence is used for the simulation. Finally, we calculate the power consumption of the processor core based on the toggle information obtained through the post-layout simulation. We evaluate the power consumption and area overheads of three different processor designs which respectively employ the following three techniques in their register circuits.

**CLK Gating** Conventional clock gating (Base of the evaluation)
**CONV GFF** Conventional gated flip-flop [4]
**UGFF** Unified gated flip-flops (Our proposal)

### 4.2 Layout Synthesis Results

Table 2 shows the number of flip-flops used in the three types of processor designs described in the previous subsection. Since our proposed unified gated flip-flops have large C-Q delay and wide latch window, the critical path delay may be very large if those flip-flops are used in all register circuits. Therefore, we use the normal flip-flops as well in all designs so that the normal flip-flops are used in critical paths. This helps reduce the average power consumption without increasing the critical path delay of the target circuit.

Figure 7 shows a layout image and layout areas of the processor core. Logic synthesis and place-and-route are performed so that the layout area is minimized under a specific clock-cycle time constraint. In this experiment, we use 10ns and 12ns as the clock-cycle time constraints. In other words, the target clock frequencies for the optimizations are 100MHz and 83MHz as shown in Figure 7. Bar charts show the layout results of the processor core which employs normal flip-flops, conventional gated flip-flops and unified flip-flops, respectively. Left three bars and right three bars in the chart show layout areas of the processor core designed targeting 100MHz and 83MHz, respectively. As can be seen from the results, our unified gated flip-flop does not increase the layout area of

**Table 2.** The number of flip-flops used in the processor

| | Normal FF | CONV GFF | 2-bit UGFF | 3-bit UGFF | 4-bit UGFF |
|---|---|---|---|---|---|
| **CLK Gating** | 3,732 | - | - | - | - |
| **CONV GFF** | 966 | 2,766 | - | - | - |
| **UGFF** | 908 | 970 | 16 | 36 | 428 |

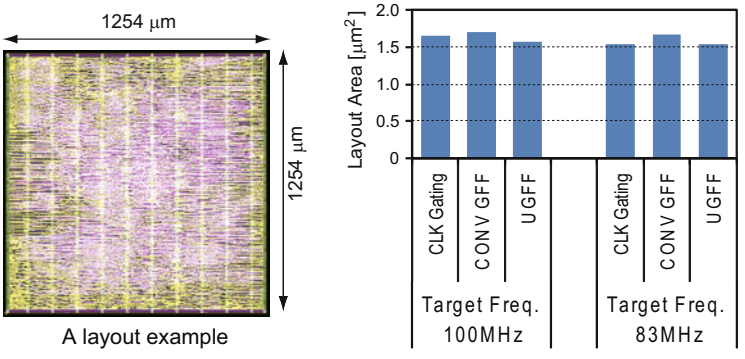



**Fig. 7.** Layout Synthesis Results

the processor core. The results also demonstrate that the unified gated flip-flop does not cause an degradation of the processor performance although the performance of an individual unified gated flip-flop is lower than that of a normal flip-flop.

### 4.3 Power Consumption Results

Figure 8 shows the power consumption results obtained using post-layout simulation. As can be seen from the Figure, our approach widely reduces the power consumption of clock trees. This is mainly due to a decrease in the clock tree size thanks to the reduction of the fanout loads in the leaves of the clock tree. As a result, our approach reduces the power consumption of the processor core by 25% on an average and by 33% at the best case. If compared to the processor employing the clock gating only, the power reductions are 32% on an average and 42% at the best case.

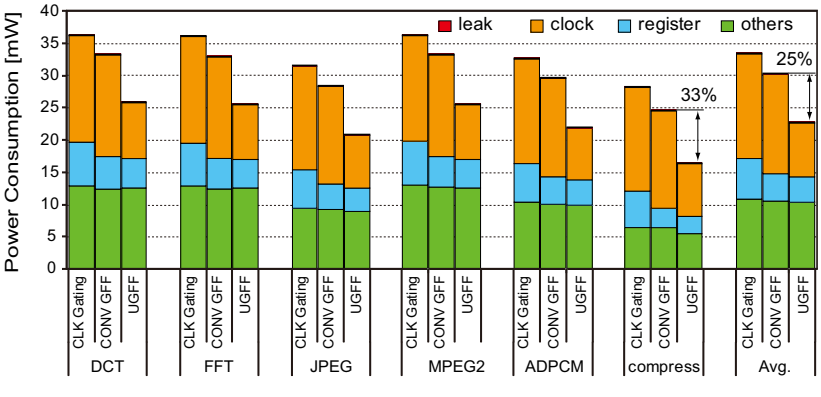



**Fig. 8.** Power Consumption Results

## 5 Conclusion

This paper presents our idea of unifying multiple gated flip-flops for reducing the power consumption in register circuits. Chip measurement results demonstrate that our unified gated flip-flops work correctly. Experimental results obtained using post-layout simulation of a commercial embedded processor demonstrate that our technique reduces the power consumption of a processor core by 25% on an average and 33% at the best case compared to the processor employing the conventional gated flip-flops. Our future work will be devoted to selectively apply appropriate types of unified gated flip-flops to registers with taking the state transition probability of each flip-flop into consideration.

**Acknowledgment.** The authors would like to thank JST CREST ULP project and JSPS NEXT program for their support to this work.

## References

1. Pedram, M.: Power Aware Design Methodologies. Kluwer Academic Publishers, Norwell (2002)
2. Pedram, M.: Power Minimization in IC Design: Principles and Applications. ACM Transactions on Design Automation of Electronic Systems 1(1) (1996)
3. Raghavan, N., Akella, V., Bakshi, S.: Automatic Insertion of Gated Clocks at Register Transfer Level. In: Proc. of VLSI Design, pp. 48–54 (January 1999)
4. Strollo, A.G., Napoli, E., De Caro, D.: New Clock-Gating Techniques for Low-Power Flip-Flops. In: Proc. of ISLPED, pp. 114–119 (July 2000)
5. Jiang, I.H.-R., Chang, C.-L., Yang, Y.-M.: INTEGRA: Fast Multi-Bit Flip-Flop Clustering for Clock Power Saving Based on Interval Graphs. In: Proc. of ISPD, pp. 115–121 (March 2011)
6. Brooks, D., Martonosi, M.: Value-Based Clock Gating and Operation Packing: Dynamic Strategies for Improving Processor Power and Performance. ACM Transactions on Computer Systems 18(2) (May 2000)
7. Choi, J., Jeon, J., Choi, K.: Power Minimization of Functional Units by Partially Guarded Computation. In: Proc. of ISLPED, pp. 131–136 (July 2000)
8. Kondo, M., Nakamura, H.: A Small, Fast and Low-Power Register File by Bit-Partitioning. In: Proc. of Int'l Symposium on High-Performance Computer Architecture, pp. 40–49 (February 2005)
9. Nogawa, M., Ohtomo, Y.: A Data-Transition Look-Ahead DFF Circuit for Statistical Reduction in Power Consumption. IEEE Journal on Solid-State Circuits 33(5), 702–706 (1998)
10. Hamada, M., Terazawa, T., Higashi, T., Kitabayashi, S., Mita, S., Watanabe, Y., Ashino, M., Hara, H., Kuroda, T.: Flip-flop Selection Technique for Power-Delay Trade-Off. In: ISSCC Digest of Technical Papers, pp. 270–271 (February 1999)
11. Mizuno, A., Kohno, K., Ohyama, R., Tokuyoshi, T., Uetani, H., Eichel, H., Miyamamori, T., Matsumoto, N., Matsui, M.: Design Methodology and System for a Configurable Media Embedded Processor Extensible to VLIW Architecture. In: Proc. of ICCD, pp. 2–7 (September 2002)
12. Okuhira, T., Ishihara, T.: Unification of Multiple Gated Flip-Flops for Saving the Power Consumption of Register Circuits. In: Proc. of ICESIT, p. 115 (February 2010)

# C-elements for Hardened Self-timed Circuits

Florent Ouchet, Katell Morin-Allory, and Laurent Fesquet

TIMA Laboratory, Grenoble INP, UJF, CNRS, France
firstname.lastname@imag.fr

**Abstract.** Self-timed circuits are slope sensitive: when the voltage of one input or internal node changes too slowly, the interconnected logical blocks might loose their local one-to-one synchronization. This phenomenon often leads to unwanted global dead-locks of the entire circuit. The deep-submicronic manufacturing process mismatches might create such situations where one logical block is significantly slower than the others. We applied two known solutions for ensuring the correct C-element behavior whatever the slopes are: the transistors are resized and the supply voltage is reduced in order to guarantee the overall chip correctness taking into account the process variations.

## 1 Introduction

The self-timed circuits are well known for their intrinsic robustness against Process-Voltage-Temperature (PVT) variations [1] [2]. The PVT variations have consequences on propagation delays and on transition speeds (slope). The self-timed circuit structure (see Section 2) ensures their insensitivity to the propagation delay variations in the digital gates as well as in the communication wires. However, the design flow of self-timed circuits usually ignores the transition speeds.

The behavior of conventional (synchronous) integrated circuit (IC) in the context of slow slopes has previously been studied in [3]: it has been demonstrated that slower slopes imply longer computational delays. Because synchronous IC are not robust against delays, they will not behave correctly in the context of slow slopes. For self-timed circuits, their robustness against delays should make their behavior immune to slow slopes. Unfortunately, digital gates will variously interpret a given analog voltage when their logical threshold voltages (see Section 3) have mismatches [4].

The simulations in [5] underline ideal conditions without process variations. The PVT variations modify these logical threshold voltages: they may change their ordering and the chip behavior consequently. In this paper, we design with the sufficient safety margins to ensure the correct threshold voltage ordering even in the presence of process variations.

## 2 Self-timed Circuit Structures

The design of self-timed IC differs from conventional synchronous ones by the lack of global synchronization. In synchronous IC, the clocks synchronize the

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 247–256, 2011.

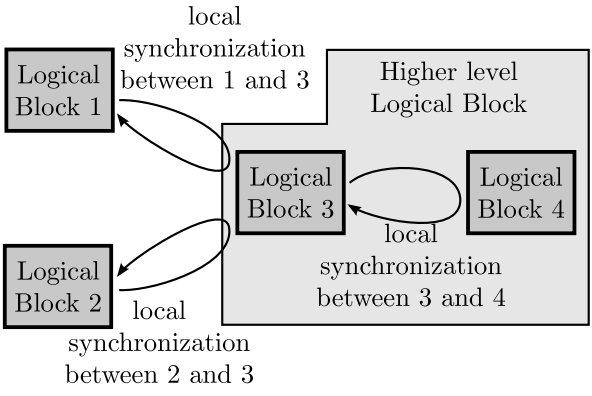



**Fig. 1.** Local synchronization between logical blocks

data over the chip. In self-timed circuits, the data are propagated through logical blocks via communication channels ensuring the one-to-one local synchronizations between the two blocks (Figure 1).

This structure can either represents Globally Asynchronous Locally Synchronous (GALS) or self-timed circuits. In GALS, the logical blocks have their own clock and internally run synchronously with respect to this internal clock. Conversely, self-timed circuits process without any clock.

Each logical block runs at its own speed and all data exchanges are safely executed when both blocks are ready. The self-timed local blocks can safely be assembled to build a higher level block as shown in Figure 1: the local one-to-one synchronization guarantees the overall delay insensitivity. However, most real-life self-timed circuits are not entirely delay insensitive (DI) because of intrinsic area and delay costs. The Quasi-Delay-Insensitive (QDI) self-timed circuits are widely considered as a good trade-off between data transfer safety and structural costs. Their structure ensures delay insensitivity except on some specific wire forks, called isochronic forks, where a timing assimption is made (the propagation delays on the fork wires are supposed to be equal) [4].

The communication channels between the logical blocks implement the handshake protocols which is responsible for the data transfer. Unlike event-based synchronizations (clock), the handshake communication protocols are state-based. Figure 2 depicts the typical timeline for the 4-phase communication protocol between two logical blocks. In phase ①, both blocks are ready for one data exchange. The sender block emits the data along with the request signal in phase ②. The receiver acknowledges the data transfer in phase ③. Finally, the sender removes the data and the request signals in phase ④: the communication channel is ready for the next data transfer.

The QDI timing assumptions are really weak but this is not sufficient in order to guarantee the chip safety in the context of slow slopes [5]. The corner cases exhibits the self-timed IC wrong behavior: the communication channels loose their synchronization when one of the two blocks, either the sender or

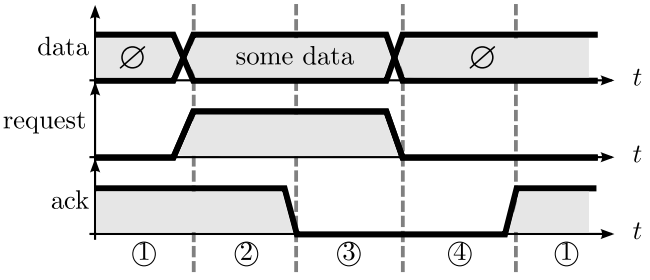



**Fig. 2.** 4-phase communication protocol timeline

the receiver, runs significantly faster than the other. The local speed variations impact the block computation times (delays) as well as the transition times (slopes) on their interconnects. These speed gaps can be the consequence of electromagnetical disturbances for instance when the chip runs in an aggressive environment. In such situations, the sender and receiver logical blocks do not see the same amount of data on the communication channels: data are duplicated and may be altered.

Manufacturing processes with significant variability can lead to the same effect and to the same consequences: one part of the manufactured chip has optimal performance while the others may be slower. Therefore two logical blocks located in different chip areas will run at different rated speeds.

## 3 C-elements

The level-based handshake communication protocols are nicely implemented using C-elements (also called Muller gates). Indeed, the C-elements implement a rendez-vous between the sender request and the receiver acknowledgement wires.

Figure 3(a) denotes the symbol that is usually associated with 2-input C-elements and its truth-table. There exist several solutions for implementing the C-element function at the transistor level. Our study focuses on two static C-element structures:

- the 2-input weak-feedback C-element (denoted WM2), as shown in Figure 3(b) [6];
- the 2-input conventional C-element (denoted CM2), see Figure 3(c) [7];

For these gates, the input stage is built by stacking the transistors $T_{N1}$, $T_{N2}$, $T_{P1}$ and $T_{P2}$. The memory point flips depending on the input values of these four transistors. The transistors $T_{N3}$ and $T_{P3}$ are part of the output stage and of the memory. Finally, the feedback stage (for the memory) is made of the remaining transistors.

All these structures do not exhibit the same behavior when analog phenomena are taken into account: for instance their robustness against Single Event Upsets (SEU) highly depends on the standard-cell structure. In addition, these different

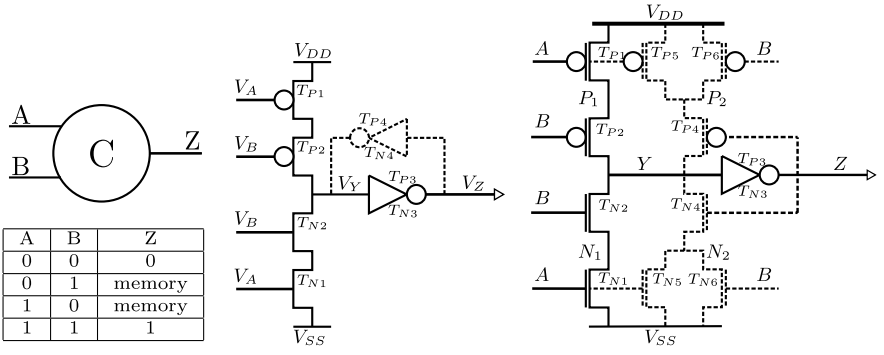



**Fig. 3.** (a) C-element symbol and truth-table (b) WM2 transistor network (c) CM2 transistor network

structures have intrinsically different threshold voltages [5]. At analog level, the input threshold voltages are defined as follows:

- $V_{TH}$ is defined as the minimum voltage needed on one C-element input, when the other input is tied to $V_{DD}$, for making its output switching to $V_{DD}$;
- $V_{TL}$ is the maximum voltage on one C-element input, when the other input is tied to $V_{SS}$, for making its output switching to $V_{SS}$.

These threshold voltages depend on the C-element structure, on the manufacturing process and on its transistor widths and lengths. The original C-elements are standard-cells taken from the TIMA Asynchronous Library [8]. The transistor sizings were elaborated with constraints on area, delay and consumption while the threshold voltages were not taken into account. The resulting $V_{TL}$ and $V_{TH}$ inducted some gates where $V_{TH} < V_{TL}$. In such cases, the implementation of the rendez-vous function is broken: the two blocks are not synchronized anymore. The sender and receiver run at their own speed and data tokens are inserted when the receiver runs intrinsically faster than the sender.

The C-element standard cells have to be carefully designed in order to always avoid cases where $V_{TH} < V_{TL}$. Because these threshold voltages depend on transistor sizings and on the manufacturing process, they also depend on PVT variations.

## 4 C-element Simulations

The C-element gates are characterized by several automated benchmarks. First, the output level is forced to $V_{SS}$ (resp. $V_{DD}$) by the stimulations of the appropriate voltage on its inputs. When the voltages are stabilized, one of its inputs is switched to the opposite logical level $V_{DD}$ (resp. $V_{SS}$). According to the C-element truth table, its output remains unchanged. Then, the other input is

stimulated by a slow rising slope to $V_{DD}$ (resp. down to $V_{SS}$). The gate output voltage flips when this input voltage reaches $V_{TH}$ (resp. $V_{TL}$). When $V_{TH} < V_{TL}$, the gate might fire several times in a closed loop structure. These characterizations are based on Monte Carlo simulation in order to model global process variations and local mismatches. In the following Figures and Tables, $\overline{V}$ denotes the average value for the respective voltage $V$ and $\sigma_V$ its standard deviation.

The gates are designed targeting different manufacturing technologies from STMicroelectronics:

- general purpose high speed 130 nm;
- low power standard $V_T$ 45 nm;

Figure 4 depicts the threshold voltage density functions for the 130nm high-speed weak-feedback C-element when the supply voltage is set to $V_{DD} = 1.2V$. The threshold voltage density functions can be approximated by Gaussian distribution. Ideal Gaussians have infinite support. However, it is accepted that 99.73% of $V$ is in the range $[\overline{V} - 3\sigma_V, \overline{V} + 3\sigma_V]$ (three-sigma rule). Therefore, the density of values outside this range is considered as null. This assumption holds in the next sections of this paper.

The C-element behavior is correct provided $V_{TH} > V_{TL}$. When the two threshold voltages are not correlated, this condition can be rewritten to: the density functions for $V_{TH}$ and $V_{TL}$ shall not overlap and $\overline{V_{TH}} > \overline{V_{TL}}$. Otherwise, when the two threshold voltages are correlated, one has to compute the difference $\Delta V_T = V_{TH} - V_{TL}$ for each measure and make sure this value is never negative.

Moreover, the density function of $\Delta V_T$ is the convolution of the density functions of $V_{TL}$ and $V_{TH}$ if and only if (*iff*) the two voltages are not correlated. This

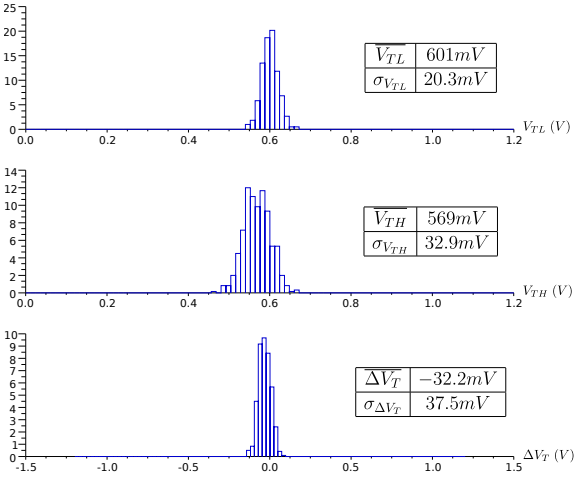



**Fig. 4.** 130nm HS WM2 threshold voltage densities

can be verified analytically and experimentally. The resulting standard deviation is $\sigma_{\Delta V_T} = \sqrt{(\sigma_{V_{TH}})^2 + (\sigma_{V_{TL}})^2}$.

In Figure 4, the average values for $V_{TH}$ and $V_{TL}$ of the 130nm HS WM2 are the same values as obtained during conventional simulations without process variations. We also obtained $\sigma_{\Delta V_T} < \sqrt{(\sigma_{V_{TH}})^2 + (\sigma_{V_{TL}})^2}$. Therefore, it implies that the two threshold voltages are not independent values but they are somehow correlated.

$V_{TH}$ (resp. $V_{TL}$) is measured when the C-element output switches to $V_{DD}$ ($V_{SS}$). At this switching state, the node $V_Y$ switches to $V_{SS}$ ($V_{DD}$). This transition on $V_Y$ is caused by the NMOS transistors $T_{N1}$ and $T_{N2}$ (PMOS $T_{P1}$ and $T_{P2}$) that are driving more current to node $Y$ than the PMOS transistors $T_{P4}$, $T_{P5}$ and $T_{P6}$ (NMOS $T_{N4}$, $T_{N5}$ and $T_{N6}$. Consequently, each transition to $V_{SS}$ and $V_{DD}$ requires both NMOS and PMOS transistors. The PMOS and NMOS transistor characteristics and moreover their variability and mismatches are parameters of both $V_{TL}$ and $V_{TH}$. Therefore a modification in characteristics either of NMOS or of PMOS transistors has consequence on both threshold voltages.

Figure 5 shows $V_{TH}$ as function of $V_{TL}$ for the 130nm HS WM2: the two values are not closely correlated. Indeed, the correlation coefficient of $V_{TH}$ and $V_{TL}$ is $r = \frac{1}{n-1}\sum_{i=0}^{n}(\frac{V_{TLi}-\overline{V_{TL}}}{\sigma_{V_{TL}}})(\frac{V_{THi}-\overline{V_{TH}}}{\sigma_{V_{TH}}}) = 0.0663$ where $n$ is the number of measures ($n = 500$ in this case). The linear regression is drawn in solid green. Its approximation is $V_{TH} = 0.108V_{TL} + 0.504$: a lower $V_{TL}$ occurs more frequently associated to a lower $V_{TH}$ than to a higher $V_{TH}$.

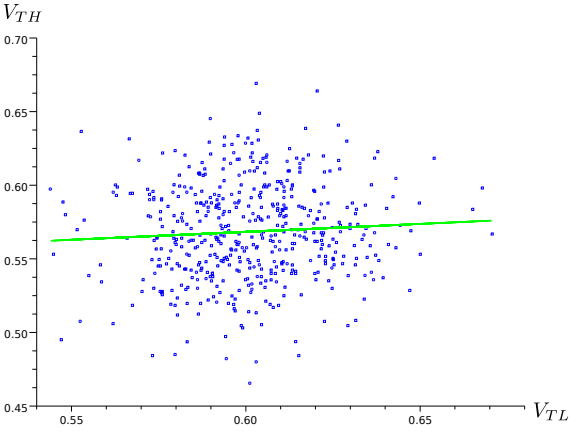



**Fig. 5.** 130nm HS WM2: $V_{TH} = f(V_{TL})$

## 5 Consequence on C-element Design

Two methods are usually experimented to fix the gate behavior [9, 5]:

1. the width of the feedback transistors may be increased;
2. the supply voltage might be reduced for the entire standard-cell.

### 5.1 Fix Methods

The first method produces good results when applied on the weak-feedback C-element structure. The scale factor $fb_{scale}$ denotes the scale factor that multiplies the width of $T_{P4}$ and $T_{N4}$ together. The behavior of the 130-nm high-speed weak-feedback C-element is correct when $fb_{scale} > 1.1$. Figure 6 shows how the difference of threshold voltages $\Delta V_T$ scales with $fb_{scale}$. The dotted curve denotes the highest $3\sigma$ boundary; these specific variations would give the best reliability. The solid curve is the $\Delta V_T$ average value. Finally, the dashed curve denotes the lowest $3\sigma$ boundary: this is the worst case at given $fb_{scale}$. The gate behavior is correct when the lowest $3\sigma$ boundary is always positive: this is true when $fb_{scale} > 1.34$.

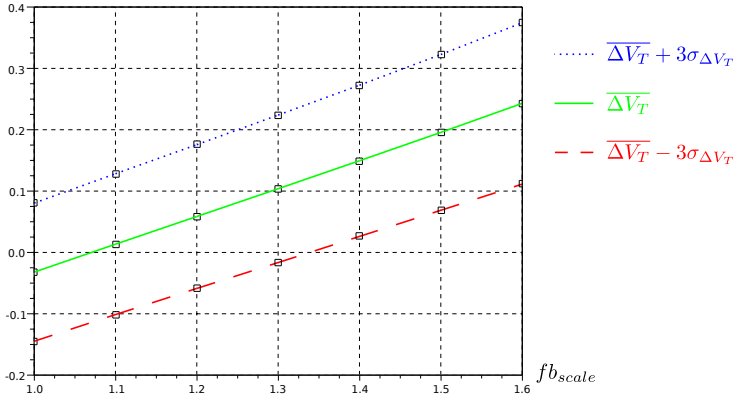



**Fig. 6.** 130nm HS WM2: Threshold voltages as function of $fb_{scale}$

The C-element structure does not cope well with the first fix method [5]. The second fix method is well-suited for the conventional C-element structure. Figure 7 draws the $\Delta V_T$ skeleton as a function of the supply voltage. $\sigma_{\Delta V_T} = 37mV$ when $V_{DD} = 1.2V$ and $\sigma_{\Delta V_T} = 44mV$ at $V_{DD} = 0.5$. When $V_{DD}$ is reduced, $\Delta V_T$ increases: the process variations have more effects when $V_{DD}$ is low. During our simulations, we could not reduce the supply voltage below $0.5V$ because the gate output is not able to switch to $V_{SS}$ anymore (the required $V_{TL}$ would be less than $0V$). Therefore, the second method does not give good results alone.

### 5.2 Costs

The transistor resizing method may affect standard-cell area: when the transistors are too wide, a single gate finger structure may not fit in the standard-cell design patterns. In this case, the designer has to draw additional gate fingers to create the equivalent transistor. Each additional finger increase standard-cell size. However, the keeper transistor widths are initialy very small for both the conventional and weak-feedback structures. Consequently, they fit in the initial standard-cell areas even if $fb_{scale}$ is 3.0.

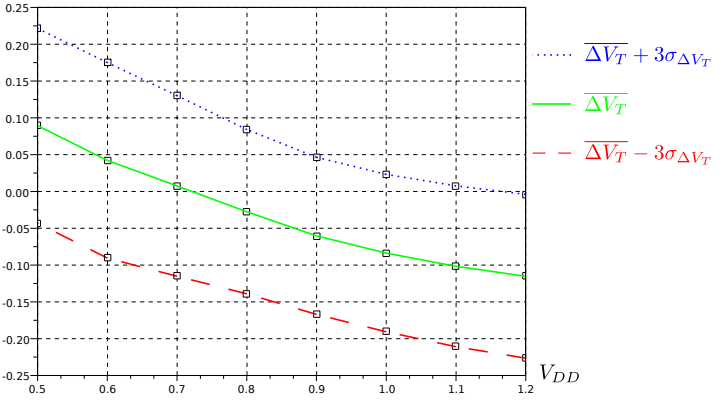



**Fig. 7.** 45nm LS CM2: Threshold voltages as functions of $V_{DD}$

Both methods impact on the propagation delays: we measured that the transistor resizing method effects are limited. On the opposite, the $V_{DD}$ scaling drasticaly slows down the circuit. A good trade-off between the two methods could guarantee the C-element correctness without a significant performance penalty.

### 5.3 Trade-off between the Two Methods

The two methods are based on different modifications that may be used complementary. Figure 8 draws $\Delta V_T$ as function of $V_{DD}$ and $fb_{scale}$ for the 45nm LS CM2. This 3D curve is obtained by the Monte-Carlo computations of $\Delta V_T$ with two parameters: $V_{DD}$ range is $0.6V$ to $1.2V$ and $fb_{sacle}$ from 1.0 to 3.0.

The region ① means $\Delta V_T > 0$: the threshold voltages are correctly ordered subjected to the $3\sigma$ assumption. In opposite, the region ② depicts the incorrect behavior where $\overline{\Delta V_T} + 3\sigma_{\Delta V_T} < 0$.

$\Delta V_T$ is always increasing when $fb_{scale}$ is increasing. When $fb_{scale}$ is small, $\Delta V_T$ increases when $V_{DD}$ reduces. The same variation could be expected when $fb_{scale}$ is large. In this case, the effects of the two methods would be optimaly mixed. Experimentally, we obtained the opposite: for large $fb_{scale}$, $\Delta V_T$ decreases when $V_{DD}$ decreases.

Actually, there is an unwanted side effect between the two methods. The saturation current for CMOS transistor is given by the well known formula: $I_{DS} \approx K.\frac{W}{L}(V_{GS} - V_{TR})^2$. It is the function of:

- the transistor geometrical sizing (width $W$ and length $L$),
- its node voltages ($V_{GS}$) (the value of $V_{DD}$ plays here),
- the manufacturing process characteristics ($K$ and transistor threshold voltage $V_{TR}$).

By simulation, we observed that the PMOS saturation current complies with this approximation whereas the NMOS saturation current does not. The effective

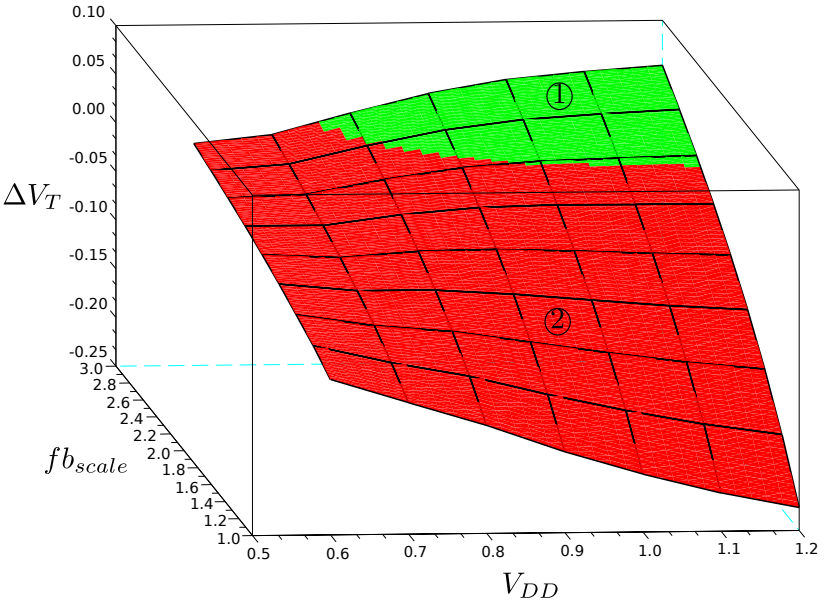



**Fig. 8.** 45nm LS CM2: $\Delta V_T$ as function of $V_{DD}$ and $fb_{scale}$

NMOS saturation current is higher than expected when $V_{DD}$ is low and their width is not the minimal one. In the studied CM2, the transistors $T_{N1}$ and $T_{N2}$ will be affected: when the two C-element inputs are set to $V_{DD}$, these transistors drive more current than expected. Consequently, the memory point output switches to $V_{DD}$ sooner than expected: $V_{TH}$ is reduced (and $\Delta V_T$ as well) when $V_{DD}$ is low.

The area ① has high $V_{DD}$ and high $fb_{scale}$. In order to guarantee the CM2 correct behavior, the keeper transistors need significantly bigger widths and the voltage scaling should be limited.

## 6 Conclusion

We studied the behavior of digital gates at analog level in order to improve the self-timed circuit robustness. The behavioral correctness is based on the ordering of threshold voltages. The process variations slightly modify these threshold voltages and may change their ordering if they are almost equal. Their variations are measured and two previously introduced methods are applied in order to guarantee a correct behavior. When the process variations are taken into account, the method based on $V_{DD}$ reduction does not give good results alone. It has to be associated to the other methods based on transistor resizing.

We managed to fix the incorrect behavior of two C-element structures. The C-element gates were simulated and process variations were taken into account

to ensure the behavioral correctness for almost all manufactured circuits. The $3\sigma$-based safety margin ensures a correct behavior which do not depend on the chip manufacturing process.

## References

1. Hamon, J., Fesquet, L., Miscopein, B., Renaudin, M.: High-level time-accurate model for the design of self-timed ring oscillators. In: 14th IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC (2008)
2. Fairbanks, S., Moore, S.: Analog micropipeline rings for high precision timing. In: Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems, ASYNC (2004)
3. Auvergne, D., Maurine, P., Azémard, N.: Modeling for designing in deep submicron technologies. In: Low-power CMOS Circuits: Technology, Logic Design and CAD Tools. CRC Press, Boca Raton (2006)
4. van Berkel, K.: Beware the isochronic fork. The VLSI Journal Integration 13, 103–128 (1992)
5. Ouchet, F., Morin-Allory, K., Fesquet, L.: Delay insensitivity does not mean slope insensitivity! In: Proceedings of the 16th IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC (2010)
6. Sutherland, I.E.: Micropipelines. Communications of the ACM 32, 720–738 (1989)
7. Martin, A.J.: Formal program transformations for VLSI circuit synthesis. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1990)
8. Folco, B., Bregier, V., Fesquet, L., Renaudin, M.: Technology mapping for area optimized quasi delay insensitive circuits. Vlsi-Soc: From Systems To Silicon 240 (978-0-387-73660-0), 55–69 (2007)
9. Al Tarawneh, Z., Russell, G., Yakovlev, A.: An analysis of seu robustness of c-element structures implemented in bulk cmos and soi technologies. In: Proceedings of the International Conference on Microelectronics, ICM (2010)

# High-Speed and Low-Power PID Structures for Embedded Applications*

Abdelkrim K. Oudjida[1], Nicolas Chaillet[2], Ahmed Liacha[1], Mustapha Hamerlain[1], and Mohamed L. Berrandjia[1]

[1] Microelectronics and Nanotechnology Division, Centre de Développement des Technologies Avancées (CDTA), Baba-Hassen, BP. 17, 16303 Algiers, Algeria
{a_oudjida,liacha,mhamerlain,mberrandjia}@cdta.dz
[2] AS2M Department, FEMTO-ST Institute, Besançon, France
chaillet@ens2m.fr

**Abstract.** In embedded control applications, control-rate and energy-consumption are two critical design issues. This paper presents a series of high-speed and low-power finite-word-length PID controllers based on a new recursive multiplication algorithm. Compared to published results into the same conditions, savings of 431% and 20% are respectively obtained in terms of control-rate and dynamic power consumption. In addition, the new multiplication algorithm generates scalable PID structures that can be tailored to the desired performance and power budget. All PIDs are implemented at RTL level as technology-independent reusable IP-cores. They are reconfigurable according to two compile-time constants: set-point word-length and latency.

