

# SOC DESIGN METHODOLOGIES

Edited by  
**Michel Robert**  
**Bruno Rouzeyre**  
**Christian Piguet**  
**Marie-Lise Flottes**



SPRINGER SCIENCE+  
BUSINESS MEDIA, LLC



---

# **SOC DESIGN METHODOLOGIES**

## **IFIP - The International Federation for Information Processing**

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- open conferences;
- working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is less rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is in information may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly. National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

# SOC DESIGN METHODOLOGIES

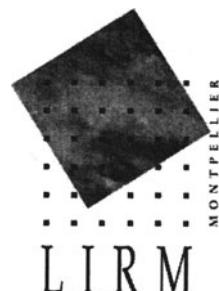
*IFIP TC10 / WG10.5*

*Eleventh International Conference on Very Large Scale  
Integration of Systems-on-Chip (VLSI-SOC'01)  
December 3–5, 2001, Montpellier, France*

*Edited by*

**Michel Robert**  
**Bruno Rouzeyre**  
**Christian Piguet**  
**Marie-Lise Flottes**

*Laboratoire d'Informatique de Robotique  
et de Microelectronique de Montpellier (LIRMM)  
UMR CNRS/Université Montpellier II  
France*



---

**Library of Congress Cataloging-in-Publication Data**

A C.I.P. Catalogue record for this book is available from the Library of Congress.

**SOC Design Methodologies**

Edited by Michel Robert, Bruno Rouzeyre, Christian Piguet, Marie-Lise Flottes

ISBN 978-1-4757-6530-4 ISBN 978-0-387-35597-9 (eBook)

DOI 10.1007/978-0-387-35597-9

---

**Copyright** © 2002 by Springer Science+Business Media New York

Originally published by Kluwer Academic Publishers in 2002

All rights reserved. No part of this work may be reproduced, stored in retrieval system, or transmitted in any form or by any means, electronic, mechanical, photo-copying, microfilming, recording, or otherwise, without written permission from the Publisher Springer Science+Business Media, LLC.

with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

*Printed on acid-free paper.*

# Contents

|                       |      |
|-----------------------|------|
| Preface               | ix   |
| Conference Committees | xiii |

## ***Architecture for Signal & Image Processing***

|  |    |
|--|----|
| Two ASIC for Low and Middle Levels of Real Time Image Processing<br><i>P. Lamaty, B. Mazar, D. Demigny, L. Kessal, M. Karabernou</i> | 3  |
| 64 x 64 Pixels General Purpose Digital Vision Chip<br><i>T. Komuro, M. Ishikawa</i>  | 15 |
| A Vision System on Chip for Industrial Control<br><i>E. Senn, E. Martin</i>  | 27 |
| Fast Recursive Implementation of the Gaussian Filter<br><i>D. Demigny, L. Kessal, J. Pons</i>  | 39 |

## ***Dynamically Re-configurable Architectures***

|  |    |
|--|----|
| A Dynamically Reconfigurable Architecture for Low-Power Multimedia Terminals<br><i>R. David, D. Chillet, S. Pillement, O. Sentieys</i>                           | 51 |
| Dynamically Reconfigurable Architectures for Digital Signal Processing Applications<br><i>G. Sassatelli, L. Torres, P. Benoit, G. Cambon, M. Robert, J. Galy</i> | 63 |
| Reconfigurable Architecture Using High Speed FPGA<br><i>L. Kessal, R. Bourguiba, D. Demigny, N. Boudouani, M. Karabernou</i>                                     | 75 |

***CAD Tools***

|   |     |
|---|-----|
| Design Technology for Systems-on-Chip<br><i>R. Camposano, D. MacMillen</i>  | 87  |
| Distributed Collaborative Design over Cave2 Framework<br><i>L. S. Indrusiak, J. Becker, M. Glesner, R. Reis</i>                           | 97  |
| High Performance Java Hardware Engine and Software<br>Kernel for Embedded Systems<br><i>M. H. Miki, M. Kimura, T. Onoye, I. Shirakawa</i> | 109 |
| An Object-Oriented Methodology for Modeling the Precise<br>Behavior of Processor Architectures<br><i>J. C. Otero, F. R. Wagner</i>        | 121 |
| Interconnect Capacitance Modelling in a VDSM CMOS Technology<br><i>D. Bernard, C. Landrault, P. Nouet</i>                                 | 133 |

***IP Design & Reuse***

|  |     |
|--|-----|
| Abstract Communication Model and Automatic Interface<br>generation for IP integration in Hardware/Software Co-design<br><i>C. Araujo, E. Barros</i>      | 145 |
| An Evolutionary Approach for Pareto-optimal Configurations<br>in SOC Platforms<br><i>G. Ascia, V. Catania, M. Palesi</i>                                 | 157 |
| Design of a Branch-Based Carry-Select Adder IP Portable<br>in 0.25 $\mu$ m Bulk and Silicon-On-Insulator CMOS Technologies<br><i>A. Nève, D. Flandre</i> | 169 |

***High Level Design Methodologies***

|   |     |
|---|-----|
| A Standardized Co-simulation Backbone<br><i>B. A. De Mello, F. R. Wagner</i>  | 181 |
| Automatic Code-Transformation and Architecture Refinement<br>for Application-Specific Multiprocessor SoCs with Shared Memory<br><i>S. Meftali, F. Gharsalli, F. Rousseau, A. A. Jerraya</i> | 193 |

***Power Issues***

|  |     |
|--|-----|
| Modeling Power Dynamics for an Embedded DSP Processor<br>Core. An Empirical Model<br><i>C.H. Gebotys, R. Muresan</i> | 205 |
| Power Consumption Model for the DSP OAK Processor<br><i>P. Guittion-Ouhamou, C. Belleudy, M. Auguin</i>              | 217 |

***Design for Specific Constraints***

|  |     |
|--|-----|
| Integration of Robustness in the Design of a Cell<br><i>J.M. Dutertre, F.M. Roche, G. Cathebras</i>  | 229 |
| Impact of Technology Spreading on MEMS design Robustness<br><i>V. Berouille, L. Latorre, M. Dardalhon, C. Oudea, G. Perez, F. Pressecq, P. Nouet</i> | 241 |

***Architectures***

|   |     |
|---|-----|
| A New Efficient VLSI Architecture for Full Search Block<br>Matching Motion Estimation<br><i>N. Roma, L. Sousa</i> | 253 |
| Design Considerations of a Low-Complexity, Low-Power Integer Turbo Decoder<br><i>S. M. Pisuk, P. H. Wu</i>        | 265 |

***Low Power, Low Voltage***

|  |     |
|--|-----|
| Low-Voltage Embedded-RAM Technology: Present and Future<br><i>K. Itoh, H. Mizuno</i>   | 277 |
| Low-Voltage 0,25 µm CMOS Improved Power Adaptive Issue Queue for Embedded Microprocessors<br><i>B. Curran, M. Gifaldi, J. Martin, A. Buyuktosunoglu, M. Margala, D. Albonesi</i> | 289 |
| Gate Sizing for Low Power Design<br><i>P. Maurine, N. Azemard, D. Auvergne</i>   | 301 |

***Timing Issues***

|   |     |
|---|-----|
| Modeling and Design of Asynchronous Priority Arbiters for On-Chip Communication Systems<br><i>J-B. Rigaud, J. Quartana, L. Fesquet, M. Renaudin</i> | 313 |
| Feasible Delay Bound Definition<br><i>N. Azemard, M. Aline, P. Maurine, D. Auvergne</i>   | 325 |

### ***Advance in Mixed Signal***

|   |     |
|---|-----|
| CMOS Mixed-signal Circuits Design on a Digital Array Using Minimum Transistors<br><i>J. H. Choi, S. Bampi</i> | 337 |
|---|-----|

|   |     |
|---|-----|
| A VHDL-AMS Case Study: The Incremental Design of an Efficient 3 <sup>rd</sup> Generation MOS Model of a Deep Sub Micron Transistor<br><i>C. Lallement, F. Pêcheux, Y. Hervé</i> | 349 |
|---|-----|

### ***Verification & Validation***

|  |     |
|--|-----|
| Speeding Up Verification of RTL Designs by Computing One-to-one Abstractions with Reduced Signal Widths<br><i>P. Johannsen, R. Drechsler</i> | 361 |
|--|-----|

|   |     |
|---|-----|
| Functional Test Generation using Constraint Logic Programming<br><i>Z. Zeng, M. Ciesielski, B. Rouzeyre</i> | 375 |
|---|-----|

### ***Test***

|   |     |
|---|-----|
| An Industrial Approach to Core-Based System Chip Testing<br><i>E. J. Marinissen</i> | 389 |
|---|-----|

|   |     |
|---|-----|
| Power-Constrained Test Scheduling for SoCs Under a “no session” Scheme<br><i>M-L. Flottes, J. Pouget, B. Rouzeyre</i> | 401 |
|---|-----|

|  |     |
|--|-----|
| Random Adjacent Sequences: An Efficient Solution for Logic BIST<br><i>R. David, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel</i> | 413 |
|--|-----|

|  |     |
|--|-----|
| On-chip Generator of a Saw-Tooth Test Stimulus for ADC BIST<br><i>F. Azaïs, S. Bernard, Y. Bertrand, M. Renovell</i> | 425 |
|--|-----|

|   |     |
|---|-----|
| Built-in Test of Analog Non-Linear Circuits in a SOC Environment<br><i>L. Carro, A. C. Nácul, D. Janner, M. Lubaszewski</i> | 437 |
|---|-----|

### ***Sensors***

|  |     |
|--|-----|
| Design of a Fast CMOS APS Imager for High Speed Laser Detections<br><i>B. Casadei, J. P. Le Normand, Y. Hu, B. Cunin</i> | 449 |
|--|-----|

|   |     |
|---|-----|
| Noise optimisation of a piezoresistive CMOS MEMS for magnetic field sensing<br><i>V. Berouille, Y. Bertrand, L. Latorre, P. Nouet</i> | 461 |
|---|-----|

|               |     |
|---------------|-----|
| Authors Index | 473 |
|---------------|-----|

|                |     |
|----------------|-----|
| Keywords Index | 475 |
|----------------|-----|

## Preface

The 11th IFIP International Conference on Very Large Scale Integration, in Montpellier, France, December 3-5, 2001, was a great success. The main focus was about IP Cores, Circuits and System Designs & Applications as well as SOC Design Methods and CAD. This book contains the best papers (39 among 70) that have been presented during the conference. Those papers deal with all aspects of importance for the design of the current and future integrated systems.

System on Chip (SOC) design is today a big challenge for designers, as a SOC may contain very different blocks, such as microcontrollers, DSPs, memories including embedded DRAM, analog, FPGA, RF front-ends for wireless communications and integrated sensors. The complete design of such chips, in very deep submicron technologies down to 0.13 mm, with several hundreds of millions of transistors, supplied at less than 1 Volt, is a very challenging task if design, verification, debug and industrial test are considered.

The microelectronic revolution is fascinating; 55 years ago, in late 1947, the transistor was invented, and everybody knows that it was by William Shockley, John Bardeen and Walter H. Brattain, Bell Telephone Laboratories, which received the Nobel Prize in Physics in 1956. Probably, everybody thinks that it was recognized immediately as a major invention. Not at all!. When Bell Telephone Laboratories announced the invention of the transistor on June 30, 1948, six months later, the general press was almost indifferently. The New-York Times carried the news on the next to the last page of the paper, with four short paragraphs. Even technical journals were slow to appreciate the inherent possibilities of the transistor.

To stir enthusiasm for the device, Bell Laboratories licensed it freely in US and publicized it extensively in seminars and papers. A free license for the transistor! This means that nobody had understood what was such a device! Its direct concurrent was the vacuum tube, a strongly established commercial product. Engineers did not like transistors; they preferred tubes. The first market pull came from the hearing aids market, for which miniaturization was a must. The first transistorized hearing aid was announced by Sonotone in February 1953; it contained 5 transistors, but still required a pair of miniaturized tubes for the input and driver stages.

How the transistor was invented? Was such a device really needed? The answer is simple; the transistor had its origin in scientific theory rather than in technological developments. From the beginning of the last century, more and more studies have been performed on solid-state physics, metals and semiconductors. In 1935, a patent was issued to O. Heil for a field effect triode, although he was not able to explain how it worked. It is ironic that the concept of field effect transistors, so marvelously simple, provides practical implementations after the invention of the far more complex bipolar transistor. After the War World II, which interrupted many studies in semiconductor materials, new research programs have been decided. Bell Labs had such a program.

The Bell Labs group, including the three inventors of the transistor, decided to limit their research to germanium and silicium, the most simple semiconductors. The group believed that the only explanation for the continued lack of understanding of semiconductors - despite intensive worldwide research - was the diffuse experimentation on so many different complex materials. Silicon and germanium, on the other hand, were elemental, simple; moreover, their atomic structure had revealed the same strong covalent bounding as a diamond, and thus their crystals tended to be striking free of defects. This is a main point to realize working devices; keep them as simple as possible.

In 1958, it was the invention of the integrated circuit. So pressing was the need for fabricating entire circuits in a single semiconductor block that had neither Jack Kilby nor Robert Noyce conceived the integrated circuit in 1958, someone else surely would have done! The device was qualified "as the most significant development by Texas Instrument since ... the commercial silicon transistor". Also in 1958, the first field-effect transistor was working. It was called "Tecnitron" by its creator, S. Teszner, working in France. In 1962, RCA was fabricating multipurpose logic block comprising 16 MOS FETs on a single chip. By 1963, RCA had fabricated large arrays of

several hundred MOS devices. They were however extremely sensitive to static charge, supply voltage was higher than those of bipolar transistors, and the speed was slower. Production was also plagued with oxide defects. In mid-1965, only two companies were producing MOS ICs - General Microelectronics and General Instruments - the other companies took a wait and see stance.

Everybody knows as the story continues: MOS technology, the first microprocessor in 1971, CISC machines, RISC microprocessors in 1981, superscalar and VLIW machines today, with a shift in CMOS technology in 1984-1985 with the 80386 and 68020. Today, the SIA Roadmap predicts in 2014 a 0.035 mm CMOS process (probably SOI) with 19 billions of transistors for high-performance microprocessors, 0.6 Volt, 180 watts and more than 10 GHz as local frequency. There are today more transistors in the world (1017) than ants (1016).

However, what is the future of microelectronics? Are we close to the end of this marvelous story? Is the future belonging to nanotechnologies that could completely replace microelectronics (although 0.035 mm transistors have a 35 nanometers length)? Nano-devices have been constructed, capable of switching a current or single electrons with a ratio between the on/off current of 1 thousand to 1 million. Such elements could be promising as their sizes of some nanometers and extremely low or no power consumption are very attractive. Carbon nanotubes, quantum dots, single electron devices or molecular switches are the most promising nano-devices. For instance, a carbon nanotube has a diameter of 1 nanometer, and depending on its diameter, is a semiconductor device (otherwise, it is a conductor, not usable as a switch). However, if one over 10 nanotubes is semiconductor, how to select and interconnect the semiconductor ones to provide a useful logic function?

Quantum dots are based on the Coulomb blockade effect and electrons are moved one by one from dots to dots. They have been constructed atom by atom by atomic force microscope. Due to noise, it is better to construct cellular automata with several dots, and to define a given state of the automat as the logic “0” and another state as “1”. Majority gates have been demonstrated, as well as AND/OR gates. The main problem is still how to interconnect these gates to provide useful functions. Furthermore, it is hard to construct atom by atom a complete chip with several billions of elements.

Design methods could be completely different from today, as nano-devices could be constructed randomly, without any predefined schematic or

layout. However, a useful function could emerge from this huge number of nano-devices, or some auto-organization could occur. It is a little bit similar to natural selection for which only the useful functions will survive. But it will be hard to design a predefined and very complex function like a Pentium microprocessor.

It is very hard to make predictions, especially for the future (Mark Twain). However, the most probable future is that microelectronics will be used until perhaps 2020. Then it will be not replaced by nanoelectronics, but both technologies, i.e. microelectronics and nanoelectronics will co-exist with probably different applications.

It means that there are many opportunities for next editions of IFIP International Conferences on Very Large Scale Integration dedicated to microelectronics. We thank all people attending to the Montpellier edition and encourage everybody to attend the next edition in 2003 in Germany.