**Keywords:** Design-Reuse, Embedded Finite-Word-Length (FWL) Controllers, Intellectual Property (IP), Linear Time Invariant (LTI) Systems, Low-Power and Speed Optimization, Proportional-Integral-Derivative (PID).

## 1 Background and Motivation

The PID is by far the most commonly used feedback controller due to its simple structure and robust performance [1]. An important feature of this controller is that it does not require a precise analytical model of the system that is being controlled, which makes it very attractive for a large class of LTI dynamic systems. However, despite the large popularity of PID controller, little attention has been paid to its optimization, either for ASIC or for FPGA integration. In [2] low-power serial and parallel multiple-channel PID architectures are proposed for small mobile robots. In this work, the optimization was carried out at macro-level considering several PIDs, rather than at micro-level (optimization of the PID itself). Nevertheless, the whole architecture will deliver much more interesting results if combined with an optimized PID. The second work [3] proposes serial, parallel, and mixed PID architectures incorporating different number

* This work was supported by "Centre de Développement des Technologies Avancées" (CDTA), Algiers, Algeria, in collaboration with FEMTO-ST institute, Besançon, France.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 257–266, 2011.

(1-3) of multiplication cores. High power consumption, even with the serial architecture, and complex control-part are the two major shortcomings of this proposal. Finally, in [4] an attractive optimized PID structure based on distributed arithmetic (DA) is presented. Although this latter exhibits interesting results in terms of resource utilization and power consumption, it suffers from three serious drawbacks: high latency (n+1 clock-cycles for n bit set-point word-length), FPGA technology-dependent as it's essentially based upon FPGA look-up-tables (LUTs), and inability to handle time-varying PID parameters since they are precomputed and stored into LUTs. Nevertheless, it's considered as a reference design against which the obtained results are confronted into the same conditions.

The objective of this paper is to design optimized FWL-PID structures that overcome all above-mentioned shortcomings, and which are especially dedicated to embedded control applications. The PID cores are described at RTL level. They are highly reconfigurable and technology-independent, offering the possibility to be mapped both on FPGA and ASIC, using a foundry standard-cell-library.

To reach such a goal, a special focus was put on the optimization of the *inner arithmetic* of PID. For that, we considered two discrete forms of PID algorithm: the commercial form [5], called also the standard or ISA form, and the incremental form. These two forms went through FPGA implementations, using a new recursive multibit recoding multiplication algorithm (RMRMA). Results show clear superiority over those provided in [4]. PID control-rate and energy-consumption savings are respectively 431% and 20% . Furthermore, RMRMA algorithm generates scalable PID structures which can be customized to fit the desired speed and power budget. Its interesting feature as a low-power multiplication algorithm makes it useful for a wide range of numeric applications.

The paper is organized as follows. In this section we outlined the main requirement specifications for embedded PID controller. Section 2 presents the two most-used discrete versions of PID algorithm. Section 3 introduces the new RMRMA algorithm and its implementation. A discussion around the obtained results is given in section 4. . And finally some concluding remarks.

## 2 The Two Most-Used Discrete versions of PID

In digital control, commercial and incremental forms are the two most-used discrete PID versions [1][5]. They are respectively denoted by recurrent equations (1) and (2), and their corresponding coefficients are grouped in Table 1.

$u(k) = P(k) + I(k) + D(k)$ (1) ; where $P(k) = A \cdot u_c(k) + B \cdot y(k)$ ;

$I(k) = I(k-1) + C \cdot e(k-1)$ ; and $D(k) = D \cdot D(k-1) + E \cdot f(k)$.

With $e(k-1) = u_c(k-1) - y(k-1)$ and $f(k) = y(k) - y(k-1)$

And

$$u(k) = u(k-1) + A \cdot e(k) + B \cdot e(k-1) + C \cdot e(k-2) \quad (2)$$

Where $e(k) = u_c(k) - y(k)$ ; $e(k-1) = u_c(k-1) - y(k-1)$ ;

$e(k-2) = u_c(k-2) - y(k-2)$.

The translation of equation (1) and (2) into architectures is depicted by Fig. 1 and 2, respectively.

**Table 1.** Coefficients of discrete recurrent equations

| Coefficients | Commercial PID | Incremental PID |
|---|---|---|
| A | $K_p\, b$ | $K_p\left(1+\frac{T_s}{T_i}+\frac{T_d}{T_s}\right)$ |
| B | $-K_p$ | $-K_p\left(1+2\frac{T_d}{T_s}\right)$ |
| C | $K_p\frac{T_s}{T_i}$ | $K_p\frac{T_d}{T_s}$ |
| D | $\frac{T_d}{T_d+NT_s}$ | – |
| E | $-\frac{K_p T_d N}{T_d+NT_s}$ | – |

$K_p$ is the proportional gain; $T_i$ and $T_d$ are respectively the integral and derivative times; $N$ is the maximum derivative gain; $b$ is the fraction of set-point in proportional term; and $T_s$ is the sampling period.

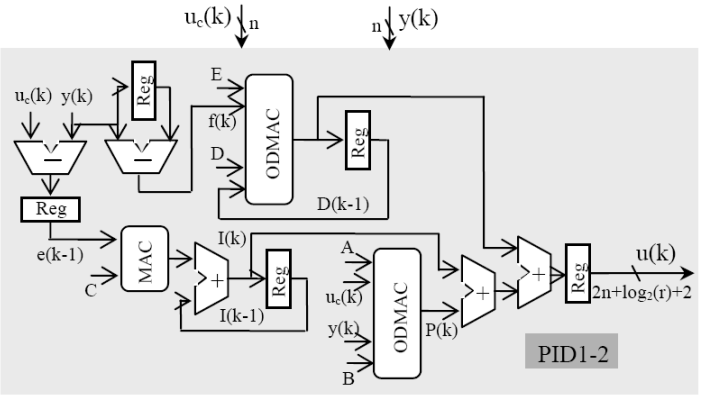



**Fig. 1.** Commercial PID architecture

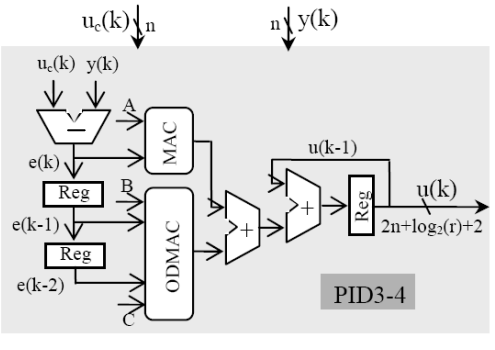



**Fig. 2.** Incremental PID Architecture

To satisfy different application cases, two IP versions are developed for each equation: with constant coefficients (PID1) and with varying coefficients (PID2). This latter requires a host side interface (HIS) to handle the runtime change of the coefficients.

The commercial version allows the three standard PID functioning modes (P, PI, PID) according to Mode input value. At the end of u(k) computation, the Done output signal toggles during one clock cycle, and the PID enters into sleep mode (whole internal activity stopped except for clocking and HIS) for maximum energy conservation.

## 3 RMRMA Based PID

Multiplication is a fundamental operation in digital design. Its speed and power requirements are two critical factors limiting the whole system performances (PID in our case). Since the publication of Booth's algorithm in 1951, a huge number of improvement attempts were proposed, especially after the publication of a generalized version of modified Booth algorithm accompanied with its proof [6]. Most of the proposals aimed to reduce the number of partial products either by employing digital optimization techniques [7][8][9] or by using larger slices (higher radices) [10]. However, experience showed [11] that beyond 4-bit slices (radix 8), the complexity to generate hard partial products can not be managed in a realistic way. In [11], three metrics are provided for comparing the tradoffs when employing higher radix Booth recodings: partial product compression factor (gain), the number of hard multiples that must be precomputed (computation complexity), and partial product generation fanin (routing complexity).

To circumvent the problem of *hard* partial products in higher radices, the idea proposed in [12] consists in applying a recursive Booth recoding on the r-bit slice. While the idea is interesting, it relies upon a complicated mathematical formulation, leading to a complex control circuitry, and especially to an exaggerated latency (2n/r).

Based on the multibit recoding algorithm presented in [6], the equation (2.1.2) of [6] is rewritten in a simpler hardware-friendly form as follows:

$$Y = \sum_{j=0}^{(n/r)-1} \left( y_{rj-1} + 2^0 y_{rj} + 2^1 . y_{rj+1} + 2^2 y_{rj+2} + \cdots + 2^{r-2} y_{rj+r-2} - 2^{r-1} y_{rj+r-1} \right) 2^{rj} = \sum_{j=0}^{(n/r)-1} Q_j 2^{rj} \tag{3}$$

Where $y_{-1} = 0$ ; $r \in \mathrm{N}^*$; and $Q_j \in \left\{ -2^{r-1}, \ldots, 0, \ldots, 2^{r-1} \right\}$

In this general case, the multiplier Y is divided into n/r slices, each of r+1 bits. Each pair of two contiguous slices has one overlapping bit. To bypass the problem of *hard* partial products, $Q_j$ terms are split into 3-bit slices (r=2) with one overlapping bit. Thus, equation (3) takes the new simpler recursive form:

$$Y = \sum_{j=0}^{(n/r)-1} \left[ \left( y_{rj-1} + y_{rj} - 2.y_{rj+1} \right) 2^0 + \left( y_{rj+1} + y_{rj+2} - 2.y_{rj+3} \right) 2^2 + \ldots \right.$$

$$+\left(y_{rj+r-5}+y_{rj+r-4}-2.y_{rj+r-3}\right)2^{2(\frac{r}{2}-2)}+\left(y_{rj+r-3}+y_{rj+r-2}-2.y_{rj+r-1}\right)2^{2(\frac{r}{2}-1)}\Bigg]\ 2^{rj} \tag{4}$$

$$Y=\sum_{j=0}^{(n/r)-1}\left[\sum_{i=0}^{(r/2)-1}\left(y_{rj-1+2i}+y_{rj+2i}-2.y_{rj+1+2i}\right)2^{2i}\right]\ 2^{rj} \tag{5}$$

$$Y=\sum_{j=0}^{(n/r)-1}\left[\sum_{i=0}^{(r/2)-1}Q_{ji}\,2^{2i}\right]2^{rj} \tag{6}$$

With $Q_{ji}\in\{-2,-1,0,1,2\}$

There is no need to prove equation (4) since it is a combination of equations (3) and modified Booth algorithm (MBA) which were both already proven in [6] and [13], respectively.

To avoid dealing with special cases, n and r must be chosen as even numbers, with r as a divider of n. Thus, the DMAC equation becomes:

$$X.Y+T.Z=\sum_{j=0}^{(n/r)-1}\left[\sum_{i=0}^{(r/2)-1}\left(Q_{ji}.X+P_{ji}.T\right)\,2^{2i}\right]\ 2^{rj} \tag{7}$$

Depending on r value ranging from 2 to n, PIDs with various levels of parallelism and latencies (n/r+1) can be automatically generated with slight control complexity. The special cases of r=n and r=2 correspond to fully-parallel and fully-sequential PID, respectively. In between (r=4,n/2), partially-parallel PIDs are obtained. The outstanding advantage of this algorithm (6) is that *hard* partial products are generated using *simple* ones (2X, X) only. For a simplified hardware and lower power consumption, the step-by-step *sign-propagate* technique is employed [14].

Obviously, equation (6) does not reduce the number of partial products, but allows a modulable space-time partitioning of the multibit recoding algorithm (equation 3), where n/r sets comprising each r/2 partial products can be generated and summed either simultaneously or iteratively. Whilst the parallel implementation of equation (6) allows an important reduction of the critical path (using a carry-save adder CSA), it requires too much power. Therefore, only the serial implementation is retained. In this case, latency drops from (n/2+1) to (n/r+1), whereas the overhead on the total critical path, which goes through $\log_2(r/2)$ adder levels and which is equal to D in the case of MBA, is slightly increased $D+d.\log_2(r/2)$, where d is a unit delay of 1-bit adder. Note that we are using a logarithmic summation tree and not a linear one (CSA like).

An illustrative serial example with r=4 is described as follows:

$$Y=\sum_{j=0}^{(n/4)-1}\left(y_{4j-1}+y_{4j}+2\,y_{4j+1}+2^2\,y_{4j+2}-2^3\,y_{4j+3}\right)2^{4j} \tag{8}$$

$$Y=\sum_{j=0}^{(n/4)-1}\left[\sum_{i=0}^{1}\left(y_{4j-1+2i}+y_{4j+2i}-2.y_{4j+1+2i}\right)2^{2i}\right]\ 2^{4j} \tag{9}$$

$$Y=\sum_{j=0}^{(n/4)-1}\left[Q_{j0}+Q_{j1}\,2^2\right]2^{4j} \tag{10}$$

$$X.Y+T.Z=\sum_{j=0}^{(n/4)-1}\left[\left(Q_{j0}X+P_{j0}T\right)+\left(Q_{j1}X+P_{j1}T\right)2^{2}\right]2^{4j} \tag{11}$$

The mapping of equation (11) into a serial architecture is shown by Fig. 3. Such a case (r=4) would have required the computation of hard partial products (7X, 6X, 5X, 3X) if the simple form of equation (8) was used. Notice that MBA is a special case of RMRMA for r=2. For r=1, equation (10) corresponds to Booth algorithm (BA).

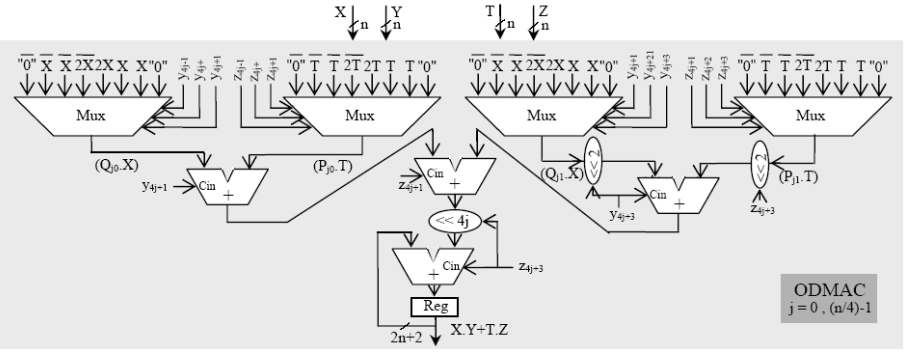



**Fig. 3.** Optimized double multiply-and-accumulate (ODMAC) architecture for r = 4

Table 2 comprises the implementation results of PIDs with n=16 and r=1,2,4,8,16. For instance, PID1 with r=4 not only achieves high improvement in latency (71%), but also maintains positive savings in power (14%) and speed (13%). These important achievements are partially due to logic-trimming performed by the synthesis tool on the constant coefficients. Such an operation is impossible in the case of PID [4] since the coefficients are stored into LUTs.

At this stage, a key question arises: among this panoply of PIDs, which one fits the best one's application case? The answer to this question is given in the next section.

**Table 2.** Implementation result comparison of RMRMA-based PID

| PID Core | Total Gate Count | Power* (mW) | Max. Clock Freq. (MHz) | Latency |
|---|---|---|---|---|
| PID [4] | 16728 | 223 | 47 | 17 |
| PID1_1 | 9286 (+44%) | 167 (+25%) | 62 (+32%) | 17 (+00%) |
| PID1_2 | 10642 (+36%) | 171 (+23%) | 62 (+32%) | 9 (+47%) |
| PID1_4 | 12443 (+26%) | 191 (+14%) | 53 (+13%) | 5 (+71%) |
| PID1_8 | 15688 (+06%) | 194 (+13%) | 44 (-06%) | 3 (+82%) |
| PID1_16 | 23545 (-41%) | 217 (+03%) | 26 (-45%) | 2 (+88%) |
| PID2_1 | 10661 (36%) | 176 (+21%) | 61 (+30%) | 17 (+00%) |
| PID2_2 | 11923 (29%) | 179 (+19%) | 61 (+30%) | 9 (+47%) |
| PID2_4 | 22962 (-37%) | 256 (-15%) | 43 (-08%) | 5 (+71%) |
| PID2_8 | 26073 (-56%) | 204 (+08%) | 37 (-21%) | 3 (+82%) |
| PID2_16 | 40327 (-141%) | 488 (-119%) | 23 (-51%) | 2 (+88%) |

*: Dynamic power consumption at 23MHz; PIDY_X: X = r
(+XY%): saving; (-XY%): overhead

## 4 Discussion

In embedded control, satisfactory control-rate (without performance degradation) at minimum power consumption is the main requirement. To select the most adequate PID for a given application, it's necessary to investigate how speed, power and hardware resources scales versus r factor for a fixed word length n. Referring to equation (7) and aided by Fig. 3, the ODMAC architecture scales as a binary tree with one stage of r mux(8:1) followed by $Log_2(r)+1$ stages of adders with a total of r adders too. Thus, the total delay cumulated by the critical path which goes through $Log_2(r)+2$ stages increases with $O(Log(r))$ complexity, whilst latency (n/r+1) decreases linearly $O(r)$, which makes the maximum control-rate increases as r increases. This is confirmed by implementation results shown in Table 3 and 4 corresponding to PID1 and PID2, respectively. The sole exception to this general rule is PIDX_n/2 which always yields to the highest control-rate compared to PIDX_n despite the numerous tests with various n values. This is justified since they exhibit very close latencies (3 and 2, respectively) and one stage difference in the critical path (n-1 and n, respectively), but an important multiplexer fanin difference (n/4 and n/2, respectively).

**Table 3.** Maximum power-consumption and control-loop-cycle of PID1

| PID Core | Power* (mW) | Max. Clock Freq. (MHz) | Latency | Max. Control Loop Cycle (MHz) |
|---|---|---|---|---|
| PID [4] | 456 | 47 | 17 | 2.76 |
| PID1_1 | 342 (+25%) | 62 | 17 | 3.65 (+32%) |
| PID1_2 | 350 (+23%) | 62 | 9 | 7.66 (+177%) |
| PID1_4 | 431 (+05%) | 53 | 5 | 10.60 (+284%) |
| PID1_8 | 365 (+20%) | 44 | 3 | 14.67 (+431%) |
| PID1_16 | 244 (+46%) | 26 | 2 | 13.00 (+371%) |

*: Dynamic power consumption at maximum clock frequency; PID1_X: X = r
Maximum control loop cycle = Maximum clock frequency / Latency

**Table 4.** Maximum power-consumption and control-loop-cycle of PID2

| PID Core | Power* (mW) | Max. Clock Freq. (MHz) | Latency | Max. Control Loop Cycle (MHz) |
|---|---|---|---|---|
| PID [4] | 456 | 47 | 17 | 2.76 |
| PID2_1 | 466 (-02%) | 61 | 17 | 3.59 (+30%) |
| PID2_2 | 475 (-04%) | 61 | 9 | 6.78 (+146%) |
| PID2_4 | 479 (-05%) | 43 | 5 | 8.60 (+211%) |
| PID2_8 | 328 (+28%) | 37 | 3 | 12.33 (+347%) |
| PID2_16 | 488 (-07%) | 23 | 2 | 11.50 (+317%) |

*: Dynamic power consumption at maximum clock frequency; PID2_X: X = r
Maximum control loop cycle = Maximum clock frequency / Latency

In terms of resource occupation, the total complexity grows linearly $O(r)$ as r multiplexers and r adders are required by ODMAC which is the most resource consuming block of PID architecture. This is also confirmed by the implementation results shown in Table 2. Note that each adder of each level of MAC and ODMAC as well as the two ones at the output of the PID (Fig. 1 and 2) are successively extended by one bit so that the total bit size of the control output u(k) becomes $2n+log_2(r)+2$. It's

necessary to do so to prevent the apparition of a possible overflow in the data-path which can cause signal clipping, limit cycles, and instabilities in the closed loop response [15].

As for power consumption, intuitively, one would expect to see PID1_16 of Table 3 as being the most rapid and the most power consumer too, for the reason that it exhibits the smallest latency and the biggest total gate count! While it is almost true for the latter (13 MHz, before the first), it is quite the opposite for the former (244 mW, the smallest one). The explanation is that power consumption ( $P = 0.5\, V_{dd}^2\, C_{sw}\, F_{clk}$ ) depends linearly on the frequency ($F_{clk}$), which is in this case 26 MHz (the smallest one) and also on the switched capacitance ($C_{sw}$) which describes the average capacitance charged during each clock period ($1/F_{clk}$). In fact, $C_{sw}$ depends on a number of parameter (circuit structure, logic function, input pattern dependence…) and not only on the total gate count (more precisely, not only on the total physical capacitance of the circuit). Furthermore, a study [16] that analyzed the dynamic power consumption in Xilinx's FPGA revealed the following share: 60% by routing, 16% by logic, and 14% by clocking. The reason is that routing is intensively segmented, using pass logic and buffers.

When both high control-rate close to 13MHz and low power are required, PID1_16 (244 mW at 13MHz) stands as the best candidate compared to PID1_8 (323 mW at 13MHz). However, it's noteworthy to mention that this comparison stands valid only for the special case of 16-bit word-length PID, for a given set of coefficients, mapped on XC2S150E-7FT256 FPGA circuit and using Xilinx's XST synthesis tool, version 9.2. Results could significantly change under other conditions, especially when considering the logic trimming process which is essentially dependant on the bit-arrangement of the coefficients. For a minimum influence of the trimming operation on the synthesized results, appropriate coefficients were used such as all *Qj* terms are represented except the null one to avoid generating null partial products that greatly simplify the circuit logic. In fact, constant coefficients PIDs (PID1) are somehow unpredictable with regard to r. They are coefficient dependant. Adversely, PID2 is not involved with the trimming process since coefficients are time varying. Implementation results comprised in Table 4 show that PID2_8 is the best at all aspects for the same reasons cited above. In sum, when high control-rate is the ultimate objective, PIDX_n/2 is the best candidate whatever n value. But in the case where both high speed and low power are required, timing and power evaluations are necessary to decide which PID to select: either PIDX_n/2 or PIDX_n.

Finally, when only low power is targeted, PIDX_1 is the best candidate. We dealt here with extreme situations only, but for a given couple (cr, pc) of control-rate and power consumption, several candidates are possible. Yet, the best PID is the one which requires the smallest gate count.

So far, speed and power have been considered in isolation to area which becomes critical, and sometimes prohibitive, for large word-length n due to the fact that PID is basically built of a set of multipliers (three or five) that scale quadratically with word length. The bigger is the area, the higher is the cost. Consequently, another advantage of RMRMA algorithm is to cope also with the cost issue as an additional constraint to speed and power.

We deliberately chose Spartan2e FPGA to compare our results with those provided in [4]. A mapping on a recent FPGA circuit (Virtex6) using XST 12.1 version of extreme PID2 delivered state-of-the-art results grouped in Table 5.

Note that control-rate scaled with an average factor of 2, while power dissipation scaled with an average factor of 45. This is not surprising, since Spartan2e and Virtex6 were fabricated with two differently scaled technology processes: 150 nm and 40 nm, respectively. Therefore, the physical capacitances of the circuit in Virtex6 are relatively too much smaller. Additionally, the supply-voltages ($V_{dd}$) used for internal core (Vccint) and for output blocks (Vcco) are respectively 1.8V and 3.3V for Spartan2e, 1V and 2.5V for Virtex6. Furthermore, the efficient advances made in CAD tools (from Xilinx ISE 9.1 to 12.1 versions) as well as in FPGA architecture, such as advanced segmented-routing, much contributed to lower the power consumption [17]. Power consumption evaluation studies [16][17] based on simulation and measurements, targeting Virtex2 and Virtex6 families revealed the following results: 5.9µW per CLB per MHz, and 1.09 mW per 100 MHz at 38% toggle rate, respectively. These studies roughly confirm our power results as proximate values are obtained.

**Table 5.** Maximum power-consumption and control-loop-cycle of PID2 mapped on Virtex6

| PID Core | Number of Slices | Power* (mW) | Max. Clock Freq. (MHz) | Latency | Max. Control Loop Cycle (MHz) |
|---|---|---|---|---|---|
| PID2_1 | 231 | 23 | 122 | 17 | 07.17 |
| PID2_8 | 1060 | 04 | 90.5 | 3 | 30.16 |
| PID2_16 | 1963 | 13 | 50.4 | 2 | 25.19 |

*: Dynamic power consumption at maximum clock frequency; PID2_X: X = r
Maximum control loop cycle = Maximum clock frequency / Latency

Timing and power evaluations were performed in the following conditions. Delays were calculated for two types of paths: Clock-To-Setup and all paths together (Pad-To-Setup, Clock-To-Pad and Pad-To-Pad.) The Clock-To-Setup gives more precise information on the delays than other remaining paths, which depend in fact on I/O Block (IOB) configuration (low/high fanout, CMOS, TTL, LVDS…). Thus, all delays (frequencies) presented so far are clock-to-setup delays with the highest speed grade of the FPGA circuit. As for power, we chose the highest Vcco voltage value (3.3 for Spartan2e and 2.5 for Virex6) with a maximum toggle activity of 50%, which means that Flip-Flops (FFs) toggle one time during each clock cycle. The reason is that only simple-edge triggered FFs are used for synthesis (no double-edge FFs).

## 5 Conclusion

Analytical scaling-complexity evaluations with respect to the couple (n,r), confirmed also by software simulations, revealed useful information which is summarized as follows:

- PIDX_n/2 is the fastest PID that yields to the highest control-rate (30 MHz for PID2_8 mapped on Virtex6, with (n,r)=(16,8) );
- PIDX_1 is the most power efficient PID when speed is not a concern;
- PIDX_n and PIDX_n/2 are the most efficient PIDs when both high control-rate and low-power dissipation are required.

Further extension to the present work is to apply the same or appropriate partitioning in conjunction with RMRMA algorithm to the set of recurrent equations of an arbitrary number of multi-loop PID controllers taken as a whole.

## References


1. Åström, K., Hägglund, T.: PID Controllers: Theory, Design, and Tuning, 2nd edn. The Instrument Society of America, Research Triangle Park, NC, USA (1995) ISBN: 1-55617-516-7, Copyright
2. Zhao, W., et al.: FPGA Implementation of Closed-Loop Control Systems for Small-Scale Robot. In: Proceedings of the IEEE 12th International on Advanced Robotics (ICAR), pp. 70–77 (2005)
3. Samet, L., et al.: A Digital PID Controller for Real-Time and Multi-Loop Control: a Comparative Study. In: Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS), vol. 1, pp. 291–296 (1998)
4. Fong, Y., Moallem, M., Wang, W.: Design and Implementation of Modular FPGA-Based PID Controllers. IEEE Trans. on Industrial Electronics 54(4), 1898–1906 (2007)
5. Wittenmark, B., Astrom, K.J., Arzenin, K.E.: Computer control: An overview. Technical Report of Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden (April 2003), `http://www.control.lth.se/kursdr/ifac.pdf`
6. Sam, H., Gupta, A.: A Generalized Multibit Recoding of Two's Complement Binary Numbers and its Proof with Application in Multiplier Implementation. IEEE Trans. on Computers 39(8) (August 1990)
7. Lamberti, F.: Reducing the Computation Time in (Short Bit-Width) Two's Complement Multiplier. IEEE Trans. on Computers 60(2), 148–156 (2011)
8. Kuang, S.R., Wang, J.P., Guo, C.Y.: Modified Booth Multipliers with a Regular Partial Product Array. IEEE Trans. on Circuit and Systems II, Express Brief 56(5) (May 2009)
9. Kang, J.Y., Gaudiot, J.L.: A Simple High-Speed Multiplier Design. IEEE Trans. on Computers 55(10) (October 2006)
10. Crookes, D., Jiang, M.: Using Signed Digit Arithmetic for Low-Power Multiplication. Electronics Letters 43(11) (May 2007)
11. Seidel, P.M., McFearin, L.D., Matula, D.W.: Secondary Radix Recodings for Higher Radix Multipliers. IEEE Trans. on Computers 54(2) (February 2005)
12. North, R.C., Ku, W.H.: $\beta$-Bit Serial/Parallel Multipliers. Journal of VLSI Signal Processing 2, 219–233 (1991)
13. Rubinfield, L.P.: A Proof of the Modified Booth Algorithm for Multiplication. IEEE Trans. On Computers C-24(10), 1014–1015 (1975)
14. Henlin, D.A., Fertsch, M.T., Mazin, M., Lewis, E.T.: A 16 bit x 16 bit Pipelined Multiplier Marcrocell. IEEE Journal of Solid-State Circuits SC-20(2), 542–547 (1985)
15. Kelly, J.S., et al.: Design and Implementation of Digital Controllers for Smart Structures Using Field Programmable Gate Arrays. Smart Material Structure Journal, PII: S0964-1726 (97) 87085-1, 559–572 (1997); printed in the UK
16. Shang, L., Kaviani, A.S., Bathala, K.: Dynamic Power Consumption in Virtex-II FPGA Family. In: Proceedings of FPGA Conference, Monterey, California, USA, pp. 157–164 (February 2002)
17. Xilinx Inc.: Virtex6 FPGA: Satisfying the Insatiable Demand for Higher Bandwidth. PN 2403, Printed in the USA, Copyright (2009), `http://www.xilinx.com/publications/prod_mktg/Virtex6_Product_Brief.pdf`

# Design of Resonant Clock Distribution Networks for 3-D Integrated Circuits

Somayyeh Rahimian, Vasilis F. Pavlidis, and Giovanni De Micheli

LSI - EPFL, CH-1015, Switzerland
{somayyeh.rahimianomam,vasileios.pavlidis,
giovanni.demicheli}@epfl.ch

**Abstract.** Designing a low power clock network in synchronous circuits is an important task. This requirement is stricter for 3-D circuits due to the increased power densities. Resonant clock networks are considered efficient low-power alternatives to conventional clock distribution schemes. These networks utilize additional inductive circuits to reduce the power consumption while delivering a full swing clock signal to the sink nodes. A design method for 3-D resonant clock networks is presented. The proposed design technique supports resonant operation for pre-bond test, an important requirement for 3-D ICs. Several 3-D clock network topologies are explored in a 0.18 μm CMOS technology. Simulation results indicate 43% reduction in the power consumed by the resonant 3-D clock network as compared to a conventional buffered clock network.

**Keywords:** 3-D integration, clock distribution networks, resonant clocking.

## 1 Introduction

Aprimary challenge in designing synchronous circuits is how to distribute the clock signal to the sequential parts of the circuit [1].This issue can be more challenging for 3-D circuits since a clock path can spread across several planes with different physical and electrical characteristics [2].

As the area of the integrated circuits increases, larger networks are required to distribute the clock signal, which results in higher capacitive loads and resistive losses of the interconnects degrading the signal integrity along these interconnects. A common solution to alleviate this problem is to insert clock buffers in the intermediate nodes of the clock network. Although buffer insertion improves clock signal integrity, clock buffers significantly increase the power consumed by the network. 3-D integration drastically decreases the interconnect length of the global wires, which can reduce the number of clock buffers and result in more power-efficient clock networks. Alternatively, thermal issues are more pronounced in 3-D integrated circuits. Clock networks consume a great portion of the power dissipated in a circuit [3].Consequently, designing low power clock networksfor 3-D circuits is aprimary challenge.

An efficient approach to eliminate the repeaters and reduce power is to use resonant clocking [8-10]. In this approach, on-chip inductance is added to the clock

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 267–277, 2011.

network and forms a resonant circuit with the interconnect capacitance, decreasing, in this way, the power consumed by the network, since the energy alternates between electric and magnetic fields instead of dissipating as heat.

Testing is another important issue in 3-D circuits. Pre-bond test, includes testing each plane before bonding to other planes and can improve the yield of 3-D systems [4]. 3-D clock networks often include several disconnected networks in some of the planes, which connect with through-silicon-vias (TSVs) to the plane where the main tree feeds the clock signal to the entire clock distribution network. To provide pre-bond test, each plane needs a complete clock tree. A technique to provide such a tree by employing additional wiring has recently been presented [5]. In another approach each disconnected clock tree is driven by a DLL, enabling pre-bond test for each plane of the 3-D system [6].

Both methods support pre-bond test for traditional clock networks by providing a means to connect the local networks within each plane. The nature of resonant clock networks, however, poses different constraints. For example, resonant operation should be achieved for each individual plane during testing irrespective of the employed pre-bond testing approach. The design of 3-D resonant clock networks and the related constraints have not been explored as compared to traditional planar clock networks [4-8]. Consider for example, a 2-D circuit that employs a monolithic *LC* tank to resonate. This design would be inadequate for a 3-D resonant clock network, since pre-bond test is not supported. The resonant 3-D clock network should be designed such that resonant operation at a specific frequency is individually achieved for each plane as well as for the entire 3-D system.

The contribution of this paper, consequently, is a novel design methodology that addresses these two objectives. The proposed 3-D resonant clock networks considerably lower the power of the clock distribution system, while pre-bond test is supported by the proper design and allocation of the *LC* tanks within each plane. In this way, resonant operation is ensured for each plane either in test or functional mode and the clock signal characteristics are maintained within each plane and for either operating mode. Different designs of a resonant clock network for up to eight-plane 3-D circuits are investigated. Simulation results indicate that using a resonant clock network can significantly decrease the power consumption of the clock tree in 3-D circuits. Furthermore, the results confirm that the power consumed in a 3-D clock network is lower than a 2-D clock network due to the shorter interconnect length.

In the following section, the design of resonant clock networks for 2-D circuits is reviewed. A design methodology for 3-D resonant clock networks supporting pre-bond test is proposed in Section 3. Simulation results are presented in Section 4 and some conclusions are drawn in the last section.

## 2 Resonant Clocking

A seminal work, introducing the concept and design of resonant transmition lines has been published in [7]. A design ofa global clock distribution network is presented in [8] in which four resonant circuits are connected to a conventional H-tree structure as illustrated in Fig. 1. Each quadrant consists of an on-chip spiral inductor that resonates with the wiring capacitance of the clock network and the decoupling

capacitor is connected to the other end of the spiral inductor.A simple lumped circuit model is utilized to determine the resonant inductance. The resonant frequency of the network is (in first-order) estimated by $f = \frac{1}{2 \cdot \pi \cdot \sqrt{LC}}$ where $C$ and $L$, respectively, denote the equivalent capacitance of the network wiring and inductance of the spiral inductors. The decoupling capacitor is employed to provide a positive voltage offset on the grounded end of the resonant inductor and adapt the voltage level to the voltage supply level of CMOS logic [11]. This capacitor should be sufficiently large to guarantee that the resonant frequency of the decoupling capacitor $f_{decap} = \frac{1}{2 \cdot \pi \cdot \sqrt{L\, C_{decap}}}$ is much lower than the desired resonant frequency of the clock network.

Based on this structure, a design methodology for resonant H-tree clock distribution networks is proposed in [9]. In this work, the clock tree is modeled with a distributed *RLC* interconnect as illustrated in Fig. 2. This electrical model is utilized to determine the parameters of the resonant circuit and the output impedance of the clock driver such that the power consumed by the network and the clock driver are minimum, while a full swing signal is delivered at the output nodes.

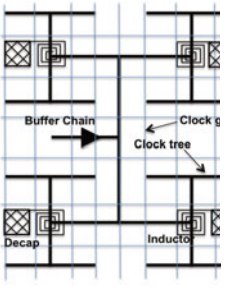



**Fig. 1.** Resonant clock network with four resonant circuits [8]

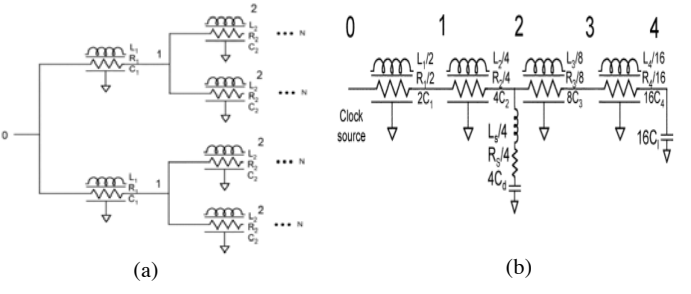



**Fig. 2.** *RLC* model of a 16-sink H-tree clock network where (a) is the distributed *RLC* model and (b) is the simplified *RLC* model of a resonant network [9]

To delivera full swing signal at the sink nodes, the magnitude of the transfer function of the network, $H_{out}$, should be close to one. This parameter is often fixed to 0.9 [9, 11] (for the remainder of the paper a "full swing signal" implies any signal swing that satisfies this specification). As shown in [9], when $|H_{out}|$ is fixed at 0.9, the driver resistance can be determined by

$$R_{driver} = \sqrt{\frac{\left|H_{\omega}(j\omega)\right|^2 \cdot \left|Z_{in\_\omega}\right|^2}{0.9^2}} - Img(Z_{in\_\omega})^2 - Real(Z_{in\_\omega}) \ , \quad (1)$$

Where $H_{\omega}$ and $Z_{in\_\omega}$ notate the transfer function and input impedance of the network.

Several resonant circuits can be utilized to improve the characteristics of the clock signal. In a symmetric H-tree clock network, the number of *LC* tanks (resonant circuits) also depends on the location of these circuits. If the resonant circuits are

placed closer to the driver, fewer circuits are needed and, alternatively, where these circuits are placed close to the sink nodes, more *LC* tanks are required. Since the equivalent inductance is the parallel combination of all the inductors as shown in Fig. 2, increasing the number of resonant circuits leads to a larger required inductance for each circuit. Using a higher number of larger inductors resultsin larger area occupied by the resonant inductors.

The number of resonant circuits also affects the output signal swing. As discussed in [11], by increasing the number of resonant circuits and placing these circuits closer to the sink nodes, each inductor resonates with a smaller part of the circuit resulting in lower attenuation of the output signal swing. Alternatively, increasing the number of resonant circuits and using larger inductors in each *LC* tank reduces the quality factor of the *LC* tanks, since in spiral inductors the effective series resistance (ESR) increases more aggressively than the inductance [9]. A lower quality factor for resonant circuits produces a higher signal loss and decreases the output signal swing.

Considering a clock network with 256 sinks driven by an ideal clock driver, $|H_{out}|$ for a different number of resonant circuits over a wide range of resonant inductance is shown in Fig. 3, where for fewer than 16 and more than 64 resonant circuits, the $|H_{out}|$ cannot meet the 0.9 signal swing depicted by the dotted line.

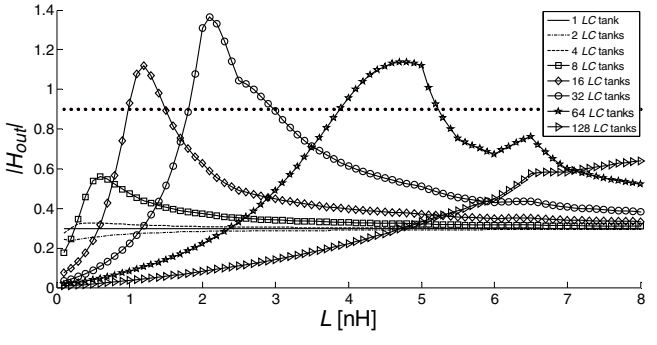



**Fig. 3.** $|H_{out}|$ for different number of resonant circuits

To determine the number of resonant circuits that maximize the output signal swing, one approach is to only consider the capacitance of the clock network and employ $f = \frac{1}{2 \cdot \pi \cdot \sqrt{LC}}$ to determine the total resonant inductance [11]. By doubling the number of *LC* tanks, the inductance of each tank is also doubled. In this approach, the inductive component of the network wires is not considered. In large clock networks with long interconnects, the inductance of the wires cannot be neglected [9]. Furthermore, this method assumes that placing the resonant circuit in different locations does not change the equivalent capacitance of the network (*i.e.* the capacitance seen by the primary clock driver). These simplifications can result in inaccurate estimation of the resonant inductance, adversely affecting the signal swing.

The signal swing for a clock network with 256 sinks using an ideal driver for different number of *LC* tanks is illustrated in Fig. 4. Employing any inductance within

the crosshatched ranges, this clock network can meet the signal swing specifications. The resonant inductance determined with the simplified approach is illustrated by the dotted lines, where due to imprecise estimation of the inductance, the clock network cannot deliver a full swing signal to the sinks (as required by the dashed horizontal line). As depicted in Fig. 4, using the simplified model from [11] can reduce $|H_{out}|$ from 0.9 to 0.65.

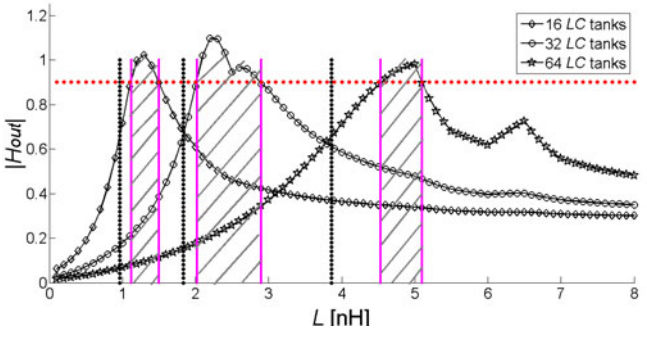



**Fig. 4.** $|H_{out}|$ for different number of *LC* tanks and resonant inductance using the model in [11] (dotted vertical lines)and the proposed approach (solid vertical lines)

In our approach, a distributed *RLC* model for the network wires is used to determine the required parameters for resonance. Different locations for the resonant circuits from the root to the sinks are investigated. For each location, the driver resistance is adapted to produce a transfer function magnitude of 0.9 for a wide range of inductor sizes using (1). The inductance for which the driver resistance is maximumor the power consumption is minimum (which do not necessarily occur for the same inductance) is determined.

## 3 Resonant Clocking for 3-D ICs

Different clock network topologies can be considered to adapt the conventional (planar) resonant clock networks to 3-D circuits [2]. In the first topology denoted as "symmetric topology", each plane contains resonant circuits and can be separately investigated. In another structure, denoted as "asymmetric topology", the resonant circuit is placed in only one plane and should resonate with the total capacitance of the 3-D stack at the desired frequency. During pre-bond test, each plane should separately resonate. Note that this requirement is an additional constraint specific to resonant networks and is completely different to the techniques that can be employed to connect the local networks [5, 6] in either a standard or resonant clock distribution approach. Consequently asymmetric structures, which can be considered as an extension of 2-D clock networks, do not support pre-bond test in a straightforward manner, since the resonant circuit is contained within only one plane.

The other important parameter in designing a resonant 3-D clock network is the number of TSVs used to connect the physical planes. From this perspective, different topologies can be explored, for example, using a single TSV in the center of each plane or by using multiple TSVs. In the multiple TSV structure, one of the planes contains a complete clock tree, where for the other planes the clock network consists of several disconnected local networks each connected to the first plane by TSVs. Increasing the number of TSVs provides more local networks increasing the area occupied by the TSVs. Four topologies for a two-plane 3-D circuit with 32 sinks are shown in Fig. 5. *RLC* models to analyze the different 3-D structures are depicted in Fig.6. In single-TSV topologies the equivalent resistance of the circuit is determined as the resistance of each plane divided by the number of planes. By increasing the number of TSVs and omitting some wires in the upper planes, the resistance of these wires is not divided by the number of planes which results in higher equivalent resistance for the 3-D circuit. Alternatively, increasing the number of TSVs results in decreased capacitance for the 3-D circuit.

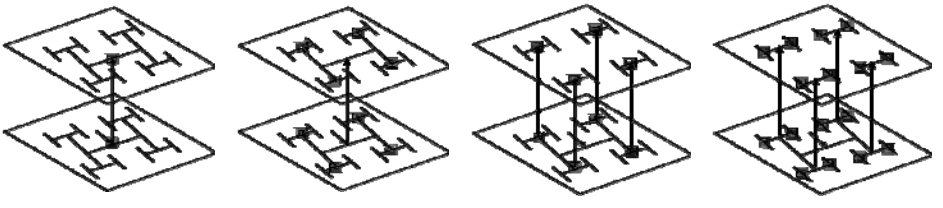



**Fig. 5.** Different topologies for a two-plane 3-D resonant clock network where (a) is a single TSV structure with one *LC* tank per plane, (b) is a single TSV structure with four*LC* tanks per plane, (c) is a four TSV structure with four *LC* tanks per plane, and (d) is a four TSV structure with eight *LC* tanks per plane.

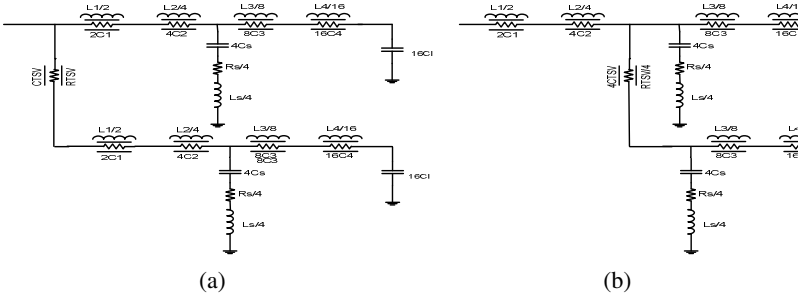



**Fig. 6.** *RLC* model for a two-plane 3-D circuit with four *LC* tanks where (a) is the model for the single-TSV and (b) is the model for four-TSV structures

To specify the parameters of resonance for a 3-D system with a specific number of TSVs, a distributed *RLC* model of the clock distribution network is employed. The

number of *LC* tanks in each plane is assumed to be more than the TSVssuch that at least one *LC* tank is connected to each local network. The location of *LC* tanks is swept from the TSVs to the sinks. The driver resistance described by (1) is plotted over the inductance to determine the resonant parameters, similar to the 2-D case. In the 3-D system the transfer function for different planes can be different due to the effect of the TSV. Not surprisingly, the last plane (the plane with the greatest distance from the clock driver) has the lowest signal swing. The driver size should be determined such that the signal swing for every plane meets the specifications. Consequently, the transfer function magnitude of the last plane should be used in (1). Following this process, the number of the *LC* tanks and the parameters of the resonant circuits are determined for normal operation.

For pre-bond test, the power consumed by the clock network within each plane is low as compared to the total power consumed by the 3-D system. Consequently, the power consumed by one plane during the pre-bond test mode is a secondary parameter and since the planes are not bonded, heat is removed faster and the thermal constraints are more relaxed. The predominant parameter in test mode is the voltage swing. The clock network should deliver a full swing clock signal to the sinks to test each plane.

During the test mode, each plane should consist of a complete clock network. Additional wiring can be used in each plane to connect local networks except for the first plane [5]. There are two important design parameters in pre-bond test, sizing the additional wires and clock drivers used only during testing. These parameters should be chosen such that a full swing signal is delivered to the sink nodes. There is a tradeoff for determining these parameters. If the wire width is decreased, a larger clock driver should be utilized. Alternatively, increasing the width of the wire results in a smaller clock driver but increases the area occupied by the redundant wires, which are used only during pre-bond test. One simplistic approach is to replicate the tree of the first plane for all the planes during the test mode and also use a driver with the same strengthas the main clock driver. This choice, however, leads to overdesign wasting silicon.

To determine these parameters, the wire width is swept and the related driver resistance is obtained by (1). The driver resistance for different wire width for a two-plane 3-D system with 8 TSVs and 16 resonant circuits is plotted in Fig. 7.

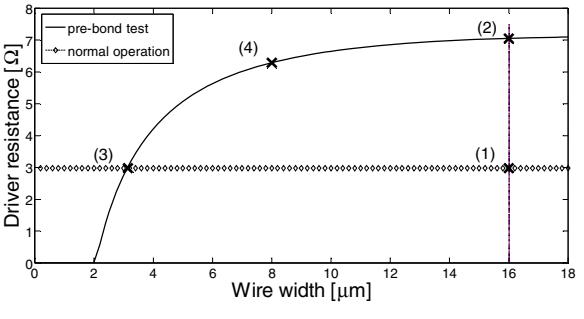



**Fig. 7.** Driver resistance *vs* wire size

The parameters estimated using the simplistic approach is shown at point 1. For points 2,3, and 4, the parameters are determined using our approach where for point 2 the driver resistance is maximum using the wire size of the first plane and for point 3 the wire size is minimum while employing a replica of the main driver. Point 4 provides a better tradeoff, since the wire width decreases to half of the wire width of point 2 andthe resistance of the driver decreases by only 15%.This resistance is twice as large as compared to the resistance of the driver used to drive the entire clock distribution network.

## 4 Simulation Results

Assuming an H-tree network as the preferred network in [9], the number of sinks is determined based on the circuit area $A$ and the load capacitance that each sink drives, $C_L$. A case study of an H-tree resonant clock network with 256 leaves is considered. The load capacitance at each node is assumed to be 20 fF and the operating frequency is 5 GHz. The interconnect length can be determined from the total network area. For a circuit area of $A = l \cdot l$ the length of the longest and shortest interconnects is $l/4$ and $l/2^{n+2}$, respectively, where $n$ indicates the number of sinks.

The PTM model for a 0.18 μm CMOS technology is used to estimate the resistance, inductance, and capacitance of the horizontal interconnects. The total area of the network is 3.4 mm $\times$ 3.4 mm. The parameters of the interconnects are listed in Table 1, where $L_1$ to $L_8$ indicate the different wire segments from the driver to the sinks.

**Table 1.** Interconnect parameters used in the investigated clock network.

| | L1-L5 | L6 | L7 | L8 |
|---|---|---|---|---|
| $R$ [ Ω/mm] | 2.75 | 5.5 | 11 | 22 |
| $L$ [nH/mm] | 0.46 | 0.6 | 0.72 | 0.82 |
| $C$ [fF/mm] | 254.6 | 175.4 | 130 | 103 |

For a conventional (non-resonant) clock network, inverters are properly inserted at the intermediate nodes to deliver a full swing clock to the output, while in the resonant clock network, resonant circuits are added to the clock tree to provide a proper clock signal at the output. The amount of the resonant inductance is determined as described in the previous section. The decoupling capacitor that should be sufficiently large not to affect the frequency of resonance is set to 60 pF. The effective series resistance (ESR) for the inductors is determined from [9].

Different topologies of 3-D circuits are explored. To form a 3-D system, the 2-D circuit is folded into several planes. The electrical and physical characteristics of the TSVs used to connect these planes are based on [13]. The number of *LC* tanks, the inductor size for each resonant circuit, and the clock driver resistance for normal and pre-bond operation are listed in Table 2. The size of the wiresin the upper planes for pre-bond test is determined and compared with the size of the wires in the first plane. The resulting decrease in wire width is also listed in this table.

Increasing the number of TSVs can result in a smaller primary driver and lower power due to the decreased capacitive load of the clock network. Alternatively, increasing the resistance of the circuit requires a larger primary driver, increasing the power consumed by the clock network. Predicting which behavior is dominant is not straightforward and strongly depends on the interconnect characteristics of the clock network. Using wide wires results in stronger capacitive behavior, while in long wires the resistive component can become dominant. For this case study, as shown in Table 2, there is not a uniform trend for the design parameters as a function of the number of TSVs.

The power consumed by different topologies for standard and resonant clock networks is listed in columns 7 and 8 of Table 2. As reported in this table, the power consumed by the resonant clock network is considerably lower than the standard network in 3-D circuits. This improvement is accompanied by an increase in the area occupied by the resonant circuits. The area of a resonant clock network increases due to these additional circuits, but alternatively, omitting the clock buffers can decrease the area of the resonant networks.

Increasing the number of planes decreases the length of the wires in the network for a specific number of sinks. Omitting long interconnects as required for some of the topologies shown in Fig. 5 reduces the resistive voltage drop and the capacitance of the network and decreases the power consumed by the network. For a 3-D circuit with two planes, the power consumption reduces up to 64%, where this reduction reaches to 70% and 72% for circuits with four and eight planes, respectively. Decreasing the equivalent capacitance of the network also results in larger resonant inductance and increases the area of the resonant clock network. Alternatively, by increasing the number of planes, the driver size and the width of the additional wires required for pre-bond test, reduces due to the smaller circuit area in each plane.

**Table 2.** Design parameters and power consumption for different topologies

| | | # *LC* tanks | *L* [nH] | $R_{driver}$ [ Ω ] | $R_{driver}$ [ Ω ] pre-bond | Wire width reduction (%) | Power [mW] | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | Standard | Resonant |
| 2-D | - | 32 | 2.2 | 1.2 | - | - | 543 | 310 |
| 3-D 2 planes | 1 TSV | 16 | 1.5 | 3.25 | 7.16 | 0 | 385 | 241 |
| | 2TSV | 16 | 1.4 | 2.95 | 6.2 | 50 | 374 | 230 |
| | 4 TSV | 16 | 1.5 | 2.77 | 6.1 | 50 | 353 | 244 |
| | 8 TSV | 16 | 1.3 | 2.98 | 5.6 | 50 | 312 | **196** |
| | 16 TSV | 32 | 3 | 2.55 | 5.7 | 50 | 347 | 218 |
| | 32 TSV | 64 | 5 | 0.97 | 4.5, 2.4 | 25, 50 | 304 | 203 |
| 3-D 4 planes | 1 TSV | 16 | 1.8 | 2.15 | 8.8 | 0 | 308 | 190 |
| | 2 TSV | 16 | 1.6 | 3.55 | 10 | 87.5 | 297 | 174 |
| | 4 TSV | 16 | 1.7 | 3.9 | 10.8 | 81.5 | 289 | **162** |
| | 8 TSV | 32 | 3.2 | 3.5 | 8.4 | 81.5 | 295 | 176 |
| | 16 TSV | 64 | 6.5 | 3 | 6.9 | 75 | 308 | 184 |
| 3-D 8 planes | 1 TSV | 8 | 1 | 2.4 | 17.8 | 0 | 299 | 173 |
| | 2 TSV | 16 | 2 | 3.5 | 20 | 87.5 | 283 | **152** |
| | 4 TSV | 32 | 3 | 3.27 | 15 | 81.5 | 311 | 188 |
| | 8 TSV | 64 | 5 | 2.65 | 9 | 50 | 307 | 181 |

The preferred 3-D structure that decreases the power consumption of the clock network by 72%, consists of eight planes which is the maximum number of planes investigated in this case study. Each plane connects to the others using 2 TSVs. 16 *LC* tanks are used in this structure (two *LC* tanks per plane) and the inductance for each *LC* tank is 2 nH. The number of planes and the number of *LC* tanks is determined such that the highest voltage swing is achieved (which implies that the Q-factor of the spiral inductors is also the highest).

## 5 Conclusions

A design methodology for resonant clock networks in 3-D circuits is proposed in this paper. The number of *LC* tanks, the resonant circuit parameters, and the driver size fornormal operation are determined such that a full swing signal is provided at the sink nodes and the power consumption of the circuit is minimized. The effect of different parameters including the number of planes and number of TSVs among the planes for designing 3-D resonant clock networks is investigated. An approach to minimize the additional wire width and clock driver size for pre-bond test is proposed. A 256-sink H-tree clock network operating at 5 GHz is considered as the case study where a power reduction of 72% is achieved for aneight-plane resonant clock network in comparison to a 2-D standard network.

**Acknowledgements.** This work is funded in part by the Swiss National Science Foundation (No. 260021_126517), European Research Council Grant (No. 246810 NANOSYS), and Intel Braunschweig Labs, Germany.

## References

1. Friedman, E.G.: Clock Distribution Network in Synchronous Digital Integrated Circuits. In: Proceedings of the IEEE 89(5), 665–692 (2001)
2. Pavlidis, V.F., Friedman, E.G.: Clock Distribution Network for 3-D Integrated Circuits. In: Custom Integrated Circuits Conference (CICC), pp. 651–654 (2008)
3. Xanthopoulos, T., et al.: The Design and Analysis of the Clock Distribution Network for a 1.2 GHz Alpha Microprocessor. In: Proceedings of the IEEE International Solid-State Circuits Conference, pp. 402–403 (2001)
4. Li, J., Xiang, D.: DfT Optimization for Pre-bond Testing of 3D-SICs containing TSVs. In: Proceedings of the IEEE International Conference on Computer Design, pp. 474–479 (2010)
5. Zhao, X., Lewis, D.L., Lee, H.H.S., Lim, S.K.: Pre-bond Testable Low-Power Clock Tree Design for 3D Stacked ICs. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design – Design of Technical papers, pp. 184–190 (2009)
6. Buttrick, M., Kundu, S.: On Testing PrebondDies ithIncomplete Clock Networks in a 3D IC Using DLLs. In: Proceedings of Design, Automation and Test in Europe Conference & Exhibition, pp. 14–18 (2011)
7. Chi, V.L.: Salphasic Distribution of Clock Signals for Synchronous Systems. IEEE Transactions on Computers 43(5), 597–602 (1994)

8. Chan, S.C., Shepard, K.L., Restle, P.J.: Design of Resonant Global Clock Distributions. In: Proceedings of the International Conference on Computer Design, pp. 248–253 (2003)
9. Rosenfeld, J., Friedman, E.G.: Design Methodology for Global Resonant H-tree Clock Distribution Networks. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 15(2), 135–148 (2007)
10. Sathe, V.S., Kao, J.C., Papaefthymiou, M.C.: A 1 GHz FIR Filter with Distributed Resonant Clock Generator. In: Symposium on VLSI Circuits, pp. 44–45 (2007)
11. Guthaus, M.R.: Distributed *LC* Resonant Clock Tree Synthesis. In: Proceedings of International Symposium of Circuits and Systems, pp. 1215–1218 (2011)
12. Katti, G., et al.: Electrical Modeling and Characterization of Through Silicon Via for Three–Dimensional ICs. IEEE Transactions on Electron Devices 57(1), 256–262 (2010)

# Power and Area Optimization of 3D Networks-on-Chip Using Smart and Efficient Vertical Channels

Amir-Mohammad Rahmani[1,2], Kameswar Rao Vaddina[1,2], Pasi Liljeberg[1], Juha Plosila[1], and Hannu Tenhunen[1]

[1] Computer Systems Lab., Department of Information Technology, University of Turku, Finland
[2] Turku Centre for Computer Science (TUCS), Finland
{amir.rahmani,vaddina.rao,pasi.liljeberg, juha.plosila,hannu.tenhunen}@utu.fi

**Abstract.** 3D NoC offers greater device integration, faster vertical interconnects and more power efficient inter-layer communication due to the beneficial attribute of short through silicon via (TSV) in 3D IC technologies. However, TSV pads used for bonding to a wafer layer, occupy significant chip area and result in routing congestions and expensive manufacturing process. This can lead to a significant reduction in 3D ICs' yield and higher power densities compared to 2D NoCs. In this paper, a power-efficient and low-cost inter-layer communication scheme is proposed as one way to mitigate these challenges. Instead of using a pair of unidirectional channels for inter-layer communication, utilizing a high-performance bidirectional channel enables a system to benefit from low-latency nature of the vertical interconnects and to remarkably reduce the number of TSVs. Additionally, we present a forecasting-based dynamic frequency scaling technique for reducing the power consumption of the inter-layer communication. Our extensive simulations demonstrate significant area and power improvements compared to a typical symmetric 3D NoC.

## 1 Introduction

The Networks-on-Chip (NoCs) paradigm, based on a modular packet-switched mechanism, was proposed to be used in complex Systems-on-Chip (SoCs) as a promising communication platform because of scalability, better throughput and reduced power consumption [1]. However, increasing the number of cores over a 2D plane is not efficient due to long interconnects. Three-dimensional (3D) integration is a viable design paradigm to overcome the existing interconnect bottleneck in integrated systems and enhance system power/performance characteristics. In 3D integration technology, multiple layers of active devices are stacked above each other and vertically interconnected using Through-Silicon Vias (TSVs). The major advantage of this emerging technology compared to 2D designs is the inherent reduction in wirelength [2][3].

The 3D NoC approach also offers a matchless platform to implement the globally asynchronous, locally synchronous (GALS) design paradigm [4]; this makes the clock distribution and timing closure problems more manageable and enables 3D technology to be suitable for heterogeneous integration. In addition, intrinsically, the GALS-based approach presents a suitable platform to implement Dynamic Power Management (DPM) techniques such as dynamic frequency scaling (DFS).

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 278–287, 2011.

In 3D NoCs, as the number of cores increases in each layer, the amount of communication between layers is also expected to grow, and consequently the number of interconnect TSVs will get higher. Since each TSV requires a pad for bonding to a wafer layer, the area footprint of TSVs in each layer should be considered. In this paper, we explore a mechanism to reduce TSV area footprint and optimize 3D NoC power consumption, and thus improving 3D IC cost, and routability. Specifically, we propose a novel technique to replace the pairs of unidirectional vertical channels between layers by a bidirectional channel that is dynamically self-reconfigurable to be used in either out-going or incoming direction. To compensate the bandwidth exacerbation, we exploit the low-latency nature of vertical TSVs by establishing high-speed inter-layer communication using bisynchronous FIFOs [5]. Moreover, we enhance the power characteristic of the proposed mechanism and present a forecasting-based dynamic power management scheme being able to adjust frequency of vertical channels based on the inter-layer communication traffic.

The rest of the paper is structured as follows. Section 2 describes the related work. The challenges and opportunities of vertical TSVs which highlight the motivation of utilizing the proposed 3D NoC is described in Section 3, while the architecture of a NoC router using Bidirectional Bisynchronous Vertical Channels (BBVC) is presented in detail in Section 4. Section 5 discusses the details of our proposed forecasting-based DFS technique. Section 6 shows the experimental results. Finally, Section 7 concludes this paper.

## 2 Related Work

One major advantage of the 3D IC paradigm is that it allows for the integration of dissimilar technologies with different speeds e.g., memory, analog, MEMS, and so forth, in different layers of a single die [6]. GALS approaches are well suited to tackle this speed difference. These systems [4] attach together a number of synchronous building blocks, and provide asynchronous facilities for the inter-block communication. In [7], a scheme to handle mesochronous communication in 3D NoCs is proposed but it devotes flexibility to inter-layer communication only in terms of clock phases differences and does not support different clock speeds.

In this work, we present an efficient technique based on the GALS concept which can remarkably improve the area, and power consumption characteristics of 3D NoCs. To the best of our knowledge, there is only one study targeting to minimize the number of TSVs in 3D NoCs [8] in which serialization of vertical TSV interconnects is proposed as a way to reduce the interconnect TSV footprint. Although, this can lead to a better thermal TSV distribution resulting in more efficient core layout across multiple layers due to the reduced routing complexity, it degrades the performance due to serialization overhead and low bandwidth utilization. It should be noted that serialization solely cannot offer any improvements in terms of dynamic power consumption. As will be shown later, our technique can achieve their improvements with less or in some cases without any performance degradation by utilizing high speed bidirectional bisynchronous vertical channels. In addition to this important enhancement, the proposed forecasting-based dynamic power management technique can considerably optimize the power consumption.

In [9], Lan et al. propose a bidirectional channel based NoC architecture to enhance the performance of 2D on-chip communication. This architecture does not support GALS-based communication and addresses only 2D communication. In addition, in order to increase the performance, they exploit ten bi-directional channels instead of five input and five output channels for a $5\times5$ mesh-based switch. To support this adaptivity, they extend the crossbar to a $10\times10$ one, add complexity to the arbitration unit, and embed a central channel control module to each router. In contrast, we avoid modifying the main router structure because the BBVC is only responsible for inter-layer communication and there is only one BBVC between two adjacent routers located in different layers. In other words, the proposed inter-layer communication scheme is independent of the intra-layer topology. It is also noteworthy that the foremost aim of this work is to reduce the TSV area footprint.

## 3 Opportunities and Challenges of Vertical TSV

TSVs in die stacking and wafer stacking technologies are one of the most important design components. To be useful for a NoC infrastructure, a vertical wire should not be used in isolation; instead, to simplify routing, it is better to create groups or buses of such wires. For a fixed chip area, as the number of stacked-layers increases, the power density within the volume could also increase, thus raising the temperature of 3D ICs. The idea is to use TSVs in a non-electrical capacity to conduct heat and alleviate thermal hot spots in 3D ICs [10], however it is noteworthy to mention that they take up more area than normal TSVs.

The major advantage of 3D ICs is the considerable reduction in the length and number of global interconnects, resulting in an increase in the performance and decrease in the power consumption and area of wire limited circuits. For the vertical interconnects, repeaters are not necessary due to the short length. Additionally, the length of the vertical communication channel for the 3D NoC is defined as the length of a TSV connecting two routers on adjacent physical planes. In contrast, horizontal interconnects need a number of repeaters and they are much longer than their vertical counterparts (assumed to be equal to square root of connected processing element area). For instance, in [2], the reported wire length for inter-layer and horizontal interconnects are $20\mu$m and 2.5mm, respectively, and consequently their delays respectively are 16 ps and 219 ps.

## 4 BBVC-Based 3D NoC Architecture

Generally the bandwidth utilization of vertical interconnects in a conventional NoC architecture is low [9], and it can be speculated that many vertical channels are idle during each cycle because of the fixed channel directions. Regarding this fact and the aforementioned pros and cons of 3D NoC design, to reduce the number of interconnect TSVs, an architecture using high-speed BBVCs is proposed. Fig. 1 shows the schematic representation of such a system. The main idea of the proposed 3D NoC system is to

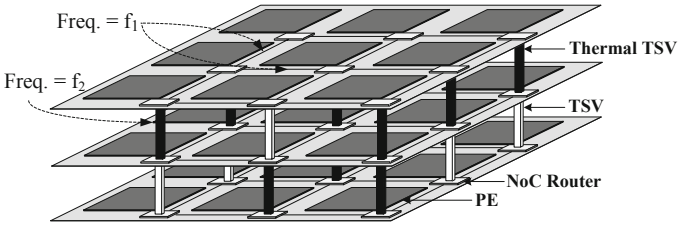



**Fig. 1.** Example of a 3D NoC with three $3 \times 3$ layers

exploit a bidirectional channel for inter-layer communication operating at a higher frequency compared to intra-layer communication ($f_2 > f_1$) and being capable of dynamically changing the channel direction between routers in neighboring layers based on the real time bandwidth requirement.

### 4.1 Router Architecture

To implement a 3D NoC architecture using BBVCs, we modify the configuration of the Up and Down input/output ports and replace their two unidirectional channels by a BBVC, as can be seen from Fig. 2. We avoid changing the structure and operating frequency of the routing modules, the arbiter, and the crossbar of a conventional router. As shown in the figure, in order to dynamically adjust the direction of vertical bidirectional channels at a run time, we add a control module to both Up and Down ports to arbitrate the authority of the channel direction. These two bidirectional channels which are composed of in-out type ports are the main difference from the typical router design in which a unidirectional channel employs a hardwired input or output port. The control module of the Up (Down) port communicates with its counterpart in the Down (Up) port of the neighbor router using a token-based communication scheme via *put*/*get-token* signals.

In order to support different clock frequencies for inter-layer communication, we replace the synchronous input FIFOs of the Up and Down ports by bisynchronous FIFOs. In addition, a bisynchronous FIFO is required for each vertical output port to enable a different (higher) clock frequency for inter-layer communication. If needed, this also offers to have 3D NoC supporting different clock frequency for each layer.

## 5 Forecasting-Based Dynamic Frequency Scaling

As mentioned before, the BBVCs can operate at a higher frequency compared to the horizontal channels. Based on our simulation results, when a BBVC has the maximum traffic load and operates with two-times faster clock frequency; it can offer almost the same performance as two unidirectional channels. If we always set the inter-layer frequency level based on the worst case scenario (heavy-load network), there will be no improvement in terms of power consumption. To this end, we propose a dynamic power management technique to dynamically determine the vertical channel frequency level. The DPM technique is based on a distributed forecasting-based policy, where

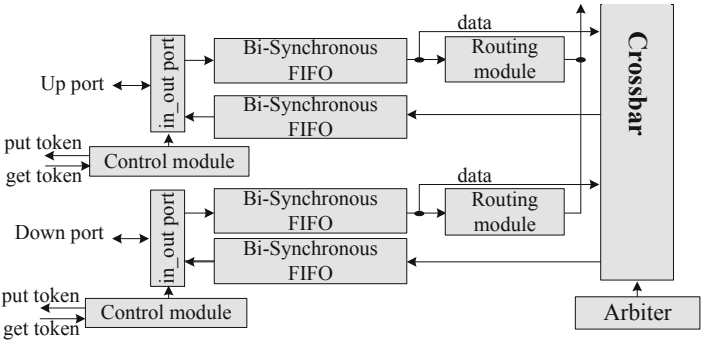



**Fig. 2.** Up and Down ports of the proposed router architecture supporting bidirectional bisynchronous vertical channels

each BBVC predicts its future communication workload and the required operating frequency based on the analysis of the prior traffic, thanks to the bi-synchronous FIFOs for enabling the DFS technique.

In order to assess the communication traffic characteristics of a channel several potential network parameters can be utilized. In this case, we employ the link utilization parameter as the network load indicator because of its simplicity and efficiency. The Link Utilization (*LU*) parameter is defined as [11]

$$LU = \frac{\sum_{t=1}^{H} A(t)}{H}, \quad 0 \leq LU \leq 1 \tag{1}$$

where *A(t)* is equal to 1, if the traffic passes through the link *i* in the cycle *t* and 0 otherwise, and *H* is the window size. The link utilization is a direct measure of the traffic workload.

In the proposed technique, the DPM unit uses the *LU* parameter for measuring the past communication traffic. Based on this analysis, the *LU* of the next period and the required frequency level are determined. To make the forecasting formula reliable, we use an exponential smoothing function. This is a simple and popular forecasting formula, which is commonly used in programming and inventory control science and is defined as [12][13]

$$LU_{predict} = LU_{past} + \alpha \times (LU_{actual} - LU_{past}) \tag{2}$$

where $\alpha$ is the forecasting weight, $LU_{predict}$ is the predicted link utilization, $LU_{actual}$ is the actual link utilization in the current history period, and $LU_{past}$ is the predicted link utilization in the previous history period. The accuracy of the prediction is a function of $\alpha$, and hence, its value should be selected such that the prediction error is minimized (see Section 6).