Prof. Christian Piguet  
CSEM & LAP-EPFL

Prof. Bruno Rouzeyre  
LIRMM

Dr. Marie-Lise Flottes  
LIRMM

Prof. Michel Robert  
LIRMM

## **Conference Committees**

### **General Chair**

Michel Robert, LIRMM, France

### **Program Co-Chairs**

Christian Piguet, CSEM, Suisse

Bruno Rouzeyre, LIRMM, France

### **Local Organization Chair**

Marie-Lise Flottes, LIRMM, France

***Technical Program Committee***

|                          |  |
|--------------------------|--|
| Nadine Azemard,          | LIRMM, France                                      |
| Dominique Borrione,      | TIMA, France                                       |
| Maciej Ciesielski,       | University of Massachusetts, USA                   |
| Luc Claesen,             | LCI-SMARTpen N.V. & K.U.Leuven, Belgium            |
| Claudionor Coelho,       | UFMG, Brazil                                       |
| Karl heins Diener,       | FhG IIS/EAS, Germany                               |
| Rolf Drechsler,          | Siemens, Germany                                   |
| Nikil Dutt, Univ.        | California Irvine, USA                             |
| Hans Eveking,            | Darmstadt Univ. of Technology, Germany             |
| Joan Figueras,           | UPC, Spain   |
| Marie-lise Flottes,      | LIRMM, France                                      |
| Masahiro Fujita,         | University of Tokyo, Japan                         |
| Patrick Girard,          | LIRMM, France                                      |
| Manfred Glesner,         | University Darmstadt, Germany                      |
| Reiner Hartenstein,      | University of Kaiserslautern, Germany              |
| John Hayes,              | University of Michigan, USA                        |
| Sybille Hellebrand,      | University of Innsbruck, Austria                   |
| Jose Luis Huertas,       | Univ. Sevilla, Spain                               |
| Heinz Hugli,             | University of Neuchâtel, Switzerland               |
| Paolo Jenne,             | Swiss Federal Institute of Technology, Switzerland |
| Andre Ivanov,            | University of British Columbia, Canada             |
| Ahmed Jerraya,           | TIMA, France                                       |
| Paul Jespers,            | Université Catholique de Louvain, Belgium          |
| Lech Jozwiak Eindhoven,  | University of Technology, The Netherlands          |
| Luciano Lavagno,         | University of Udine, Italy                         |
| Wolfgang Nebel,          | OFFIS, Germany                                     |
| Pascal Nouet,            | LIRMM, France                                      |
| Irith Pomeranz,          | Purdue University, U.S.A.                          |
| Paolo Prinetto,          | Politecnico di Torino, Italy                       |
| Ricardo Reis,            | UFRGS, Brazil                                      |
| Tsutomu Sasao Kyushu,    | Institute of Technology, Japan                     |
| Donatella Sciuto,        | Politecnico di Milano, Italy                       |
| L. Miguel Silveira,      | Technical University of Lisbon, Portugal           |
| Lars Svensson,           | SwitchCore AB, Sweden                              |
| Paulo Teixeira,          | IST - INESC-id, Portugal                           |
| Flavio Wagner,           | UFRGS, Brazil                                      |
| Neil Weste,              | Radiata Communications, Australia                  |
| Tom Williams,            | Synopsys, USA                                      |
| Hans-Joachim Wunderlich, | University of Stuttgart, Germany                   |

*Organizing Committee*

|                               |   |
|-------------------------------|---|
| <b>Publicity:</b>             | N. Azémard-Crestani<br>P. Maurine                     |
| <b>Sponsors:</b>              | M. Robert<br>D. Auvergne                              |
| <b>Plenary Sessions:</b>      | P. Girard<br>M. Renovell                              |
| <b>Finance:</b>               | L. Torres<br>D. Deschacht                             |
| <b>Registration:</b>          | G. Cathebrat<br>L. Latorre<br>V. Beroulle             |
| <b>Proceedings &amp; CD:</b>  | S. Bernard<br>Y. Bertrand<br>L. Latorre<br>A. Virazel |
| <b>Social:</b>                | C. Landrault<br>Y. Bonhomme<br>P. Faure               |
| <b>Technical Support:</b>     | J. Galy<br>B. Caillard<br>J.M. Dutertre<br>J. Pouget  |
| <b>Technical Arrangement:</b> | G. Cambon<br>I. Vogel                                 |
| <b>E-submission:</b>          | P. Nouet<br>G. Sassatelli                             |
| <b>Web:</b>                   | F. Azaïs<br>D. Navarro                                |
| <b>Local Arrangement:</b>     | M. Comte<br>X. Michel<br>V. Rahajandraibe             |

*Conference Secretariat*

Céline Berger  
LIRMM / Université de Montpellier  
161 rue Ada 34392 Cedex 5 Montpellier, France  
<http://www.lirmm.fr>

***Post-conference Volume Coordination***

Laurent Latorre, LIRMM, France

***Acknowledgements***

This book is the result of the work of many dedicated volunteers: session organizers and moderators, invited lecturers, authors of papers and sponsors.

We thank them all for their contribution and particularly: D. Auvergne, F. Azaïs, G. Cambon, G. Cathebras, D. Deschacht, J. Galy, P. Georgelin, M. Johann, P. Kalla, A. Krasniewski, C. Landrault, P. Lepinay, A. Mischenko, M. Nahvi, J-L. Paillet, L. Pierre, L. Pozzi, G. Schmacher, E. Sicard, L. Torres, N. Van Der Meijs, M. Velev and Z. Zeng who helped in reviewing the papers.

On behalf of the program committee we thank the LIRMM laboratory (Laboratoire d'Informatique de Robotique et de Microelectronique de Montpellier) from the University of Montpellier and the CNRS (French National Scientific Research Center).

---

# **SOC DESIGN METHODOLOGIES**

# Two ASIC for Low and Middle Levels of Real Time Image Processing

P. Lamaty\*, B. Mazar\*, D. Demigny\*\*, L. Kessal\*\*, M. Karabernou\*\*

\* Aérospatiale-Matra-Missiles, Département électronique, 18 rue Le Brix, 18000 Bourges cedex, [philippe.lamaty@aeromatra.com](mailto:philippe.lamaty@aeromatra.com)

\*\* ETIS, UPRESA CNRS 8051, ENSEA et Université de Cergy Pontoise, ENSEA, 6 av. du Ponceau, 95014 Cergy Pontoise cedex, [demigny@ensea.fr](mailto:demigny@ensea.fr)

**Abstract:** This paper presents the system architecture (with main details on each algorithm) of two ASIC for real time image processing designed with Aérospatiale-Matra industry. The first chip: OREC is dedicated to low level processing (edge detection) and includes a large 2D convolution filter 12x12, gradient computation, extraction of the local maxima of the gradient and thresholding. The second chip: OPNI is dedicated to intermediate level image processing. Processes or IP for edge thinning, region labeling, edge chaining (line segment extraction, line segment chaining, polygonal approximation and little chains elimination) are included in OPNI as well as a DSP core and a mathematical coprocessor based on Cordic method for trigonometric computations. Both VLSI chips have been successfully tested. They are used in a European project : obstacle detection for vehicule. The maximum frame rate reaches 25 images per second for 1024x1024 image size, and more than 110 images per second for 233x256 image size.

**Key words:** asic, image processing, edge detection, region labelling, polygonal approximation, edge chaining

## **1. INTRODUCTION**

### **1.1 Purpose**

Low and intermediate real time image processing is still a challenge for embedded system. Common multi-DSP programmed architectures are ineffective considering size and consumption criteria. If the most powerful DSP satisfy the real time constraint for one algorithm, they are not enough efficient to support the whole processes required by an application. Beyond the computation architecture, real time image processing requires high memory bandwidth. Wired architectures associated with dedicated memory organization remain the best solution. The global function of the wired architecture is the reduction of the data bandwidth which are transferred to the high level processing stage commonly build with programmable architectures.

From the collaboration between the vision and electronic departments of Aérospatiale-Matra industry and the ETIS research lab, it results the design of two VLSI chips which offer a complete solution for low and intermediate image processing.

The first chip: OREC (see section 2) is dedicated to low level processing (edge detection). It includes a large 2D convolution filter (12x12), gradient computation, extraction of the local maxima of the gradient, and thresholding of the gradient maxima.

The second circuit: OPNI (see section 3) is dedicated to intermediate level image processing. Processes or IP for edge thinning, edge closing, region labeling, edge chaining (line segment extraction, line segment chaining, polygonal approximation, elimination of little chains), corner detection, are included as well as a DSP core and a mathematical coprocessor based on Cordic method for trigonometric computations.

All the functional block have been designed as IP. This allows an easy reuse of each part of the design.

### **1.2 Constraints**

Behind the high data rate inherent to real time image processing, two main difficulties arise in this project.

- In the application planned by Aérospatiale, the image processing flow is used to control actuators which modify the camera position and then the visual scene (feedback loop). Then the latency of the computation has a strong effect on the system stability. This induces that no frame buffer can be used in front of and inside the treatments. Data flow computations must be done on the fly, when pixels are extracted (at the video rate)

from the camera. A second involvement is that the latency constraints reduce the choice of the available algorithms that can be used on each step as it will be discussed in the rest of the paper.

- The second difficulty is that all the algorithms and IP architectures must be consistent to lead to an efficient system (data format, exchange, interfaces, timing).

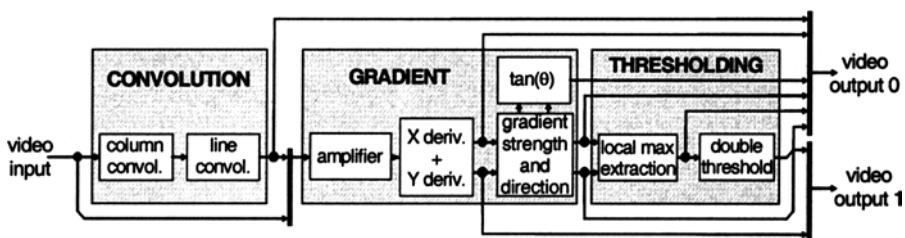
### 1.3 Design methodology

The major aspect is that architect and hardware specialists have been associated to the image processing team in the early steps of the project. So they could contribute to the algorithms choice in the way of a feasible solution and they could help the discussion about how to preserve the quality for the lowest complexity. The algorithms have been first written in C language with bit true computation. Then VHDL was used for digital design. Real images were used for the simulation tests. Results in C and VHDL implementation have been compared to ensure the functional quality of the design.

## 2. THE OREC CHIP

### 2.1 Main functions

The OREC chip is dedicated to edge detection. The figure 1 shows the different blocks and the main data flows of this chip.



*Figure 1.* OREC functionality. Two output video buses allow to extract the results of different blocks (ex. edges and gradient orientation  $\tan(\theta)$  for a fast Hough transform)

Edge are detected has the local maxima of the gradient of the intensity pixel value. In a recent work [4], we show that gradient computation can be made with a separable 2D smoothing filter (which can be computed with two 1D filters) associated to a simple 2x2 Roberts local derivating filter (for the first derivative computation). This scheme reduces the computation cost by a

factor two compared to more classical approaches. An other advantage is that the resolution of the detection is only fixed by the choice of the smoothing coefficients. The gradient and threshold parts remain unchanged when the resolution varies.

The video input are 8 bits width and the two pixel synchronized video output are 16 bits width. All the memory required for the treatment are included in the chip. Two synchronization standards (Imaging and DataCube) are allowed. All the parameters (synchronization, filter and amplification coefficients, threshold levels, output sources) can be changed on the fly. The maximum image size is 1024x1024, and the maximum pixel rate is more than 20Mhz with military constraints. The chip was realized in a CMOS 0.6  $\mu\text{m}$  technology process. It uses 100K equivalent gates and 160 Kbits of memory and includes boundary scan, full scan et bistram tests.

## **2.2 Smoothing filter**

Recursive filters as the Deriche filter [2] are often used because they offer the best noise reduction for a given computation cost. Unfortunately vertical recursivity in image processing induces a frame delay which is not compatible with our latency constraint (a few lines delay). Then, we choose to use a large kernel FIR 2D separable smoothing filter computed with two 1D convolvers. Each of them implements a symmetrical FIR filter with 12 coefficients which requires 6 multipliers and 11 adders. To avoid transient responses near the image boundaries, a mirror effect is generated on the "shadow pixels" near each horizontal or vertical image frontiers. This is controlled with a simple state machine. In order to limit the number and the size of the line buffers (remember that no frame buffer exists between the camera and the chip), the vertical convolver is placed in front of the horizontal one. To preserve the detection quality, the smoothing result is conserved without data truncation.

## **2.3 Gradient computation**

The amplifier is needed to ensure that the gradient strength will be independent on the resolution. So the same threshold can be used for all the resolutions. Small size Roberts derivative filters are used to compute the  $x$  and  $y$  components of the gradient. The following equation gives the expression of the horizontal gradient component  $G_x$  where  $s_{i,j}$  is the smoothed output (column  $i$ , line  $j$ ).

$$G_x = (s_{i+1,j} + s_{i+1,j+1}) - (s_{i,j} + s_{i,j+1}) \quad (1)$$

Inverting  $i$  and  $j$  leads to the vertical component  $G_y$  expression. Instead of using expensive square root and square computation, the gradient strength is approximated with:

$$G = \max \left[ |G_x|, |G_y|, \frac{3}{4}(|G_x| + |G_y|) \right] \quad (2)$$

which has a 6 % precision and only requires a Max computation, two incrementers and two adders. This is sufficient because the local maxima detection uses relative gradient weight and because the sensibility of the threshold is not too high. The gradient direction is computed with a  $22.5^\circ$  precision for the local maxima extraction. Furthermore a divider is used to compute  $\tan(\theta) = G_y/G_x$  or its inverse with a 8 bit precision. This allows the implementation of a fast Hough transform. The gradient orientation is used to select the location of the vote in the Hough space and considerably reduces the memory bandwidth compared to the classical implementation. Only one line buffer is needed for gradient computation.

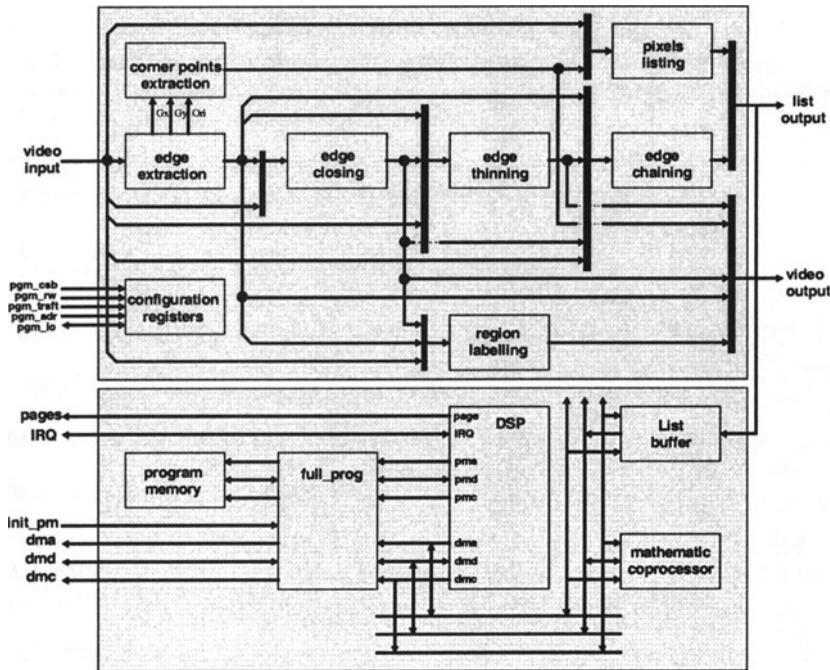
## 2.4 Thresholding

The local maxima extraction operator selects the highest gradient value in the gradient direction. the gradient direction is used to select involved pixels in a  $3 \times 3$  neighborhood. Balancing (function of the gradient direction) of the gradient strength of involved pixels is made to improve the detection. Comparisons are then computed with the gradient of the centered pixel. Then, a double thresholding of the gradient of the retained pixels is applied. A high level threshold selects the pixels which are surely edges. A low level threshold selects doubtful pixels which can help to close the contour lines on the OPNI chip.

## 3. THE OPNI CHIP

### 3.1 Main functions

The OPNI chip (see figure 2) extracts features from an input video with dedicated hardware operators localized on the higher part of the figure 2.



**Figure 2.** OPNI fonctionnalité. Two buses one for video frame an the other for data list extracted the results of the different IP. The lower part of the figure shows the DSP and its mathematical coprocessor which can be used for features processing

These features are corner points, vector extremity of polygonalized contour lines, and region labels. The two first ones are reorganized to form list of pixels which allows a more compress form than the video flow where a lot of pixels are useless. The second part is based on a DSP core associated to a mathematical coprocessor. They are used to implement high level image processing algorithms which operate on feature lists. The OPNI can be used without preprocessing when images are sufficiently noiseless. In this case, the edge extractor is used on the video input. The video output is 10 bits wide. The list output is 32 bits wide which allows to extract in parallel the pixel coordinates and their characteristics. The maximum number of detected segments is  $2^{18}$  and the maximum number of retained labelled regions:  $2^{10}$ . As for the OREC chip, two synchronization standards (Imaging and DataCube) are allowed. All the parameters can be changed on the fly. The maximum image size is  $1024 \times 1024$ , and the maximum pixel rate is more than 20Mhz with military constraints. The chip was realized in a CMOS  $0.35 \mu\text{m}$  technology process. It uses 240K equivalent gates (DSP core not included) and 200 K bits of memory. It includes boundary scan, full scan et bistrom tests. Its size is  $71 \text{ mm}^2$ . Edge extraction is based on the same architecture as in the OREC chip, so it is not more discussed here.

### 3.2 Corner points extraction

The gradient operator (strength and orientation) is applied again to the gradient of the intensity image. Then, the local maxima of the second derivatives are extracted. A threshold operation retains the edge pixels which have a high curvature value. Finally a list of the selected pixel is built. The same hardware architecture as for edge detection is used in this block.

### 3.3 Edge closing

Edge closing is based on a cellular automaton (as the game of life)[1]. 64 elementary processors implement a four state cellular automaton. Initially, each pixel can be background, strong edge (gradient is more than high threshold), weak edge (gradient is between high and low thresholds) or extremity of a strong edge. So pixels are stored with 2 bits.

The automaton rules are simple and operate on a 3x3 neighborhood:

- each weak pixels which has at least two not background neighbors and which has an extremity neighbor is change to an extremity pixel,
- each weak or extremity pixels which has less than two not background neighbors is change to a background pixel.