The network traffic profile has short-term and long-term fluctuations. The proposed technique in this work filters out the short-term traffic fluctuations, adapting the

*Algorithm 1 Forecasting-based DFS*

$LU_{predict} = LU_{past} + \alpha \times (LU_{actual} - LU_{past})$
**if** ($LU_{predict} < Threshold_1$) **then**
    *Freq_Level = 1*
    *New_BBVC_Freq = Frequency_table[Freq_Level]*
    **if** *Freq_Level = intra_layer_Freq_Level* **then**
        *Syn/Bi-Syn_Mode = Syn*
    **else**
        *Syn/Bi-Syn_Mode = Bi-Syn*
    **end if**
**else if** ($LU_{predict} > Threshold_1$ **and** $LU_{predict} < Threshold_2$) **then**
    *Freq_Level = 2*
    *New_BBVC_Freq = Frequency_table[Freq_Level]*
    **if** *Freq_Level = intra_layer_Freq_Level* **then**
        *Syn/Bi-Syn_Mode = Syn*
    **else**
        *Syn/Bi-Syn_Mode = Bi-Syn*
    **end if**
                **…**
                **…**
**else if** ($LU_{predict} > Threshold_{n-1}$) **then**
    *Freq_Level = n*
    *New_BBVC_Freq = Frequency_table[Freq_Level]*
    **if** *Freq_Level = intra_layer_Freq_Level* **then**
        *Syn/Bi-Syn_Mode = Syn*
    **else**
        *Syn/Bi-Syn_Mode = Bi-Syn*
    **end if**
**end if**
$CT_{past} = CT_{predict}$

frequency level of a BBVC based on the long-term traffic profiles. Based on the prediction, the controller decides to increase, decrease, or keep the same frequency level. The pseudo-code for our proposed forecasting-based DFS policy for the case of $n$ frequency levels is shown in Algorithm 1.

In this algorithm, when the frequency level is equal to the intra-layer frequency level, it means that the inter-layer and intra-layer communications are using the same clock frequencies. In other words, the bi-synchronous FIFOs are not needed. To avoid the power/performance overhead of the bi-synchronous FIFOs in such cases, we exploit Reconfigurable Synchronous/Bi-Synchronous (RSBS) FIFOs presented in [14] instead of simple bi-synchronous FIFOs. When the read and write clock signals have the same frequency level; the controller changes the RSBS FIFO mode to synchronous, otherwise asynchronous.

To reduce the hardware overhead, we set $\alpha$ to $3/4$, which is very close to the optimal values (see Section 6) of this parameter for the traffic profiles used in this work. We implemented the division and multiplication operations by right and left shifts, respectively. It should be noted that the number of DFS units is equal to the number of vertical links (i.e., if there are $k$ layers, and each of which contains $m \times p$ nodes, $(k\text{-}1) \times m \times p$ DFS units are needed). Finally, note that we simplify the division and multiplication operations by setting $H$ value as power-of-two numbers. Otherwise, the overhead of the DFS unit will become slightly larger.

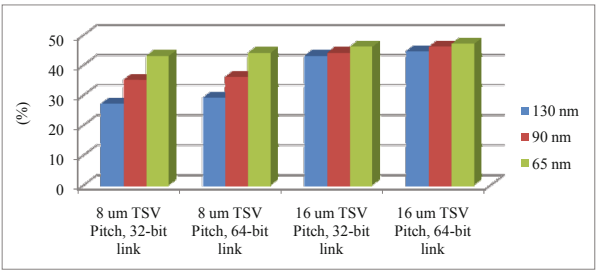


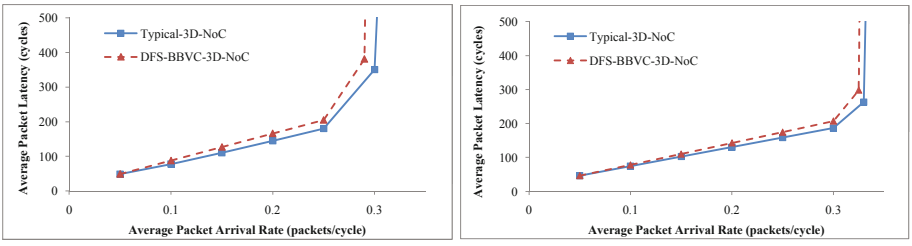

**Fig. 3.** Percentage of area saving with BBVC-based communication scheme for different TSV pitches and link sizes for 65nm, 90nm, and 130nm technologies

## 6 Experimental Results

To assess the efficiency of the proposed BBVC-based 3D NoC, several simulations were conducted. We have simulated the Typical-3D-NoC and DFS-BBVC-3D-NoC to characterize their area, latency and power consumption. The area of the BBVC-based communication scheme was computed once synthesized on CMOS 130, 90, and 65nm standard cells by Synopsys Design Compiler. Different bandwidths are used to illustrate the scalability of the architecture. In Fig. 3, the first and second groups of bars show the area savings when using the proposed communication scheme for a $5\mu m \times 5\mu m$ TSV pads and an $8\mu m$ pitch for 32-bit and 64-bit links, respectively. It can be seen that the proposed communication scheme can significantly reduce the area footprint of TSVs in 3D ICs. As technology scales, the savings in area increase as a result of lower area footprint of the control module logic.

As discussed before, it is a common practice to increase TSV pad area to enhance fault tolerance. To this end, the area saving was calculated for the $10\mu m \times 10\mu m$ TSV pad dimensions and $16\mu m$ pitch, while TSV dimensions were kept unchanged. The third and fourth groups of bars in Fig. 3 illustrate the area savings for 32-bit and 64-bit links in the case when the proposed BBVC-based communication scheme is applied. As shown in the figure, compared to the previous non fault-tolerant case, the design is more area efficient. For instance, the area savings are greater than 47% for an implementation based on a 65nm library and 64-bit links.

To demonstrate the amount of power savings and the negligible performance overhead of the proposed inter-layer communication scheme, a cycle-accurate NoC simulation environment was implemented along with two different inter-layer channel designs (Typical and DFS-BBVC-based) in HDL. In synthetic traffic analysis, the 3D NoC of our simulation environment consists of $3 \times 3 \times 3$ nodes connected as a symmetric 3D mesh. The performance of the network was evaluated using latency curves as a function of the packet injection rate (i.e., the number of packets injected into the network per cycle). For each simulation, the packet latencies were averaged over 500,000 packets. It was assumed that the buffer size of each synchronous FIFO was eight fits, and the data width was set to 64 bits. To support the bisynchronous inter-layer communication, 6-stage modular synchronizing FIFOs presented in [5] were exploited. The FIFO had a




(a) Under uniform traffic profile (b) Under NED traffic profile

**Fig. 4.** Latency versus average packet arrival rate results

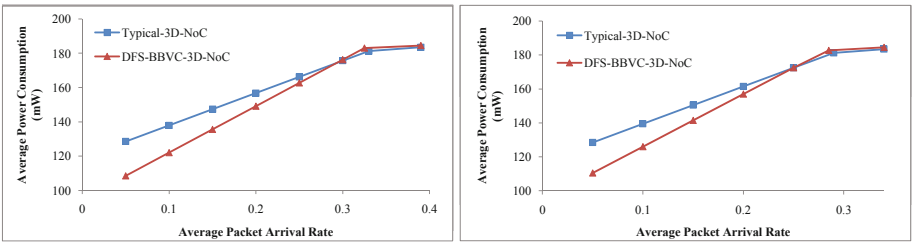
maximum read and write clock speed of 2.33GHz at full throughput. A 500 MHz clock frequency was applied to the typical NoC and for the intra-layer communication of the DFS-BBVC-Based NoC, resulting in a maximum transmission rate per link of 8 Gbps. For the inter-layer communication, following frequency levels were available to be assigned based on the DFS decision: 250MHz, 500MHz, 750MHz, and 1 GHz. Note that the clock cycle used to measure average packet latency, is based on the fixed intra-layer communication frequency.

To see the impact of $\alpha$ on the prediction accuracy, we have used the sum square error (*SSE*) [15]

$$SSE = \sum_{i=1}^{k} \sum_{j=1}^{H} (y_{ij} - \widehat{y}_{ij})^2 \tag{3}$$

where $y_{ij}$ is the actual required frequency level, $\widehat{y}_{ij}$ is the predicted required frequency level, and *k* is the total number of the prediction windows that the simulation has been run. To perform the simulations, we used an XYZ wormhole routing algorithm under uniform and Negative Exponential Distribution (NED) [16] traffic patterns. For the uniform traffic, the optimum forecasting weights for *2FreqLevels*, *4FreqLevels*, and *8FreqLevels* were equal to 0.86, 0.76 and 0.82, respectively while for the NED traffic, the optimum forecasting weights for *2FreqLevels*, *4FreqLevels* and *8FreqLevels* were equal to 0.85, 0.77 and 0.82, respectively. The error, however, is not much higher than the minimum value if we select $\alpha$ as 0.75 for all the cases. This can simplify the implementation of the DFS unit. Fig. 4(a) and 4(b) show that our DFS-BBVC-3D-NoC with half of vertical channels has still negligible performance overheads when compared to the Typical-3D-NoC under both uniform and NED traffic patterns. The overhead originates mainly from the prediction error. It can be seen from Fig. 4 that as the injection rate increases, the performance overhead slightly grows.

To estimate the power consumption, we adapted the high level NoC power simulator presented in [17] to support the 3D NoC architectures. The average power consumption of the routers (getting average of each router and its connected links) which are based on 35nm standard CMOS technology are presented in Fig. 5. As the results reveal, the average power consumption of the routers using DFS-based BBVCs are considerably lower than those of the corresponding conventional routers at low traffic loads. The reason is that when the forecasting-based DFS technique is used, vertical links and the connected buffers operate at a lower frequency level, leading to some power saving.




(a) Under uniform traffic profile (b) Under NED traffic profile

**Fig. 5.** Average power consumption versus average packet arrival rate results

As the traffic load increases the difference between the average power consumptions of the routers decreases till they eventually become almost the same at the congestion injection rate.

The area of the main components was computed once synthesized on CMOS 65nm *LPLVT STMicroelectronics* standard cells using Synopsys Design Compiler. Different subcomponents were also synthesized to illustrate the area overhead of the controllers and the extra hardware. The layout area of the typical 7-port NoC router, the proposed router, and the utilized controllers are listed in Table 1. The figures given in the table reveal, the area overheads of the proposed forecasting-based DFS unit and BBVC control module are negligible.

**Table 1.** Hardware implementation details

| **Component** | **Area**($\mu m^2$) |
|---|---|
| Typical-3D-NoC 7-Port Router | 43506 |
| DFS-BBVC-3D-NoC 7-Port Router | 46722 |
| BBVC Control Module | 304 |
| Forecasting-Based DFS Unit | 1392 |

## 7 Conclusion

In this paper, a low-cost and power-aware 3D NoC architecture based on bidirectional bisynchronous vertical channels is proposed as a solution to reduce the number of TSVs and mitigate high power densities of 3D NoCs. Dynamically self-configurable vertical channels, which can transmit flits in either direction, enable a system to benefit from a high-speed bidirectional channel instead of a pair of unidirectional channels for inter-layer communication. In addition, we introduce a technique to manage power consumption of the inter-layer communication using a forecasting-based DFS technique. The approach forecasts the inter-layer communication traffic and adjusts frequency level of the respective vertical channel accordingly. Our Simulation results show that the proposed architecture can remarkably reduce interconnect TSV area footprint and NoC power consumption.

**Acknowledgment.** The authors wish to acknowledge the financial support by the Academy of Finland and Nokia Foundation during the course of this project.

## References


1. Jantsch, A., Tenhunen, H. (eds.): Networks on Chip. Kluwer Academic Publishers, Dordrecht (2003)
2. Feero, B.S., Pande, P.: Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation. IEEE Transactions on Computers 58(1), 32–45 (2009)
3. Rahmani, A.-M., et al.: Congestion Aware, Fault Tolerant, and Thermally Efficient Inter-Layer Communication Scheme for Hybrid NoC-Bus 3D Architectures. In: Proc. of the NOCS 2011, pp. 65–72 (2011)
4. Muttersbach, J., et al.: Practical design of globally-asynchronous locally-synchronous systems. In: Proc. of the ASYNC 2000, pp. 52–59 (2000)
5. Ono, T., Greenstreet, M.: A modular synchronizing FIFO for NoCs. In: Proc. of the NOCS 2009, pp. 224–233 (2009)
6. Lu, Y.-C.: 3D technology based circuit and architecture design. In: Proc. of the ICCCAS 2009, pp. 1124–1128 (2009)
7. Loi, I., et al.: Developing Mesochronous Synchronizers to Enable 3D NoCs. In: Proc. of the DATE 2008, pp. 1414–1419 (2008)
8. Pasricha, S.: Exploring serial vertical interconnects for 3D ICs. In: Proc. of the DAC 2009, pp. 581–586 (2009)
9. Lan, Y.-C., et al.: BiNoC: A bidirectional NoC architecture with dynamic self-reconfigurable channel. In: Proc. of the NOCS 2009, pp. 266–275 (2009)
10. Cong, J., Zhang, Y.: Thermal via planning for 3-D ICs. In: Proc. of the ICCAD 2005, pp. 745–752 (2005)
11. Shang, L., et al.: Dynamic voltage scaling with links for power optimization of interconnection networks. In: Proc. of the HPCA 2003, pp. 91–102 (2003)
12. Tersine, R.J.: Principles of Inventory and Material Management. Prentice Hall PTR, Englewood Cliffs (1994)
13. Rahmani, A.-M., et al.: Forecasting-Based Dynamic Virtual Channel Management for Power Reduction in Network-on-Chips. Journal of Low Power Electronics 5(3), 385–395 (2009)
14. Rahmani, A.-M., et al.: Power and performance optimization of voltage/frequency island-based networks-on-chip using reconfigurable synchronous/bi-synchronous FIFOs. In: Proc. of the CF 2010, pp. 267–276 (2010)
15. Bowker, A.H., Lieberman, G.J.: Engineering Statistics. Prentice Hall PTR, Englewood Cliffs (1972)
16. Rahmani, A.-M., et al.: NED: A Novel Synthetic Traffic Pattern for Power/Performance Analysis of Network-on-Chips Using Negative Exponential Distribution. Journal of Low Power Electronics 5, 396–405 (2009)
17. Guindani, G., et al.: NoC Power Estimation at the RTL Abstraction Level. In: Proc. of the ISVLSI 2008, pp. 475–478 (2008)

# Worst-Case Temperature Analysis for Different Resource Availabilities: A Case Study

Lars Schor, Hoeseok Yang, Iuliana Bacivarov, and Lothar Thiele

Computer Engineering and Networks Laboratory,
ETH Zurich, 8092 Zurich, Switzerland
firstname.lastname@tik.ee.ethz.ch

**Abstract.** With three-dimensional chip integration, the heat dissipation per unit area increases rapidly and may result in high on-chip temperatures. Real-time constraints cannot be guaranteed anymore as exceeding a certain threshold temperature can immediately reduce the systems reliability and performance. Dynamic thermal management methods are promising methods to prevent the system from overheating. However, when designing modern real-time systems that make use of such thermal management techniques, the designer has to be aware of their effect on the maximum possible temperature of the system. This paper proposes an analytic framework to calculate the worst-case temperature of a system with general resource availabilities. The event and resource model is based on real-time and network calculus so that the method is able to handle a broad range of uncertainties in terms of task arrivals and available computational power. In various case studies, the proposed framework is applied to an advanced multimedia system to analyze the impact of dynamic frequency scaling and thermal-aware scheduling techniques on the worst-case temperature of an embedded real-time system.

**Keywords:** Real-Time Systems, Worst-Case Peak Temperature, Thermal Analysis, Thermal Management.

## 1 Introduction

Three-dimensional stacked multi-processor systems are a promising approach to keep pace with the ever-increasing demand of computational performance by reducing the interconnect delay and effective chip footprint. However, the increase in performance imposes a major rise in heat dissipation per unit area, which in turn threats the reliability and performance of a system [7].

As modern embedded processors support several operation frequencies and deep power down (DPD) states for power efficient design, dynamic thermal management (DTM) based on dynamic voltage/frequency scaling is considered as a promising technique to prevent the system from overheating. Consequently, thermal management has been a hot topic in research in recent years including thermal-constraint scheduling to maximize the performance [2, 3, 9] or peak temperature reduction to meet performance constraints [1]. Specifically, in [3],

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 288–297, 2011.

the workload is maximized under thermal constraints for discrete DVS speeds and in [9], a convex optimization technique for temperature-aware frequency assignment is proposed. Bansal et al. [1] explore peak temperature reduction by adopting the on-line algorithm for energy efficiency proposed by Yao et al. [16].

Unfortunately, none of the previous work has studied the effect of general resource availabilities on the worst-case temperature of a system under all possible scenarios of task executions, even thought this knowledge is desirable for any modern real-time system. In [10], a system-level analytic technique has been proposed, which calculates the worst-case temperature of a real-time system. However, as the work assumes work-conserving systems with full service availability, the method is not able to analyze DTM techniques. Recently, the method has already been applied to DTM techniques, in particular leaky bucket shapers [5].

Consequently, we extend the base technique proposed in [10] to general resource availability. This enables us to study the effect of dynamic frequency scaling (DFS) and TDMA scheduling on the worst-case temperature of the system. Our framework considers general event arrivals modeled by arrival curves from real-time and network calculus [6, 12]. An arrival curve constraints an upper bound on the workload that might arrive to the system in any time interval. Similarly, the computational power provided by a processor in any time interval is also constrained by a service curve set. The contributions of this paper can be summarized as follows:

- The considered system is formally described by corresponding thermal and computational models.
- The technique proposed in [10] is extended to general resource availabilities based on service curves from real-time and network calculus [6, 12].
- We implemented the proposed technique in MPA [14] and used it in various experiments to study the effect of DFS and TDMA on the worst-case temperature of a real-time system.

## 2 System Model

This section introduces the models to analyze a single computing component for its worst-case temperature. The model proposed in [10] is extended to consider general resource availabilities, such as frequency modulation.

### 2.1 Computational Model

The computational model of a component follows the idea of network and real-time calculus [6, 12]. Consequently, we suppose that in time interval $[s, t)$, events with a total workload of $R(s, t)$ time units arrive at processing component. The arrival curve $\alpha$ upper bounds all possible cumulative workloads:

$$R(s,t) \leq \alpha(t-s) \quad \forall s < t \tag{1}$$

with $\alpha(0) = 0$. Suppose that several independent workload functions $R_k$ are individually bounded by arrival curves $\alpha_k$. In case a component concurrently processes multiple workload functions, the accumulated workload can be bounded as $R(s,t) \leq \sum_{\forall k} \alpha_k(t-s) = \alpha(t-s)$.

A processing component is given $W(s,t)$ time units computing resources in time interval $[s,t)$. Events that have not yet been completed are queued in the component's ready queue while waiting for further resources. The resource availability $W$ is upper and lower bounded using a service curve

$$\beta^l(t-s) \leq W(s,t) \leq \beta^u(t-s) \quad \forall s < t \tag{2}$$

with $\beta^l(0) = \beta^u(0) = 0$. The accumulated computing time $Q(s,t)$ describes the amount of time units that a component is spending to process an incoming workload of $R(s,t)$ time units. The accumulated computing time $Q(s,t)$ in time interval $[s,t)$ is

$$Q(s,t) = \inf_{s \leq u \leq t} \{W(s,t) - W(s,u) + R(s,u)\} \tag{3}$$

provided that there is no buffered workload in the ready queue at time $s$ [12]. The upper bound on the accumulated computing time can be determined according to [13] as

$$Q(t-\Delta,t) \leq \gamma(\Delta) = \min\{(\alpha \otimes \beta^u) \oslash \beta^l, \beta^u\}, \tag{4}$$

where $(f \otimes g)(\Delta) = \inf_{0 \leq \lambda \leq \Delta}\{f(\Delta-\lambda)+g(\lambda)\}$ and $(f \oslash g)(\Delta) = \sup_{\lambda \geq 0}\{f(\Delta+\lambda)-g(\lambda)\}$.

Because of (3) and (4), the accumulated computing time $Q(s,t)$ for any fixed $s$ and its upper bound $\gamma(t-s)$ are monotonically increasing. Consequently, the operating mode of a component can be expressed by the mode function

$$S(t) = \frac{\mathrm{d}Q(s,t)}{\mathrm{d}t} \in [0,1] \tag{5}$$

for any $s < t$. $S(t) = 1$ and $S(t) = 0$ denote that the processing component is fully utilized and idle, respectively.

### 2.2 Power Model

It is well known that the dynamic power consumption $P_{\mathrm{dyn}}$ growths quadratically with the supply voltage $v$ and linearly with the operating frequency $f$:

$$P_{\mathrm{dyn}} \propto v^2 \cdot f. \tag{6}$$

For the sake of simplicity, the supply voltage is assumed to be given as fixed value, so that the model can be applied to DFS without loss of generality. In addition, we model the temperature dependency of leakage power by means of a linear approximation [8], i.e. $P_{\mathrm{leak}} = \phi \cdot T + \psi$. With fixed $v$, the operating frequency $f$ is proportional to the mode $l$ and the total power consumption can be derived as

$$P = \phi \cdot T + \rho \cdot l + \psi. \tag{7}$$

### 2.3 Thermal Model

A widely used duality to analyze the heat transfer of a modern VLSI system is to model the heat flow as current passing through a thermal resistance, to model the thermal difference as the corresponding voltage, and to model the heat capacity as an electrical capacity, see [4, 11, 15]. In particular, temperature will follow a first order linear differential equation of the form

$$C \cdot \frac{\mathrm{d}T}{\mathrm{d}t} = P - G \cdot (T - T_{\mathrm{amb}}) \tag{8}$$

where $C$, $P$, $G$ and $T_{\mathrm{amb}}$ denote the thermal capacity, the generated power, the thermal conductance and the ambient temperature, respectively. Rewriting (8) with (7) leads us to

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -g \cdot T + h, \quad \text{with } g = \frac{G - \phi}{C}, \quad h = \frac{\rho \cdot l + \psi + G \cdot T_{\mathrm{amb}}}{C} \tag{9}$$

with time-dependent temperature $T$. A closed-form solution to the above yields

$$T(t) = T^{\infty} \cdot \left(1 - e^{-g \cdot (t - t_0)}\right) + T(t_0) \cdot e^{-g \cdot (t - t_0)} \tag{10}$$

where $T^{\infty} = h/g$ is the steady-state temperature in a single mode of slope $l$.

Throughout this paper, we restrict our analysis to *proper* thermal models, in which the following reasonable conditions are satisfied:

- We have $g > 0$, i.e. $G > \phi$.
- The steady-state temperature is not smaller in a single mode of slope $l$ than in a single mode of slope $l = 0$, i.e. we have $T_l^{\infty} \geq T_{l=0}^{\infty}$ or $l \geq 0$.

Moreover, according to the thermal model, the component has the *thermal monotonicity* property.

**Lemma 1.** *(Monotonicity) Suppose we consider two equal timed sequences of operation modes in a time interval from $s$ to $t$. Then, the sequence with higher temperature at time $s$ leads to a higher component temperature at time $t$.*

*Proof.* This lemma comes from (10) by taking $h$ as a time-dependent function. □

### 2.4 Problem Definition

Now, we can formulate the worst-case peak temperature analysis problem:

> Given is the computing model in (3), the power model in (7), the distributed thermal model in (9). Then, the objective is to determine the peak temperature $T^*$ for any cumulative workload $R$ that complies with a given sub-additive arrival curve $\alpha$ and any resource availability $W$ within service curve $\beta$.

## 3 Thermal Analysis

In this section, we will construct a worst-case accumulated computing time $Q^*(0,t)$ of the processing component that leads to an upper bound $T^*$. The upper bound $T^*$ can later be obtained by simulating the system with this accumulated computing time $Q^*(0,t)$. As a first prerequisite, we will show in Lemma 2 that allowing more power later in time (closer to $\tau$) will always increase the temperature $T(\tau)$.

**Lemma 2.** *(Shifting) Given is a proper system model according to (7) and (9) as well as some time instance $\tau$. In addition, we consider two mode functions $S(t)$ and $\overline{S}(t)$ defined for $t \in [0,\tau)$ which are only different in a time interval $[\sigma, \sigma+2\delta)$ with $\sigma+2\delta < \tau$. In particular, we have $S(t) = l_1$ for all $t \in [\sigma, \sigma+\delta)$, $S(t) = l_2$ for all $t \in [\sigma+\delta, \sigma+2\delta)$. $\overline{S}(t)$ is $\overline{l_1}$ and $\overline{l_2}$ for $[\sigma, \sigma+\delta)$ and $[\sigma+\delta, \sigma+2\delta)$ respectively, where $\overline{l_1} = l_1 - \Delta$, $\overline{l_2} = l_2 + \Delta$, and $0 \leq \Delta \leq l_1$. In other words, we allow more power at the second time interval of $[\sigma, \sigma+2\delta)$ while keeping the total power consumption as the same. Then, if $\overline{T}(0) = T(0)$, we have $\overline{T}(\tau) \geq T(\tau)$, i.e. mode function $\overline{S}(\tau)$ results in a higher temperature at time $\tau$ when $\delta$ is small enough.*

*Proof.* As the mode functions satisfy $S(t) = \overline{S}(t)$ for all $t \in [0,\sigma)$ and $\overline{T}(0) = T(0)$, we clearly have $\overline{T}(\sigma) = T(\sigma)$. As $S(t) = l_1$ for $t \in [\sigma, \sigma+\delta)$ and $S(t) = l_2$ for $t \in [\sigma+\delta, \sigma+2\delta)$, we find

$$T(\sigma+2\delta) = T^{\infty}_{l=l_2} \cdot \left(1 - e^{-g\delta}\right) + T^{\infty}_{l=l_1} \cdot \left(1 - e^{-g\delta}\right) \cdot e^{-g\delta} + T(\sigma) \cdot e^{-2g\delta} \quad (11)$$

Furthermore, as $\overline{S}(t) = \overline{l}_1$ for $t \in [\sigma, \sigma+\delta)$ and $\overline{S}(t) = \overline{l}_2$ for $t \in [\sigma+\delta, \sigma+2\delta)$, we find

$$\overline{T}(\sigma+2\delta) = T^{\infty}_{l=\overline{l}_2} \cdot \left(1 - e^{-g\delta}\right) + T^{\infty}_{l=\overline{l}_1} \cdot \left(1 - e^{-g\delta}\right) \cdot e^{-g\delta} + \overline{T}(\sigma) \cdot e^{-2g\delta} \quad (12)$$

Then, for a proper thermal model, we have

$$\overline{T}(\sigma+2\delta) - T(\sigma+2\delta) = \frac{\rho}{G-\phi} \cdot \Delta \cdot \left(1 - e^{-g\delta}\right)^2 \geq 0 \quad (13)$$

where we used the fact that $\Delta \geq 0$. With the thermal monotonicity and $S(t) = \overline{S}(t)$ for all $t \in [\sigma+2\delta, \tau)$, the condition $\overline{T}(\sigma+2\delta) \geq T(\sigma+2\delta)$ also implies that $\overline{T}(\tau) \geq T(\tau)$. □

The next Lemma shows that we obtain a higher temperature at some time $\tau$ if in *any* interval ending at $\tau$ the component has larger accumulated computing time. This Lemma provides the foundation for the main theorem.

**Lemma 3.** *(Mode Functions Comparison) Given is a proper thermal model as well as some time instance $\tau$. In addition, we consider two accumulated computing time functions $Q$ and $\overline{Q}$ which satisfy*

$$\overline{Q}(\tau - \Delta, \tau) \geq Q(\tau - \Delta, \tau) \quad (14)$$

*for all $0 \leq \Delta \leq \tau$. Then, if $\overline{T}(0) = T(0)$ we have $\overline{T}(\tau) \geq T(\tau)$, i.e. mode function $\overline{S}(\tau)$ results in a higher temperature at time $\tau$.*

*Proof.* Because of space limitations, only a sketch of the proof is provided. First note that because of (5), the condition of the Lemma translates equivalently to

$$\int_{\tau-\Delta}^{\tau} \overline{S}(t)\,\mathrm{d}t \geq \int_{\tau-\Delta}^{\tau} S(t)\,\mathrm{d}t. \tag{15}$$

In other words, in case of $\overline{Q}$, the component in *any* interval ending at $\tau$ has larger accumulated computing time.

Now, we will stepwise transform $S(t)$ into $\overline{S}(t)$ and in each step, the temperature will not decrease because of Lemma 2. In order to simplify the proof technicalities, we suppose discrete time, i.e. $S(t)$ and $\overline{S}(t)$ may change values only at multiples of $\delta$. In other words, $S(t)$ and $\overline{S}(t)$ are constant for $t \in [k\delta, (k+1)\delta)$ for all $k \geq 0$. Let us define $\tau = k_{\max}\delta$. We now execute the following algorithm:

1. Determine the smallest $1 \leq k_1 \leq k_{\max}$ such that $S(\tau - k_1\delta) < \overline{S}(\tau - k_1\delta)$. If there is no such $k_1$, then $S(t) = \overline{S}(t)$ for all $0 \leq t \leq \tau$ and therefore, $\overline{T}(\tau) = T(\tau)$ and the algorithm stops.
2. Determine the smallest $k_2$ with $k_1 < k_2 \leq k_{\max}$ such that $S(\tau - k_2\delta) \neq 0$. In case such a $k_2$ does not exist, it is trivial that $\overline{T}(\tau) \geq T(\tau)$ and the algorithm stops. Otherwise,
   (a) set $\Delta = \min\{S(\tau - k_2\delta), \overline{S}(\tau - k_1\delta) - S(\tau - k_1\delta)\}$,
   (b) add $\Delta$ to $S(t)$ for $t \in [\tau - k_1\delta, \tau - (k_1 - 1)\delta)$,
   (c) and subtract $\Delta$ from $S(t)$ for $t \in [\tau - k_2\delta, \tau - (k_2 - 1)\delta)$.
3. If $S(\tau - k_1\delta) < \overline{S}(\tau - k_1\delta)$, continue with step 2. Otherwise, go to step 1.

Now, one can simply prove that after each iteration, $T(\tau)$ does not decrease until it reaches $\overline{T}(\tau)$ and therefore, the initial $T(\tau)$ was not larger than $\overline{T}(\tau)$. □

Based on the above Lemma we will show the main result of this section. The following Theorem provides a constructive method to determine the worst-case accumulated computing time $Q^*$ for any processing component whose computational power is bounded by a service curve set.

**Theorem 4.** *(Worst-case Computing Time) Given is a proper thermal model according to (7) and (9). The component has the computational model (3) with the operation modes and power defined in (5) and (7), respectively. Then the following holds:*

- *Suppose that the accumulated computing time function $Q^*(0, \Delta) = \gamma(\tau) - \gamma(\tau - \Delta)$ for all $0 \leq \Delta \leq \tau$ leads to temperature $T^*(\tau)$ at time $\tau$. Then $T^*(\tau)$ is an upper bound on the highest temperature $T^*(\tau) \geq T(\tau)$ for all feasible workload traces (4) that are bounded by the arrival curve $\alpha$ and service curve $\beta$ according to (2) and (3) respectively.*
- *If in addition $T(0) \leq T^{\infty}_{l=0}$ holds in general, where $T^{\infty}_{l}$ is a steady-state temperature of the operation mode $S(t) = l$, then for any feasible workload trace we find $T^*(\tau) \geq T(t)$ for all $0 \leq t \leq \tau$.*

*Proof.* We apply Theorem 3 to the proof of Theorem 4 of [10]. □

**Table 1.** Parameters of the video conferencing application

| | Video | Audio | Network |
|---|---|---|---|
| period | 50 ms | 30 ms | 30 ms |
| jitter | [20, 90]ms | 10 ms | 10 ms |
| min. interarrival | 1 ms | 1 ms | 1 ms |
| execution demand | 6 ms | 3 ms | 2 ms |
| deadline | 50 ms | 30 ms | 30 ms |

**Table 2.** Thermal and power parameters of the considered embedded system architecture

| $G$ | $C$ | $\phi$ | $\rho$ | $\psi$ |
|---|---|---|---|---|
| $0.3\,\frac{\mathrm{W}}{\mathrm{K}}$ | $0.03\,\frac{\mathrm{J}}{\mathrm{K}}$ | $0.1\,\frac{\mathrm{W}}{\mathrm{K}}$ | 14.0 W | −25.0 W |

Note that the worst-case scenario for the temperature is often completely different from the conventional critical instance scenario that is used in real-time analysis in order to determine the worst-case timing behavior. For example, for periodic tasks with jitter, the worst-case peak temperature scenario is to first warm up the system with periodic arrivals and then heat up the system with burst arrivals and jitters.

## 4 Experimental Analysis

In this section, we analyze the effect of a reduced resource availability on the worst-case temperature and provide hints on how to design a system which is schedulable and meets temperature constraints at the same time.

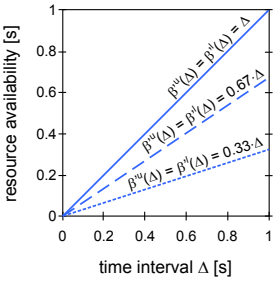
### 4.1 Experimental Setup

Two different example applications executing on a single-core embedded processor are considered in this section. The first example consists of a single process with a period of 200 ms and a execution demand of 50 ms. The second example is a multi-processing video-conferencing application, which includes a video code, an audio code and a network process for communication management. For illustration purpose, we use a period-jitter-delay model[1], with parameters summarized in Table 1. The system parameters are borrowed from [10] as summarized in Table 2 and in all our experiments, we start simulation with the initial temperature $T(0) = T^{\infty}_{l=0} = 325.00\,\mathrm{K}$, calculated from the parameters given in Table 2. The observation time $\tau$ is set to 1.0 s in all experiments.

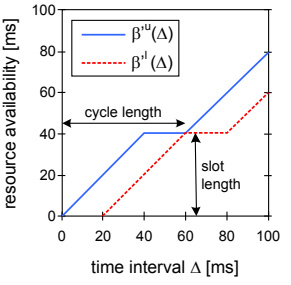
### 4.2 Dynamic Frequency Scaling

In the first experiment, we study the effect of frequency scaling on the peak temperature of the system. Figure 1(a) shows several services curves for different

[1] See [13] for the detail of all the parameters.

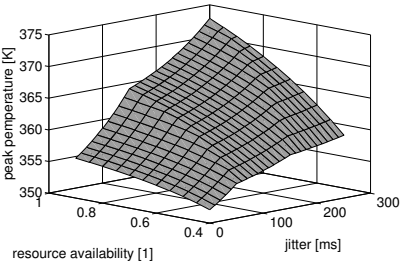



(a) Frequency modulated processor.

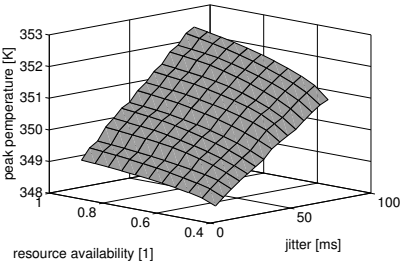



(b) TDMA.

**Fig. 1.** Service curves for various resource availabilities




(a) Single process system.

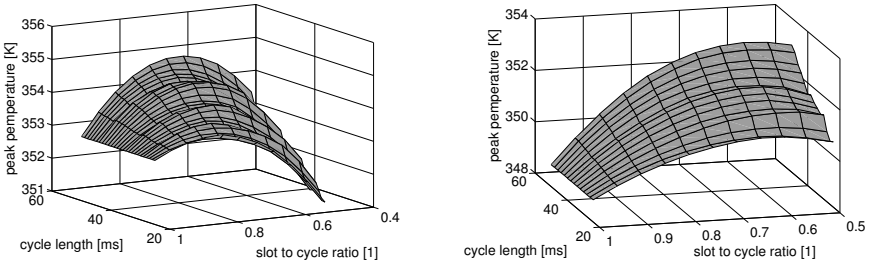



(b) Video-conferencing system.

**Fig. 2.** Worst-case temperature as a function of both resource availability and jitter

operation frequencies. The full service curve ($\beta(\Delta) = \Delta$) corresponds to full operation frequency while the two partially utilized service curves ($\beta(\Delta) = 0.67\Delta$ and $\beta(\Delta) = 0.33\Delta$) correspond to operation frequencies of 67 %, and 33 %, respectively.

Figures 2(a) and 2(b) outline the worst-case temperature as a function of resource availability and jitter for the single task application and the video conference application, respectively. For the single process application and a jitter of 50 ms, a reduction of the operation frequency by 50 % lowers the peak temperature by 4.23 K. Similarly, for a jitter of 300 ms, such a reduction of the operation frequency lowers the peak temperature by 14.50 K. As the power consumption and the inverse execution time of a process scale linearly with operation frequency, the steady-state temperature calculated by the average power consumption of an application is independent of the operation frequency. That is the reason why lowering the operation frequency has little influence on the peak temperature for relatively small jitters. As larger jitters cause the length of the burst to increase, the operation frequency has higher influence on the peak temperature for large jitters. During the burst, the component is continuously processing and the temperature increases towards the steady-state temperature $T_l^\infty$ of slope $l$, which in turn depends on the operation frequency.




(a) Single process system. (b) Video-conferencing system.

**Fig. 3.** Worst-case temperature as a function of cycle length and slot to cycle ratio

### 4.3 Thermal-Aware Scheduling

A widely proposed method for temperature reduction is to place idle service intervals so that only a certain amount of time units are assigned to an application for computation. This service availability can be described by a TDMA schema, where the processor is always available for $S$ consecutive time units during every cycle $C$. We call $C$ the cycle length and $S$ the slot length. Figure 1(b) outlines the service curves for a TDMA resource with $C = 60\,\text{ms}$ and $S = 40\,\text{ms}$.

Figures 3(a) and 3(b) outline the worst-case temperature as a function of cycle length and slot to cycle ratio. In both examples, the maximum jitter is 20 ms. Both figures show that longer idle intervals do not necessarily lead to a lower peak temperature, which is due to the buffering effect. In case the resource is not fully available, workload that arrives during idle intervals is buffered in the queue and cause the bursty shot later.

## 5 Conclusion

In this paper, we presented an analytical approach to determine the worst-case temperature for real-time systems with general resource availability. The accumulated workload arrival from all task invocations is characterized by an arrival curve, i.e. by an upper bound on the sum of task execution times arriving in any time interval. Similarly, we model the resource availability by service curves.

By applying the proposed method to various resource availability scenarios, we could show that frequency modulation is only a viable method for peak temperature reduction if the system has a large non-determinism. Analogously, placing idle service intervals does not always result in a lower peak temperature, as shown in the experiments. Consequently, our analysis methods can be used by system designer to choose proper resource parameters that reduce the peak temperature and guarantee real-time constraints.

**Acknowledgments.** This work was supported by EU FP7 projects EURETILE and PRO3D, under grant numbers 247846 and 249776.

## References


[1] Bansal, N., Pruhs, K.: Speed Scaling to Manage Temperature. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 460–471. Springer, Heidelberg (2005)

[2] Chantem, T., Dick, R.P., Hu, X.S.: Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In: Proc. Design, Automation and Test in Europe (DATE), pp. 288–293 (2008)

[3] Chantem, T., Hu, X.S., Dick, R.P.: Online Work Maximization under a Peak Temperature Constraint. In: Proc. Int'l Symposium on Low Power Electronics and Design (ISLPED), pp. 105–110 (2009)

[4] Krum, A.: Thermal Management. In: Kreith, F. (ed.) The CRC Handbook of Thermal Engineering, pp. 1–92. CRC Press, Boca Raton (2000)

[5] Kumar, P., Thiele, L.: Cool Shapers: Shaping Real-Time Tasks for Improved Thermal Guarantees. In: Proc. of Design Automation Conference, DAC (2011)

[6] Thiran, P., Le Boudec, J.-Y.: Network Calculus. LNCS, vol. 2050. Springer, Heidelberg (2001)

[7] Li, F., Nicopoulos, C., Richardson, T., Xie, Y., Narayanan, V., Kandemir, M.: Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In: Proc. Int'l Symposium on Computer Architecture (ISCA), pp. 130–141 (2006)

[8] Liu, Y., et al.: Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy. In: Proc. Design, Automation and Test in Europe, DATE (2007)

[9] Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., De Micheli, G.: Temperature-Aware Processor Frequency Assignment for MPSoCs using Convex Optimization. In: Proc. Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 111–116 (2007)

[10] Rai, D., Yang, H., Bacivarov, I., Chen, J.-J., Thiele, L.: Worst-Case Temperature Analysis for Real-Time Systems. In: Proc. Design, Automation and Test in Europe, DATE (March 2011)

[11] Skadron, K., Stan, M.R., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D.: Temperature-Aware Microarchitecture: Modeling and Implementation. ACM T. Arch. and Code Opt. 1(1), 94–125 (2004)

[12] Thiele, L., Chakraborty, S., Naedele, M.: Real-Time Calculus for Scheduling Hard Real-Time Systems. In: Proc. IEEE Int'l Symposium on Circuits and Systems (ISCAS), vol. 4, pp. 101–104 (2000)

[13] Wandeler, E., Maxiaguine, A., Thiele, L.: Performance Analysis of Greedy Shapers in Real-Time Systems. In: Proc. Design, Automation and Test in Europe (DATE), pp. 444–449 (2006)

[14] Wandeler, E., Thiele, L., Verhoef, M., Lieverse, P.: System Architecture Evaluation using Modular Performance Analysis: A Case Study. Int'l J. on Software Tools for Technology Transfer (STTT) 8, 649–667 (2006)

[15] Wang, S., Bettati, R.: Reactive Speed Control in Temperature-Constrained Real-Time Systems. Real-Time Systems 39, 73–95 (2008)

[16] Yao, F., Demers, A., Shenker, S.: A Scheduling Model for Reduced CPU Energy. In: Hájek, P., Wiedermann, J. (eds.) MFCS 1995. LNCS, vol. 969. Springer, Heidelberg (1995)

# A Framework for Architecture-Level Exploration of 3-D FPGA Platforms

Harry Sidiropoulos, Kostas Siozios, and Dimitrios Soudris

School of Electrical and Computer Engineering,
National Technical University of Athens, Athens, Greece
{harry,ksiop,dsoudris}@microlab.ntua.gr

**Abstract.** Interconnection structures in FPGAs increasingly contribute more to the delay and power consumption. Three-dimensional (3-D) chip stacking is touted as the silver bullet technology that can keep Moore's momentum and fuel the next wave of consumer electronics products. However, the benefits of such a technology have not been sufficiently explored yet. This paper introduces a novel 3-D FPGA, where logic, memory and I/O resources are assigned to different layers. Experimental results prove the efficiency of our architecture for a wide range of application domains, since we achieve average performance improvement and power saving of 30% and 10%, respectively.

## 1 Introduction

Field-Programmable Gate Arrays (FPGAs) have become the implementation medium for the vast majority of modern digital circuits. This situation makes the FPGA paradigm to grow in importance, as there is a stronger demand for faster, smaller, cheaper, and lower-energy devices.

For decades, semiconductor manufacturers have been shrinking transistor size to achieve the yearly increases in performance described by Moore's Law, which exists only because the *RC* delay was negligible, as compared to the signal propagation delay. For sub-micron technology, however, the *RC* delay becomes a dominant factor. For instance, in the 130nm CMOS technology, approximately 51% of microprocessor power is consumed by interconnect, with a projection that without changes in design philosophy, in the next 5 years up to 80% of microprocessor power will be dissipated at interconnect [11]. This has generated many discussions concerning the end of device scaling as we know it, and has hastened the search for solutions beyond the perceived limits of current 2-D architectures.

An emerging solution to this problem is the usage of 3-D integration, which replaces a large number of long interconnects (needed in 2-D structures) with shorter ones. More specifically, 3-D integration provides: (i) higher logic density in the same footprint area, (ii) shorter interconnections, (iii) reduced signal propagation delay, (iv) greater versatility and resource utilization, and (v) lower power consumption. The shift from horizontal to vertical stacking of circuits has the potential to rewrite the conventions of electronics design.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 298–307, 2011.

The benefits of using 3-D integration in logic chips are especially great for designing FPGAs since these devices already exhibit performance limitations that are tightly firmed to increased wire-length. However, in order such a technology to be widely accepted, several challenges need to be satisfied. Among others, new methodologies that support architecture-level exploration under different optimization goals are essential to design efficient 3-D architectures. Since this procedure is rather a complex task, there is also an increasing demand for developing CAD tools.

Up to now, the majority of research related to 3-D integration can be summarized in three categories, namely: (i) the manufacturing and fabrication processes that is mainly guided by industrial research, (ii) the development of CAD algorithms and tools from the academia to support the design of emerging 3-D technologies, and (iii) the case studies about the design of general-purpose 3-D architectures.

A number of 3-D FPGA architectures have been proposed until now both from academia and industry. For instance, previous efforts to model 3-D architectures affects both homogeneous (all the layers contain only logic blocks) [1][2][13], as well as heterogeneous approaches where each of the layers is specialized (i.e. memory, switches, logic, etc) [3]. Even though 3-D architectures with identical layers (first case) are suitable for designing 3-D FPGAs (due to their inherent regularity), the employment of a heterogeneous 3-D architecture potentially can provide additional performance improvement. In addition to that, there is a number of commercially available 3-D FPGAs. Typical instantiations are the 3-D FPGAs provided by Tezzaron Corp. [4], as well as the devices from Xilinx (e.g. Virtex-7 [5]).

In this paper we propose a novel architecture template for designing heterogeneous 3-D FPGAs with layers that contain blocks of different type. More specifically, our case study affects a 3-D architecture consisted of three layers, where logic, memory and I/O blocks are assigned to different layers. Furthermore, we introduce also a novel framework that software supports the architecture-level exploration task, as well as the application mapping onto these 3-D FPGAs. To the best of authors knowledge, this is the first time in literature where such a software-supported architecture level exploration of heterogeneous 3-D FPGAs is presented. Even though in this paper we study 3-D FPGAs consisted of three layers, the proposed methodology and CAD tools are also applicable to any other 3-D device (e.g. with more layers and/or that contain different type of blocks).

The rest of the paper is organized, as follows: Section 2 describes the motivation behind this work, whereas section 3 introduces the architectural template of proposed 3-D FPGA, as well as the supporting tool framework. Experimental results that prove the efficiency of proposed solution are shown in section 4. Finally, conclusions are summarized in section 5.

## 2 Motivation

The application's complexity usually introduce constraints to the architecture of target FPGA. More specifically, the performance of many application domains

(e.g. telecommunication, encryption and image-video processing) usually depends on the availability of sufficient communication bandwidth.

Previous studies [13] have already highlighted that communication between logic and memory blocks is significantly improved by the usage of 3-D integration. Hence, it is almost obvious that memory blocks have to be assigned into a different layer, as compared to logic infrastructure (e.g. slices). However, none of these works study up to now the impact of communication bottleneck between I/O blocks and the rest architecture.

Existing FPGAs incorporate a limited number of I/O blocks, which are usually spatially assigned to the periphery of the device [7]. However, such a topological arrangement imposes that application's networks that provide signal transmission among logic and I/Os exhibit increased wire-lengths, resistance and capacitance values. This problem becomes even more important for communication intensive applications, where the majority of utilized interconnection resources affect routing paths between logic (CLB) and I/O blocks.

In order to depict this limitation, Fig. 1(a) shows the successful placement for a cryptographic application (named *bigkey*) [6]. The target architecture is a Virtex-based 2-D FPGA and the P&R was performed with timing-driven VPR tool [7]. The utilized logic blocks in this figure are depicted with grey color squares, whereas the lines correspond to routing paths between them (as they retrieved after global routing).

Based on Fig. 1(a) we can conclude that the majority of connections affect routing paths between logic resources (distributed over the entire architecture) and a few I/O blocks (placed mostly for this application at the bottom-right corner of the device). These routing paths exhibit increased wire-length, and hence they result among others to considerable delay and power overheads.

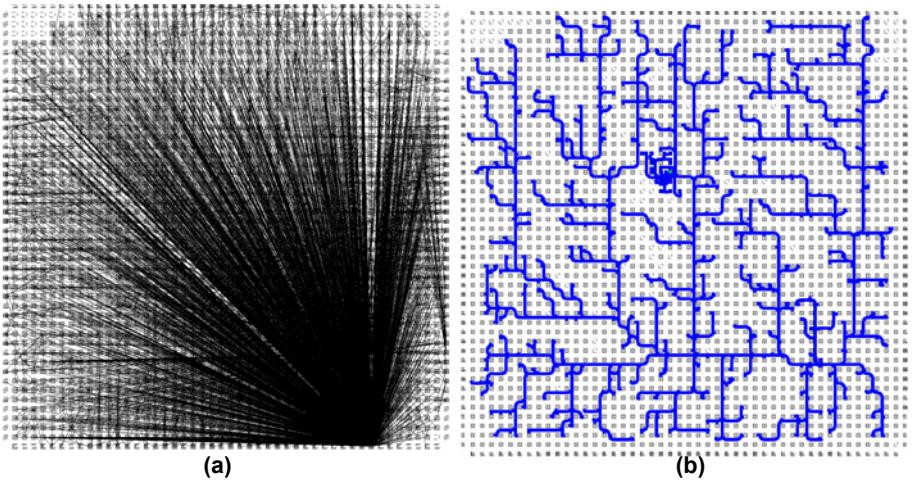



**Fig. 1.** Benchmark *bigkey* (a) after placement and global routing with VPR and (b) a random network with at least one I/O block after detailed routing

Furthermore, networks that contain at least one I/O block are more likely to be scattered across the FPGA, as compared to the rest networks. This is also depicted in Fig. 1(b), where we highlighted with blue color lines the distribution of a randomly selected routing path that contain at least one I/O block. This info was retrieved after successful detailed routing of *bigkey* benchmark with VPR tool [7].

Based on previous conclusions, we can claim that existing way for assigning I/O blocks at the periphery of FPGA device introduce considerable performance limitations, especially for communication intensive applications. Hence, alternative architecture can be inspired. For this purpose, our introduced heterogeneous 3-D FPGA assumes that I/O blocks are assigned to a dedicated layer. Note that such an enhancement is complementary to the previous one (affecting logic and memory blocks to separate layers).

## 3 Proposed 3-D FPGA Architecture and Tool Flow

This section introduces the proposed architectural template for 3-D FPGAs, as well as the software framework for enabling architecture-level exploration and application mapping onto these devices. More specifically, logic, memory and I/O resources at this architecture are assigned to different layers, as it is depicted in Fig. 2. Note that the number of total layers is tunable in order to better match application's requirements.

The communication between resources assigned to different layers is provided through TSVs, which are actually implemented either inside vertically aligned switch boxes (SBs), or connection boxes (CBs). For this purpose we assume that these blocks are assigned to the same $(x, y)$ co-ordinates, whereas the interlayer connections are depicted in Fig. 2 with dotted lines. Regarding the modeling of these TSVs, we used the electrical equivalent characteristics found in a recent technical published report for Tezzaron 3-D FPGA [12].

In order these routing resources to be aware about the third dimension, they appropriately extended similar to the way discussed in [8]. More specifically, connectivity between I/O and logic layers (Layers 1 and 2 at Fig. 2) is implemented inside 3-D CBs, whereas the 3-D SBs are employed for implementing connectivity between layers with CLBs and memories (Layers 2 and 3 at Fig. 2) [8].

Next, we introduce the design framework for supporting architecture-level exploration, as well as application mapping onto the proposed 3-D FPGAs with heterogeneous layers. The new tool framework, named 3-D NAROUTO, is public available through [10] for download and/or additional improvements. We have to mention that our solution differs compared to similar approaches (e.g. [1][2][13]), since it supports heterogeneous resources (e.g. logic, memories, DSPs, etc) which can be assigned to different layers. More details about the features of NAROUTO framework can be found in [10].

Fig. 3 gives an abstract view of the 2-D and 3-D NAROUTO frameworks. Note that during the development of 3-D NAROUTO, only the tools that are aware about the third dimension were appropriately extended, whereas the rest tools (e.g. synthesis and technology mapping) are identical between flows.

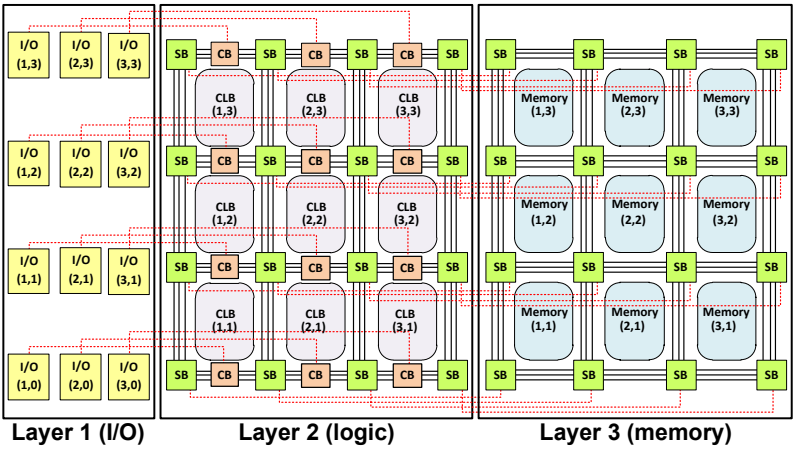



**Fig. 2.** Architectural template of our proposed 3-D FPGA

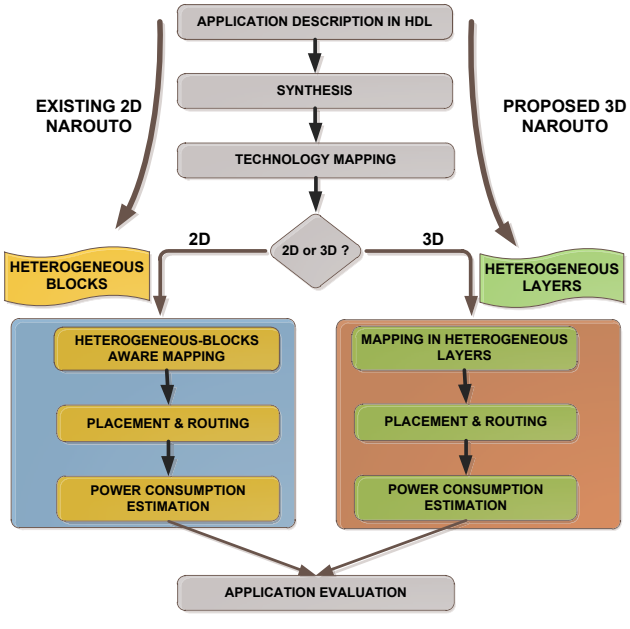



**Fig. 3.** 2-D and 3-D NAROUTO design framework

After synthesis and technology mapping we deal with application P&R. Initially, this involves to generate an instance for the target 3-D FPGA based on the architectural template discussed in Fig. 2. For simplicity reasons, we assume (without affecting the generality of introduced solution) that all the layers occupy the same area (3-D cube architecture).

The employed P&R algorithms at 3-D NAROUTO are based on VPR tool [7], but they have extensively modified in order to be aware about: (i) the additional flexibility imposed by the 3-D integration, and (ii) the heterogeneous blocks (e.g. memories) found in target architecture. Similarly, the application's routing is based upon an extended version of Pathfinder algorithm [7], which repeatedly rips-up and re-routes each net of the design, until all the congestions to be resolved. Since interlayer connections exhibit shorter wire-lengths compared to the routing tracks found in the same layer but their number is significantly limited, our routing algorithm penalties their non-efficient utilization. For this purpose we encode TSV connections as high cost edges at the platform graph. Hence, their usage is prohibited whenever they do not used at timing critical networks.

## 4 Experimental Results

This section provides a number of comparison results that prove the efficiency of our solution with the usage of MCNC [6] and Altera QUIP [14] benchmark suites. In order to software support the application implementation onto 2-D and 3-D FPGAs, we employ the VPR [7] and our proposed 3-D NAROUTO framework, respectively. Finally, for shake of completeness we have to mention that both 2-D and 3-D architectures incorporate the same number of logic (CLBs), memory and I/O blocks.

In order to study all the potential architectural solutions, in this section we provide results about three alternative scenarios which are summarized, as follows:

- Study the impact of I/O bottleneck: We evaluate the efficiency of a 3-D FPGA, where logic and I/O blocks are assigned to different layers.
- Study the impact of stacking memory onto logic: Our 3-D FPGA consists of two layers, where the first layer contains logic and I/O blocks, whereas memories are assigned to second layer.
- Study a combination of the previously mentioned architectures: This scenario involves a 3-D FPGA, where logic, memory and I/O blocks are assigned to different layers.

The first set of experimental results quantifies the efficiency of assigning I/O blocks to a different layer, as compared to logic resources. For this purpose, our 3-D architecture is composes by two layers, while the evaluation is performed by employing a well established benchmark suite [6]. Note that during this study we need benchmarks without memory blocks, since they introduce additional (artificial) overhead at routing networks.

In order to depict that our 3-D architectures is superior as compared to conventional 2-D FPGAs, Fig. 4 gives the average wire-length for networks that contain at least an I/O block (blue color bars), or not (red color bars).

Based on this figure, we can conclude that the average wire-length for networks with at least one I/O is about 92% higher, as compared to the rest

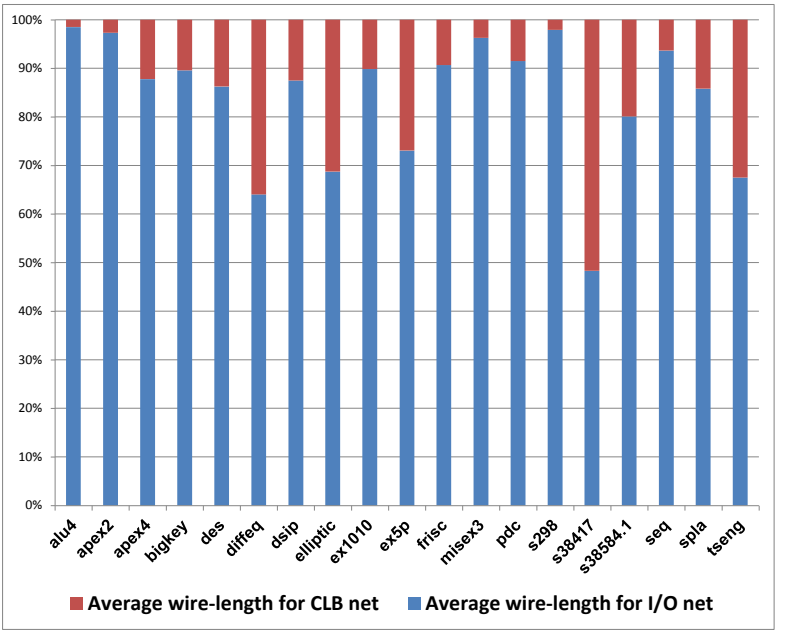



**Fig. 4.** Average wire-length for routing paths that contain (or not) I/O blocks

networks (those that include only logic resources). Furthermore, this trend is common for all the benchmarks. This proves our inspiration discussed in section 2, that networks with I/Os are significantly longer. Hence, by assigning these I/O blocks to a different layer, as our 3-D architecture proposes, we expect reasonable wire-length, delay and power improvements.

In conjunction to this conclusion, Table 1 gives the maximum operation frequency and power consumption for such a 3-D architecture, as compared to the corresponding 2-D FPGA. Note that both 2-D and 3-D architectures contain the same number of logic, I/O blocks and routing tracks per channel.

From this table it is evident that the average performance improvement achieved by the usage of proposed 3-D architecture is 22%, without any power consumption overheads. Furthermore, in conjunction to Fig. 4, there are some applications that achieve considerable delay reduction due to better manipulation of networks that contain at least one I/O block and span the entire architecture. Typical examples for this conclusion are the *bigkey*, *des*, *dsip* and *ex*1010 benchmarks, which increase maximum operation frequency by 88%, 46%, 42% and 58%, respectively. Note that additional performance improvement is feasible in case we alleviate the constraint regarding the same power budget.

After quantifying the efficiency of improving I/O bandwidth, we study the impact of assigning memory blocks to a different layer, as compared to logic resources. For this purpose, we employ a number of benchmarks from QUIP suite [14]. Table 2 evaluates the maximum operation frequency and the power

**Table 1.** Evaluation of I/O improvement between 2-D and the proposed 3-D FPGA

| Benchmark | Max. Operation Frequency (MHz) | | | Power Consumption (mW) | | |
|---|---|---|---|---|---|---|
| | 2-D FPGA | 3-D FPGA | Gain (%) | 2-D FPGA | 3-D FPGA | Gain (%) |
| alu4 | 126.58 | 128.21 | 1.28% | 47.29 | 46.12 | -2.47% |
| apex2 | 101.01 | 111.11 | 10.02% | 63.4 | 62.35 | -1.66% |
| apex4 | 120.48 | 129.87 | 7.79% | 37.78 | 36.65 | -2.99% |
| bigkey | 104.17 | 196.08 | 88.24% | 22.79 | 22.89 | 0.44% |
| des | 105.26 | 153.85 | 46.15% | 62.71 | 62.59 | -0.19% |
| diffeq | 116.28 | 140.85 | 21.13% | 35.89 | 35.89 | 0.00% |
| dsip | 156.25 | 222.22 | 42.22% | 80.96 | 80.75 | -0.26% |
| elliptic | 65.36 | 84.75 | 29.66% | 46.02 | 45.18 | -1.83% |
| ex1010 | 43.10 | 68.49 | 58.90% | 121.01 | 120.86 | -0.12% |
| ex5p | 121.95 | 131.58 | 7.89% | 44.38 | 43.89 | -1.10% |
| frisc | 55.25 | 57.80 | 4.63% | 130.61 | 129.74 | -0.67% |
| misex3 | 131.58 | 149.25 | 13.43% | 55.84 | 51.67 | -7.47% |
| pdc | 59.52 | 80.00 | 34.40% | 84.18 | 83.46 | -0.86% |
| s298 | 75.19 | 76.92 | 2.308% | 48.6 | 51.05 | 5.04% |
| s38417 | 81.30 | 83.33 | 2.50% | 76.06 | 80.19 | 5.43% |
| s38584 | 96.15 | 104.17 | 8.33% | 63.8 | 71.19 | 11.59% |
| seq | 113.64 | 129.87 | 14.29% | 58.81 | 58.54 | -0.45% |
| spla | 84.03 | 94.34 | 12.264% | 235.58 | 232.47 | -1.32% |
| tseng | 161.29 | 178.57 | 10.71% | 25.75 | 25.93 | 0.71% |
| **Average:** | **100.97** | **122.17** | **21.90%** | 70.61 | 70.60 | 0.10% |

**Table 2.** Evaluation of memory bandwidth optimization between 2-D and the proposed 3-D FPGA

| Benchmark | Max. Operation Frequency (MHz) | | | Power Consumption (mW) | | |
|---|---|---|---|---|---|---|
| | 2-D FPGA | 3-D FPGA | Gain (%) | 2-D FPGA | 3-D FPGA | Gain (%) |
| oc_aes_core_inv | 101.58 | 131.96 | 29.91% | 0.723 | 0.821 | -13.48% |
| ata_ocidec | 141.67 | 248.27 | 75.25 | 0.148 | 0.107 | 27.76% |
| os_blowfish | 93.55 | 94.17 | 0.67% | 0.250 | 0.221 | 11.75% |
| oc_hdlc | 176.40 | 207.26 | 17.50% | 0.182 | 0.167 | 8.22% |
| oc_minirisc | 131.55 | 156.25 | 18.78% | 0.066 | 0.057 | 13.63% |
| **Average:** | **128.95** | **167.58** | **28.42%** | **0.31** | **0.27** | **9.58%** |

dissipation, when these benchmarks are mapped onto 3-D FPGAs with two layers (logic and memory).

Experimental results from this study shown that our proposed 3-D stacking improves the performance of target architectures, since it results to higher bandwidth between memory and logic resources. More specifically, there is an average delay and power reduction by 29% and 10%, respectively, as compared to the corresponding 2-D FPGAs. These improvements occur mainly due to shorter wires that provide signal communication between logic (CLBs) and memory blocks. Note that usually, these connections are formed from wider routing channels in order to achieve the desired data-path implementations, and hence their impact is much more important than existing implementations of 3-D FPGAs (e.g. [1][2][13]) which assume that different layers are formed solely from CLBs.

**Table 3.** Evaluation of the proposed 3-D FPGA architecture

| Benchmark | Max. Operation Frequency (MHz) | | | Power Consumption (mW) | | |
|---|---|---|---|---|---|---|
| | 2-D FPGA | 3-D FPGA | Gain (%) | 2-D FPGA | 3-D FPGA | Gain (%) |
| oc_aes_core_inv | 101.58 | 129.40 | 27.41% | 0.723 | 0.783 | -8.30% |
| ata_ocidec | 141.67 | 243.49 | 71.87% | 0.148 | 0.137 | 6.74% |
| os_blowfish | 93.55 | 115.08 | 23.017% | 0.250 | 0.274 | -9.74% |
| oc_hdlc | 176.40 | 198.53 | 12.55% | 0.182 | 0.172 | 5.65% |
| oc_minirisc | 131.55 | 152.58 | 16.01% | 0.066 | 0.071 | -7.37% |
| **Average:** | **128.95** | **167.82** | **30.16%** | **0.27** | **0.29** | **-2.61%** |

Finally, we evaluate application mapping onto 3-D FPGAs, where both I/Os and memories are assigned to different layers, as compared to logic resources. For this purpose, Table 3 summarizes the impact of implementing these benchmarks onto an architecture with three layers. From the results summarized in Table 3 we can conclude that our architecture achieves performance improvement by 30% with a negligible penalty in power dissipation (increase by 2.6%).

## 5 Conclusion

This paper introduces a novel 3-D FPGA architecture, where I/O and memory blocks are assigned to different layers as compared to logic resources. The new architecture is software supported by a public available software framework, named 3-D NAROUTO. The evaluation procedure proves the efficiency of proposed hardware/software solution, as it achieves application implementation with significant shorter wire-lengths, which in turn leads to mentionable delay and power improvements. More specifically, average delay and power reduction by 30% and 10%, respectively, is feasible as compared to conventional 2-D FPGAs.

## References

1. Das, S., Chandrakasan, A., Reif, R.: Design tools for 3-D integrated circuits. In: Proc. of ASP-DAC, pp. 53–56 (2003)
2. Ababei, C., Mogal, H., Bazargan, K.: Three-dimensional Place and Route for FPGAs. IEEE Trans. on CAD 25(6), 1132–1140 (2006)
3. Lin, M., El-Gamal, A., Lu, Y., Wong, S.: Performance benefits of monolithically stacked 3-d fpga. IEEE Trans. on CAD 26(2), 216–229 (2007)
4. 3-D FPGA from Tezzaron, `http://www.tezzaron.com/about/PhotoAlbum/Products/3D_FPGA.html`
5. Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency. White paper: Virtex-7 FPGAs, Xilinx Inc. (October 2010)
6. Standard Cell Benchmark Circuits from the Micro-electronics Center of North Carolina (MCNC), `http://vlsicad.cs.binghamton.edu/gz/PDWorkshop91.tgz`
7. Betz, V., Rose, J., Marquardt, A.: Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, Dordrecht (1999)

8. Hsu, C., et al.: Interlaced switch boxes placement for three-dimensional FPGA architecture design. Int. Journal of Circuit Theory and Applications, doi: 10.1002/cta.739
9. Jang, D., et al.: Development and evaluation of 3-D SiP with vertically interconnected Through Silicon Vias (TSV). In: Proc. of ECTC, pp. 847–852 (2007)
10. Electronic document, `http://proteas.microlab.ntua.gr/ksiop/software.html`
11. Magen, N., et al.: Interconnect-Power Dissipation in a Microprocessor. In: ACM System-Level Interconnect Prediction Workshop (February 2004)
12. Gupta, S., et al.: Techniques for Producing 3-D ICs with High-Density Interconnnect. In: Int. Conf. VLSI Multi-Level Interconnection Conference (2004)
13. Siozios, K., Pavlidis, V., Soudris, D.: A Software-Supported Methodology for Exploring Interconnection Architectures Targeting 3-D FPGAs. In: Proceedings in Design, Automation and Testing in Europe (DATE), Nice, France, pp. 172–177 (April 2009)
14. Altera QUIP Benchmark suite, `http://www.altera.com`
15. Sidiropoulos, H., Siozios, K., Soudris, D.: A Methodology and Tool Framework for Supporting Rapid Exploration of Memory Hierarchies in FPGAS. In: International Conference on Field Programmable Logic and Applications (FPL), September 5-7 (2011)

# Variability-Speed-Consumption Trade-off in Near Threshold Operation

Mariem Slimani[1], Fernando Silveira[2], and Philippe Matherat[1]

[1] Institut TELECOM, TELECOM-ParisTech, LTCI-CNRS
46, rue Barrault, 75634 - Paris Cedex 13, France
{slimani.mariem,philippe.matherat}@telecom-paristech.fr
[2] Universidad de la República, Instituto de Ingeniería Eléctrica,
J. Herrera y Reissig 565, 11300 Montevideo, Uruguay
silveira@fing.edu.uy

**Abstract.** Sub-threshold operation is an efficient solution for ultra low power applications. However, it is very sensitive to process variability which can impact the robustness and effective performance of the circuit. On the other hand this sensitivity decreases as we move towards near-threshold operation.