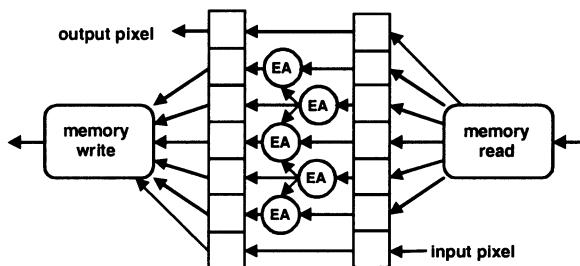


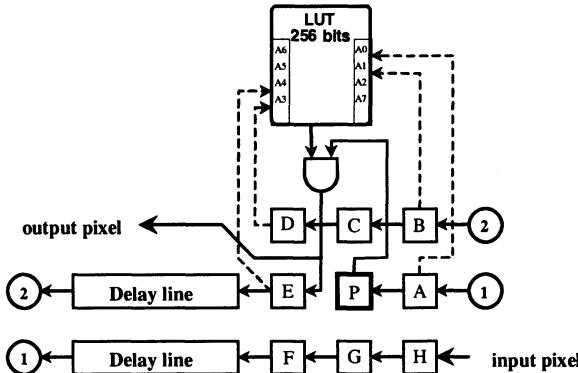
Figure 3. Organization of the closing edge operator. AE are automaton elements

The iteration of the automaton rules propagates along the weak edges a closing wave starting from the extremities. The waves which reach a strong edge are finally conserved. At the end, extremity pixels are changed to strong edge pixels, weak edge pixels are changed to background. To applied the automaton rules 32 times on each pixel, 33 lines (2 bits wide) are stored in FIFO. The figure 3 shows and example with 5 iterations and 6 stored lines. This operator scans the image in a classical video flow. At each cycle, a column of 32 pixels is read from the line memory, the current pixel of the same column is input, 32 pixels can change their state and will be stored in the line memory, one pixel is definitely output. This dataflow edge closing

operator can be implemented very simply and easily pipelined. Full implementation details can be found on the reference [1].

### 3.4 Edge thinning

Edge chaining requires that edges will be one pixel wide. Because various edge detectors (as morphological operators) can be used in front of the OPNI chip, we have implemented an edge thinning operator. A simple filter which operates on a  $3 \times 3$  neighborhood. Edge pixels are retained if and only if they are useful for edge connectivity. Results for each possible configuration of the neighborhood is stored in a look up table (see figure 3). The main advantage of this scheme is that it is a simple dataflow algorithm. The major drawback is that resulting edges are pushed in the scan direction. However, The misslocation is reduced when the edge thickness is small.



*Figure 4.* Details of the edge thinning operator. P is the processed pixel, its value is changed as a function of its neighborhood. The change is immediately taken into account in the context of the following pixel

### 3.5 Edge chaining and polygonalization

This part is composed of four successive blocks: segment extraction, segment chaining, polygonal approximation, little chains elimination.

Segment extraction is based on a simple automaton which works on a  $3 \times 3$  pixel neighborhood during video scanning. The informations about each segment detected are stored in a queue memory. The segment chaining links the connected segments and build a list of segments stored in memory with dynamic allocation. Each segment is then described by the coordinates of its extremities. This part is based on a state machine with 24 states which triggers memory pointers modification.

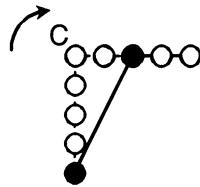


Figure 5. An Example of a bad approximation when corner points are not taken into account

The polygonal approximation [3] is made by an iterative approximation between corner pixels. This is needed in order to conserve the global shape of the features. The figure 5 shows the bad approximation result when the corner point C is not detected. So, corner pixels are detected has the vertex of two segments which strongly differ in direction. To do that, the distance (vertically and horizontally) between two successive vertex is computed. Successive distance results are combined and compared to a threshold in order to retain high curvature vertex.

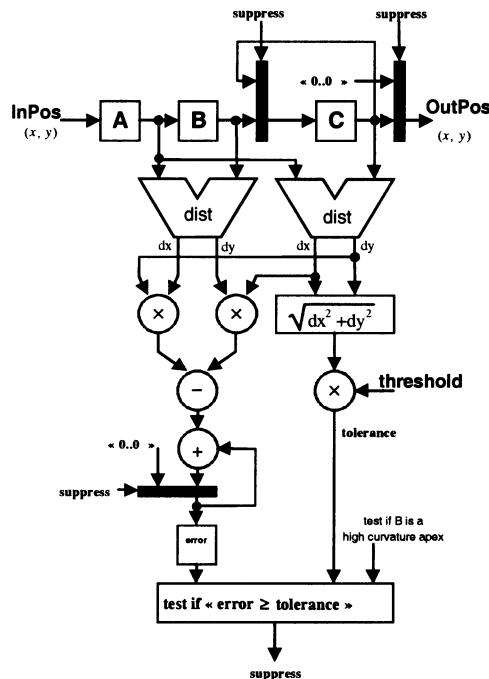


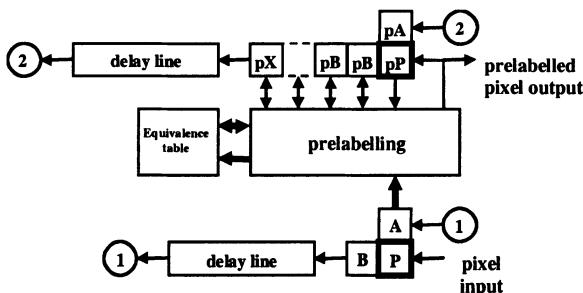
Figure 6. Iterative approximation for the edge polygonalization

The iterative approximation starts from a corner point or an extremity and includes (step by step) successive vertex. At each step, the approximation error is refreshed.

When the error becomes greater than a given tolerance, the approximation process of the present vector stops. Details of the iterative approximation are given in figure 6. Here, we try to suppress the point stored in register B. C is the memorized starting point of the approximation vector. The output of the subtractor is the area generated when B is suppressed. The following adder accumulate successive area errors. To take the decision, the error is relatively ponderated by a fraction (threshold) of the vector length computed with  $(dx^2 + dy^2)^{1/2}$ . B is finally suppressed if it is not a high curvature vertex and if the global area error is less than the tolerance. Little chain elimination is simply realized by a new distance computation between vertex. An other particularity of our implementation is the use of Freeman code in order to reduce the memory size.

### 3.6 Region labelling

This algorithm is based on the original algorithm from Rosenfeld [5]. It involves not only "pixel computation" but also global computation to solve equivalence between identical regions which first appeared as different because of the video scanning order. Most of the computation cost occurs when a edge is encountered but no work has to be done for "background" pixels. So, it is not a good idea to synchronize the algorithm on the pixel rate, because in this case, the computation cost cannot be distributed with regularity. To avoid this problem, we have introduced RLE compression and uncompression before and after the region labelling. Algorithms and architecture have been modified consequently.



*Figure 7. Architecture for the region labelling*

The architecture is based on two parts. The first part scans the edge image and build a fictitious prelabelled image. We store only the equivalence between region which have been initially supposed different

during the scan and the edges image. At the end of the image scan, a scan from low to high labels of the equivalence table computes a new table which contains final labels. Then the second part of the architecture is used to scan again the edge images but now, when a new region arises, the final label is find in the equivalence table. Compare to more classical scheme, this architecture does not need to store prelabelled image but only the edge image. The figure 7 gives the organization of the prelabelling scheme (first part). Here, the pixel P is computed. Two neighborhoods are used:

- at the bottom of the figure from the edge image, the pixel B at the left side of P and the pixel A at the up side of P,
- at the top of the figure, prelabelled values  $p_A$ ,  $p_B$ ,  $p_P$  of pixels A, P, B and of some of the predecessors of B.

Configuration (edge or background) of the pixels of the neighborhood decided that  $p_P$  is or is not a new region. Usually, only the pixels connected to P (A,B) are used. Here, we temporary store the label results of the last pixels (in the left of the top neighborhood). After a fixed delay the final decision is made for all the stored pixels. This scheme reduces by a factor of four the equivalence memory size.

The same architecture is used for the second part (final labelling) except that the prelabelled neighborhood is reduced to (A,B,P) and that the equivalence table is just read (not written).

## 4. CONCLUSION

Both circuits have been successfully tested and are used in different systems. The maximum frame rate reaches 25 images per second for 512x512 image size in a prototype designed by Aérospatiale-Matra. A PC card was also build to designed a prototype used in an european project (obstacle detection in front of cars). More than 110 images per second for 233x256 image size are computed by this PC card. More recently, all the IP of the OREC and OPNI have been included in a Xilinx Virtex II 3M gates FPGA. Sixty images of size 1280x1024 per second are computed by the FPGA. This implementation outperforms all the designs based on DSP architectures for speed, compactness and consumption criteria and prove the reusability of the IP designed in this project.

## 5. REFERENCES

- [1] J. Devars, D. Demigny, and J.F. Quesne. Boundary closing with asynchronous cellular automata. IEEE Conf. on Computer Architecture for Machine Perception, pages 81 – 88, 1991.
- [2] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. Int. J. Computer Vision, pages 167 – 187, 1987.
- [3] P. Garnesson, and G. Giraudon. Chaînage efficace de contours. Rapport de recherche INRIA, n° 1621, 1992.
- [4] F. Garcia Lorca, L. Kessal, and D. Demigny. Efficient ASIC and FPGA implementations of IIR filters for real time edge detection. Proc. International Conference on Image Processing, n° 2, pages 406 – 409, Santa Barbara, october 1997.
- [5] A. Rosenfeld, and J.L. Pfaltz. Sequential operations in digital picture processing. Journal of ACM, volume 13, pages 471 – 494, 1966.

# **64 x 64 Pixels General Purpose Digital Vision Chip**

Takashi Komuro and Masatoshi Ishikawa

*Department of Information Physics and Computing, Graduate School of Information Science and Engineering, The University of Tokyo, Japan (e-mail: kom@k2.t.u-tokyo.ac.jp)*

**Abstract:** Conventional image processing has a critical limit of the frame rate, which is a result of serial transmission of the video signal. In order to overcome this limit, a vision chip in which photo detectors and parallel processing elements are integrated together has been proposed. In this paper, the general purpose vision chip with digital processing elements and the 64x64 pixels prototype chip we developed will be described.

**Key words:** vision chip, image sensor, parallel processing, visual feedback

## **1. INTRODUCTION**

The expectation for real-time sensory information processing is increasing in the field of robot vision, intelligent transportation systems, high speed inspection, and so on. High speed visual feedback control is especially useful in performing advanced tasks in the real world. A feedback rate of over 1kHz makes maximum use of actuator performance.

However, conventional vision systems use serial transmission of video signals from an image sensor such as CCD to a processing device such as DSP. In such a system, the frame rate is limited by the video signal such as NTSC (33ms).

On the other hand, a device called a vision chip has been proposed, in which photo detectors (PDs) and processing elements (PEs) are connected in each pixel. This chip captures and processes images in pixel-parallel without serial transmission. Therefore, this chip achieves high speed visual feedback at over 1kHz. Also, by integrating a sensor and a processor together in one chip, we expect benefits of being a system VLSI such as compactness, low cost, and low power.

## 2. DIGITAL VISION CHIP

There has been considerable research on vision chips, but most of them use analog circuits in the processing element[1]. Analog circuits require less area and therefore are easier to integrate than digital ones. However, they have a drawback in that the functions are few and almost fixed once they are designed. To make a general purpose vision chip that can process various algorithms in a changing environment, it is necessary to develop a programmable processing element that can be controlled by software. Digital circuits are effective in accomplishing this purpose. We call such vision chips containing digital circuits "digital vision chips".

Considering the above mentioned requirements, Ishikawa et al. proposed an architecture for a digital vision chip[2]. Using a scaled-up model of the vision chip, various application systems have been developed[3-6]. These systems have achieved high speed visual feedback at a sampling period of 1ms and have achieved very high speed real-time control that can't be achieved in existing systems.

Research on digital vision chips has been carried out at a few other laboratories. In France, a compact digital vision chip called the programmable artificial retina has been proposed[7], and 65x76 pixels and 128x128 pixels chips have been developed. In Sweden a unique digital vision chip based on their near-sensor image processing concept was proposed and a 32x32 pixel chip has been developed[8].

One of the problems in designing a digital vision chip is that it is difficult to integrate many pixels on a single chip since the digital circuits requires more area than analog circuits.

To integrate a large number of pixels in one chip, we have to make an effort to design a compact PE whose area is as small as possible.

## 3. BASIC ARCHITECTURE

Following the design concept described above, Komuro et al. designed a new architecture S<sup>3</sup>PE (Simple and Smart Sensory Processing Element) [9] with the goal of further high integration. Figure 1 shows the architecture of S<sup>3</sup>PE.

Each PE has an ALU, which has bit-serial operation structure and 24 bit local memory in which each bit can be randomly accessed. It communicates with the photo detector and the neighbor PEs via an 8 channel I/O which is mapped to memory address space.

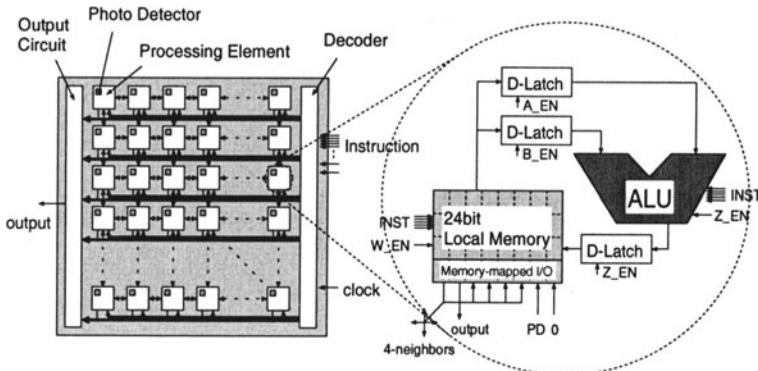


Figure 1. Basic Architecture of General Purpose Vision Chip

The ALU has a simple structure consisting of a full-adder, a carry register and some multiplexers. It can process one of 10 different kinds of logical and 8 kinds of arithmetic operations at each cycle. Bit-serial operation is used that the ALU performs each bit of multi-bit operations in order. This is slower than bit-parallel operation, but has the advantage of compact circuitry and the capability of variable bit length data operation

In the local memory, 24 bit random access memory (RAM) and 8 bit I/O ports are allocated to the same address space. I/O ports are connected to up, down, left and right PEs, input from the sensor, and zero signal. With the introduction of the memory mapped I/O, all processing, including not only load/store operations but also I/O operations are performed by accessing the local memory.

As a result, the instruction code has a simple form : all instructions consist of read address A, B (5 bits each), operation code (5 bits), and write address (5 bits). This 20 bit instruction is divided into four steps, and is transmitted and processed in order. This vision chip can process most of the early vision algorithms such as edge detection, smoothing, and convolution filtering in the order of a  $\mu$ s, which is much faster than conventional image processing systems and is fast enough for visual feedback control of a robot. Gray-scale image processing algorithms can also be implemented as well as binary ones since the PE has enough memory and arithmetic operation function. This is a big advantage to the other existing chips mentioned above. Table1 shows the steps and processing time. The cycle time per instruction is supposed to be 100ns. Practically, to achieve this speed, a controller which can transmit control signals to the vision chip at very high speed is needed.

*Table 1.* Steps and Processing Time of Early Vision Algorithms

| Algorithm                 | steps | time  |
|---------------------------|-------|-------|
| Edge Detection (binary)   | 8     | 0.8μs |
| Smoothing (binary)        | 14    | 1.4μs |
| Edge Detection (6bit)     | 47    | 4.7μs |
| Smoothing (6bit)          | 41    | 4.1μs |
| Thinning (binary) *       | 12    | 1.2μs |
| Convolution (6bit)        | 986   | 99μs  |
| Poisson Equation (6bit) * | 65    | 6.5μs |

\* repeating operation

## 4. BASIC ARCHITECTURE

In addition to the effort in designing the compact architecture, the rapid progress of semiconductor integration technology in recent years has enabled development of a general purpose digital vision chip with a large number of pixels. Our first goal is to develop a general purpose vision chip with 64x64 pixels, which is considered to be the minimum requirement for many industrial applications.

Based on the designed architecture, we have developed some test chips with 8x8 pixels[9] and 16x16 pixels[10]. Also we developed a 64x64 pixels chip but it has some defects in memory access and photo detector sensitivity.

Then, we re-designed the circuit. We made some modifications as described below and realized further integration.

### Global wiring of the control line

In a multi-layer process, the area taken up by wires is smaller than that of transistors. In this design, the decoders of control signal and address signals are taken out of the PE. All signals are transmitted to the PE after they are decoded. Those global wires are placed not in the exclusive area but over the circuits.

### Single bit line SRAM

Usually, an SRAM has two bit lines - bit signal and its compliment. Writing data is performed by giving an input signal and its complement respectively to the bit lines, and reading data is performed by taking a differential of two bit lines using a sense amp. This structure is adopted mainly in order to solve the problem of speed. Many SRAM elements are connected to a bit line and the parasitic capacitance of the line is so large that an SRAM element can't rapidly reverse the value.

However, in the vision chip, area has priority over speed. Also only four SRAMs are connected to a bit line. Moreover, the power consumption in the sense amp has been an obstacle against further integration. Therefore, we adopted single line SRAM.

Consequently, the SRAM elements are small, the row-select multiplexers of the memory are half size, and the sense amp is removed.

In former designs, the AD conversion in the PD is performed by counting time until output of an inverter is reversed. In this design, the inverter is replaced by a comparator and the voltage is compared to the reference voltage  $V_{th}$ .

The circuit using the inverter is smaller than that using the comparator, but there is a problem in that direct-current flows at the time around the inverter's reversal and the total power consumption is enormous. The circuit using the comparator has an advantage in that power consumption is reduced considerably and the threshold voltage is changeable in AD conversion. Figure 2 shows the photo detector circuit and the method of AD conversion. The slope of the voltage of the comparator input is steep in proportion to the light intensity and the time until it reaches the threshold is short.

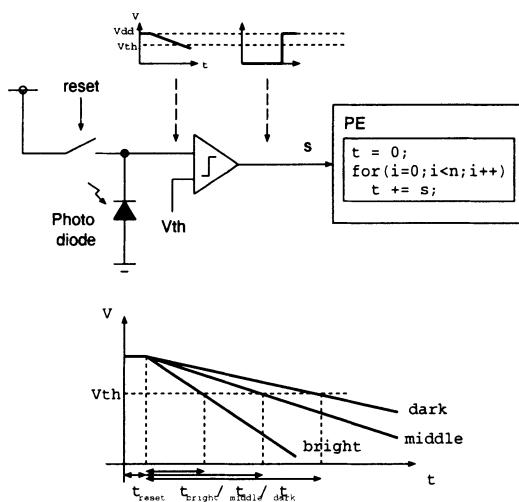
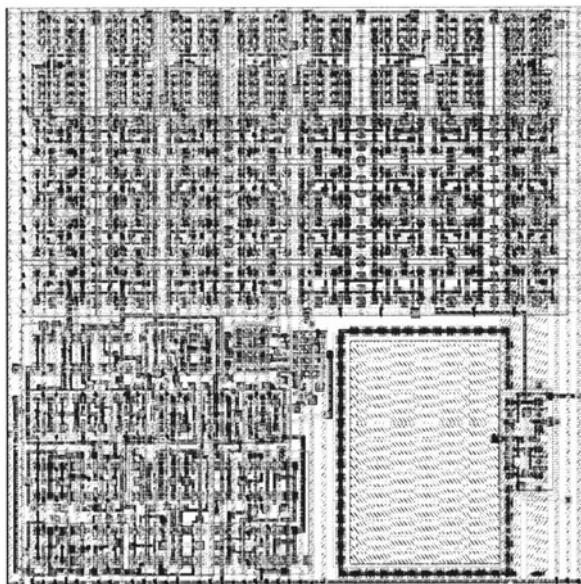


Figure 2. Photo Detector and AD Conversion

In addition to these modifications, some new functions including global feature extraction are introduced in this design but they are not described in this paper.

The layout of the chip has 64x64 pixels in a 5.4mm x 5.4mm area using 0.35 $\mu$ m TLM CMOS process. The number of transistors per PE is about 400 and the PE area is as small as 67.4 $\mu$ m x 67.4 $\mu$ m. This means that 256x256 pixels can be integrated in about a 1.8cm x 1.8cm chip and the standard pixel number as an image processing device is achievable. Figure 3 shows the layout of the pixel. It is confirmed that the chip works at least with the control speed of 10MHz.



*Figure 3. Layout of the Pixel*

At that control speed, the cycle time per instruction is 2.2 $\mu$ s. It seems to be slow but the potential performance of the vision chip is much higher. At present the working speed is limited by the performance of the controller which transmit control signals to the vision chip. With faster controller the working speed will be greatly improved. Now we are developing such a controller.

## 5. BASIC ARCHITECTURE

Using this design, we have developed a prototype chip of 64x64 pixels. Figure 4 shows a photograph of the chip. This chip is fabricated via CMP in France.

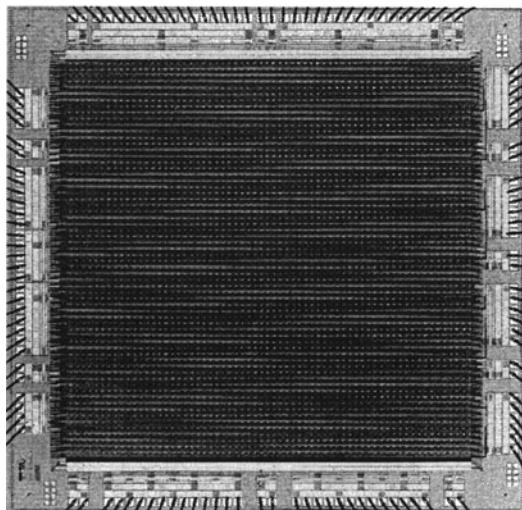
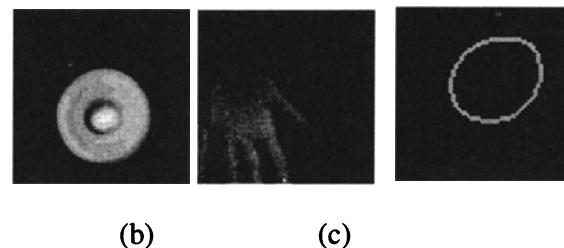


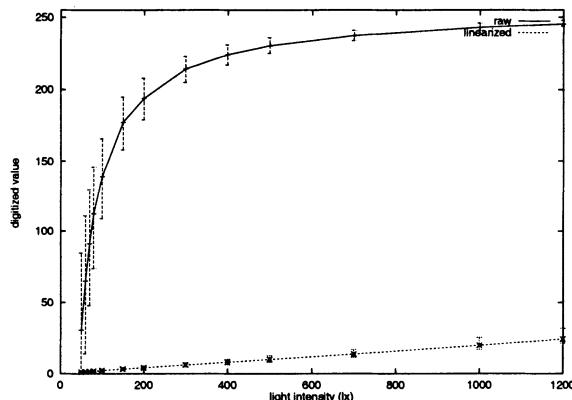
Figure 4. Photograph of General Purpose Vision Chip

Using this chip, experiments in image capturing and some processing with a lens attached were performed. The result is shown in Figure 5. In (a) the chip is irradiated directly by a spherical light source. In (b) a human hand was irradiated. The images were captured in 4bit gray-scale. In (c) an image of the same light was binarized and edge-detected. The integration time is 2.8ms. The threshold voltage of the AD conversion  $V_{th}$  is 1.5V in (a) and (c), and 3.3V in (b). The higher the  $V_{th}$  is, the higher the sensitivity of the PD becomes but the lower the quality of the image (S/N) becomes.



*Figure 5. Sample Output*

We did an experiment to evaluate the performance of the sensor. The chip surface was flatly irradiated by an LED array lighting without a lens. Figure 6 shows the graph of maximum, average, and minimum value in a screen when the image was captured in 8bit gray-scale and  $V_{th}$  was set to 2.4V. The integration time of the PD was 4.6ms.



*Figure 6. Sensor Characteristics*

The graph shows that the average of output is a smooth function of light intensity. Owing to the AD conversion method of time counting, the output value is not linear with light intensity. But the output, which is linearized using the conversion equation  $p' = p_{max}/(p_{max} - p)$ , shows good linearity. From the difference between the maximum (or minimum) and average, effective resolution is about 3bit. The main reason for the variation at the same intensity is fixed pattern noise (FPN) owing to the variation of transistor characteristic. As FPN appears as the offset of voltage, the effect is strong at low values when it is transferred to the time scale by AD conversion.

To evaluate the quality of image quantitatively, we calculated the root-mean-square (RMS) noise using the expression below. Figure 7 shows the graph of the RMS noise.

$$RMS = \sqrt{\frac{\sum(p_i - \hat{p})^2}{\sum p_i^2}} \times 100[\%] \quad (1)$$

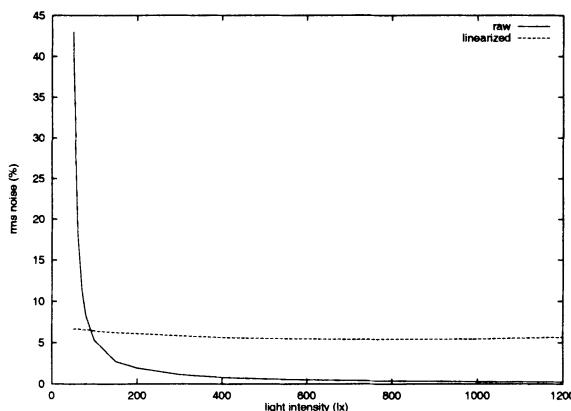


Figure 7. RMS Noise of the Sensor

This graph confirms that the FPN's effect is strong at low values. When the scale is linearized, the FPN's effect is almost flat at any value.

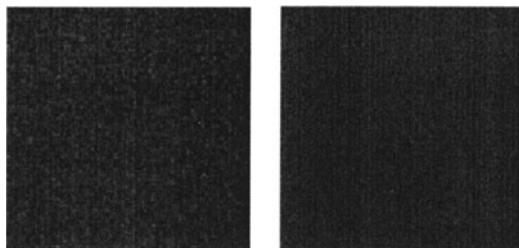
From the result of these experiments, the sensitivity and resolution are not particularly good, but still can be used in some applications.

It is proved that the AD conversion method used in the PD is weak against FPN as the voltage is transferred to time.

Overcoming the quality-of-image degradation of FPN, may be accomplished with the introduction of a good PD circuit with pixel-level AD conversion such as Yang et al. has proposed[11].

In this time, however, we tried to reduce the noise only by software. The program to download the offset and correct the input value by the offset is implemented. As mentioned above, the FPN's effect is strong at low values. It is proved theoretically and practically that the effect is proportional to the reversed input value. So the program calculates the product of the reversed input value and the offset and then subtract it from the input value. All of these calculations are processed in the vision chip.

Figure 8 shows the result of the software noise reduction. The chip was flatly irradiated by an LED array lighting without a lens. The image was captured in 8bit gray-scale and V<sub>th</sub> was set to 2.4V. The integration time of the PD was 4.6ms. The offset was made from the output of 70lx irradiation. The images in Figure 8 is the output of 100lx irradiation.



*Figure 8. The Result of Software Noise Reduction  
(left: before correction, right: after correction)*

Figure 8 shows the result of the software noise reduction. The chip was flatly irradiated by an LED array lighting without a lens. The image was captured in 8bit gray-scale and V<sub>th</sub> was set to 2.4V. The integration time of the PD was 4.6ms. The offset was made from the output of 70lx irradiation. The images in Figure 8 is the output of 100lx irradiation.

## 6. FURTHER INTEGRATION

Semiconductor technology has been progressed rapidly in recent years and it is expected that it will continue to progress in the future. It is estimated in the technology roadmap that the gate length of the transistor will decrease by one-half in five years and the number of transistors that can be integrated in a unit area will increase. Also chip size tends to increase and the number of transistors that can be integrated in one chip will go to the order of Giga. In the future it is expected that Mega-pixel vision chips can be easily realized.

However, as the semiconductor process goes to a smaller scale, some design problems will appear. The problems in the case of further integration are examined in this section.

Concerning the effect on logic circuits, the existing design as it is can be shrunk with a process scale. For example, the circuit designed on the basis of 0.8μm process can be shrunk in 0.35μm process to  $(0.35/0.8)^2$ , or about 1/5th the area. Moreover, with increases in routing layers, more integration can be realized by wire globalization as described in section 4. According to the well-known scaling rule, if the device size is 1/K and the power supply

voltage is  $1/K$ , the power consumption will be  $1/K^2$  and the number of devices that can be integrated in a unit area will be  $K^2$  so the power consumption per unit area will be constant. But, as it is difficult to decrease the power supply voltage following the scaling rule, the power consumption will increase. Additionally with the increase in chip area and clock frequency, the power consumption will increase exponentially. This increased power consumption may have a bad effect on a vision chip integrated with image sensor. Therefore the architecture that tends toward low power consumption will be desired.

Concerning the effect on sensor circuits, using a standard CMOS and submicron process, some problems such as an increase in leakage current and decline in sensitivity will appear[12]. Therefore, the process modification will be needed. Moreover, with deep submicron processes, the power supply voltage will decrease according to the scaling rule and parasitic current will increase exponentially. The problem of power supply voltage is especially serious for with mixed-signal circuits. With a small-scaled process, a photodiode can be shrunk similar to other circuits, but with the limit of optical diffraction, the minimum size is about  $4\mu\text{m}$ . Figure 9 shows the shrink trend of the vision chip circuits.

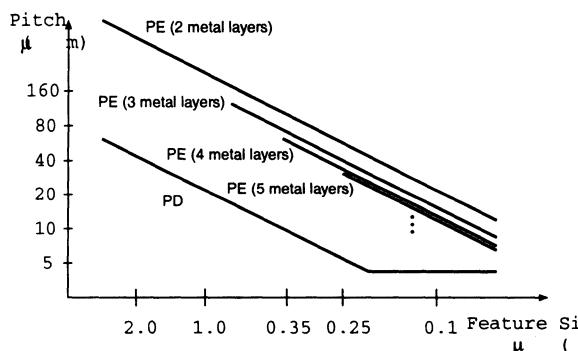


Figure 9. Circuits Shrink with Processing Scaling

## 7. CONCLUSION

We have designed the vision chip architecture for high speed image processing. Based on this architecture a 64x64 pixels prototype chip has successfully been developed. Further integration of vision chips in the future is also estimated. This vision chip will link image processing and real-time

control, and open a new world of robot vision, intelligent transportation systems and many other applications

## 8. REFERENCES

- [1] Alireza Moini : VISION CHIPS, Kluwer Academic Publishers (2000)
- [2] M.Ishikawa, A.Morita, and N.Takayanagi : High Speed Vision System Using Massively Parallel Processing, Proc. Int. Conf. on Intelligent Robots and Systems, pp.373- 377 (1992)
- [3] Y.Nakabo, and M.Ishikawa : Visual Impedance Using 1ms Visual Feedback System, Proc. IEEE Int. Conf. Robotics and Automation, pp.2333-2338 (1998)
- [4] Hiromasa Oku, Idaku Ishii, and Masatoshi Ishikawa: Tracking a Protozoan Using High-Speed Visual Feedback, Proc. of 1st Annual Int. IEEE-EMBS Special Topic Conf. on Microtechnologies in Medicine & Biology, pp.156-159
- [5] Akio Namiki, Yoshihiro Nakabo, Idaku Ishii, Masatoshi Ishikawa: 1ms Sensory-Motor Fusion System, IEEE Trans. on Mechatronics, Vol.5, No.3, pp.244-252 (2000)
- [6] M.Ishikawa : 1ms VLSI Vision Chip System and Its Application, Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp.214-219 (1998)
- [7] T.M.Bernard, B.Y.Zavidovique and F.J.Devos : A Programmable Artificial Retina, IEEE Journal of Solid State Circuits, Vol.28, No.7, pp.789-797 (1993)
- [8] J.E.Eklund and C.Svensson and A.Astrom : VLSI Implementation of a Focal Plane Image Processor -- A Realization of the Near-Sensor Image Processing Concept, IEEE Trans. VLSI Systems, Vol.4, No.3, pp.322-335 (1996)
- [9] Takashi Komuro, Idaku Ishii, and Masatoshi Ishikawa : Vision Chip Architecture Using General-Purpose Processing Elements for 1ms Vision System, Proc. Int. Workshop on Computer Architecture for Machine Perception, pp.276-279 (1997)
- [10] M.Ishikawa, K.Ogawa, T.Komuro, and I.Ishii : A CMOS Vision Chip with SIMD Processing Element Array for 1ms Image Processing, Proc. IEEE Int. Solid-State Circuits Conf., pp.206-207 (1999)
- [11] David X.D. Yang , Boyd Fowler and Abbas El Gamal, A Nyquist-Rate Pixel-Level ADC for CMOS Image Sensors, IEEE Journal of Solid-State Circuits, Vol.34, No.3, pp. 348-356 (1999)
- [12] Hon-Sum Phillip Wong and Abbas El Gamal : Single-Chip CMOS Imaging Systems, 1999 ISSCC Tutorial (1999)

# A vision system on chip for industrial control

*From light to smart vision*

Eric Senn and Eric Martin

*L.E.S.T.E.R., University of South Brittany, BP 92116, 56321 Lorient Cedex, France*

**Abstract:** This paper describes the building of an original vision system on chip. Our smart sensor performs motion detection and pattern recognition with only one single line of pixels. The maximum cross-correlation number is processed by the processors network. A peculiar methodology was defined to tackle the sensor's processing part design. Relying on our architectural synthesis tool GAUT, it leads to an optimal matching between the application specifications, the sensor's architecture and the processor's. The elementary processor's architecture is detailed. Its CMOS VLSI implementation is sketched, as well as the sensor's analog part and the light to byte conversion. The circuit's final structure and floorplan are outlined. Its performances are exhibited.

**Key words:** Vision System On Chip, Active Pixel Sensor, Industrial Control, Motion Detection, Pattern Recognition

## 1. INTRODUCTION

The least a vision system must include is one optical sensor and one processor. The sensor issues images to the processor that performs several low and high level operations to extract relevant information from the image flow. The amount of data to be transferred between those two components is huge. This transfer is thus very time and power consuming. To save some precious energy (as precious as the system ought to be mobile), it is necessary to reduce the bandwidth requirement between sensor and processor. This is achieved when some processing capabilities are given to the sensor, making it undertake every low-level task (*Figure 1*). As a second effect, the main processor load is alleviated; this processor can run slower, consumes less, and has plenty more time to take the high-level decision.

Thus, Vision Systems On Chip (VSOC) actually permit to reduce the vision system's overall cost. The system is cheaper to build since a simpler interface between sensor and processor is needed, even the processor could be slow-down. It is cheaper to use because some energy is saved in the data transfer, or could be saved in the processor clocking.