In this paper, the impact of variability on sub-threshold and near-threshold circuit performance is investigated through analytical modeling and circuit simulation in a 65 nm industrial low power CMOS process. We show that variability moves the effective minimum energy point towards the near threshold region. Thus, we demonstrate that when variability is taken into account, a complete model that includes the near threshold (moderate inversion) region is necessary in order to correctly model circuit performance around the minimum energy point. Finally, we present the resulting speed-consumption trade-off in a variability-aware analysis of sub-threshold and near-threshold operation.

**Keywords:** Sub-threshold logic, Near-threshold operation, Variability, Modeling.

## 1 Introduction

Over the last decade, sub-threshold logic has been used as an ideal option to achieve Ultra Low Energy consumption for applications with low demand in speed requirements. Here, sub-threshold term refers to the weak inversion (WI) region where the minimum energy can be achieved using a supply voltage $V_{DD}$ well below the threshold voltage.

As the interest for ultra low energy has increased, research related to sub-threshold logic has attained considerable importance. Modeling and characterization of sub-threshold operation for standard CMOS cell designs have been investigated for energy and performance analysis[1][2].
However, several works have shown that variability in subthreshold logic, especially intrinsic one due to Random Dopant Fluctuations (RDF), is a critical limit

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 308–316, 2011.

to achieve robust ultra low energy devices as it cannot be compensated through adaptive body biasing (ABB)[3][4][5]. As currents in weak inversion region exponentially depends on threshold voltage ($V_t$), random $V_t$ variations significantly affect the on and off currents and gate delays and can affect the output swings and result in functional failures of some gates. Moreover, the minimum energy point can be strongly affected by variability. As reported in [6], device variability leads to 20% of minimum energy increase.

Models considering variability have been developed in previous works specifically for use in subthreshold, considering, therefore, just the weak inversion region. In this paper, we will show through 1K-point Monte Carlo Spice simulations in a Low Power (LP) technology from an industrial foundry that variability moves the effective minimum energy point towards the near-threshold region (Moderate Inversion). Thus, restrictive weak inversion models can no longer model circuit performance around the minimum energy point.
Therefore, we apply a complete and compact transistor model valid from weak to strong inversion in order to model the delay and energy performance over the weak and moderate inversion regions. This will allow us to correctly model the circuit performance in a variability aware analysis.

This paper is organized as follows. Section 2 presents the concept of subthreshold operation and investigates the impact of process variations on minimum energy point. In section 3, we propose a complete model that includes the near threshold region. We show through Monte Carlo Spice simulations that the new model is necessary to correctly model the circuit performance in a variability aware analysis.

## 2 Sub-Threshold Circuit Design

In this section, the concept of sub-threshold operation is briefly explained. Variability impact on sub-threshold circuit performance through Monte Carlo Spice simulations of an inverter chain implemented in a Low Power technology from an industrial foundry is then presented.

### 2.1 Sub-Threshold Operation

Sub-threshold operation consist in reducing the power supply voltage $V_{DD}$ below the threshold voltage $V_T$ in order to achieve minimum energy consumption. The concept is simple: as $P_{dyn}$ is proportional to the square of $V_{DD}$, a small reduction in supply voltage causes quadratic decrease in dynamic power consumption at the cost of a significant increase in delay. This results in an increase in leakage energy that leads to a minimum energy point achieved at an optimum supply voltage. In[7], authors provide an analytical solution for the optimum $V_{DD}$ that minimize the energy for a given operating frequency.

However, lowering the power supply voltage will expose the circuit to the effect of process variations which can impact the functionality of the circuit and result in a functional failure of the gate. Thus, a process corners analysis

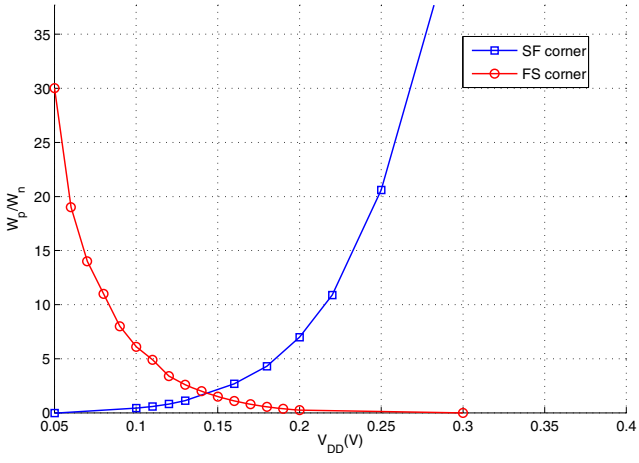



**Fig. 1.** Worst case corners analysis of the inverter for different supply voltage

which determines the minimum operation voltage and the minimum transistor sizing which ensures a good functionality of the gate is the first step to do, as described in [2]. We have applied this analysis to an inverter from a 65nm Low Power industrial library. Figure 1 shows the limits for the ratio of the pMOS width to nMOS width, which must be between the minimum set by the Fast 0-Slow 1 (FS) corner and the maximum set by the Slow 0-Fast 1 (SF) corner in order to assure an output swing of 10%-90% of the supply voltage.

We observe that this restriction determines a minimum possible operating voltage, which for this technology occurs at 143mV by sizing the PMOS width to be 1.83 the NMOS one. We see that the inverter of the standard cell library is guaranteed to operate down to 160mV. Thus, if the minimum energy point is achieved by a supply voltage between 143mV and 160mV, a modified logic cell library should be created.

### 2.2 Variability Effect on Subthreshold Circuit

Process variations is a critical limit of sub-threshold circuits. Intrinsic variations due to Random doping fluctuation (RDF) is considered to be the dominant source of variability in sub-threshold circuits [5].

To show the impact of variability on sub-threshold circuit performance, Monte Carlo Spice simulations with 1000 points have been performed for different values of $V_{DD}$. The considered benchmark circuit is a chain of inverters where the first inverters are not considered in order to correctly calculate the static current, increased due to the degeneration of stable states as mentioned in [1].

Since delay variability depends on the logic depth of the circuit, an appropriate choice of the logic depth is essential. For our case study, we have choosen a logic depth of 23 like in [8] where the circuit is a standard 8-bit ripple-carry-array (RCA) multiplier.

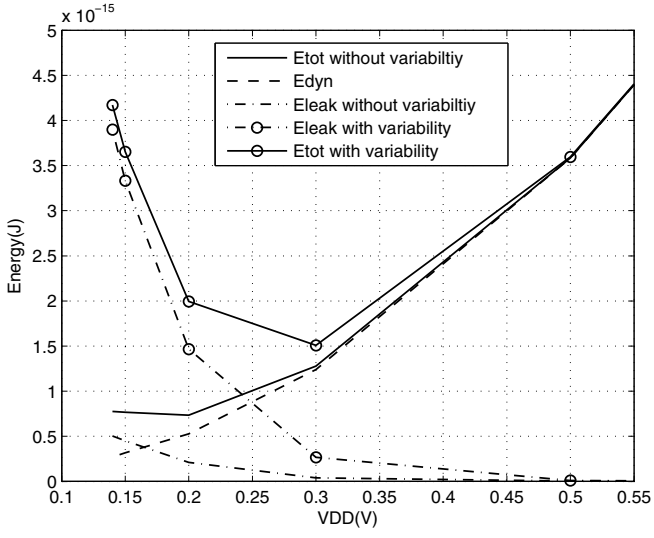



**Fig. 2.** Leakage and total energy evolution with and without variability

Figure 2 shows the evolution of total energy consumption without and with variability effect, considering typical and $3\sigma$ worst case delay, respectively.

We observe that variability leads to a considerable increase in leakage energy. This results in the increase of the minimum energy point by 50%, located now in the moderate inversion region.

## 3 Variability Aware Circuit Model

In[1],[5] and [6], models in WI region are applied to describe sub-threshold operation. However, as we have seen in section 2, variability considerably affects the minimum energy point which moves in the moderate inversion region. In[1] the impact of operation in moderate inversion applying an all region transistor model is also considered, but the impact of variability is not analyzed in this case. [9] is another prior work where authors suggest to work in the near threshold voltage region in order to recover some of delay performance at the expense of a little energy increase. An energy-delay modeling framework that extends over all inversion regions is developed in this paper. But variability is still not considered in these models.

In this section, we present a variability aware model that extends over the weak and moderate inversion region. This model is based on the EKV model expressions [1]. The main contribution here is that we consider $V_T$ and $\beta$ variations in our analysis. We show through Spectre and Matlab simulations that the WI model in no longer sufficient to model the performance of a system exposed to process variations.

### 3.1 Current and Delay Model under Variability Analysis

In weak and moderate inversion region, the drain current can be expressed as[1]:

$$I_{DS} = I_S(ln^2[1 + exp\frac{V_{GS} - V_T}{2nU_T}] - ln^2[1 + exp\frac{V_{GS} - V_T}{2nU_T} exp\frac{-V_{DS}}{2U_T}]) \quad (1)$$

Where $n$ is the subthreshold slope factor, $V_{GS}$ and $V_{DS}$ are respectively the gate to source and drain to source voltages and $V_T$ is the threshold voltage. $I_S$ is the specific current given by

$$I_S = 2n\mu C_{ox} U_T^2 W/L = 2n\beta U_T^2 \quad (2)$$

Where $\mu$ is the mobility, $C_{ox}$ is the oxide capacitance, $U_T$ is the thermal voltage and W/L denotes the channel width-length ratio of the transistor.
Equation 1 tends to the classical exponential WI model when $V_{GS} - V_T$ is negative.

The expression of delay can be derived from the current model as follows:

$$T_d = \frac{C_L V_{DD}}{I_{on}} \quad (3)$$

Where $C_L$ is the load capacitance, $V_{DD}$ is the supply voltage and $I_{on}$ is the saturated on-current.

Leakage current is also determined from the $I_{DS}$ expression when $V_{GS} = 0$. The model of Equation 1 is a long channel model that does not include effects such as mobility reduction, velocity saturation and drain induced barrier lowering (DIBL). The former two are mainly of impact in the strong inversion region and that is why, for simplicity sake, were not considered in this work. The last one (DIBL), has some impact on the performance in the WI and MI regions, as analyzed in [5]. In our case we considered the effect of DIBL when extracting the parameters for the model of Equation 1 by considering the drain current data of the 65nm Standard Threshold Voltage, Low Power (SVTLP) NMOS transistor with drain voltage equal to $V_G$. To extract model parameters, we have applied the method based on the gm/ID curve described in [10]. The following values were obtained:

- n = 1.22;
- $V_T = 0.38$(V);
- $\beta = 4.83e^{-4}(A/V^2)$.

Figure 3 shows the $I_D$ versus $V_{GS}$ for NMOS transistor determined with the weak inversion model, the complete model and spice simulations. As expected, the weak inversion model is not sufficient to model the current in near-threshold region. We observe that the complete model is not so accurate in strong inversion region due to the lack of modeling of mobility reduction and velocity saturation[10].

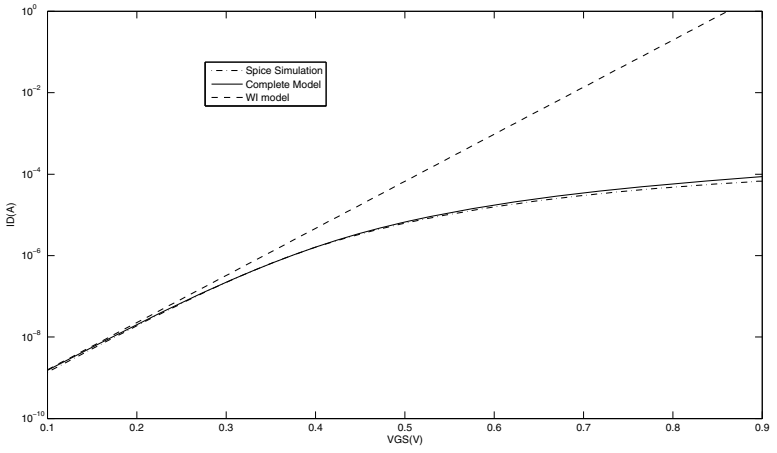



**Fig. 3.** $I_D$ versus $V_{GS}$ curves for NMOS transistor

For variability analysis, we will consider just random $V_T$ and $\beta$ variations, modeled as a normal distributions $(\mu_{V_T}, \sigma_{V_T}, \mu_\beta, \sigma_\beta)$, determined through Monte Carlo simulations or through analytical expressions given in [5]. Current model considering process variations is derived from Equation 1 by replacing $\beta$ and $V_T$ by values that follow the normal distribution described in the previous paragraph.

The evolution of typical and $3\sigma$ Worst Case(WC) delay is presented in Figure 4 (left). We observe that the complete model with and without variability consideration follow perfectly Monte-carlo Spice simulations whereas the delay of the WI model deviates from $0.3V$ of $V_{dd}$.

Figure 4 (right) plots delay variability versus the supply voltage. The simulated variability, obtained through Monte-Carlo simulations, shows that variability decreases as the supply voltage increases. We remark that delay variability, obtained with the WI model, remains constant at different supply voltage, while, the one obtained with the complete model presents the same shape as spectre simulations and has even close values.

We conclude that the WI model is a restrictive model that can not be used to model process variations and that the model developed, considering $V_T$ and $\beta$ variations, is a good model for variability analysis.

The delay model presented in Equation 3 is valid for a simple gate. For a circuit with a logic depth $L_D$, the delay will be $T_{circuit} = L_D.T_d$. If $T_d$ is a normal distribution defined by $(\mu_{delay}, \sigma_{delay})$, $T_{circuit}$ will be a normal distribution too defined by $(L_D.\mu_{delay}, \sqrt{(L_D)}.\sigma_{delay})$. Thus, the delay variability of the circuit can be obtained as follow:

$$var_{circuit-delay} = (1/\sqrt{(L_D)}).var_{gate-delay} \tag{4}$$

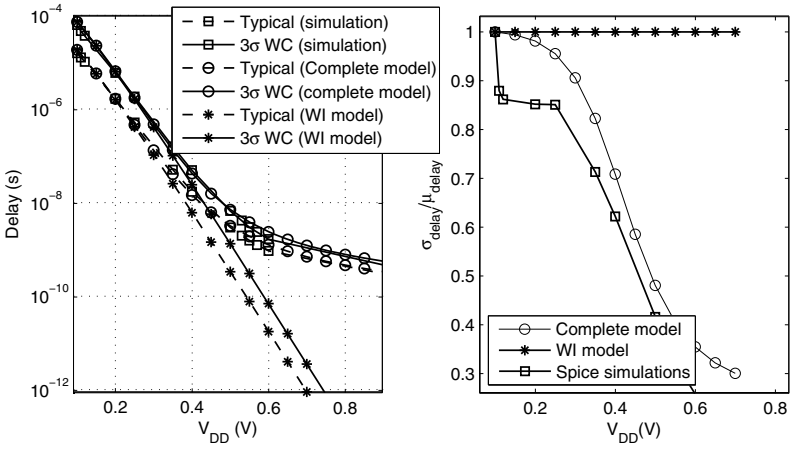



**Fig. 4.** Evolution of typical and WC delay (left), and normalized delay variability (right) for different $V_{DD}$

Table 1 lists delay variability for different Logic depth at $V_{DD} = 0.2V$.

**Table 1.** Delay variability for different logic depth at $V_{dd}$=0.2 V

| $L_D$ | Delay variability ($\sigma_{delay}/\mu_{delay}$) | |
|---|---|---|
| | **Spice simulation** | **Analytical model** |
| 1 | 0.86 | 0.99 |
| 7 | 0.28 | 0.36 |
| 15 | 0.25 | 0.26 |
| 23 | 0.205 | 0.209 |

As expected, the delay variability of a circuit decreases as its logic depth increases, and the decrease follows perfectly $1/\sqrt{(L_D)}$ law.

### 3.2 Energy Model under Variability Analysis

The total energy consumed by the circuit is the sum of the dynamic energy $E_{dyn}$, required to charge and discharge parasitic capacitances during logic transitions, and static energy $E_{stat}$ due to leakage currents $I_{leak}$. This can be summerized in the following expression :

$$E_{tot} = \alpha C_L V_{DD}^2 + V_{DD} I_{leak} T_{circuit} \tag{5}$$

Where $\alpha$ is the switching activity of the circuit and $C_L$ is the load capacitance.

Here, we consider $Just - in - time$ operation [11], where the circuit works in its maximum frequency, i.e., the period is set to be the critical path delay of the circuit.

To consider variability in energy analysis, $3\sigma$ worst-case delay and mean $I_{leak}$ are considered in the static energy calculation as follows

$$E_{stat} = V_{DD}\mu_{I_{leak}}\mu_{delay}(1 + 3 * \frac{\sigma_{delay}}{\mu_{delay}}) \quad (6)$$

Figure 5 shows the consumed energy under process variations consideration.

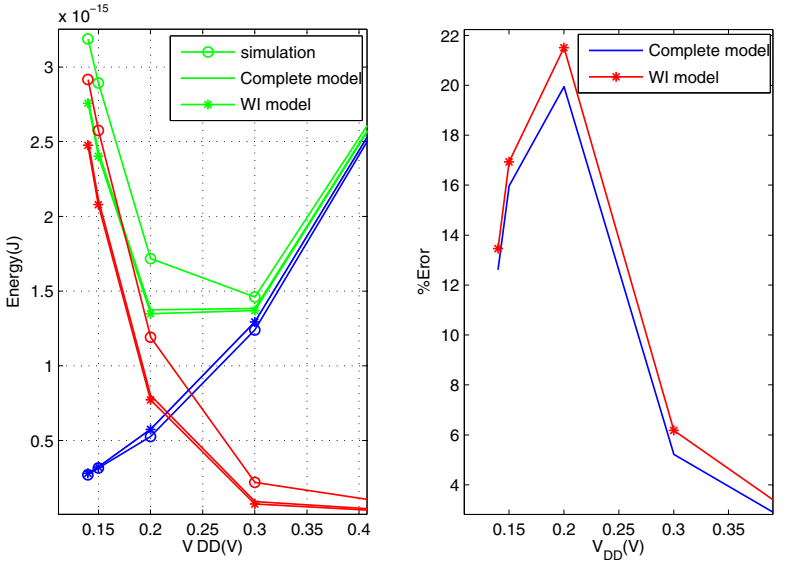



**Fig. 5.** Consumed energy under process variations (left), and % of complete and WI model error compared to Spice simulations (right)

We observe that the total energy consumed is slightly different from that determined by the models. The error of the energy on the minimum point is of 5% and 6%, when obtained by the complete and the WI model, respectively.
Not what we expect, the error of the WI model is comparable to that of the complete model as shown in Figure 5 (right). This can be explained by the inverse tendency of variability and delay determined by the WI model. On the one hand, the variability of the WI model is constant whatever the value of the supply voltage. It is therefore overestimated in the moderate and strong inversion regions. On the other hand, the delay obtained by the WI model is underestimated with respect to the one obtained by Spice simulations as observed in Figure 4.
As the energy contains the product of variability and the delay, there is a compensation that let the WI model remain a good model of energy consumption even under variability analysis.

## 4 Conclusion

In this paper, modeling under variability analysis is investigated. We show that the Weak Inversion model, normally used to describe sub-threshold circuit, is a

restrictive model that can no longer model circuit delay around the minimum energy point of circuit exposed to process variations.

We develop a complete model that extends over the weak and moderate inversion regions. Through Monte Carlo Spice simulations of a chain of inverters in a Low Power technology, we show that the new model is necessary to correctly model the variability of the circuit.

Nevertheless, instead of what may be expected, the Weak Inversion model remains a good model of energy consumption even under variability analysis. This is due to the compensation of delay and variability errors resulting when the WI model is applied.

**Acknowledgments.** The authors would like to thank STIC AmSud Program for the financial support given to this work through the NanoRadio project.

## References

1. Vittoz, E.: Weak inversion for ultimate low-power logic. In: Piguet, C. (ed.) Low-Power CMOS Circuits. CRC, Boca Raton (2006)
2. Wang, A., Chandrakasan, A.: A 180-mV subthreshold FFT processor using a minimum energy design methodology. IEEE Journal of Solid-State Circuits 40(1), 310–319 (2004)
3. Hanson, S., Zhai, B., Blaauw, D., Sylvester, D., Bryant, A., Wang, X.: Energy optimality and variability in subthreshold design. In: Proceedings of the 2006 International Symposium on IEEE, Low Power Electronics and Design, ISLPED 2006, pp. 363–365 (2007)
4. Verma, N., Kwong, J., Chandrakasan, A.: Nanometer MOSFET variation in minimum energy subthreshold circuits. IEEE Transactions on Electron Devices 55(1), 163–174 (2007)
5. Bol, D., Ambroise, R., Flandre, D., Legat, J.: Interests and limitations of technology scaling for subthreshold logic. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 17(10), 1508–1519 (2009)
6. Bol, D., Kamel, D., Flandre, D., Legat, J.: Nanometer MOSFET effects on the minimum-energy point of 45nm subthreshold logic. In: Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 3–8. ACM, New York (2009)
7. Calhoun, B., Chandrakasan, A.: Characterizing and modeling minimum energy operation for subthreshold circuits. In: Proceedings of the 2004 International Symposium on IEEE, Low Power Electronics and Design, ISLPED 2004, pp. 90–95 (2005)
8. Bol, D.: Pushing ultra-low-power digital circuits into the nanometer era. Ph.D. dissertation, University of California (2008)
9. Markovic, D., Wang, C., Alarcon, L., Rabaey, J.: Ultralow-power design in near-threshold region. Proceedings of the IEEE 98(2), 237–252 (2010)
10. Jespers, P.: The Gm/ID Methodology, a Sizing Tool for Low-voltage Analog CMOS Circuits. In: The Semi-empirical and Compact Model Approaches. Springer, Heidelberg (2009)
11. Seok, M., Hanson, S., Sylvester, D., Blaauw, D.: Analysis and optimization of sleep modes in subthreshold circuit design. In: Proceedings of the 44th Annual Design Automation Conference, pp. 694–699. ACM, New York (2007)

# High Level Synthesis of Asynchronous Circuits from Data Flow Graphs

Rene van Leuken, Tom van Leeuwen, and Huib Lincklaen Arriens

Circuits and Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands
t.g.r.m.vanleuken@tudelft.nl

**Abstract.** This paper presents a toolbox for the automatic generation of asynchronous circuits starting from a data flow graph description. The toolbox consists of a scheduling and code generation tool. We use traditional scheduling algorithms as for synchronous circuits, but have replaced the implied synchronous controller for an asynchronous distributed control network. The control circuit allows for true asynchronous operation of all digital resources and as a result of its scalable distributed topology allows unlimited resource sharing. The distributed controllers can be created by connecting a small number of pre-designed sub-controllers which are presented in this paper. Prototype IP-blocks of these sub-controller circuits have been designed in a 90nm ASIC design process. Our toolbox is a capable to generate large complex asynchronous solutions, with upto 20 percent power saving, and as least as good latency performance as of synchronous solutions.

## 1 Introduction

Digital circuits use a clock signal to synchronize operations, the so called synchronous circuits. Although this clock signal makes the design convenient, especially since practically all commercial synthesis tools assume a synchronous design, some advantages can be exploited when using asynchronous circuits (circuits without clock signal). Those advantages can include typical case performance, low power consumption, less sensitive to variability, lower EMI admittance and protection against differential power analysis attacks. Disadvantages of asynchronous circuits include the lack of synthesis tools, their sensitivity to hazards and in some cases performance loss. To assist a designer in his/her attempts to convert a behavior level description of a compute function to be implemented in digital hardware, we have developed an toolbox, which is capable of scheduling and mapping operations on hardware resources. These operations are currently limited to ALU functions like multiplication, subtraction, comparison and addition. However, since many computational and signal processing functions consist of only these functions, many of them can be implemented. Our design methodology uses traditional scheduling algorithms as for synchronous

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 317–330, 2011.

circuits to generate a asynchronous solution. To achieve this, we replace the by the scheduling software implied synchronous controller with an asynchronous distributed control network.

## 2 Related Work

Behavioral synthesis is widely explored in the past, mostly targeting synchronous circuits. Scheduling, the process of allocating operations to time slots, is a well-known method for behavioral synthesis. A large number of scheduling algorithms are available, as well as control network topologies. In this paper, standard scheduling algorithms for synchronous circuits are used, but a new control network is created, targeting asynchronous circuits. For behavioral synthesis of asynchronous circuits, a number of methods for scheduling and resource allocation are published [1] [8]. However, these publications do not include the synthesis of the control network. Also, a number of behavioral synthesis methods for asynchronous circuits including the control network synthesis are published. In [3], distributed controllers for asynchronous scheduled data flow graphs are proposed, similar to our method, but each distributed controller is specified in a separate Signal Transition Graph (STG). STG's are hard to synthesize because they should operate hazard free. In our method, only a few small STG's have to be synthesized, which can then be reused to create the larger distributed controller. In [5], a high level synthesis method using a bundled-data centralized controller is proposed. The centralized controller neglects some of the advantages of asynchronous operation, since all operations are synchronized by the controller. Also, their method is limited to bundled-data implementations, while our method can easily be converted to any completion detection method. In [2], Cortadella et. al. propose a method for de-synchronization. De-synchronization is the process of converting a (synthesized) synchronous circuit to an asynchronous circuit. Although this method does not target high-level synthesis and prevents resource sharing, the theory of de-synchronization is used in this paper since our method uses scheduling results for synchronous circuits.

## 3 Background

The starting point for behavioral synthesis is a behavioral description of the circuit. Our method uses a State Sequencing Graph (SSG) as input, which is then converted to a bundled-data asynchronous circuit. The conversion from a behavioral description to an SSG is not explained in this paper in detail, since the same method is used for synchronous designs, so only the relevant issues are explained in this paper. More details about scheduling and resource allocation, the process of converting the behavioral description to a SSG, can be found in [7].

### 3.1 Data Flow Graph

A Data Flow Graph (DFG) is a graph of operations, represented by nodes, and data-dependencies represented by directed edges. Additionally, there are

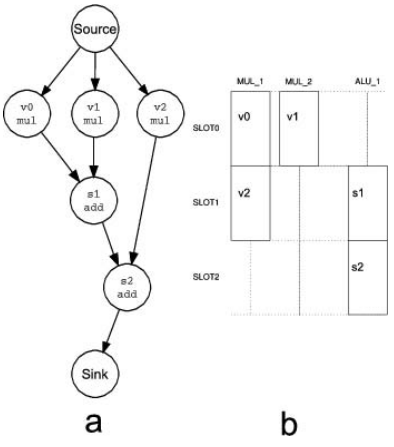



**Fig. 1.** a: Data Flow Graph of a 3rd order FIR filter, b: Resource Mapping

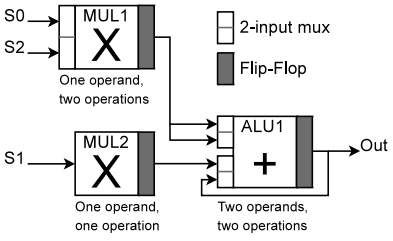



**Fig. 2.** Datapath of FIR3 filter with flip-flop

two extra nodes, the *source* and *sink* node. Those are used to represent data-dependencies from and to the environment. When an operation processes input data, it depends on the source node, and when the output of an operation is used by the environment, the sink node depends on this operations. An example is shown in Figure 1-a, where the DFG of a 3rd order Finite Impulse Response (FIR) filter is depicted.

### 3.2 Scheduling and Resource Sharing

When the data-dependencies are identified using the DFG, a scheduling algorithm can map each operation to a time slot. Then, operations can be allocated to resources like Multipliers and ALU's. Each resource can execute a number of operations from the DFG, but it can only execute one operation per time slot. Each operation is scheduled on a resource that is able to execute the operation. Data can be saved for more than one cycle in the flip-flop of a resource, but when the data needs to be saved after a new operation is executed, a register is used which is also considered a resource. This paper will not go into detail about scheduling and resource allocation, since well-known algorithms for synchronous circuits are used. The results of scheduling and resource allocation for the FIR filter can be found in Figure 1-b. It should be noted that most scheduling algorithms support multiple clock cycles per operation. Using a large number of clock cycles for each type of resource allows the synchronous scheduling algorithm to approximate asynchronous behavior at the cost of computational time [8]. As stated, each resource is scheduled to execute a number of different operations from the DFG. In the intended synchronous circuit, this is handled by a multiplexer (MUX) at the input of each resource. A flip-flop with an enable signal on its output makes sure the data is available as long as intended. The

flip-flop of a resource loads new data at the end of an operation scheduled to that resource. For example, if resource $A$ is scheduled to do two operations, $ax$ and $ay$, during cycle 1 and 3 respectively, then the results of operation $ax$ is available during cycle 2 and 3, and the result of operation $ay$ is available from cycle 4 to the last cycle. The datapath of the intended synchronous implementation of the FIR filter can be found in Figure 2. Since the second input to the multiplier is a constant in the FIR, these are hardcoded in the multiplier and not shown in the datapath. There are a number of requirements which have to be satisfied in order to create a valid scheduling for the intended synchronous datapath. These requirements are used later on:

- A result of an operation can only be used after it is produced. An operation $X$ that has a data-dependency from operation $Y$ in the DFG should be scheduled at least one time slot later than operation $Y$.
- A result should be available until the last operation that depends on it has consumed it. If the results from operation $X$ have data-dependencies to $Y$, the resource which executes operation $X$ cannot execute a new operation in a time slot earlier than the time slot in which $Y$ is executed. (unless a register is used, which acts as a new resource).

### 3.3 Bundled-Data

Asynchronous circuits indicate themselves when an operation is finished. There are several ways for an operation to indicate the completion, but the most common ways is by a matched delay element. If the operation starts, the input of the delay element is toggled. When the output of the delay element also toggles, the operation is assumed to be finished. The output of the delay element can thus be used to indicate that the succeeding operation can start [9]. A matched delay element is not data-dependent, and thus the delay is matched to the longest path in the operation. Although average-case performance can not be achieved with bundled data, performance improvements can be achieved by delay matching between the delay element and the operation since the variation between gates within a (part of a) chip is smaller than the maximum variation taken into account by the design of synchronous circuits [6]. Note that our method is not limited to Bundled-data, it can be converted to any completion detection method.

### 3.4 Signal Transition Graphs

Signal Transition Graphs (STG's) are a subset of Petri nets where all transitions are signal transitions [4]. In this paper, STG's are used to model Speed-Independent controllers. Speed-Independent circuits operate hazard-free under certain assumptions [9]. An STG contains transitions, places and directed edges which can connect a transition and a place in both directions. A directed edge cannot connect two places or two transitions. Every place can contain a token. A transition is enabled when all input places (places with an edge to the transition)

contain a token. A transition can be an input transition which *can* be fired by the environment when enabled, or a transition of an output or internal signal (non-input transition) which *will* be fired by the circuit when it is enabled. If a transition is fired, the tokens from the input places are removed and a token is added to each of its output places. To simplify the drawings, a place can be made implicit when it has exactly one incoming and one outgoing edge. The two edges and the place are then replaced by one edge between two transitions. This edge can now contain a token. *Marked Graphs* (MG) are a subset of STG's, where each place has exactly one incoming and one outgoing edge. When drawing a marked graph, all places are usually implicit. *Directed circuits* are a closed cycle in a marked graph where the direction of the arcs is respected. *A strongly connected MG* is a MG which is strongly connected when there is a path from each transition in the graph to every other transition.

### 3.5 De-synchronization

De-synchronization is the process of replacing all flip-flops for latches and the clock tree for latch controllers. This method is proposed by Cortadella et all [2]. Replacing the flip-flops for latches is a technique also used in synchronous designs. The process is trivial since a flip-flop, which is normally composed of two latches, is now explicitly created with two latches. Additionally, when the circuit is latch-based, the circuit can be retimed since the two latches are independent, i.e. the latches can be moved through logic blocks as long as the timing requirements are met. For de-synchronization, the clock signal for the latches is replaced by latch controllers. The controllers for de-synchronization, discussed in detail by Cortadella, ensure that the circuit is equivalent to the synchronous counterpart. Since we use scheduling results which are valid for synchronous circuits, we can use the theory of de-synchronization to prove that our asynchronous circuit is able to implement any valid scheduling results. However, the controllers proposed by Cortadella do not allow resource sharing, so new controllers are designed based on the theory of de-synchronization. In this paper, we use Marked Graphs to model the operation of the latches. Marked Graphs are also used by Cortadella to prove that the de-synchronization method is valid. We use Marked Graphs to prove the two properties, liveness and flow equivalence, which together show that the circuit is a valid replacement for the synchronous counterpart [2], in our case the intended synchronous circuit represented by the scheduling results.

**Liveness.** Liveness indicates that the circuit cannot enter a deadlock state, a state which it cannot leave. A strongly connected Marked Graph is live *if* each directed circuit contains at least one token.

**Flow Equivalence.** An asynchronous circuit is flow-equivalent to the synchronous counterpart, or in our case the scheduling results, if the data in each latch of the asynchronous circuit is equal to the data of the corresponding latch in the synchronous counterpart.

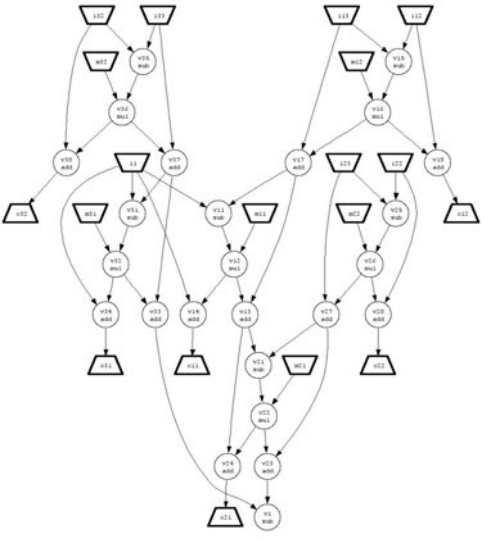


**Fig. 3.** A DFG of an example (6th order Lattice Wave Digital Filter) designers may use to experiment with scheduling tools and implementation

## 4 Proposed Method

### 4.1 Toolbox

To assist a designer in his/her attempts to convert a behavior level description of a compute function to be implemented in digital hardware, we have developed an toolbox, which is capable of scheduling and mapping operations on hardware resources. These operations are currently limited to some ALU functions, such as multiplication, addition, subtraction and comparison. The tool lets designers enter a data flow graph description and after specifying some attributes like type and number of resources, the tool creates a scheduled data flow graph (Figure 3) and maps the operations to resources. The tool is set up as a collection of (Matlab) functions, some of which are accessible through a GUI (Figure 4). The functions can be used for testing an algorithm both in the Matlab environment as a reference, as well as for supplying the VHDL test benches. Currently, designers can choose from the ASAP and ALAP scheduling methods (minimum number of clock states, unlimited resources), a Force Directed scheduling method (minimum number of clock states, minimum number of resources which are optimally distributed over the available clock states) and a List scheduling method (user defined number of resources that determine the number of clock states needed).