A great number of VSOC, also called Active Pixel Sensors, or Smart Sensors, are made of a 2D array of pixels (a pixel is an elementary light sensor), connected to a 2D array of elementary processors, each processor being dedicated one or more pixels in the focal plane. Classically, every processor runs the same instruction on different data: the processors' network control is SIMD [4]. Such systems have a very bad filling ratio: the resolution has to be drastically decreased to make enough room for the processing part [3]. Indeed, the design of smart pixel sensors is strongly limited by the maximum transistor count and the available silicon area [8]. To include the SIMD array control inside the circuit is then not even thinkable. Our approach is quite different. We managed to keep VSOC' advantages by removing pixels from the chip. More silicon area is then available for more control and more processing capabilities. According to this idea, we designed smart sensors with only one single or two orthogonal lines of pixels, for motion detection and pattern recognition in the frame of industrial control.

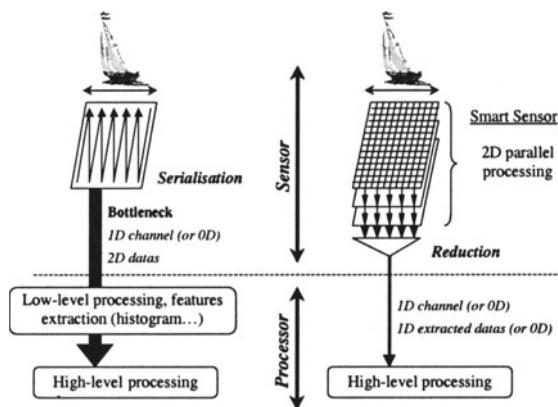
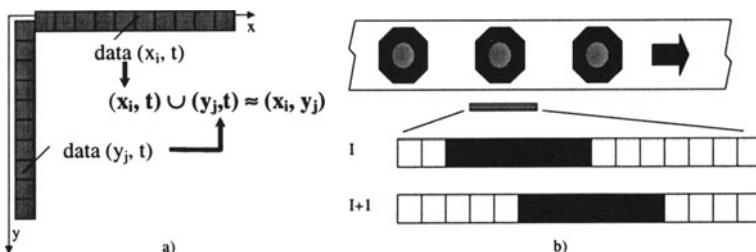


Figure 1. Vision systems: sensor plus processor (left), smart sensor (right)

In the following, we first explain how efficient image processing can be done with such sensors. Then we outline the methodology we defined to handle the smart sensor's processing part design. The resulting circuit's architecture is detailed afterward, and next the processor's. We finally sketch the sensor's VLSI implementation and give its foreseen performances.

## 2. PRINCIPLE

Assuming a few hypothesis depending on the application, it can be shown that the same amount of information is obtained with a full 2D array of pixels than by merging data from two orthogonal lines of pixels [1]. Consider *Figure 2-a*. The pixel's value  $p_i = I(x_i, y_i)$  of image  $I$  is retrieved by merging the two projections  $x_i$  and  $y_i$  of  $p_i$  at time  $t$ . An algorithm that handles a 2D data flow can thus be translated in an algorithm that operates on two 1D data flows, each of them being associated with the time variable  $t$ .



*Figure 2.* (a) 2D data flow equivalent to  $2 \times 1$ D data flows with time  $t$  - (b) Motion detection and pattern recognition for industrial control

This principle only holds on applications with bounded and known behaviour. If a “normal” motion can be determined, it becomes possible to detect a rupture in the motion or in the object shape [2]. In the frame of industrial control, motion detection and pattern recognition can even be done with a single line of pixels. On *Figure 2-b*, items of same shape and size are passing before the sensor at speed  $v$  (in pixels per second). The motion’s direction is known in this case.

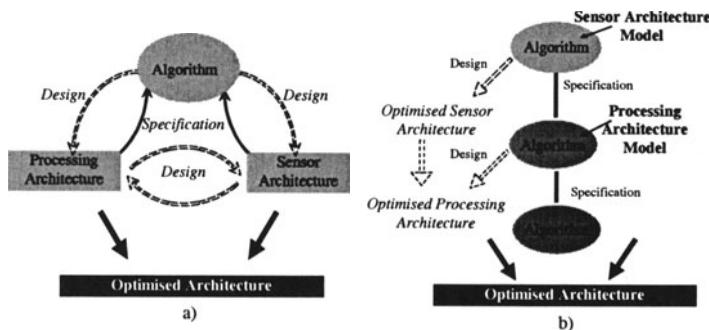
Our purpose is to measure this speed and to determine if an item has a fault in its shape. The two images  $I$  and  $I+1$  on *Figure 2-b* are sampled at a time interval  $T$ . The cross-correlation number  $C_{xx}$  is calculated for every value of  $\tau$  from 0 to  $N-1$  ( $N$  is the number of pixels on the line).

The value of  $\tau$  where  $C_{xx}$  is maximal gives the item’s motion, which is easily related to its speed.  $C_{xx_{Max}}$ , the maximal value of  $C_{xx}$ , informs on the item’s shape. Given a shape,  $C_{xx_{Max}}$  will indeed always be the same. Whenever  $C_{xx_{Max}}$  changes significantly, the item’s shape is altered. The target’s speed is  $v = \tau / T$  (pixels per second):

$$C_{xx}(\tau) = \sum_{j=0}^{N-1} X_t(j) \cdot X_{t-T}(j - \tau)$$

### 3. METHODOLOGY

Methodologies for algorithm/architecture matching classically start with the application description, both an architectural model and an implementation method, then issue a solution that can be software, hardware, or mixed. Never would they tackle sensors' implementation. Acquisition capabilities are however of a great use in many systems. The methodology we proposed copes with systems that include sensors, like VSOC. Such a system can be decomposed in three parts: algorithm, processing, and sensor (*Figure 3-a*).



*Figure 3. (a) Algorithm/Architecture matching (b) Design flow*

Each part has a bi-directional “design/specification” relation with any other. To build a linear design flow involves to break these relations as shown in *Figure 3-b*. The flow is re-iterated until resulting architectures meet the specifications. The final design-flow is shown on *Figure 4*.

As stated on this figure, every architectural model entails a new formulation for the algorithm. Once its correctness checked, this algorithm is used as an entry to GAUT.

GAUT is an architecture synthesis tool for dedicated signal processors [6]. It issues a RTL netlist from an algorithm and an associated timing constraint (area constraint can be specified too). The timing constraint we choose was to obtain a processor as fast as the light to byte conversion. The amount of treatment for one processor to achieve depends on the number of pixels P it copes with. GAUT firstly selects operators depending on the timing constraint. The necessary hardware resources and the involved silicon area are computed. The optimal number of pixels P is determined in this very first step [2]. P is set to minimize the processing area per pixel Sp. With St the total processing area and Spe the processor's area, one gets:

$$Sp = \frac{St}{N} = \frac{Spe}{P}; \text{ indeed: } St = Spe \cdot \frac{N}{P}$$

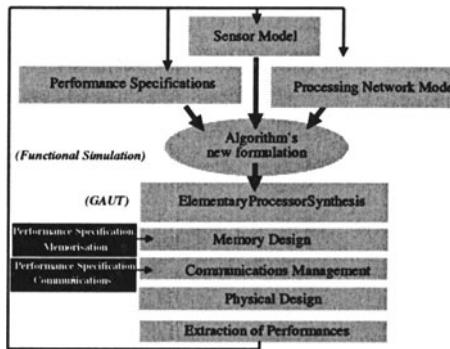


Figure 4. Detailed design flow

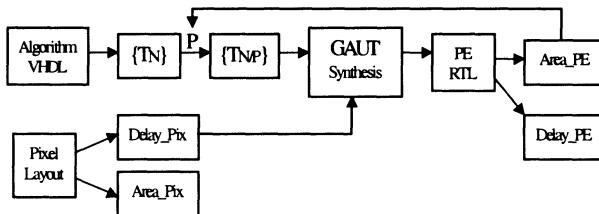


Figure 5. Synthesis tool GAUT and the methodology

The processor's RTL netlist is obtained afterward and processed by a commercial logic synthesizer to target VLSI. *Figure 5* sketches the methodology. The algorithm is written in VHDL. The set of treatments  $\{T_N\}$  represents the work to be done in the algorithm, for all pixels. The subset of treatments  $\{T_{N/P}\}$  represents the elementary processor's work and depends on the parameter  $P$ . It is synthesised by GAUT with the pixel's delay as timing constraint. The resulting processor's area is computed. The processus is repeated with different values for  $P$  until the minimum PE area is found. The computed PE delay matches the timing constraint.

## 4. CIRCUIT'S ARCHITECTURE

The correlation algorithm is strongly regular and can be easily distributed between every processor on a SPMD network :

```

Iterate
For "all new image"
Sequential
For  $\tau = [\tau_{\min}; \tau_{\max}; \tau+1]$ 
    PARALLEL for all  $k = 1, \dots, N/p$ 
     $C_{xx}^k(\tau) = 0$ 
    For  $j = [(k-1)p; kp; j+1]$ 
         $C_{xx}^k(\tau) = C_{xx}^k(\tau) + X_{II}(j) \cdot X_{I0}(j-\tau)$ 
    End  $j$ ; End  $k$ 
    SEQUENTIAL for  $k = 1, \dots, N/p$ 
     $C_{xx}(\tau) = 0$ 
     $C_{xx}(\tau) = C_{xx}(\tau) + C_{xx}^k(\tau)$ 
    End  $k$ 
End  $\tau$ 
Max [ $C_{xx}(\tau)$ ] => Motion =  $\tau$ 

```

The cross correlation number  $C_{xx}(\tau)$  is calculated for every value of  $\tau$  (from 0 to a parameter  $\tau_{\max}$ ).  $N$  is the number of pixels on the line,  $P$  the number of pixels by elementary processors in the circuit.  $X_{II}(j)$  is pixel  $j$  from image  $I_1$ . Image  $I_0$  follows  $I_1$ . During the parallel part, each processor computes a part of the cross-correlation number for  $N/P$  pixels. For instance, processor  $k$  tackles pixels  $(k-1)p$  to  $kp$ . The last sequential part is necessary to add up all the  $P$  parts of the cross-correlation number and to finally compute its maximal value among all the  $\tau$  values from  $\tau_{\min}$  to  $\tau_{\max}$ .

Our purpose here is to determine the optimal number of pixels per elementary processor, in other words, to match the algorithm with an architecture, considering two constraints: silicon area and circuit's speed.

According to the methodology, several high level synthesis were performed using GAUT with different values for  $P$ , and the pixel cell's speed as timing constraint (see the VLSI implementation section). The minimum processing area per pixel was obtained with  $P=8$  [9]. 4 processors are thus necessary to build our 32 pixels prototype. The circuit's architecture is outlined on *Figure 6*.

Every processor stores two successive images ( $I_0$  and  $I_1$ ) and computes the partial sum  $C_{xx}^k(\tau)$  from these images and its right neighbour's partial sum. Result is send to the processor's left neighbour. The last processor of the line finally issues the cross-correlation coefficient for the chosen  $\tau$  value. Note that  $C_{xx}$  is computed from right to left in the processors line. The same result would have been obtained from left to right. The  $\tau$  value where  $C_{xx}(\tau)$  is maximal gives the item's motion. The MAX module keeps the maximal value for  $C_{xx}$ :  $C_{xx\max}$ .

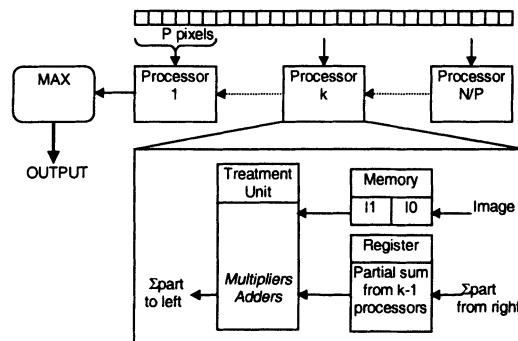


Figure 6. The circuit's architecture

## 5. PROCESSOR'S ARCHITECTURE

As shown on *Figure 7*, the processor contains sixteen, 8 bits registers (parallel and serial load), two 8 bits multipliers, and one 16 bits adder.

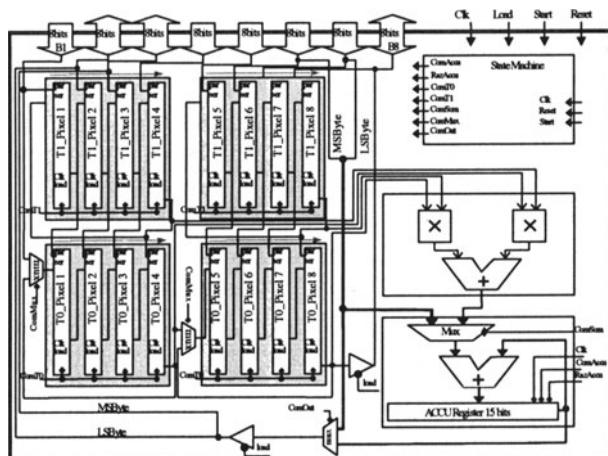


Figure 7. The processor's architecture

Pair of registers are connected vertically to make eight 2 stages FIFO, which store the pixels value for images I0 and I1. I0 is the oldest image and is stored in the 8 lowest registers called T0-pixel1 to T0-pixel8. I1 is the last sampled image. It is stored in the 8 upper registers on the figure, called T1-pixel1 to T1-pixel8. T1 is the first stage of the FIFO, T0 the second stage.

Horizontally, registers are connected by four in a ring in which data can shift to reach the processing unit. This minimises the multiplexers count.

Eight data busses reach the elementary processor. Their main purpose is to input the pixels' value from the light sensor and the analog to digital converter. However, most of them are multi-purpose. B1 also inputs the register T0-pixel1 value from the processor's left neighbour. B2 and B3 issues the processor's partial sum to the left neighbour. B6 and B7 get this partial sum from the right and B8 issues the right register T0-pixel8 value to the right neighbour.

The circuit's behaviour is summarised below:

Step1, *Initialisation*: Sample image. Store image to registers T1.

Step2, *Shift and store*: Sample image. Shift T1 to T0 (vertical FIFO). Store image to registers T1. At this stage I0 is in T0, and I1 in T1.

Step3, *Cross correlation calculation*: Read pixels' values in T0 and T1, ( $\tau=0$ ). Horizontally shift pixels value to reach multipliers (internal four registers rings are used). Resulting partial sum is accumulated in the ACCU register.

Propagate partial sum across the line. Issue  $C_{xx}(\tau)$ .

Step4, *Increment  $\tau$  value*: Horizontally shift to the right every registers T0 in all processors. Internally shift-right registers T0. Send T0\_pixel8 register's value to the right. Receive T0\_pixel1's value from the left neighbor.

Step5: If  $\tau <= \tau_{\text{Max}}$  go to step3.

Step6, *Compute  $C_{xx_{\text{Max}}}$* : Issue  $C_{xx_{\text{Max}}}$  and the corresponding  $\tau$  value.

Step7, *next image*: Go to step2.

## 6. VLSI DESIGN

A great care was taken in designing the sensor's analog part. This part is absolutely essential for the first thing a vision sensor has to do is to convert light to byte. This conversion is not so easy to achieve for light has a very broad dynamic, and because the sensor could be used in many different places, with different light average and max/min levels. The basic light sensor we used is a  $60 \times 60 \mu\text{m}$  photo diode. It provides an output current  $i_L$  ranging from  $10^{-11}\text{A}$  to  $10^{-6}\text{A}$  for a light level from 0.01 (dark) to  $100 \text{ W/m}^2$  (broad daylight) [10]. *Figure 8-a* shows the pixel cell's structure.

Transistor M0 converts current  $i_L$  in voltage  $v_1$ . Wired like a diode, it behaves in the subthreshold region and has a logarithmic response [11]. Transistor M1 acts as an amplifier. First, Exp and Preb are low so that  $V_{out}=V_{dd}$ . Then Preb is high and Cout is discharged across M2 while Exp is high.  $V_{out}$  final value is linear with  $\ln(i_L)$  (see *Figure 9*). When Exp is

toggled high, charge sharing induces a parasitic current from Exp, via M2 and M1, to M1's gate. For very low  $i_L$ , this current increases  $v_2$  during exposure time, and finally leads to bad output values. To keep the sensor's response linear, M2's length was increased to  $4\mu\text{m}$ .

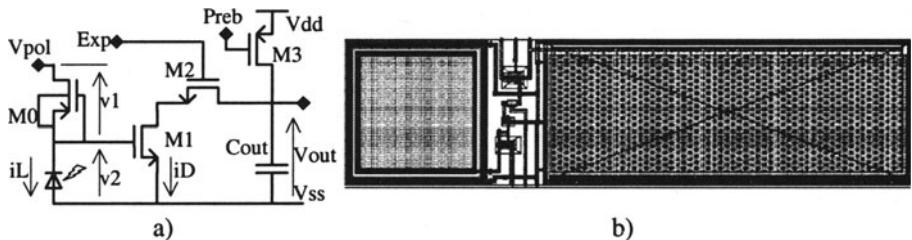


Figure 8. (a) The pixel cell's structure: photo-diode and response “linearization” - (b) The pixel's layout

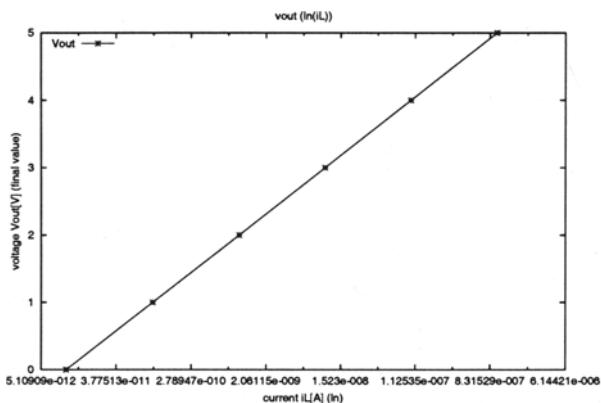


Figure 9. Simulated  $V_{out}(\ln(i_L))$

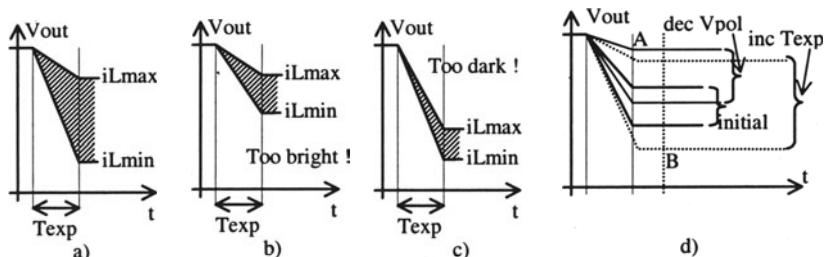


Figure 10. Output dynamic for (a) normal (b) over exposed scene - (c) under exposed scene - (d) Trimming procedure for  $V_{pol}$  and  $T_{exp}$

$T_{exp}$  is high during time  $T_{exp}$ .  $T_{exp}$  is somewhat like a camera's exposure time. Along with  $V_{pol}$ , it allows for trimming the sensor's output dynamic. Indeed, transistor M1 converts  $v_2$  to  $iD$ . Since  $v_2$  equals  $V_{pol}$  minus  $v_1$ ,  $iD$ 's output range actually depends on  $V_{pol}$ .

As shown on *Figure 10*, the sensor's output dynamic is maximal only if the image has good contrast. It is poor if the scene is too bright or too dark. In the first case, output dynamic can be restored merely by increasing  $T_{exp}$ . It is restored by decreasing  $V_{pol}$  in the second case. Both of these actions can be used, as shown on *Figure 10-d* where the initial output dynamic is too narrow. Hence the trimming procedure : first to decrease  $V_{pol}$  down to A, then to increase  $T_{exp}$  up to B.

The final pixel's layout is given on *Figure 8-b* ( $65.1 \times 256.8\mu\text{m}$ ). The photo receptor is the square on the left ( $64.6 \times 64.6\mu\text{m}$ ).  $C_{out}$  is the large rectangle on the right ( $65.1 \times 166.8\mu\text{m}$ ). It was set to  $15\text{pF}$  to overcome the cumulated input capacitance of the multiplexing part to the ADC. Indeed,  $C_{out}$  is set to reduce charge sharing during multiplexing [9]. It must not be too big to avoid wasting room on the floorplan, and to keep the sensor's response time small.

The processor's layout can be found in [9]. It is  $1.1\text{mm}$  wide, its surface area is  $1.25\text{mm}^2$ . This processor performs one cross-correlation in  $200\text{ns}$ . The AMS CYX  $0.8\mu\text{m}$  process (2 polysilicium, 2 metal layers) was used.

The floorplan is constrained by the pixels' location on chip. Pixels have to be placed in a line. The analog to digital converter (ADC) we used, the ADC02 ( $612 \times 358\mu\text{m}$ ), comes from the AMS library and is almost as big as the processor itself. To avoid wasting place, we multiplexed one single ADC to all pixels.

To bring sampled pixels' values to the processors, a pipelined bus was designed, as shown on *Figure 11-a*. This bus behaves somehow like a FIFO. The first sampled data has to cross all the stages to reach the last processor. The operational amplifier (OP5B's input capacitance is  $0.1\text{pF}$ ) is needed to hide the converter's huge input capacitance ( $8\text{pF}$ ) to the transmission gates which supports multiplexing. Extra input and output are added for test purpose. It is so possible to read or write the pixels' pipelined bus from outside the circuit. The pixel-max and pixel-min modules store the maximal and minimal pixels' values in an image. These values are used to bias the sensor's output dynamic afterward. If the max value is too low,  $V_{pol}$  should be decreased. For the analog to digital converter's output is 8 bits wide, a too small max value could be when  $\text{pixel\_max} < 250\text{d}$ . Also, if the min value is too high,  $T_{exp}$  should be increased. For the same reason, a too high min value could be  $\text{pixel\_min} > 05\text{d}$ .

Final performances actually depend on the multiplexing ratio. The AMS ADC02 performs one conversion in  $50\mu\text{s}$ ; it will do the whole line in

$1600\mu\text{s}$ . The chip's sampling frequency is then  $\text{Fs}=625$  images per second. It is not bounded by the processor's speed: one cross-correlation is issued every 200ns and the max across 32  $\tau$  values is found in  $6.4\mu\text{s}$ . The chip's performances are neither settled by the processor's speed, nor by the pixel cell's acquisition time ( $T_{\text{exp}}$  is about  $0.5\ \mu\text{s}$ ). They are mainly settled by the ADC's response time and the multiplexing ratio. The sampling frequency could be set to 2500 images per second if four ADC were used instead of one. No multiplexing is needed in that case.

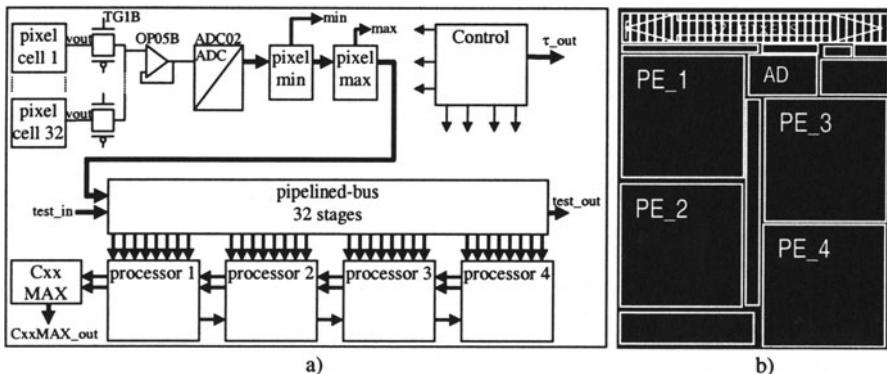


Figure 11. (a) The circuit's synopsis (control lines are not represented) - (b) The floorplan shows the sizes of processor (PE), analog to digital converter (ADC) and pixels (Pix)

The maximum target's speed is related to the sampling frequency. If the target's motion is more than 32 pixels in  $1600\mu\text{s}$ , it will not appear on the next sampled image. Hence, the maximum speed is  $32/1600^{\text{E}-6} = 20000$  pixels/second. The circuit's floorplan is sketched on *Figure 11-b*. The core is about 3mm x 2.4mm.

## 7. CONCLUSION

This paper shows that motion detection and pattern recognition are possible with only one single line of pixels, assuming a few hypothesis, as in the presented frame of industrial control. Moreover, by removing pixels, enough room is made on chip for the whole processing and control resources. As a result, dedicated complex algorithms can be implemented though the sensor remains fully independent. Our methodological approach was very useful in designing the vision system and could be re-used in many smart sensors' design. Given timing constraints, the optimal number of processors per pixel was found. The sensor's light to byte conversion

exhibits original mechanisms to balance the analog part's output dynamic, depending on the input average light level. The pixel's response time,  $0.5\mu s$ , was used as a timing constraint to the processor's architecture synthesis. This synthesis was performed with our architectural synthesis tool GAUT. The processor's RTL netlist is implemented in CMOS VLSI thanks to commercial logic synthesis tools. The processing time for 32 pixels is  $6.4\mu s$ . To reduce the circuit's surface area, one single analog to digital converter was used and multiplexed between all pixels. As a result, the circuit's maximal sampling frequency is decreased to 625 images per second.

## 8. REFERENCES

- [1] Emzivat D., Gagnadre C. and Martin E. "Vision sensor for the industrial quality control". *Proceedings of the 7<sup>th</sup> International Conference on Image Processing and its Application*, Manchester, UK, 12-15 July 1999.
- [2] Emzivat D., Gagnadre C. and Martin E. "Optical integrated circuit for the quality control". *International Symposium on Microelectronic Manufacturing Technologies*, Edimbourg, 19-21 May 1999.
- [3] Gruev V. and Etienne-Cummings R.R. "A programmable spatiotemporal image processor chip". *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, May 28-31 2000.
- [4] Ishikawa M., Komuro, T., Ogawa K. and Ishii I. "A CMOS vision chip with SIMD processing element array for 1ms image processing". *Abst. 1999 Dig. Tech. Papers of 1999 IEEE Int. Solid-State Circuits Conf.*, pages 206-207.
- [5] Komuro T., Ishii I., Ishikawa M. and Yoshida A. "High speed target tracking vision chip". *Proceedings of the Fifth IEEE Int. Workshop on Computer Architecture for Machine Perception*, Padova, Italy, September 11-13 2000, pages 49-56.
- [6] Martin E., Sentieys O., Dubois H. and Philippe J.L. "GAUT, An architectural synthesis tool for dedicated signal processors". *Proceedings of the EURO-DAC93*, 1993.
- [7] Paillet F., Mercier D., Bernard T.M. and Senn E. "Low power issues in a digital programmable artificial retina". *IEEE Alessandro Volta Memorial International Workshop on Low Power Design*, Como, Italy, March 4-5, 1999.
- [8] Paillet F. and Mercier, D. "Design solutions and techniques for vision system on a chip and fine-grain parallelism circuit integration". *Proceedings of the Thirteenth Annual IEEE International ASIC/SOC Conference*, Arlington VA, USA, September 13-16 2000.
- [9] Senn E., Emzivat D. and Martin E. "A smart pixel sensor for industrial control". *International Symposium on Signals, Circuits and Systems*, Iasi, Romania, July 10-11 2001.
- [10] Sicard G. "From biology to silicium: A bio-inspired analog retina for smart vision sensor". *PhD Thesis*, Institut National Polytechnique de Grenoble, 1999.
- [11] Uyemura J.P. "Circuit design for CMOS VLSI". Kluwer Academic Publishers, 1992.

# Fast Recursive Implementation of the Gaussian Filter

D.Demigny, L. Kessal, J. Pons

*ETIS, UPRESA CNRS 8051, ENSEA et Université de Cergy Pontoise, ENSEA, 6 av. du Ponceau, 95014 Cergy Pontoise Cedex, demigny@ensea.fr*

**Abstract:** On one hand, the convolution with the truncated impulse response (IR) of Gaussian filters leads to a high computation cost when the standard deviation  $\sigma$  of the Gaussian increases. On the other hand, recursive filters that approximate the Gaussian filters reduce the computation cost but can only be applied to finite length signals due to the infinity of the IR on both sides.

In this paper, we present a new filter: PAOG. Its IR is a polynomial which is an accurate approximation of the truncated Gaussian IR. Then, we derive a simple and fast recursive implementation of the PAOG filter. The computation cost is independent on  $\sigma$ . PAOG is particularly suitable for very high speed hardware architecture because the recursive part of the filter uses only adders (without multiplier) which can be implemented in carry-save. This filter can be used with signal of infinite duration (real time). Simultaneously, the first and second derivative of the Gaussian filter are computed without any extra computation cost.

**Key words:** Gaussian filter, recursive filter, polynomial filter

## 1. INTRODUCTION

Gaussian filters and their first and second derivative are often used in image processing for example for denoising and edge detection. The 2D Gaussian filter is the only symmetric filter which is separable. This property reduces the computations to two 1D Gaussian filters used vertically and horizontally. Then our study will use 1D filters without lack of generality.

In this section, we will examine first the finite impulse response (IR) approximations. Then we will discuss some aspects of recursive implementations. Finally we will introduce the PAOG (Polynomial

Approximation Of Gaussian) filter. The second section will discuss the polynomial approximation and its accuracy. The third section will present the recursive implementation of PAOG and its computation cost. The fourth section will be about hardware implementation.

## 1.1 Finite IR approximation of the Gaussian filter

Usually, the Gaussian infinite IR is truncated to the symmetric  $2M+1$  center terms. This leads to a convolution scheme for the filter computation. Traditionally, a good approximation requires that  $M > 3\sigma$ . Then the computation cost and the memory bandwidth increase as  $\sigma$ . The number of multiplications and additions is more than  $(3\sigma x, 6\sigma +)$ .

Canny in [1,2] has suggested to use a binomial approximation. The corresponding filter, defined by its z transform:  $H(z) = (1 + z^{-1})^n$ , is computed with  $n$  iterations of the simple equation  $y_i = x_i + x_{i-1}$  where  $y$  and  $x$  are the output and the input of the elementary filter. Only  $n$  additions are needed. But  $\sigma$  increases as  $\sqrt{n}/3$ . So the computation cost is:  $(9\sigma^2 +)$  which increases very fast with  $\sigma$ . This approximation is also not accurate for small value of  $n$ .

## 1.2 Infinite IR approximation of the Gaussian filter

Because the Gaussian has an IR that extends to infinity in both sides, the recursive implementation must employ two recursive filters moving in opposite directions (stability constraint). Each of them has an IR which is approximately a half Gaussian. In [2], Canny has suggested to approximate the half Gaussian by a damped exponential cosine, which leads to a z transform of the filter with two poles and two zeros. Canny used two passes with the same filter to obtain a good approximation of the Gaussian. The computation cost is then  $(12x, 12+)$ . The major advantage of the recursive implementations is that the computation cost becomes independent on  $\sigma$ . Canny said that this implementation has the lowest cost and must be employed in the future. We will see in this paper that the PAOG filter has a lower computation cost and offers the main advantage to require a move in only one direction which permits its use with real time signals.

## 1.3 The PAOG filter

Canny has defined three criteria to derive the equation of an optimal filter for edge detection in the case of continuous signal. In our previous work [3,4,5], we extend the Canny criteria to sampled signals. Recently, we proved that the optimal Canny first derivative filter can be approximated by

a simple polynomial IR filter. Because Canny has suggested that its filter can be approximated by the first derivative of Gaussian filter, it means that the integral of our polynomial filter should be a good approximation of the Gaussian.

The PAOG IR has a symmetric shape. Its half IR is a polynomial of degree equal to 4 inside the window size (finite IR). Then, the z transform of PAOG can be express with a recursive expression. The five poles of this filter are equal to 1 (polynomial property). This particularity permits the computation with only one direction move. Then, the recursive implementation of PAOG is usable for real time signals. Because the IR is finite, each pole is compensated by a zero of the same value. Details of the PAOG implementation are given in section 3. We will show that the computation cost is only (4 x ,11+). In the next section, we will discuss the quality of the Gaussian approximation with the PAOG filter.

## 2. POLYNOMIAL APPROXIMATION

### 2.1 The PAOG impulse response

The size of the IR is defined as  $2w+1$ . The expression of the IR  $h^w$  of the PAOG filter is:

$$h_k^w = C_p \cdot (w + 2 - |k|)(w + 1 - |k|)(-3k^2 + (2w + 3)|k| + w(w + 3)) \quad (1)$$

where  $C_p$  is a normalization constant chosen to obtain a sum of the filter coefficients equal to one. This choice sets the amplification to one for low frequency signals independently of  $w$ .

$$C_p = \frac{5}{2w(w+1)(w+2)(w+3)(2w+3)} \quad (2)$$

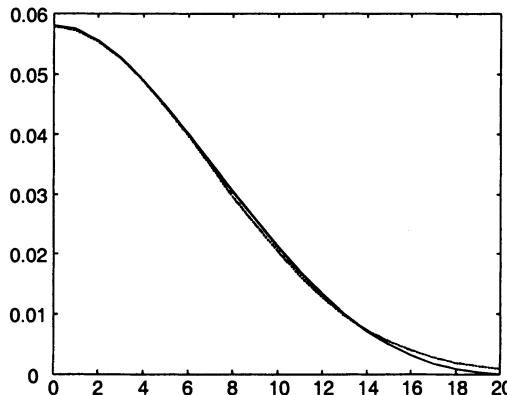
The Gaussian IR is:

$$g^\sigma = C_g e^{-\frac{k^2}{2\sigma^2}} \quad (3)$$

For each value of  $w$ , it exists a value of  $\sigma$  for which  $h^w$  will be an accurate approximation of the Gaussian IR  $g^\sigma$  as explained below. Using the minimization of the absolute quadratic error, we find:

$$\sigma = 0.3217w + 0.481 \quad (4)$$

On figure 1, we show the half positive IR  $h^w$  and  $g^\sigma$  for  $w=20$  and  $\sigma=6.915$ . We see that the largest difference is near the extremity of the window, *i.e.* for the lowest values of the response.



*Figure 1.* Half impulse responses of the Gaussian filter (dashed) with  $\sigma=6.915$  and of the PAOG filter (solid) with  $w=20$

## 2.2 Extension to different $\sigma$ values

Usually, for image processing, it is not necessary to use accurate values of  $\sigma$ . Then the fact that only a finite number of  $\sigma$  values is available since  $w$  is an integer is not a drawback. However, it is possible to obtain different  $\sigma$  values if a cascade of PAOG filters is used. For a cascade of  $n$  filters, if the filter  $i$  corresponds to  $\sigma_i$ , the global  $\sigma$  is defined as:

$$\sigma^2 = \sum_{i=1}^n \sigma_i^2 \quad (5)$$

## 2.3 Accuracy of the Gaussian approximation

We note  $g_w^\sigma$  the restriction of the function  $g_w$  to the window of size  $(2w+1)$ . Three different measures are used to study the error inside the window.