### 4.2 Datapath

For the conversion of the synchronous scheduling results to a latch-based design, the flip-flop is replaced by two latches. Using re-timing, one latch can be placed

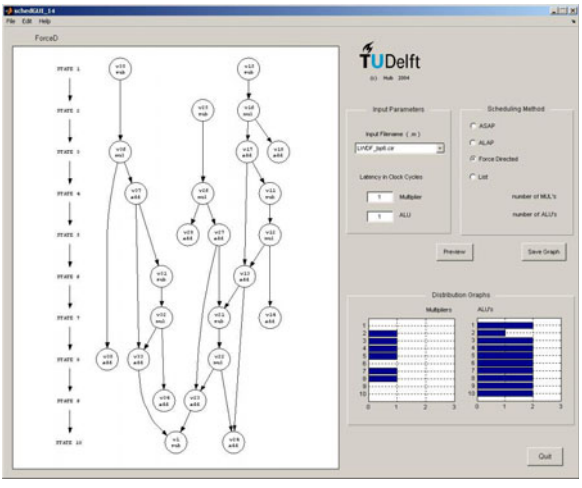



**Fig. 4.** The user interface of the scheduling and mapping tool

between the MUX and the input of the operation, as shown in Figure 5. A latch is located at the output of the MUX, the MUX and input latch can be combined using dynamic logic. In our simulations, dynamic logic is used for the combination of the input MUX and input latch.

### 4.3 Control Network

The controllers are based on the fall-decoupled model from. This model is live and flow-equivalent to synchronous circuits. However, this model does not allow hardware reuse, so a new model is created which allows hardware reuse, but is still live and flow-equivalent to the synchronous scheduling results.

**Fall-decoupled Model.** In Figure 6, the fall-decoupled model is shown. $A$ and $X$ indicate even- and odd latch control signals. The $A+$ transition will make latch $A$ transparent, while $A-$ will make latch $A$ opaque. In this model, even and odd latches alternate. In [2], it is proven that this model live and flow-equivalent to a synchronous counterpart when each flip-flop is replaced by two latches and latch controllers.

**Resource Sharing.** To be able to implement the scheduling results, the Fall-decoupled model has to be extended to implement resource sharing. For each operand, a separate handshake signal is introduced unless the data is an input from the environment or a constant, which are available during the entire operation of the circuit. The communication with the environment should also contain a form of handshaking to indicate that new input data is available and that the output data is ready. The start and done signal are introduced to represent the

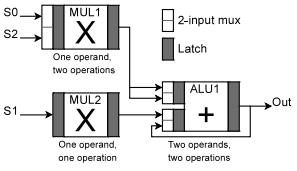



**Fig. 5.** Datapath of FIR3 filter with latches

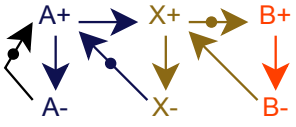



**Fig. 6.** Fall decoupled latch controllers (A and B are even, X is odd)

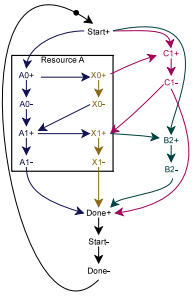



**Fig. 7.** Marked Graph of Fall-decoupled model extended with resource sharing

validity of inputs to and outputs from the asynchronous circuit respectively. When the start signal goes high, it indicates that all input data is valid, and when the done signal goes high, it indicates that the output data is available. The signals have to go low in the same order to reset the handshake signals to the initial state, i.e. it is a 4-phase handshake. Combining the Fall-decouple model with resource sharing results in the controller model shown in Figure 7. In this marked graph, each transition is a latch control signal except Start and Done. The letter A, B and C indicate the input latch control signals for three different resources, while X represents the output latch control signal for the resource with input latch A. The numbers associated with the latch control signal represent the time slot in which the operation is scheduled. If a resource has no operation scheduled for a certain time slot, the numbers will not be subsequent, but the numbers are always strictly increasing, e.g. no two operations can be scheduled on one resource at the same time and the order in time is honored by the marked graph. In the rest of this section, we focus on the implementation of the Control Network.

### 4.4 Liveness

Initially, there is only a token at the positive event of the start signal. To prove that the model is Live, we have to prove that any directed circuit includes the start signal. To prove that all directed circuits include the positive event of the start signal, the edges are followed backwards from any given event in the circuit:

1. The positive event of the start signal is preceded by the positive event of the done signal via the two negative events of those signals.
2. The positive event of the done signal is preceded by either the last negative event of an odd latch, or the last negative event of an even latch.
3. Any negative event of an odd latch control signal (Xn-) is always fired by the positive event of the odd latch control signal from the same cycle (Xn+).

4. The positive event of an odd latch (Xn+) is always fired by an event of an preceding (An+) or succeeding (Bn-m- where $m >= 0$) even latch at the same cycle or a lower cycle; The even latch from the same resource belongs to the same operation, and thus the same cycle. The negative event from the succeeding even latch (Bn-m-) has to be from the same cycle or a lower cycle, because the negative event indicates that the data in the odd latch from the previous cycle can be overwritten, because it is saved in the succeeding even latch. In the synchronous scheduling results, it is also assumed that previous output data is also available until the end of the next operation.
5. A negative event of an even latch (An-) is always fired by the positive event of that even latch (An+) from the same cycle.
6. A positive event of an even latch is either fired by the start signal containing a token, or by a latch event *at least one cycle earlier*. The positive event of an even latch (An+) is preceded by the negative event of the same latch from the previous operation (An-m- where $m >= 1$) which is at least one cycle earlier, from the negative event of the odd latch (Xn-m-) of the previous operation on the same resource, or from the positive event of a preceding resource (In Figure 7 shown as Xn-m+ with respect to Bn). The positive event of the preceding odd latch indicates that data is ready for an operation. The data for every operation should originate from an operation at least one cycle earlier, because the scheduling assumes that a result of an operation can only be consumed after it is produced.

Following the directed circuit backwards as indicated will always end in event 6, where the cycle number is decreased by one, from where it can be traced back to item 3, 4 or 5 where the cycle number stays equal or decreases and ends in event 6 again. Consequently, any directed circuit ends in cycle 0 of an even latch. Cycle 0 of an even latch is fired only by the start signal which contains a token. Thus, every directed circuit contains a token and the model is live.

### 4.5 Handshaking

To implement the proposed controllers in a circuit, handshaking is used to communicate between latch controllers. Each operand is coupled with one set of handshake signals. Inside each resource, the even latch controller and odd latch controller also communicate with one set of handshake signals. If output data for a certain operation is used more than once, the handshake is forked to all succeeding operations. A delay element is required for each latch, which results in two delay elements per resource. The required delay for the operation can be added to one of those delays. To save area and improve delay matching, the handshake signals for all operations scheduled on a particular resource should share the same delay element.

### 4.6 Handshake Blocks

To create an automated design flow which can implement the proposed handshaking, a number of IP-blocks have been designed. In this section, the topology

of the IP-blocks is explained. There are three main blocks (inputselect, odd latch controller and outputselect) and a few support blocks (Fake request, fake acknowledge, fork). The topology of those blocks can be found in Figure 8.

**Inputselect.** The inputselect block controls the even latch and input MUX. The active inputselect block which has control over the resource, indicated by the start signal, will send a request out and make the even latch is transparent when input data is ready, which is indicated by an input request. After the delay, the inputselect block will receive an acknowledge out from the latch controller and the even latch will be made opaque again. When the odd latch is also opaque (indicated by a low acknowledge out), a finish signal is send, to hand over control to the next inputselect block.

**Odd Latch Controller.** The odd latch controller makes the odd latch transparent when new data is ready and the old data is latched by all succeeding resources, indicated by a request in and low output acknowledge respectively. When the latch is transparent, a request out is send and after the delay, an acknowledged out is received, indicating that the data has propagated through the odd latch, so it can be made opaque again. Also, when the request in is high, an acknowledge in is send immediately to indicate that the data has propagated through the even latch. The acknowledge in can only go low when the odd latch is transparent.

**Outputselect.** The outputselect block does not control any latch or MUX, but is used to unfold the subsequent requests from the odd latch controller. The odd latch controller has only one output request signal, but the resource is shared so output data should be coupled with different handshake signals, which is taken care of by the outputselect block. When a request arrives at the active outputselect block (activated by the start signal), an acknowledge is send immediately to indicate that the odd latch can go opaque again, and an output request is send. The acknowledge is made low when the data is latched by the subsequent resource, indicated by a high acknowledge out signal. When the request in is low again, the next outputselect block is activated using the finish signal.

**Fake Request.** When an operand is provided by the environment, there is no handshake associated with the data and the data is valid during the entire operation of the circuit. For these cases, a fake request block is designed, which replaces the inputselect block when the operand is an input from the environment.

**Fake Acknowledge.** When a result is not an operand for any operation, e.g. when it is merely an output to the environment, there is no handshake associated with the result. For these cases, a fake acknowledge block is designed, which replaces the outputselect block.

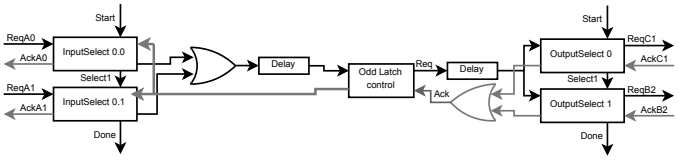



**Fig. 8.** Handshake blocks implementing a distributed controller

**Fork.** When a result of is an operand for multiple operations, the handshake should be forked. A fork is created for this purpose, which forks the request out to all destinations and uses a muller-C element to join all acknowledge signals.

**Join.** When a resource has more than one operand, each operation is assigned two inputselects block and delay elements. The datapath includes an extra MUX and even latch for the second operand. The odd latch controller is modified to include an extra handshake input.

## 5 Results

To test the asynchronous control flow, a number of high-level descriptions were synthesized. The implemented circuits include a 5th order LWDF low-pass filter (Figure 11) and an 18-point IMDCT (Figure 10). The circuits were scheduled using the List scheduling algorithm. It is assumed that an ALU with two latches and a MUX has 70% of the delay of an MUL with two latches and a MUX, so during scheduling the ALU was assigned 7 cycles and the MUL was assigned 10 cycles. During synthesis, the delay constraints for the ALU was set to 3.5 ns and the delay for the MUL was set to 5 ns. The circuits were implemented in UMC90 with a gate library produced by the Faraday corporation. The netlist of the IP-blocks was created using the Technology Mapping function in Petrify and the layout of the IP-blocks is designed using Cadence Encounter. The IP-blocks are then used to implement the control network for the scheduling results using our scheduling toolbox. Synopsys Design Compiler is used to compile the datapath and select delay elements to match the datapath latency. Then, the datapath, control network and delays are combined in Cadence Encounter and the placement and routing of the IP-blocks and datapath completes the layout. The delay of the data operations were distributed over the latch delay elements instead of using an extra delay element. The typical delay of the controllers and delay elements were matched to the worst-case delay of the datapath, while the simulations were run in typical case conditions. For each circuit, a synchronous counterpart was also generated. The same scheduling algorithm and resources are used, however, one clock cycle of 5ns is assigned to both the MUL and ALU.

Power simulations (Table 1) were performed using Cadence Encounter with back-annotated activity from netlist simulations. The same input was used for the synchronous and asynchronous circuits, except for the clock signal. The speed of the synchronous circuit was matched to that of the asynchronous circuit. For

**Table 1.** Power consumption and gate area

| Circuit | Synchronous | | Asynchronous | |
|---|---|---|---|---|
| | Power $(mW)$ | Gate area $(um^2)$ | Power $(mW)$ | Gate area $(um^2)$ |
| 5th order LWDF filter | 1.46 | 37687 | 1.72 | 93049 |
| 11th order WDF filter | 3.37 | 332590 | 2.70 | 493602 |
| 18-point IMDCT core | 13.72 | 86973 | 11.43 | 138622 |

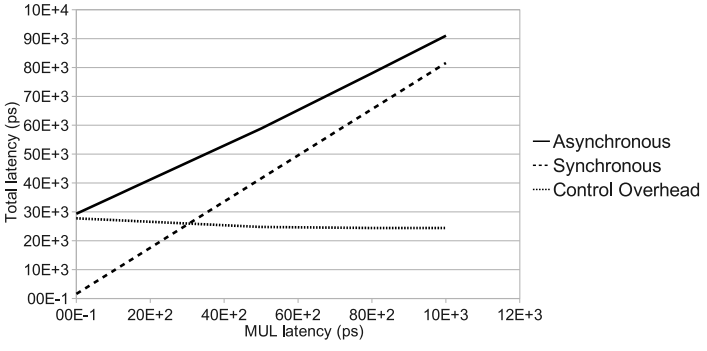



**Fig. 9.** Latency of asynchronous and synchronous LWDF filter with different multiplier latencies

a small 5th order LWDF filter with 32-bit operations, the synchronous circuit consumes less power. The extra power consumption for the asynchronous circuit can be attributed to the power consumption of the control network, which consumes more power than the synchronous control network per resource. For larger circuits, where more operations are scheduled per resource, the power of the control network becomes less significant and the asynchronous circuits use considerably less power than the synchronous counterparts. Table I also shows the area size of the designs. The numbers show that the additional area required to implement the distributed control network is substantional for small digital designs. The longest path of the LWDF circuit at different multiplier latencies is shown in Figure 9. The delay of the ALU is set at 70% of the multiplier delay. It can be observed that the absolute value of the controller overhead increases when the delay of operations decrease. This is a result of controller paths which are not delayed by the delay element that will become part of the critical path, while they would normally be shorter than a different path delayed by the delay element with the same endpoint. The slope of the synchronous circuit is equal to the number of cycles times the multiplier delay, since the multiplier delay fixes the clock period. The latency of the asynchronous circuit is the controller overhead plus the datapath latency. The datapath latency is equal to the results of the fine-grained scheduling. From Figure 9, it can be concluded that the control overhead is a significant part of the critical path when reasonable values for

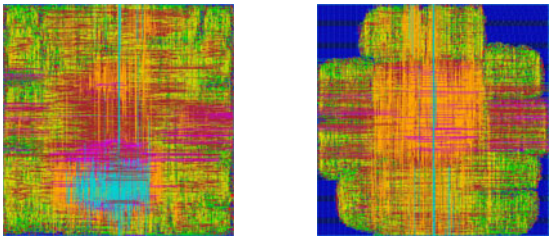


**Fig. 10.** The asynchronous design of a IMDCT (left) and the synchronous version at the right side

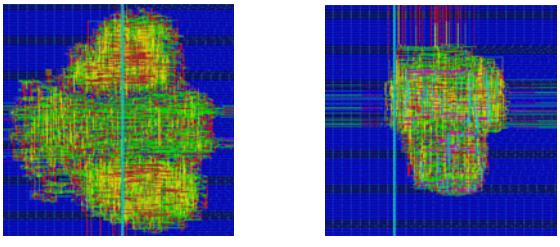


**Fig. 11.** The asynchronous design of a LWDF (left) and the synchronous version at the right side

the multiplier delay are used. To outperform a synchronous design when using multiplier latencies of 5ns, the control overhead should be reduced to 30% of its current value.

## 6 Conclusion

In this paper we have presented a toolbox for the automatic generation of asynchronous circuits starting from van data flow graph description. The toolbox consists of a scheduling and code generation tool. We use traditional scheduling algorithms as for synchronous circuits, but have replaced the implied synchronous controller for an asynchronous distributed control network. We have also presented an asynchronous distributed control network based which based upon a number of pre-designed and optimized IP-blocks. Compared to synchronous designs, a significant reduction (upto 20 percent) in power consumption can be achieved for larger circuits, while maintaining good latency figures. The area size cost is still high. However, some improvements are still possible such as flip-flop based register file designs. More importantly, the design asynchronous digital circuits has become a lot easier, since our high level synthesis toolbox automatically generates asynchronous circuit implementations of a given set of data flow graphs. But also, our toolbox is capable to synthesize large and very large complex circuits. To our knowledge, this was not possible before.

## References

1. Bachman, B., Zheng, H., Myers, C.: Architectural synthesis of timed asynchronous systems. In: International Conference on Computer Design, ICCD 1999, pp. 354–363 (1999)
2. Blunno, I., Cortadella, J., Kondratyev, A., Lavagno, L., Lwin, K., Sotiriou, C.: Handshake protocols for de-synchronization. In: Proceedings of 10th International Symposium on Asynchronous Circuits and Systems, pp. 149–158 (April 2004)
3. Cortadella, J., Badia, R.: An asynchronous architecture model for behavioral synthesis. In: Proceedings of 3rd European Conference on Design Automation, pp. 307–311 (March 1992)
4. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers (1996)
5. Hamada, N., Shiga, Y., Saito, H., Yoneda, T., Myers, C., Nanya, T.: A behavioral synthesis method for asynchronous circuits with bundled-data implementation (tool paper). In: 8th International Conference on Application of Concurrency to System Design, ACSD 2008, pp. 50–55 (June 2008)
6. Imai, M., Nanya, T.: A novel design method for asynchronous bundled-data transfer circuits considering characteristics of delay variations. In: 12th IEEE International Symposium on Asynchronous Circuits and Systems, pp. 10–77 (2006)
7. de Micheli, G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill Higher Education, New York (1994)
8. Saito, H., Hamada, N., Jindapetch, N., Yoneda, T., Myers, C., Nanya, T.: Scheduling methods for asynchronous circuits with bundled-data implementations based on the approximation of start times. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E90-A, 2790–2799 (2007), http://portal.acm.org/citation.cfm?id=1521680.1521697
9. Sparsø, J., Furber, S.: Principles of Asynchronous Circuit Design. Kluwer Academic Publishers, Dordrecht (2001)

# A Secure D Flip-Flop against Side Channel Attacks

Bruno Vaquie, Sebastien Tiran, and Philippe Maurine

LIRMM UMR 5506-CNRS
161, Rue Ada, 34085 Montpellier, France
firstname.lastname@lirmm.fr

**Abstract.** Side Channel Attacks (SCA) are a serious threat against security of cryptographic algorithms. Most of the countermeasures proposed to protect cryptosystems against these attacks, are efficient but present a significant area and power consumption overhead. The registers being the main weakness of cryptosystems, the source of leaks the more easily exploitable, we proposed a secure DFF which reduces leaks. In this paper, we present this countermeasure which considerably increases the robustness of cryptographic algorithms against side channel attacks. Moreover, the area and power overhead of our secure DFF in a cryptosystem is attractive.

**Keywords:** Side-Channel Attacks, hardware countermeasure, Secure D Flip-Flop, Data Encryption Standard (DES).

## 1 Introduction

Since Differential Power Analysis (DPA) [1], a lot of hardware countermeasures have been proposed to protect cryptographic devices against Side Channel Attacks (SCA). SCA are efficient because they allow the attackers to find secret keys of cryptographic algorithms by correlating processed data and side channel informations such as computing time, electric consumption or electromagnetic emissions. For example, Differential Power Analysis is based on the analysis of dependencies between intermediate data computed by an algorithm and the current consumption. By knowing the algorithm, DPA allows linking the current measured in the device to a theoretical model of power consumption in order to find the secret key. This kind of attack is very powerful because it requires few resources and little technical knowledge.

To protect the cryptographic devices against the SCA, designers have developped coutermeasures. The goal of a countermeasure is to remove this correlation by masking or hiding the internal data activity of cryptographic devices. We can sort the countermeasures into three categories:

- **redundant logics**
  Such secure logics aim at normalizing the power consumption by rendering the activity rate of all nets in the design constant and independent of the

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 331–340, 2011.

processed data. This is typically acheived by adopting dual or triple rail encoding of data [2–4].

- **randomisation**
  The masking countermeasure aims at rendering all intermediate values of the algorithm processed by the secure integrated circuit (IC) unpredictable by an attacker. This is typically achieved by mixing the input data with random data that are unknown for the attackers. There are two types of masking:
  - boolean masking : it is mainly used in symmetric algorithms. It consists in applying a XOR between the data and a random number generated on chip at each computations [5].
  - arithmetic masking : this type of masking is mainly used in asymmetric algorithms. This countermeasure uses the algebraic structure of the algorithm by adding random values to sensitive data [6].
- **desynchronisation**
  An underlying assumption to all SCA is that all attackable intermediate values processed by a secure IC are always computed at the same time. The goal of desynchronisation based countermeasures is to break this assumption by ramdomly spreading the critical computations in time. Ending so, Random Process Interrupts (RPI) [7] or random clock frequency [8] have been proposed as efficient countermeasures.

Despite their efficiency, the main drawback of these countermeasures is their area and power consumption overheads (Table 1). Such overheads forbid the use of such countermeasures in several applications like secure RFID tags or other low cost or low power products. It is thus mandatory to develop low power and low area overhead countemeasures.

**Table 1.** Area overhead of several hardware countermeasures applied to cryptosystems

| Countermeasure | Area |
|---|---|
| Sense Amplifier Based Logic (SABL) [2] | 73% |
| Wave Dynamic Differential Logic (WDDL) [3] | 240% |
| Secure Triple Track Logic (STTL) [4] | 455% |
| Boolean Masking [5] | $\geq 100\%$ |

In this paper, we propose the use of secure D Flip-Flop (DFF). The D Flip-Flop, as explained in section 2, constitutes the main source of leakage.

## 2 Cryptographic Devices Leakages

In this paper, we focus on symmetrical cryptosystems. During a cryptographic computation, sources of leaks are multiple, and occur at specific times. However, we may highlight the two most important ones.

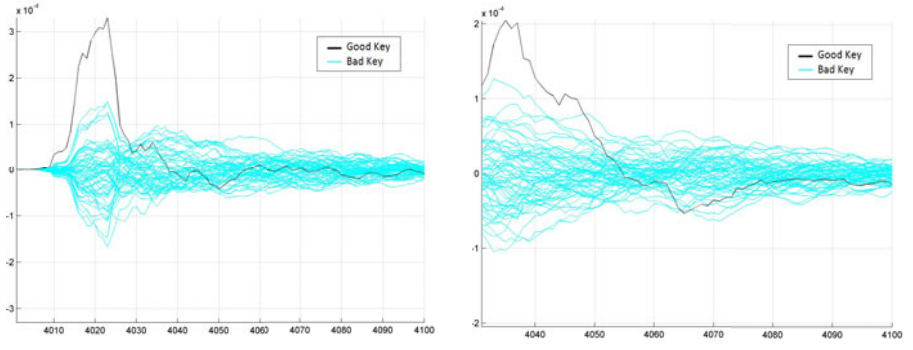
The main one is undoubtly the DFF banks or registers. Several reasons explain this fact. First, they usually sample, at each clock edge (thus in a perfectly synchronised way), either in a slave or master stage, the intermediate values targetted by the attackers.

Second, their power consumption significantly differs depending on if the data to be sampled is the opposite to that sampled during the previous rising edge of the clock or not. Indeed, if D has changed during the last clock cycle, then both master and slave have to switch, while none switch if D has been keep constant during the last clock cycle.

As a result, the amplitude of the current to be supplied to the secure IC, during an edge of the clock, is proportional to the number of DFF that have updated their content. This gives rise to the so called Hamming Distance Model [1].

The second main source of leakage is the first logic layers of standard cells after the DFF. Indeed, the switching activity of these gates is highly correlated to that of DFF and also remains quite synchronous [9]. Note however that this second source is less significant than the other.

Figure 1 gives evidences of this. It represents the difference of means (DoM) [1] after a simulated DPA, considering (a) a time windows embedding the switching of both DFF and the first layers of cells, and (b) a time windows that embed only the switching of the first layers of gates. As shown, it is necessary to process respectively 100 and 1500 power traces to disclose the key when considering or not the DFF activity.




(a) Switching of DFF and gates (b) Switching of gates

**Fig. 1.** Difference of Means (a) after 100 traces - (b) after 1500 traces

So DFF are the critical security issue in a CMOS cryptoprocessor. In order to provide low power and low area countermeasures, we introduce below a secure DFF.

# 3 Secure D Flip-Flop

## 3.1 Secure DFF Implementation

To normalize the power consumption, we propose to double the master-slave stage (DFF1 and DFF2) and to add a detector-generator of switching. This latter block detects switches in the DFF1, and provokes a switch in DFF2, when DFF1 does not switch.

As shown Figure 2, the detector-generator of switching is built so that at the clock's rising edge:

- when $Q1 \neq D1$, the DFF 1 switches.
- when $Q1 = D1$, the DFF 2 switches.

As a result, independently of the data processed by the secure DFF, there is always one and only one switching. Such a behaviour results in normalizing the power consumption of sequential elements as dual rail logic does for combinational elements.

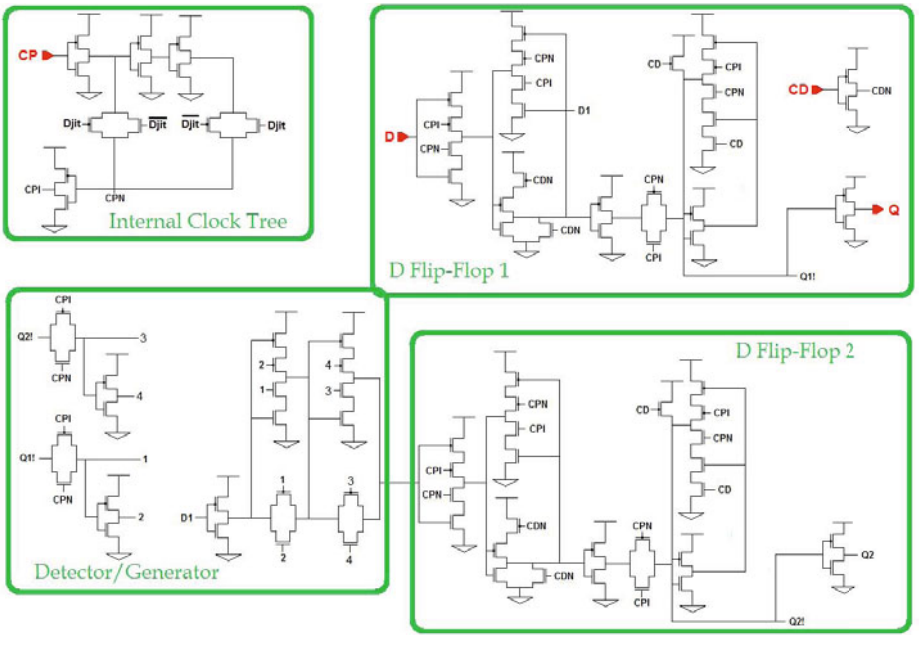



**Fig. 2.** Implementation of the secure D Flip-Flop

To go further, we also propose to add a jitter in the internal clock tree of our D Flip-Flop to improve its robustness. The figure 3 shows our internal clock tree with a jitter. The figure in full line is a standard internal clock tree while the dotted line shows our modifications.

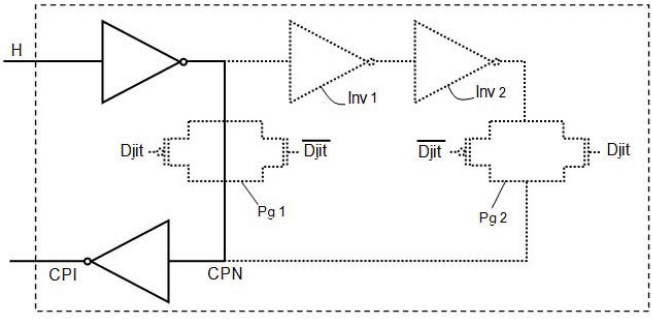



**Fig. 3.** clocktree with jitter of the D Flip-Flop

The jitter is composed of two inverters (Inv1 and Inv2) and two path gates (Pg1 and Pg2). The two path gates are controlled in phase opposition by $Djit$ and $\overline{Djit}$ by complementary. This signal is provided by a True Random Number Generator (TRNG).

When the signal Djit is low, the path gate Pg1 is closed and the path gate Pg2 is opened. The jitter is desactivated and the clock tree works like a standard clock tree (full line). When the signal Djit is high, the path gate Pg1 is opened and the path gate Pg2 is closed. The jitter is activated and the signal H goes through the two inverters Inv1 and Inv2 (dotted line).

Therefore, thanks to the signal Djit, we can modulate the time delay of the clock tree. This delay can take two distinct values, depending on the number and the size of inverters used to generate the signal CPI; which is two or four in our clock tree. The delay is well below the clock period. In our secure DFF, it is about half the propagation delay between input signal H and the Q output of a standard D Flip-Flop, that is to say between 50 ps and 100 ps for the 130 nm technology under consideration.

The jitter has two effects on the security.

- While the attacker captures the traces of power consumption, the jitter spreads in time and wreathes the traces.
- Because of the variation in the response time of the output of the DFF, the consumption of first layers of gates is desynchronized.

The jitter can therefore improve the robustness of cryptosystems by reducing leakages both in the DFF and in the first layers of gates.

### 3.2 Standard Characterisation

We wish to know the cost in area, consumption and timing of our secure D Flip-Flop, compared to a D Flip-Flop without countermeasures.

**Area and power consumption considerations.** We estimated that the surface of our secure DFF is four times bigger than that of a standard DFF. The power consumption has been simulated. Results show it is six time bigger than

that of a standard DFF. That can be explained by the addition of a second master-slave stage, of the logic circuit detector/generator and of the jitter. the latter considerably increases the power consumption. In comparison, the same secure DFF without the jitter ability consumes three times more than a standard DFF, due to high switching activity.

**Timing Considerations.** Tables 2, 3, and 4 give timing metric comparisons between our secure DFF and a standard DFF for the 130 nm technology.

**Table 2.** Clock to Output Time

| | | |
|---|---|---|
| Standard DFF | → | 85 to 97 ps |
| Secure DFF with inactive jitter | → | 180 to 205 ps |
| Secure DFF with active jitter | → | 296 to 317 ps |

**Table 3.** Setup Time

| $\tau_D$ \ $\tau_H$ | 50 ps | 100 ps | 150 ps | 200 ps |
|---|---|---|---|---|
| 50 ps | 46<br>365<br>245 | 40<br>360<br>230 | 46<br>350<br>225 | 41<br>350<br>160 |
| 100 ps | 51<br>370<br>250 | 55<br>360<br>238 | 51<br>356<br>230 | 45<br>354<br>225 |
| 150 ps | 65<br>375<br>250 | 61<br>370<br>240 | 56<br>3365<br>235 | 61<br>360<br>231 |
| 200 ps | 71<br>385<br>201 | 66<br>376<br>195 | 61<br>367<br>180 | 67<br>360<br>176 |

**Table 4.** Hold Time

| $\tau_D$ \ $\tau_H$ | 50 ps | 100 ps | 150 ps | 200 ps |
|---|---|---|---|---|
| 50 ps | -25<br>36<br>155 | -20<br>50<br>171 | -16<br>56<br>175 | -11<br>61<br>180 |
| 100 ps | -31<br>31<br>150 | -26<br>46<br>155 | -21<br>51<br>170 | -16<br>55<br>175 |
| 150 ps | -46<br>25<br>135 | -31<br>31<br>150 | -36<br>46<br>155 | -31<br>50<br>160 |
| 200 ps | -40<br>21<br>130 | -36<br>26<br>136 | -41<br>41<br>150 | -36<br>44<br>155 |

Table 2 shows the clock to output (Ck to Q) time for a DFF without countermeasures and our secure DFF with jitter active or not. Compared to a standard DFF, the Ck to Q time for our secure DFF is two time bigger when the jitter is inactive and three times when it is active.

Tables 3 and 4 display respectively the setup and the hold times versus the rise time of the clock and the input D of the DFF. In each box of the tables, the first line represents the setup or the hold time for a standard DFF, the second represents our secure DFF with inactive jitter, and the third represents our secure DFF with active jitter.

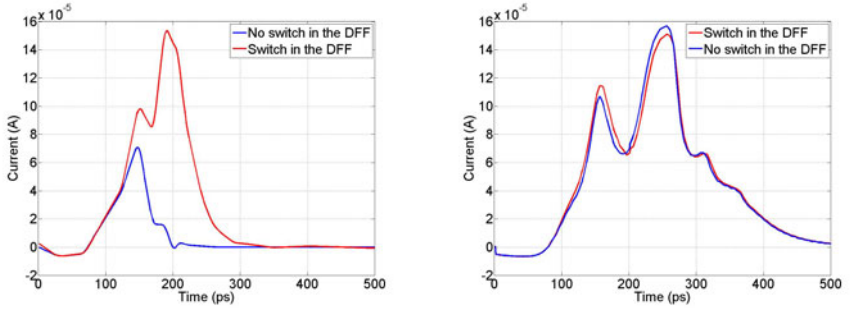
As a result of this comparison, we may conclud that our secure DFF prevents some acceptable timing metrics but exhibits a significant power overhead at cell. However, this overhead remains small compared to that reported on Table 1 for a whole cryptosystem.

### 3.3 Secure Characterisation

To evaluate the robustness of our DFF, we designed a Data Encryption Standard (DES) [10] with our secure DFF using a 130nm technology.

**Area and Power Consumption Considerations.** We want to know the area and power consumption overhead of our secure DFF in an algorithm like the DES. We estimated that the surface of a DES with our secure DFF is 30% bigger considering that of a DES with standard DFF. Results show that the power consumption is also 22% bigger compared to a DES with standard DFF. Once again, these values have to be compared to that of Table 1.

**Power Consumption Model.** Figures 4 and 5 show the impact of our countermeasure.




(a) Standard DFF (b) Secure DFF without jitter

**Fig. 4.** Power consumption model

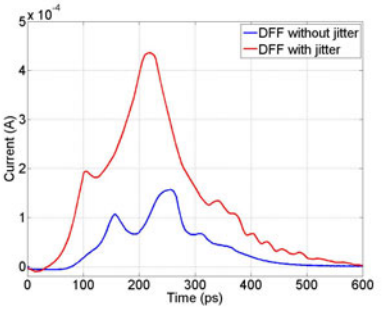



**Fig. 5.** Power consumption model of our DFF with jitter or not

Figure 4 shows the differences in power consumption model between a standard DFF (a) and our secure DFF (b). In (a), there are two distinct power consumption models: the red line is when the DFF switches and the blue continues dash is when

the DFF does not switch. In (b), both red and blue lines are the same; the power consumption models are identical. As shown, the addition of a second master slave stage quasi normalise power consumption of our secure DFF.

Figure 5 shows the power consumption model of our secure DFF with jitter. It represents the mean of several hundred power consumption traces. Compared to Figure 4, we notice that the power consumption is spreaded in time due to the effect of the jitter.

**Evaluation Against Side Channel Attacks.** In order to estimate the robustness of our countermeasure, we attacked a DES with our secure DFF with three different side channel attacks. We applied a Differential Power Analysis (DPA) [1], a Correlation Power Analysis (CPA) [11], and a Mutual Information Analysis (MIA) [12] with a Hamming Distance (HD) model.