- a) The mean of the relative quadratic error:  $mrq$

$$mrq = \frac{1}{2w+1} \sqrt{\sum_{-w}^{+w} \left(1 - \frac{h^w}{g_w^\sigma}\right)^2} \quad (6)$$

- b) The mean of the absolute quadratic error normalized by the value of  $h^w_0$ :  
 $maq$

$$maq = \frac{1}{h^w_0(2w+1)} \sqrt{\sum_{-w}^{+w} (g_w^\sigma - h^w)^2} \quad (7)$$

- c) The maximum of the absolute quadratic error normalized by the value of  $h^w(0)$ :  $Maq$

$$Maq = \frac{\max_k |g_w^\sigma(k) - h^w(k)|}{h^w(0)} \quad (8)$$

The figure 2 shows the errors in % for  $w$  between 1 and 20 ( $0.8 < \sigma < 6.91$ ).  $mrq$  is less than 6 %,  $Maq$  is approximately constant (2 %) and  $maq$  is less than 0.5 %.

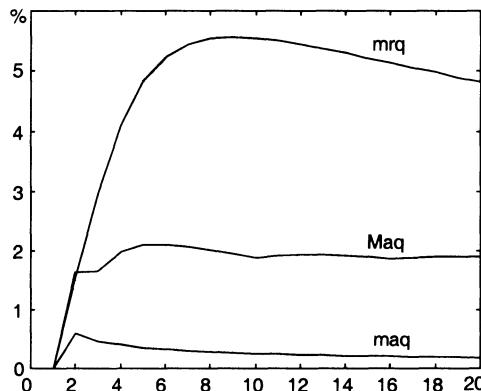


Figure 2. Mean absolute (maq), maximum absolute (Maq) and mean relative (mrq) quadratic error between  $g^\sigma$  and  $h^w$  in %

An other measure looks for the truncative effect. The ratio between the sum of the filter coefficients of  $g^\sigma$  outside the window and the sum of the filter coefficients of  $g^\sigma$  inside the window is less than 1 % for all the  $w$  values. All these results confirm that the PAOG filter is a good approximation of the Gaussian filter.

### 3. RECURSIVE IMPLEMENTATION

#### 3.1 Z transform

The z transform of  $h^w$  can be cut in:

$$H^w(z) = \sum_{k=-w}^0 h_k^w z^{-k} + \sum_{k=0}^{+w} h_k^w z^{-k} - h_0^w \quad (9)$$

To compute the second sum part of (9) we have first expanded the polynomial (1) as:

$$\sum_{k=0}^{+w} h_k^w z^{-k} = \sum_{i=0}^4 a_i \sum_{k=0}^{+w} k^i z^{-k} \quad (10)$$

where  $a_i$  are functions of  $w$ . The second sum in (10) can be analytically evaluated:

$$\sum_{k=0}^{+w} k^i z^{-k} = \frac{N_i^w(z^{-1})}{(1-z^{-1})^i} \quad (11)$$

where  $N_i^w(z^{-1})$  is a polynomial of degree  $i$  of the variable  $z^{-1}$ . Introducing (11) in (10) and using the same method to compute the first sum part of (9), one obtains the expression of  $H^w(z)$ :

$$H^w(z) = C_p \frac{z^{-2}(1+z^{-1})N^w(z)}{(1-z^{-1})^5} \quad (12)$$

with:

$$\begin{aligned} N^w(z) &= w(z^{(w+2)} - z^{-(w+2)}) + (w+3)(z^{-(w+1)} - z^{(w+1)}) \\ &\quad + (2w+3)(z - z^{-1}) \end{aligned} \quad (13)$$

and:

$$C_p = \frac{30}{w(w+1)(w+2)(w+3)(2w+3)} \quad (14)$$

### 3.2 Stability

Because the poles are located on the unit circle, the stability of the filter is obtain if and only if the zeros of the numerator  $N^w$  compensate exactly the poles. So, any computation error on  $N^w$  leads to instability ! This prohibits the use of floating point numbers for the filter computation because summation of two numbers very different in value leads to an approximated result. The computations must be made with integer numbers. The input data and the filter coefficients are integer numbers too. The normalization by the constant  $C_p$  must be made at the end of the computation and, in this case, the floating point notation is authorized (after and outside the recursive loop). This is not so restrictive because the input data in image processing are usually represent by integer values. It is also a great advantage for the implementations on the cheaper DSP or on hardware (ASIC or FPGA).

### 3.3 Computation algorithm

We define:

$$a_0 = w, \quad a_1 = w + 3, \quad a_2 = 2w + 3 \quad (15)$$

The current data input is  $x_i$  and one wants to compute the output  $y_i$ . The computation progresses towards increasing values of  $i$ . All the  $t_j$  and  $n$  are simple temporary variable.

$$\begin{aligned} n &= a_0(x_{i+w+2} - x_{i-w-2}) + a_1(x_{i-w-1} - x_{i+w+1}) + a_2(x_{i+1} - x_{i-1}) \\ t_1 &= t_1 + n \\ t_2 &= t_2 + t_1 \\ t_3 &= t_3 + t_2 \\ t_4 &= t_4 + t_3 \\ t_5 &= t_5 + t_4 \\ y_i &= C_p(t_6 + t_5) \\ t_6 &= t_5 \end{aligned} \quad (16)$$

### 3.4 Initialization of the algorithm

For image processing, because the number of data on each line (or each column) is finite, attention can be made for the initialization to avoid

transient response near each side. More important, because five integrations are applied on the signal, bad initialization leads to overflow and instability !

To avoid instability and transient responses, we proceed as follow: near the left side (or at reset for real time signals), to avoid transient response near  $x_0$ , one supposes that  $x_0$  is a continuous value for negative  $i$  indexes.  $t_{1..4}$  and  $t_6$  are zeroed and  $t_5$  is initialized to  $2x_0/C_p$  which is always an integer when  $x_0$  is an integer. At iteration  $i$ , unavailable input data  $x_{i+j}$  are replaced in (16) with  $x_0$ . Input data  $x_{i+j}$  are replaced by their right values as soon as they become available. For the right side (image processing), if  $N$  is the index of the last pixel, the data  $(x_{i+w+2}, x_{i+w+1}, x_{i+1})$  can be replaced by  $x_N$  when they become unavailable.

### 3.5 Performances of the programming version of PAOG

The PAGG algorithm uses only 11 additions and 4 multiplications. An other advantage is the reduced memory bandwidth needed. Only 5 memory accesses by pixel computed are necessary. To compare the performances, we have implemented the Gaussian filter with a simple convolution and the PAOG filter for the same window size  $w$ . We have also tested the recursive solution proposed by Canny, see §1.2. Programs are written in C language on a PC, and integer numbers are used. The PAOG filter is 1.37 times faster than the Canny recursive filter. Both have a computation time independent on  $\sigma$ . The table I shows the speed gain of the PAOG filter compared to the convolution as a function of  $w$ .

| $w$  | 2    | 6    | 10   | 14   | 18   | 20   |
|------|------|------|------|------|------|------|
| gain | 1.77 | 4.25 | 6.71 | 9.17 | 11.6 | 12.9 |

Table 1. Speed improvement

The gain increases as  $w$  or  $\sigma$ . The superiority of the recursive scheme of PAOG on the convolution scheme is clear.

## 4. HARDWARE IMPLEMENTATION

### 4.1 Functional description

The figure 3 shows the main details of the hardware implementation. Delayed data are obtained at the output of the registers and of the two memories A and B used as FIFO of size  $w$ . The time consuming operators are essentially the multipliers. Because they are only used in the no recursive part, retiming and pipelining can be easily employed to increase the speed.

The recursive part uses only adders. So, this part can be easily accelerated with carry-save or bit pipelined adders. Looking to figure 3, it will be remarked that in the same time, the filter computes the fourth first derivatives ( $y^{(1)}$  to  $y^{(4)}$ ) of the output without extra computation !

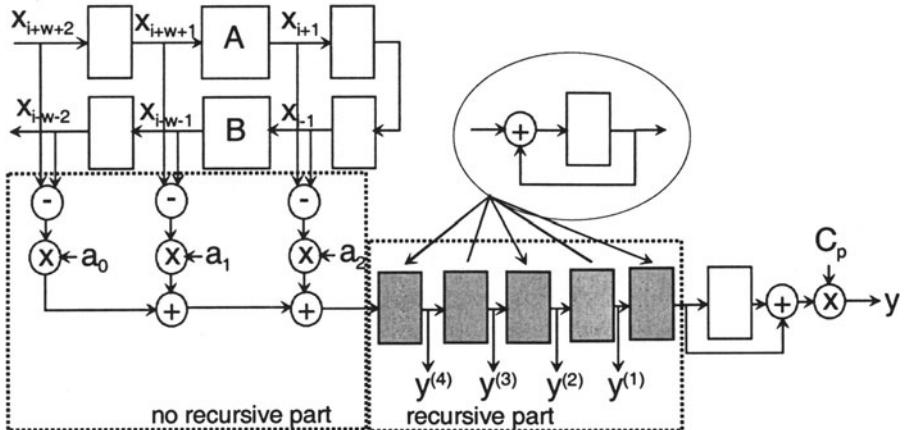


Figure 3. Hardware design. Pipelined registers are not detailed on this figure

## 4.2 Sizing of data and computations

The presence of the integrators in the recursive loop complicates the operators and registers sizing. To ensure the filter stability, one must avoid the saturation of the computations. One supposes that input data are  $M$  bits unsigned numbers. Multipliers have low size because  $M$  is about 8 bits in image processing and the highest coefficient is  $2w+3$ . If this coefficient is coded with 8 bits,  $w=126$  and the impulse response is extend to 253 pixels !

The output of the no recursive part is a signed integer. It is coded with:

$$M + 3 + \log_2(w + \frac{3}{2}) \text{ bits} \quad (17)$$

It is impossible to size an integrator with a reasoning starting from input and progressing towards output. One must proceed from output towards input. Because the global amplification is one, the  $y$  result on figure 3 as an integer part coded with  $M$  bits. The output size of the last adder is unsigned and coded with:

$$M - \log_2 C_p \text{ bits} \quad (18)$$

Because the static amplification of the last adder is 2, the output of the last integrator uses one bit less than (18). An integrator with input  $u$  and output  $v$  computes:

$$v_n = u_n + v_{n-1} \quad (19)$$

Inverting this equation leads to:

$$u_n = v_n - v_{n-1} \quad (20)$$

The input of an integrator uses one more bit than its output (worst case). Then, it is possible to compute an over-estimation of the  $y^{(4)}$  register size (figure 3) which is an unsigned integer:

$$M + 3 - \log_2 C_p \approx M - 1 + 5\log_2(w + 3) \quad (21)$$

The other operators and registers size can be easily deduced from (21). For a 32 bits DSP implementation and for 8 bits input data, (21) leads to  $w_{max}=29$  or  $\sigma_{max}=9.8$ .

An other way to compute the register size uses a property of the addition with two's complement numbers (thanks to J. Liénard which remember me this property). There is no need to take care of the overflow of the partial results in a chain of adders. If  $O$  is the number of bits of the output of the last integrator:

$$O \approx M - 5 + 5\log_2(w + \frac{3}{2}) \quad (22)$$

This result is less restrictive than the previous one. The same size  $O$  can be used for each integrator. For example, 32 bits computations allow a maximum of  $w=47$  for 8 bits input data or  $w=17$  for 16 bits input data.

### 4.3 FPGA implementation

We have estimated the maximum sampling data rate for an implementation a Xilinx VIRTEX XC2V1000-5 FPGA. This device includes fast 18x18 bits multipliers which can compute 8x9 bits multiplications in less than 3ns. Registers and adders of our implementation are over-estimated to 32 bits width. We assume that pipelining registers are inserted between each operator of the no-recursive part. In this case, the maximum sampling rate (144 Mhz) is limited by the accumulators (integrators). This sampling rate

can be double by the use of block adders (each 32 bits adder is cut in four 8 bits adders laterally pipelined).

## 5. CONCLUSION

We have shown that the PAOG filter is a good approximation of the Gaussian filter. It has a finite impulse response, but on the contrary to the conventional approaches (convolution), we have proposed to implement the PAOG filter with a recursive scheme. Usually, recursive filters on image processing require computation moves in both directions (left to right and right to left for horizontal computation). The very unusual choice of a polynomial impulse response leads to a limit case. It derives that only one move is necessary. Then the PAOG filter can be employed with real time data (signal processing) and no frame storage is needed for the vertical computation (image processing). Usually, one thinks that floating point computations are more accurate than fixed point or integer ones. Surprisingly, in the PAOG case, it is the opposite. Only the integer numbers ensure the filter stability ! As strange as it is, this simple filter permits very fast computations both in hardware and software.

## 6. REFERENCES

- [1] J. Canny, "A computational approach to edge detection", IEEE Transactions on Pattern Analysis And Machine Intelligence, vol. 8, pp. 679 – 714, 1986.
- [2] J.F. Canny, "Finding edges and lines in images", Tech. Rep. AI-TR-720, MIT Artificial Intell. lab., 1983.
- [3] D. Demigny, and T. Kamleh, "A discrete expression of Canny's criteria for step edge detection performances evaluation", IEEE Pattern Analysis and Machine Intelligence, vol. 19, n° 11, pp. 1199 – 1211, november 1997.
- [4] D. Demigny, F. Garcia Lorca, and L. Kessal, "Evaluation of edge detector performances with a discrete expression of Canny's criteria", Proc. International Conference on Image Processing, Washington, october 1995, IEEE ICIP, n°2, pp. 169 – 172.
- [5] D. Demigny, "Extension of Canny's discrete criteria to second derivative filters: towards a unified approach", Proc. International Conference on Image Processing, Chicago, october 1998, IEEE ICIP.

# A Dynamically Reconfigurable Architecture for Low-Power Multimedia Terminals

Raphaël David, Daniel Chillet, Sébastien Pillement, Olivier Sentieys  
ENSSAT - LASTI - University of Rennes 1, 6 rue de Kérampong, 22300 - France,  
[name@enssat.fr](mailto:name@enssat.fr)

**Abstract:** Within the framework of third generation telecommunication domain, reconfigurable architectures are becoming more and more popular thanks to both their flexibility and performances. In order to define a system that combines high-performance and low-energy consumption, a dynamically reconfigurable architecture, designed with energy awareness is proposed. This paper presents the main features of the DART architecture along with results from the application domain implementations. These results validate the architectural choices and demonstrate the adequacy between DART and next generation telecommunication applications.

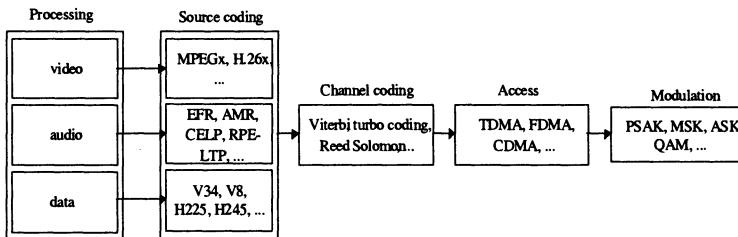
**Key words:** Reconfigurable Architecture, High-Performance, Low-Power, UMTS

## 1. INTRODUCTION

In addition to the high performance requirements (more than 12 GOPS) inherent to multimedia processings or to access techniques such as W-CDMA (Wide-band Code Division Multiple Access) and to the low-energy constraints associated to portable devices (less than 500 mW), a third generation mobile communication system (Figure 1) brings new constraints to the semiconductor design world. In particular, the success of the Universal Mobile Telecommunication System (UMTS) will be linked to a greater flexibility of the standard than that of the Global System for Mobile Communication (GSM) or Interim Standard (IS)-95.

In fact, since UMTS integrates all the current generation networks, different types of processing will have to be supported by multimedia terminals. For example, speech signals has to be coded according to the GSM norm with an Enhanced Full Rate (EFR) coder but also with a more

powerful and adaptive AMR (Adaptive Multi Rate) coder which is recommended for the UMTS. At the same time, a multimedia terminal will have to support the evolution of different standards and services, mentioned in Figure 1, and the integration of new services which still have to be imagined.



*Figure 1. Block diagram of a third-generation transmission system*

From an architectural point of view, furthermore this kind of flexibility which is usually referred to as software, the third-generation (3G) systems must have another kind of flexibility which is far more problematic: That of the Hardware. In fact, multimedia terminals will have to ensure successively the execution of very different applications in terms of calculation and data access patterns. For example, a Viterbi coder working at the bit level could follow an MPEG-2 coder working on 8-bit data. These processing modifications are then much more problematic since it will be necessary, in order to be efficient, to dynamically adapt the architecture to these changes.

Because of the lack of flexibility in ASICs and the low level of performance associated with the high energy consumption of the DSPs, the reconfigurable architectures are more and more taken into consideration to answer the problems associated with the 3G-telecommunications [11]. However, in spite of the quantity of projects on reconfigurable computing [8], none of them ambition to solve all the problems listed above. Some of these architectures are dynamically reconfigurable, others are multi-granularity or low-power but none of them is really adapted to this application domain.

Among these projects, some architectures related to our work can however be distinguished. The Pleïades project for example [1] is an architecture template supporting various granularity of calculation. Although this architecture has been designed under low-power constraints, it does however not met all our requirements. In fact, even if this architecture associates low-energy and high-performance, its flexibility is limited since it is domain specific. The dynamic reconfiguration proposed by the Chameleon™ processor [13] allows a flexibility which does not limit its use to dedicated processings. This architecture, thanks to the large amount of operators that are integrated, supports the 3G-telecommunication complexity

but in spite of its flexibility and its performances, it is unusable in an embedded system because of its high-energy consumption.