We designed three different DES in technologie 130 nm:

- a standard DES without any countermeasure.
- a DES with our secure DFF
- a DES with three versions of our secure DFF, each one is characterized by a specific jitter value.

**Table 5.** Robustness Comparisons on SCA

| | DPA | | CPA | | MIA | |
|---|---|---|---|---|---|---|
| | MTD | Stability | MTD | Stability | MTD | Stability |
| Standard DES | 96 | 593 | 98 | 606 | 970 | 1680 |
| DES with secure DFF | 832 | 10502 | 490 | 2472 | 3512 | 15230 |
| DES with 3 different secure DFF | 504 | 39150 | 492 | 27350 | 2309 | 13223 |

Table 5 shows the comparison of results obtained for the three DES, respectively against simulated DPA, CPA, and MIA. The traces used in these attacks are without noise and are obtained thanks to the simulation tool NanoSim. The Minimum Trace to Disclosure (MTD) is defined as the minimal number of traces needed to correctly find the secret key. The stability is the number of traces required to recover the full key at least 100 consecutive times. The latter metric suggests that the secret key is definitely broken.

According to Table 5, we can observe that the DES with our secure DFF offers a better resistance against DPA, CPA, and MIA than the DES with standard DFFs. Furthermore, the addition of two more DFF with different jitter considerably increases the robustness on the DES against DPA and CPA. These results highlight the importance of using DFF with different jitters in DES.

To go further, we wanted to estimate the impact of noise on the SCA results. We added a Gaussian noise of mean zero and Variance V equal to a percentage of the maximum peak current of the DES. We reapplied simulated DPA and CPA. Table 6 and Table 7 show the results of the attacks versus the percentage of noise added in the traces.

**Table 6.** Stability on traces with noise on DPA

| % noise, V= | 0,15% | 1,5% | 15% | 30% | 50% |
|---|---|---|---|---|---|
| Standard DES | 605 | 602 | 1203 | 2032 | 3032 |
| DES with secure DFF | 10510 | 11000 | 15000 | - | - |
| DES with 3 different secure DFF | 40000 | 41000 | - | - | - |

**Table 7.** Stability on traces with noise on CPA

| % noise, V= | 0,15% | 1,5% | 15% | 30% | 50% |
|---|---|---|---|---|---|
| Standard DES | 637 | 673 | 853 | 1873 | 4324 |
| DES with secure DFF | 2500 | 3000 | 13000 | - | - |
| DES with 3 different secure DFF | 27000 | 29000 | - | - | - |

According to Table 6 and Table 7, we can see that the addition of noise complicates the attacks. The attacks become ineffective on the DES with our secure DFF when the noise reaches a certain level, 30% for the DES with our secure DFF and 15% for the DES with 3 different secure DFF.

In completing, the element which breaks in first in the DES despite our countermeasure is the first layers of gates after our secure DFF.

## 4 Conclusion

Security is a major concern in many applications. To offer a high level of security, many countermeasures have been proposed and proven efficient. However, most of them are impacted by extremely large area and power consumption overheads. As a result, they cannot be used in low power applications such as RF tags. Within this contrast, we proposed a secure DFF to indrease significantly the level of security of low power and secure products. Its use implies a power overhead of 22% only.

## References

1. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
2. Tiri, K., Akmal, M., Verbauwhede, I.: A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In: Proceedings of 28th European Solid-State Circuits Conf. ESSCIRC 2002 (2002)
3. Tiri, K., Verbauwhede, I.: A Logic Level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: Proceedings of DATE 2004, pp. 246–251 (February 2004)
4. Lomne, V., Maurine, P., Torres, L., Robert, M.: Evaluation on FPGA of Triple Rail Logic Robustness against DPA and DEMA. In: DATE, pp. 634–639 (2009)

5. Standaert, F.-X., Rouvroy, G., Quisquater, J.-J.: FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks. In: Proceedings of FPL 2006 (2006)
6. Coron, J.-S., Goubin, L.: On Boolean and Arithmetic Masking against Differential Power Analysis. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 231–237. Springer, Heidelberg (2000)
7. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
8. Akkar, M.-L., Bévan, R., Dischamp, P., Moyart, D.: Power analysis, what is now possible... In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 489–502. Springer, Heidelberg (2000)
9. Suzuki, D., Saeki, M., Ichikawa, T.: DPA Leakage Models for CMOS Logic Circuits. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 366–382. Springer, Heidelberg (2005)
10. National Bureau of Standards, Data Encryption Standard. Federal Information Processing Standards Publication 46 (January 1977)
11. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
12. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)

# Convex-Based Thermal Management for 3D MPSoCs Using DVFS and Variable-Flow Liquid Cooling

Francesco Zanini[1], David Atienza[2], and Giovanni De Micheli[1]

[1] Integrated Systems Laboratory (LSI), EPFL, Lausanne, Switzerland
[2] Embedded Systems Laboratory (ESL), EPFL, Lausanne, Switzerland
name.surname@epfl.ch

**Abstract.** In this work, we propose a novel online thermal management approach based on *model predictive control* for 3D multi-processors system on chip (*MPSoCs*) using microfluidic cooling. The controller uses dynamic voltage and frequency scaling (*DVFS*) for the computational cores and adjusts the liquid flow rate to meet the desired performance requirements and to minimize the overall MPSoC energy consumption (MPSoC power consumption+cooling power consumption). Our experimental results illustrate that our policy satisfies performance requirements and maintains the temperature below the specified threshold, while reducing cooling energy by up to $50\%$ compared with traditional state-of-the-art liquid cooling techniques. The proposed policy also keeps the thermal profile up to $18^{\circ}C$ lower compared with state of the art 3D thermal management using variable-flow liquid cooling.

## 1 Introduction

Power and thermal management are important challenges for multicore systems. Since power density is increasing, heat extraction is becoming more difficult. Moreover, in 3D stacked chips it is even more complex to develop efficient cooling mechanisms. However, liquid cooling has emerged as a potential solution to address the high temperatures in 3D chips [3], due to the higher heat removal capability of liquids in comparison to air. Liquid cooling is performed by attaching a cold plate with built-in microchannels, and/or by fabricating microchannels in the silicon layers of the 3D-MPSoC architectures. Then, a coolant fluid is pumped through the microchannels to remove the heat. The flow rate of the pumps can be altered dynamically, and the pump power consumption increases quadratically with the increase in flow rate [3]. Thus its contribution to the overall system energy is not negligible [16], and new thermal management policies must be developed to exploit this new cooling technology while considering the pumping power overhead.

The main contribution of this work is a novel thermal management approach for 3D stacks that controls both DVFS and a variable-flow liquid cooling using convex optimization to meet the desired performance and minimal energy requirements. The optimization process is applied at run-time using the convex-solver proposed by [14]. At this stage the convex solver finds the optimum frequency assignment for the inputs of the MPSoC system that will maximize performance under temperature constraints.

J.L. Ayala et al. (Eds.): PATMOS 2011, LNCS 6951, pp. 341–350, 2011.

We perform experiments on a 3D multicore architecture case study based on Niagara T1 UltraSparc2 cores [4] using benchmarks ranging from web-accessing to playing multimedia. Results show that the proposed method guarantees that scenarios with dangerous thermal profiles are avoided while satisfying the application performance requirements. Moreover, cooling energy is reduced by up to 50% compared with state of the art liquid cooling policies. In addition, the proposed policy keeps the average thermal profile up to $18°C$ lower compared with state of the art polices using variable-flow liquid cooling, like [16].

## 2 Related Work

Accurate thermal modeling of liquid cooling is critical in the design and evaluation of systems and policies. HotSpot [5] is a thermal model tool that calculates transient temperature response given the physical and power consumption characteristics of the chip. The latest versions of HotSpot include 3D modeling capabilities and liquid-cooled systems as well [6]. Finally, 3D-ICE [7] is a new thermal modeling tool specifically designed for 3D stacks, and includes inter-layer liquid cooling modeling capabilities.

Many researchers in computer architecture have recently focused on thermal control for *Multi-Processor System on Chips* (MPSoCs) [11], [8]. Processor power optimization and balancing using DVFS have been proposed in several works [8], [19]. However in all aforementioned policies there is not a guarantee to avoid hotspots by performing this optimization, because the policy targets power optimization and not hotspot avoidance.

More advanced solutions apply the concepts of model-predictive control to turn the control from open-loop to closed-loop [9], [10]. In [18] a similar concept is tailored for multi-modal video sensor nodes. In [15] a convex optimization-based approach is presented. The advantage of our technique over these methods is the new degree of freedom given by active liquid cooling. In [1] and [16], thermal management methods for 3D MPSoCs using a variable-flow liquid cooling have been proposed. These policies use simple heuristics to control the temperature profile of the 3D MPSoC, so there is not formal guarantee of optimality using this approach.

## 3 Modeling 3D Systems with Liquid Cooling

This paper deals with 3D MPSoCs stacking two or more dies. As an example, Figure 1(a),(b),(c) shows a 3D system consisting of 4-tiers. There are four silicon layers (A, B, C, D) (with various functional units grouped into $p$ islands with independent clock frequency and voltage supplies), where *microchannels* are etched in silicon bulk for liquid cooling. The model abstracts the interconnect on chip as copper layers (A, B, C, D). For every silicon layer there is a total of $nc$ linear microchannels $Ch_1 \ldots Ch_{nc}$. Microchannels are assumed to be equal in dimensions and a uniform coolant flux is assumed in channels of the same layer. All microchannels belonging to the same layer are connected to a pump. In the model shown in Figure 1(c) there is a total of 4 pumps connected to the microchannels of the 4 silicon layers. Fluid flows through channels belonging to different layers with different flow rates, according to the power of each pump. The liquid flow rate provided by each pump can be dynamically altered at runtime.

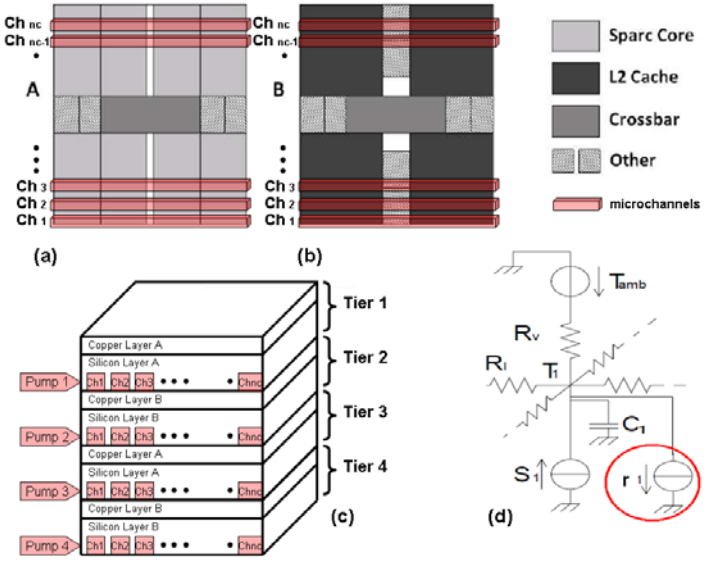



**Fig. 1.** 3-D stacked MPSoC with liquid cooling: (a)silicon layer type-A, (b)silicon layer type-B, (c)overall MPSoC view, (d)resistive network model

### 3.1 3D Heat Propagation Model

We model heat propagation in the 3D stacks using a network of thermal resistances and capacitances. To model the architecture shown in Figure 1(a),(b),(c) we propose an extension of the model presented in [15]. In particular, the active cooling (for cell $i$) is modeled by a current sink $r_i$, as highlighted by the circle in Figure 1(d). This current sink models the capability of the cooling system to remove heat in a specific location of the MPSoC. Following [15], we model the heat propagation process as:

$$\mathbf{t_{\tau+1}} = \mathbf{A}\mathbf{t}_\tau + \mathbf{B}\mathbf{p}_\tau \tag{1}$$

$$\tilde{\mathbf{t}}_\tau = \mathbf{C}\mathbf{t}_\tau \tag{2}$$

We assume that the total number of cells in all layers of the 3D MPSoC structure is $n$, the total number of cores is $p$ and the total number of pumps is $z$. Matrices $\mathbf{A} \epsilon \Re^{\mathbf{n}\times\mathbf{n}}$ and $\mathbf{B} \epsilon \Re^{\mathbf{n}\times(\mathbf{p}+\mathbf{z})}$ describe the heat propagation properties of the MPSoC. At time $\tau$, the temperature of the next simulation step of cell $i$, i.e. $(\mathbf{t_{\tau+1}})_\mathbf{i}$ can be computed thanks to Equation 1. The vector $\mathbf{p} \epsilon \Re^{\mathbf{p}+\mathbf{z}}$ is the power input vector. The first $p$ entries are the normalized power consumption for each of the $p$ frequency islands (cores), while the remaining $z$ entries are the normalized cooling power for each of the $z$ pumps.

The relation between the frequency assignment at time $\tau$, $\mathbf{f}_\tau \epsilon \Re^{\mathbf{p}}$, and the power consumption is assumed to be quadratic [5]. Equation 2 describes the choice of temperature sensors inside the MPSoC. Matrix $\mathbf{C} \epsilon \Re^{\mathbf{s}\times\mathbf{n}}$ relates the temperature value of each cell

to the temperature measurement of a particular sensor location in order to represent realistic IC designs that can only contain a discrete number of $s$ temperature sensors. The law that relates the microchannel flow-rate to heat extraction has been taken from [7]. Thus, we consider that the amount of heat $r_i$ extracted in cell $i$ by the fluid in the microchannel controlled by pump $j$ can be approximated by: $r_i = m_j \cdot \gamma_{i,j} \cdot (t_i - t_{fluid})$ where the fluid temperature is $t_{fluid}$, $t_i$ is the temperature of cell $i$ and $\gamma_{i,j}$ is the constant modeling the channel heat extraction properties. Vector $\mathbf{m} \epsilon \Re^{\mathbf{z}}$ is the normalized amount of heat that can be extracted for each of the $z$ independent pumps. Hence, by varying vector $\mathbf{m}$, the cooling power (flow rate of the cooling liquid) is varied to achieve the desired heat extraction. In our model, we used the temperature mapping from [7] to derive $\gamma_{i,j}$. Experiments have shown that by updating $\gamma_{i,j}$ every time the policy is applied ($10ms$ in our simulation setup), our approximation leads to a maximum error up to $\pm 5\%$.

### 3.2 Workload Model

The workload is an abstraction of what the operating system generates from higher-level software layers. For each $p$ clock islands (cores), the workload is defined as the minimum value of the clock frequency that the functional unit should have to execute the required tasks within the specified system constraints.

The workload requirement at time $\tau$ is defined as a vector $\mathbf{w}_\tau \epsilon \Re^{\mathbf{p}}$, where $(\mathbf{w}_\tau)_\mathbf{i}$ is the workload requirement value for input $i$ at time $\tau$. $(\mathbf{w}_\tau)_\mathbf{i}$ is the frequency that cores associated with input $i$ from time $\tau$ to time $\tau + 1$ should have in order to satisfy the desired performance requirement coming from the scheduler.

We assume a continuous control on the frequency ranging from the minimum frequency possible by each core ($\mathbf{f}_{\min}$) to their maximum frequency value ($\mathbf{f}_{\max}$), namely:

$$\mathbf{f}_{\min} \preceq \mathbf{w}_\tau \preceq \mathbf{f}_{\max} \ \forall \tau \tag{3}$$

When $(\mathbf{w}_\tau)_\mathbf{i} > (\mathbf{f}_\tau)_\mathbf{i}$, the workload cannot be processed and so it needs to be stored and rescheduled in the following clock cycles. The way we measure the performance of the system in achieving the requested workload requirements at time $\tau$ is given by the vector $\mathbf{u}_\tau \epsilon \Re^{\mathbf{p}}$ as follows:

$$\mathbf{u}_\tau = \mathbf{w}_\tau - \mathbf{f}_\tau \tag{4}$$

Therefore, we call $\mathbf{u}_\tau$ the undone workload at time $\tau$, which expresses the difference at time $\tau$ between the requested workload and the actual one executed by the MPSoC.

## 4 Policy Computation

The proposed thermal management approach uses both DVFS and variable-flow liquid cooling to meet the desired requirements, which are represented by a two-term cost function. The first one is related to power minimization (3D-MPSoC power consumption and liquid cooling pumping system power consumption) and the second one to the performance loss (undone work). The solution of following minimization are the

3D MPSoC frequencies and cooling pumps speeds necessary to meet the desires requirements. The control problem is formulated as the following convex optimization problem:

$$J = \sum_{\tau=1}^{h} \left( \|\mathbf{R}\mathbf{p}_\tau\|_{\mathbf{j}} + \|\mathbf{T}\mathbf{u}_\tau\|_{\mathbf{b}} \right) \tag{5}$$

$$min\ J \tag{6}$$

$$subject\ to: \ \mathbf{f}_{\min} \preceq \mathbf{f}_\tau \preceq \mathbf{f}_{\max}\ \forall\, \tau \tag{7}$$

$$\mathbf{x}_{\tau+\mathbf{1}} = \mathbf{A}\mathbf{x}_\tau + \mathbf{B}\mathbf{p}_\tau\ \forall\, \tau \tag{8}$$

$$\tilde{\mathbf{C}}\mathbf{x}_{\tau+\mathbf{1}} \preceq \mathbf{t}_{\max}\ \forall\, \tau \tag{9}$$

$$\mathbf{u}_\tau \succeq \mathbf{0}\ \forall\, \tau \tag{10}$$

$$\mathbf{u}_\tau = \mathbf{w}_\tau - \mathbf{f}_\tau\ \forall\, \tau \tag{11}$$

$$\mathbf{l}_\tau \succeq \mu \mathbf{f}_\tau^\alpha\ \forall\, \tau \tag{12}$$

$$-\mathbf{w} \preceq \mathbf{m}_{\tau+\mathbf{1}} - \mathbf{m}_\tau \preceq \mathbf{w}\ \forall\, \tau \tag{13}$$

$$0 \preceq \mathbf{m}_\tau \preceq \mathbf{1}\ \forall\, \tau \tag{14}$$

$$\mathbf{p}_\tau = [\mathbf{l}_\tau; \mathbf{m}_\tau]\ \forall\, \tau \tag{15}$$

It is important to highlight that the matrices $\mathbf{A}$, $\mathbf{B}$ used in previous equations are constant during the $h$ time steps the system tries to minimize the cost function $J$, and are then updated every time the policy is applied. In our optimization problem formulation, h is the time horizon [9](or number of time steps) to minimize the cost function J. Then, matrices A, B are constant during these next h time steps, and are then updated every time the predictive policy is applied.

Function $J$ is expressed by a sum where the summation index $\tau$ ranges from 1 to $h$. The first term $\|\mathbf{R}\mathbf{p}_\tau\|_{\mathbf{j}}$ is the $j$ norm (in our implementation $j = 1$) of the power input vector $p$ weighted by matrix $\mathbf{R}$. Power consumption is generated here by two main sources: the voltage-frequency setting of the 3D MPSoC and the liquid cooling pumping power. Vector $p$ is a vector containing normalized power consumption data of both the cores and the cooling pumps. Matrix $\mathbf{R}$ contains the maximum value of the power consumption of both the cores (first $p$ diagonal entries) and the cooling pumps (last $z$ diagonal entries).

The second term $\|\mathbf{T}\mathbf{u}_\tau\|_{\mathbf{b}}$ is the $b$ norm (in our implementation $b = 1$) of the amount of predicted required workload that has not been executed. The weight matrix $\mathbf{T}$ quantifies the importance that executing the workload required from the scheduler has in the optimization process.

Inequality 7 defines the range of working frequencies that can be used. It enables a continuous range of frequency settings but this does not prevent from adding in the optimization problem a limitation on the number of allowed frequency values. Equation 8 defines the evolution of the system according to the present state and inputs. Equation 9 states that temperature constraints should be respected at all times and in all specified locations. Since the system cannot execute jobs that have not arrived, every entry of $\mathbf{u}_\tau$ has to be greater than or equal to 0 as stated by Equation 10. The undone work at time $\tau$, $u_\tau$ is defined by Equation 11. Equation 12 defines the relation between the power

vector $\mathbf{l}$ and the working frequencies. $\mu$ is a technology-dependent constant. Because of the fact that all constraints in the minimization problem must be convex functions, we relaxed the original power equation to the convex inequality of Equation 12. By doing this operation we changed the original minimization problem to the problem described by the convex Equations 5-12. It can be shown that the resulting relaxed convex problem is equivalent to the original problem with the equality constraint [20].

Equation 15 defines formally the structure of vector $\mathbf{p}$ as described in Section 3.1. Vector $\mathbf{l} \epsilon \Re^{\mathbf{p}}$ is the power input vector, where $p$ is the number of frequency islands composing the 3D-MPSoC. Vector $\mathbf{m} \epsilon \Re^{\mathbf{z}}$ contains the normalized amount of cooling power for each of the $z$ independent pumps. Equations 13-14 define constraints on the liquid cooling management. Equation 14 states that $\mathbf{m}$ is a normalized value and it can range from 0 to 1. Equation 13 defines the maximum increment/decrement that the normalized pump can have between two consequent applications of the policy. In other terms this value takes into account the mechanical time dynamics of the pump. Their values are stored in vector $\mathbf{w} \epsilon \Re^{\mathbf{z}}$.

The result of the optimization is an optimal sequence of future control moves (i.e., frequency settings for the cores of the 3D MPSoC which are stored in vector $\mathbf{f}$). To increase the performance of our proposed policy, history information about the task arrival process are exploited by the proposed algorithm. Matrix $\mathbf{T}$ is chosen accordingly to the reliability of the workload prediction. We have selected these parameters to achieve a good prediction, according to empirical studies performed on different benchmarks [12].

## 5 Experimental Setup

### 5.1 3D MPSoC Model

The MPSoC structure we are considering is presented in Figure 1(a),(b),(c). The floorplan has been modelled using technological parameters and coefficients taken from [1] and [4]. This structure has a maximum operating frequency of 1.2 GHz and the maximum power consumption of each core at this frequency is 5 W.

To implement the voltage and frequency scaling techniques, we use frequencies ranging from $\mathbf{f}_{\min}$ to 1.2GHz, see [4] for details. In this range, only specific values of frequencies are allowed. These values are generated from the integer division of the maximum clock frequency by scaling factors as proposed in [17].

We compute the leakage power of processing cores as a function of their area and the temperature. We assume a base leakage power density of $0.25 W mm^2$ at $383°K$ for $90nm$, as in [19]. $T_o$ accounts for the temperature effects on leakage power and we use the model proposed in [1]. In this case, the leakage power at a temperature $T_o$ $°K$ is given by: $P(T) = Po \cdot e^{\beta(T-383)}$, where $P_o$ is the leakage power at $383°K$, and $\beta$ is a technology dependent coefficient. Finally we set $\beta = 0.017$ [16].

The number of independent flow rates is 4 and the spacing between two microchannels on the same layer is $100\mu m$. We assume that a pump connected to all microchannels of the same layer, such as a centrifugal pump [16], is responsible for the fluid injection to the whole system. This pump has the capability of producing large discharge rates at small pressure heads. Liquid is injected to the stacks from this pump via

a pumping network. To enable using different flow rates for each of the 4 stacks, the cooling infrastructure includes valves in the network. Cooling microchannels parameters and cooling pump power consumption values are taken from [16].

### 5.2 Policy Setup

According to the general model of Equations 5-12, the problem formulation is the following. Matrix $\mathbf{T}$ is set to be an identity matrix while matrix $\mathbf{R}$ contains the maximum value of the power consumption of both the cores and the cooling pumps (power values from [1]). The policy minimizes the sum of all contributions to the 3D MPSoC power consumption as well as the undone workload. For this reason, we set both the norms $b$ and $j$ to 1.

All the others constraints expressed by Equations 7-12 are considered inside the problem formulation. The policy is applied every $T_{pol} = 10ms$, while the simulation step for the discrete time integration of the RC thermal model has been set to $200\mu$s. The maximum temperature limit is set to $370°K$. The room temperature and $t_{fluid}$ are set to $300°K$. In the problem formulation, we used $\alpha = 2$ to establish the relation between the frequency setting and the power consumption. The linear predictor has been designed using a $3^{rd}$ order polynomial equation, an observation window of $600ms$ and a prediction length equal to $50ms$ in the future. The optimization process is done online using the convex solver proposed in [14]. These operations, have been performed on standard processors (i.e., Core Duo @ 2GHz) in few tenth of microseconds. This time is 3 orders of magnitude smaller compared with the time the policy is applied (i.e.10ms). The time constants needed by the mechanical dynamics of the cooling pumps to go from 0 to maximum power is set to $400ms$.

We use workload traces collected from real applications running on an UltraSPARC T1. We record the utilization percentage for each hardware thread at every second using *mpstat* for several minutes for each benchmark. We use various real-life benchmarks including web server, database management, and multimedia processing.

## 6 Experimental Results

In our experiments, we compare the proposed 3D thermal management method with state-of-the-art thermal management techniques based on DVFS, load balancing and variable flow liquid cooling [1], [16], [11], [13], [2].

Dynamic load balancing (**LB**) [11] balances the workload by moving threads from a core's queue to another if the difference in queue lengths is over a threshold. Temperature-triggered task migration (**TTTM**) [13] moves tasks from a core if that core exceeds the threshold temperature. TTTM has an impact on performance resulting from the time overhead required to move tasks between the cores (e.g., context switch overhead and cold start effects). In this work we assume a $1ms$ overhead when a thread is migrated to a new core [1], [2]. For previously mentioned polices, if the temperature goes higher than $420°K$, the system shuts down until the maximum MPSoC temperature returns below $250°K$. In temperature triggered DVFS (**TTDVFS**) [11] the voltage and frequency settings are reduced to the $10\%$ of the maximum value when

the maximum MPSoC temperature exceed the threshold value set to $370°K$. TTTM and TTDVFS can also be combined into a joint policy called (**TTTM _TTDVFS**) [2].

We experiment with both air-cooled (**AC**) and liquid-cooled (**LC**) systems for comparison purposes. In **LC_LB**, we apply $100\%$ of the maximum flow rate (0.0323 l/min per cavity [16]).We also consider in the comparison the latest state-of-the-art liquid cooling methods proposed in proposed in [1] and [16]. These methods employ a variable-flow liquid cooling combined with DVFS. We refer to the first method as **LC_VF** and to the second one as **LC_Fuzzy**.

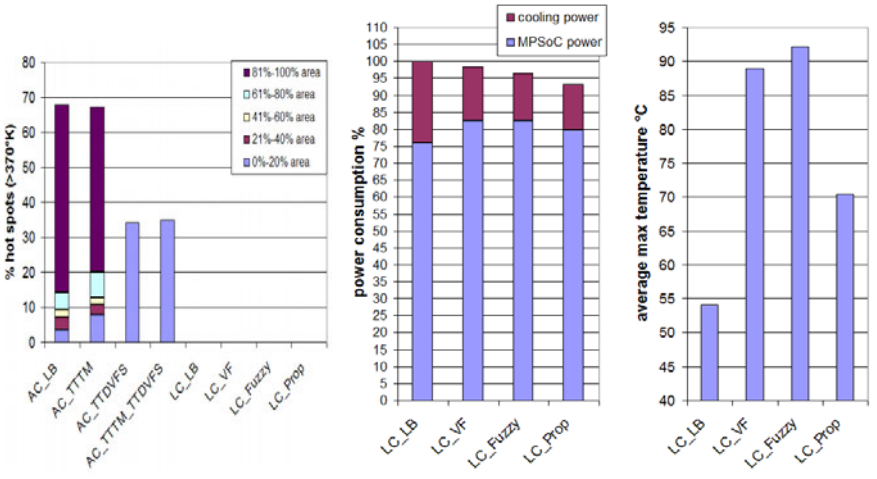
Thermal impact of all the policies on the system is shown in Figure 2(a). This figure compares the percentage of run-time execution where the maximum MPSoC temperature is higher than $370°K$. The hotspot area is labeled as the overall MPSoC area percentage.

The first four policies are air cooled methods, while the last four are liquid cooled. As Figure 2(a) shows, the first ones are not able to avoid hot spots. **AC_LB** and **AC_TTTM** present hot spots for up to $67\%$ of the execution time, and in addition to that, these hot spots affect more than $80\%$ of the total MPSoC area. Methods using temperature-triggered DVFS show a better performance. This is shown for **AC_TTDVFS** and **AC_TTTM_TTDVFS**.

Hence, they present hot spots for only $34\%$ and $35\%$ of the execution time, respectively. In addition, these hot spots cover less than $20\%$ of the overall MPSoC area.

Nevertheless, overall air cooled policies do not completely avoid hot spots. The reason is because the 4-tier stacked architecture is unable to dissipate the heat of inner layers by using only a heat spreader. These results indicate the benefits of inter-tier liquid cooling techniques to avoid hotspots scenario, as they remove the heat directly from the inner layers of the 3D-MPSoC (cf. Figure 1). In addition to that, liquid cooling policies provides a value of undone workload that is less than $1\%$ of the overall executed workload. However, air cooled polices provide values ranging from $24\%$ to $31\%$ in the case of **AC_LB** and **AC_TTDVFS**, respectively.

Previous results show the reason why there is a need for liquid cooling for 3D-MPSoC structures. Because of the fact that we are interested in techniques that avoid hot spots while satisfying performance requirements, we restrict from now on our comparison to liquid cooling methods. The following paragraphs compare the proposed policy versus state of the art liquid cooling methods. The left graph of Figure 2(b) shows the overall energy consumption of the 3D MPSoC. It is divided here into two contributions. The first one is the one absorbed by the cooling network (pumps and valves) while the second is the energy absorbed by the MPSoC activity (switching and leakage). The simplest policy **LC_LB** shows the highest energy consumption. The value of the cooling power here represents $24\%$ the overall 3D MPSoC energy consumption. For this reason, **LC_VF** [1] and **LC_Fuzzy** [16] have been proposed to reduce the power consumption of the cooling system. We tested these policies on our experimental setup. They show a reduction in the cooling power consumption by approximately $30\%$ and $50\%$ respectively. The proposed technique has a cooling and an overall 3D MPSoC power consumption that is respectively $50\%$ and $7\%$ lower compared with **LC_LB**. If we compare our policy with **LC_Fuzzy**, we see approximately the same saving in terms of cooling power and a $3\%$ additional saving in the overall MPSoC consumption.




(a) (b)

**Fig. 2.** (a):Percentage of run-time execution where the maximum MPSoC temperature is higher than the threshold($370^\circ K$). The area of the hotspot is also provided as a percentage of the overall MPSoC area.(b) left graph: energy consumption of the overall system: 3D MPSoC power consumption and cooling network. Values are normalized to **LC_LB**;(b) right graph: average maximum 3D MPSoC temperature [$^\circ C$].

Finally, Figure 2(b) shows the average maximum 3D MPSoC temperature for all the policies under comparison. The lowest thermal profile among the compared policies is generated by the **LC_LB**. In this case the maximum MPSoC temperature has an average value of $54^\circ C$. **LC_LB** and **LC_Fuzzy** show a thermal profile having an average maximum temperature of $89^\circ C$ and $92^\circ C$, respectively. The reason is because both these systems save energy by reducing the cooling cost and by having the system working at a temperature close to the threshold set to $97^\circ C$. However, the proposed policy is able to save as much energy as **LC_Fuzzy**, while being able to keep the thermal profile $18^\circ C$ lower. The main reason is because the predictive problem formulation of the proposed method is able to satisfy performance requirements by acting in advance and this allows the policy a smoother control on the system and SAVES active power. Therefore, the 3D MPSoC thermal profile is colder and more thermally-balanced overall.

## 7 Conclusion

The contribution of this paper is a novel online thermal management approach that exploits the use of variable-flow liquid cooling on a 3D-MPSoC. In particular, we propose a thermal manager that uses DVFS and adjusts the liquid flow rate to meet the desired performance requirements, while minimizing the overall MPSoC energy consumption (MPSoC frequency setting and liquid cooling) and preventing hot spots. Our experimental results illustrate that our policy satisfies performance requirements, maintains the temperature below the specified threshold, while reducing cooling energy by up to

$50\%$ compared with traditional state-of-the-art liquid cooling techniques. The policy also keeps the thermal profile approximately $18°C$ lower compared with state of the art polices using liquid cooling.

## References


1. Coskun, A.K., et al.: Energy-Efficient Variable-Flow Liquid Cooling in 3D Stacked Architectures. In: DATE 2010 (2010)
2. Coskun, A.K., et al.: Modeling and dynamic management of 3D multicore systems with liquid cooling. In: VLSI-SoC (2009)
3. Brunschwiler, T., et al.: Interlayer cooling potential in vertically integrated packages. Microsyst. Technol. (2008)
4. Kongetira, P., et al.: Niagara: A 32-way multithreaded SPARC processor. IEEE Micro (2005)
5. Skadron, K., et al.: Temperature-aware microarchitecture: Modeling and impl. In: TACO (2004)
6. Coskun, A.K., et al.: Modeling and dynamic management of 3D multicore systems with liquid cooling. In: VLSISOC (2009)
7. Sridhar, A., et al.: 3D-ICE: Fast compact transient thermal modeling for 3D-ICs with inter-tier liquid cooling. In: ICCAD (2010)
8. Mukherjee, R., et al.: Physical aware frequency selection for dynamic thermal management in multi-core systems. In: ICCAD (2006)
9. Bemporad, A., et al.: The explicit linear quadratic regulator for constrained systems. Automatica (2002)
10. Zanini, F., et al.: Multicore Thermal Management with Model Predictive Control. In: ECCTD (2009)
11. Donald, J., et al.: Techniques for multi-core thermal management: Classif. and new exploration. In: ISCA (2006)
12. Coskun, A.K., et al.: Proactive temperature balancing for low cost thermal management in MPSoCs. In: ICCAD (2008)
13. Coskun, A.K., et al.: Temperature management in multiprocessor SoCs using online learning. In: DAC (2008)
14. Grant, M., et al.: CVX: Matlab software for disciplined convex programming, `http://www.stanford.edu/~boyd/cvx/`
15. Zanini, F., et al.: Online Convex Optimization-Based Algorithm For Thermal Management of MPSoCs. In: GLSVLSI (2010)
16. Sabry, M.M., et al.: Fuzzy Control for Enforcing Energy Efficiency in High-Performance 3D Systems. In: ICCAD (2010)
17. Ruggiero, M., et al.: MPARM: Exploring the Multi-Processor SoC Design Space with SystemC. The Journal of VLSI Signal Processing (2005)
18. Magno, M., et al.: Adaptive power control for solar harvesting multimodal wireless smart camera. In: ICDSC (2009)
19. Bose, P.: Power-efficient microarchitectural choices at the early design stage. In: Keynote Address on PACS (2003)
20. Boyd, S., et al.: Convex Optimization. Cambridge University Press, Cambridge (2004)

# Author Index