Moreover these two examples, many other projects on reconfigurable computing are based on the use of FPGA. Some of these projects, like GARP [9] or NAPA [12], associate a reconfigurable circuit to a programmable processor or controller. Furthermore, other architectures such as Piperench [6] or RAPID [4] can be reconfigured at a higher level in a more efficient way, respectively at the operator and at the functional level. However, they do not meet all the requirements that come from the application domain. In fact, some are low power, others are very flexible or very powerful, but none of them associates the performance to the flexibility and the energy efficiency needed by portable multimedia terminals.

In order to answer to the overall problem of 3G telecommunications, a new architecture, called DART, is proposed. The aim of this paper is to present this architecture and to demonstrate its potential via some implementation examples. The next section focuses on the DART architecture. In the section 3, on the basis of key component study, we will estimate the level of performance and the energy efficiency of DART as well as its adequacy with next generation telecommunication domain. This paper will finally evoke the software tools associated to DART in the conclusion.

## 2. THE DART ARCHITECTURE

So that the user can use only one development platform and do not have to worry about the interfacing of the architecture with the rest of the system, DART is fully autonomous. This architecture has been designed to get the programming model as simple as possible. This is obtained by organizing the architecture into a hierarchy which allows the partitioning of the development flow. This hierarchy concerns the calculation, the storage, the interconnection and the control resources.

### 2.1 Processing primitives

The diversity of the calculation granularity levels in a 3G data processing sequence led us to integrate two kinds of operator in DART. For bit-level operations, we use an FPGA core, which allows a reconfiguration at the gate level. For arithmetic processings, we use some Reconfigurable DataPath (DPR), with a reconfiguration at the functional level. In order to optimize the architecture according to the application, we exploit the dynamic reconfiguration at both level of granularity.

The arithmetic processing primitives in DART are the DPRs fitted in Figure 2. They are organized around functional resources and memories, interconnected according to a very powerful communication network.

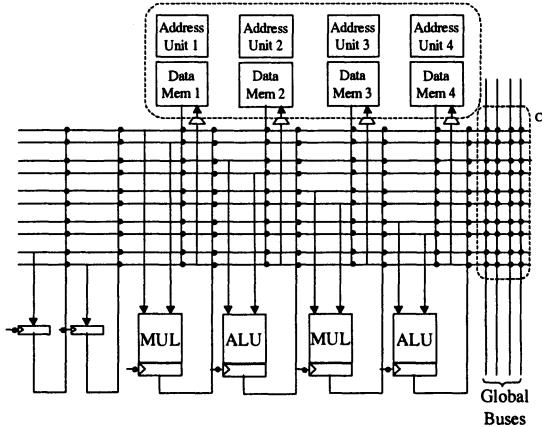


Figure 2. *Architecture of a DPR*

Every DPR have 4 functional units followed by a register, namely 2 multipliers ( $16 \times 16 \rightarrow 32$ ) and 2 reconfigurable ALUs ( $32+40 \rightarrow 40$ ), supporting Sub-Word Processings. They are working on data stored in 4 local memories ( $256 \times 16$ bits) which permit 4 read/write per cycle. In addition to these memories, 2 registers are also available in every DPR. These registers are particularly useful for data flow oriented applications where the different functional units are working on the same data flow but on samples delayed from one iteration to the following. In fact, in that case, these registers will be used to build delay chains that will allow the time-sharing of the data. All these resources are connected via an entirely connected network. The right part of the Figure 2 facts also of appearing some connections with global buses which permit to connect several DPRs for the massively parallel processings.

For the low-level processings, DART integrates an FPGA core. It will, for example, be very effective for the generation of Gold or Kasami code in W-CDMA [5], just as for channel coding algorithms. The operators are now LUTs, dynamically reconfigured as well as their interconnection network.

## 2.2 Cluster organisation

The processing primitives previously mentioned are integrated within the clusters of DART represented on figure 3. They integrate an FPGA core and 6 DPRs interconnected via a segmented network. Thus, every DPR can work independently from the others or be connected to them thanks to Switching Boxes that allow some flexibility in their connections. Every cluster also

integrates a data memory which is shared between all the processing elements, and a configuration memory dedicated to the FPGA. For its last, the reconfiguration will be done in a serial manner, thanks to the DMA controller.

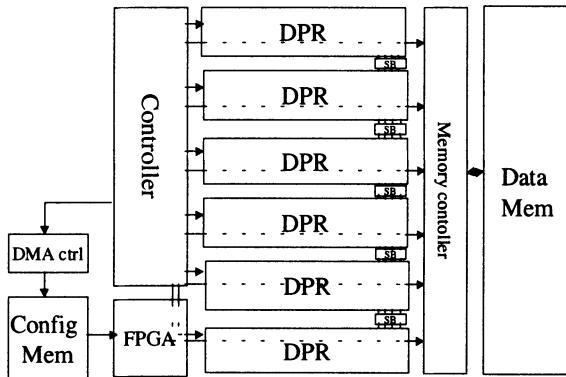


Figure 3. Architecture of a cluster

The cluster controller manages the DPRs and their reconfigurations. Its primary task is so to sequence the reconfiguration instructions of the cluster. Its architecture is similar to that of a controller in a typical DSP processor. However, it sequences configurations rather than instructions and so, it does not need to access an instruction memory at every cycle, but only when a reconfiguration occurs. This allows very significant energy savings.

## 2.3 Dynamic Reconfiguration

| Resource     | Bits/resource | Resource/DPR | Bits/DPR   | Bits/cluster |
|--------------|---------------|--------------|------------|--------------|
| Interconnect | 92            | 1            | 92         | 592          |
| Clock guard  | 1             | 10           | 10         | 60           |
| Multiplier   | 3             | 2            | 6          | 36           |
| ALU          | 11            | 2            | 22         | 132          |
| <b>total</b> |               |              | <b>130</b> | <b>820</b>   |

Table 1. Reconfiguration instruction width

The DPRs are dynamically reconfigured due to instructions carried out from the cluster controller. The Table 1 resumes the targets of the reconfiguration and the number of bits necessary to this operation. This table shows that the dynamic reconfiguration of a cluster will require more than 800 bits. Wishing to be able to reconfigure the cluster in one cycle, to be effective in both regular and irregular applications, a disproportionately large amount of memory would have to be used. A more thorough examination of Table 1 led us to discern two kinds of processing.

### 2.3.1 Hardware reconfiguration

The regular processings are typically those that can be found in loop kernels. They are realized during long periods of time and are composed of very few operations. In order to optimize the datapath for the calculation pattern during these regular processings, a hardware reconfiguration will be done. The configuration being used the time of the processing, its modification is very occasional and it can so require a large amount of data without disturbing the execution of the entire processing. Moreover, the reconfiguration periods of the different DPRs in the cluster will be disjointed unless these DPRs are executing the same task.

The reconfiguration can so concern only one DPR per cycle without lowering the performances. This property allows the controller to manage only one instruction per cycle and so to be less complex while authorizing the simultaneous reconfiguration of several DPRs in the cluster. In that case, every DPR concerned by the reconfiguration will have its datapath optimized for the same processing pattern. We define this as the *Single Configuration Multiple Data* (SCMD) concept. This hardware reconfiguration will require between 4 and 9 instructions of 52-bit width.

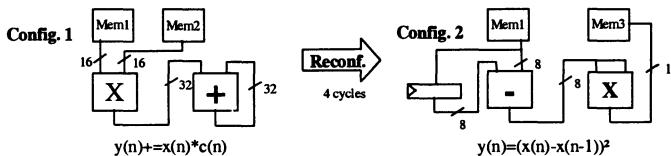


Figure 4. Hardware reconfiguration

This kind of configuration can for example be illustrated by the Figure 4. In this figure, the datapath is optimized at first in order to compute a filtering based on Multiply-ACcumulate operations. Once this configuration has been specified, the computation model is of dataflow type and no other instruction memory readings are done during the time of the filtering. At the end of the computation, after a reconfiguration step which needs 4 cycles, a new datapath is specified in order to be in adequacy with the computation of the square of the difference between  $x(n)$  and  $x(n-1)$ . Once again, no control is necessary to conclude this processing.

### 2.3.2 Software reconfiguration

On the other hand, all the processings in UMTS are not regular and DART must be able to execute very different processings, from one cycle to the following. The efficiency of the processing is thus less important since it will be executed once. This property imposes to minimize the amount of data

necessary to the reconfiguration and therefore, the flexibility of the DPRs. It has been decided to adopt a calculation pattern of *Read-Modify-Write* type, such as those that are used in conventional DSPs.

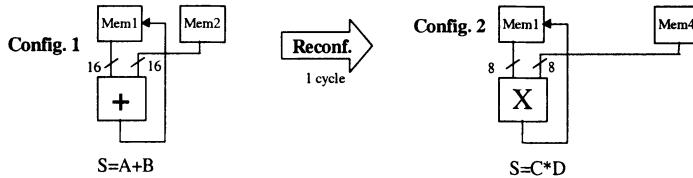


Figure 5. Software reconfiguration

This software reconfiguration thus concerns only the functionality of the operators and the origin of the data handled by the application. Thanks to these flexibility limitations, the DPR may be reconfigured at each cycle with only one 52-bit instruction, as illustrated in Figure 5. Like in hardware reconfiguration, the controller handles only one instruction per cycle which is sufficient since irregular processings have little parallelism.

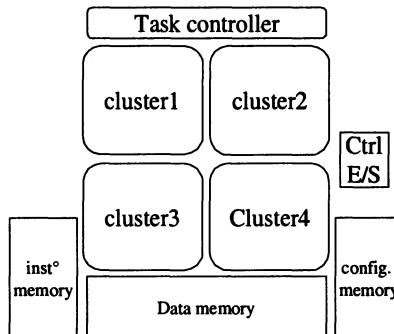
## 2.4 The address generation units

Since the controller task is limited to the management of the reconfigurations, DART must integrate some dedicated resources for address generation. These units must provide the addresses of the data handled in the DPRs for each memory during the dataflow tasks. In order to be efficient in a large variety of application, they support numerous addressing pattern (bit-reverse, modulo, pre/post increment,...). These units are built around a module in charge of sequencing the accesses to an instruction memory (64x16-bits). In order to minimize the energy consumption, these accesses will take place only when an address has to be generated. For that, the sequencer may be put in Wait State thanks to an instruction which will moreover specify the number of Wait State. Another module is then in charge of waking up the sequencer after the number of cycle specified in the instruction. Even if this method needs some additional resources, its interest is largely justified with the energy savings.

Once, the instruction has been read, it is decoded in order to control a small datapath which will supply the addresses. On top of the four address generation units of each DPR (one per memory), a module will provide a zero-overhead loop support. Thanks to this module, up to four levels of nesting will be supported, each loop kernel being able to contain up to eight instructions, without any additional cycle for their management.

## 2.5 System level architecture of DART

As it has been said below, DART is fully autonomous. Hence, DART integrates a task controller which manage 4 clusters accessing a same data memory space. At this system view (Figure 6), the clusters can now be considered as the processing primitives of DART and the task controller is in charge of assigning the different tasks to be executed on the clusters according to urgency and resource availability constraints. It has to support a Real-Time Operating System. The configurations of the clusters are realized dynamically, since the task controller as only to specify to the cluster controller, which task has to be executed. The configuration support is thus only an address bound which corresponds to the location of the program in the cluster memory. In the same time, the data have obviously to be loaded into the cluster memories.



*Figure 6. System level architecture of DART*

## 3. DART EVALUATION

In order to validate the architecture and to evaluate its potential, some key applications of the UMTS have been implemented on DART. To estimate the performance and the energy efficiency of DART, its main parts have been synthesized on a 1.95V ST 0.18 $\mu$ m technology thanks to the Synopsys design tool framework. These synthesis have permitted to extract the key features of our design and to integrate them into the DART simulator which has been developed in systemC. The results described bellow are coming from this simulator.

### 3.1 Some key applications

Because of the complexity of W-CDMA [10], its implementation in a multimedia terminal has to be very effective. In order to test DART for such application, a subset of the norm has been implemented, a finger of a rake receiver which realizes the complex despreading, represented by the equations 1 and 2, with a spreading factor of 256.

$$I_{NB} = \sum_{j=0}^{SF-1} I_{WB} * C_P(j + \tau_i) + Q_{WB}(j) * C_Q(j + \tau_i) \quad (1)$$

$$Q_{NB} = \sum_{j=0}^{SF-1} I_{WB} * C_Q(j + \tau_i) + Q_{WB}(j) * C_P(j + \tau_i) \quad (2)$$

To illustrate the video processings we also have implemented a Discrete Cosine Transform, which is working on 8x8 pixels MacroBlocs, since this kind of algorithm is nearly systematic in video compression's standards like MPEGx or H.26x [7]. The 2 loop kernels of this algorithm are based on a *Multiplication-ACcumulation*. For MacroBloc Raws, the loop kernel is :

$$value[y] = \sum_{j=0}^7 coef[y][x] * bloc[j + x * 8] \quad (3)$$

The multimedia terminals of third generation will still be used for the transmission of speech between two distant persons and so need very effective speech coders [2]. To illustrate these kind of processings, we have implemented an autocorrelation, on a signal of 240 samples, preamble of Levinson-Durbin algorithm which permits the adaptation of the prediction filter coefficients in speech coders such as the EFR or the AMR. The mathematical description of this processing is given by equation 4.

$$r(p) = \sum_0^{239} x(n) * x(n - p) \quad \text{for } p \in [0..239] \quad (4)$$

### 3.2 Experimentation results

The Table 2 resumes the implementation results and reveals the potential of DART in two ways. The columns of this table specify for each application: the number of DPRs needed for the implementation; its number of operation; the number of execution cycles; the number of accesses to the instruction and to the data memories and finally the energy consumed for the execution of the algorithm.

| Application         | DPR | Operations | Cycles | Inst. Read | Data read | energy  |
|---------------------|-----|------------|--------|------------|-----------|---------|
| Complex Despreading | 2   | 2048       | 258    | 4          | 1032      | 435.8nJ |
| DCT 2-D             | 4   | 2048       | 85     | 6          | 1088      | 60.3 nJ |
| Autocorrelation     | 6   | 57600      | 2543   | 43         | 5040      | 3.15μJ  |

Table 2. Implementation results on DART

### **3.2.1 Performance analysis**

A synthesis of the DPR estimate the operating frequency of DART at 130 MHz. Running at 130 MHz, DART is thus able to provide 260 MMACS ( $260 \cdot 10^6$  Multiplication-ACcumation per second) on each DPR when handling 16 bits data. Integrating 6 DPR per cluster, 1.56 GMACs/cluster can be achieved and these figures are doubled when handling 8-bits data as in video-coding [6]. From an instruction point of view, DART may deliver up to 520 MIPS/DPR or 3.12 GIPS/cluster. As an instruction includes an address generation, a memory access and up to 2 operations per multiplier (1 shift + 1 multiply) or 3 operations per ALU (2 shift + 1 ALU operation), this may be translated in 10.9 16-bit GOPS/cluster or 18.7 8-bit GOPS/cluster.

On each application previously quoted, the flexibility of the DPRs permits to obtain very good performances. The Table 2 shows for example that a finger of a rake receiver may be implemented on 2 DPRs which is allowing us to integrate 3 fingers in each cluster for a processing power bigger than 3.6 GOPS.

The connection of the DPRs being made thanks to a segmented mesh network, they can work independently or together. This property may be particularly useful. For example, to compute simultaneously a 2-D DCT and a finger of a Rake Receiver, within a same task, it would be possible to have the two elementary computations running concurrently on a single cluster. On the other hand, the massively parallel processings such as the autocorrelation should occupy all the DPRs, for a very effective execution.

### **3.2.2 Energy consumption analysis**

If there are several architectures that can reach such level of performance e.g. the Chameleon [3], DART differs from its competitors by allowing a very significant energy saving thanks to the data sharing and the memory accesses minimization. In fact, the energy efficiency of ASIC is obtained by integrating operators that are not any larger or more complicated than they need to be and so, that consume a minimum of energy. Moreover, since they realize only one operation, the energy overhead of the instruction distribution is avoided. On the contrary, to be flexible, the programmable processors need general-purpose circuit blocks that are more complex to support the execution of several operations. Moreover, to control these general-purpose units, an instruction has to be read and decoded at each cycle. Since the instructions and the data are stored in very large memories, the energy waste in data access and control distribution widely exceed that which is useful to perform a computation. These two points are however, much less problematic in DART as shown in Table 2.

This table shows that only 43 instructions permit the control of the autocorrelation. On a conventional DSP processor more than 57,000 readings from the instruction memory would have to be done. Given the cost in energy of a memory access, the gain in energy consumption is therefore very important.

The second source of energy savings in DART is the data sharing. For example, on the autocorrelation, DART allows to divide by 12 the number of accesses to the data memory and so the energy waste due to this accesses (which is typically very high). This data sharing is made easier by the high degree of flexibility of the interconnection network and the use of the registers in data flow oriented applications.

In addition to the minimization of the memory accesses, the energy consumption of DART is lowered by the minimization of transistor activity. This point is essential in an architecture like DART since it must integrate, in order to be effective, a large amount of resources. However, all these resources will not be used at every cycle. DART has so to avoid making them to work when they are not in use to the execution thanks to guarded clocks. The energy is also saved in DART by optimizing the operators at the bit level and by scaling the voltage and the operating frequency of the clusters according to the complexity of the task to be implemented.

Hence, DART provide more than 9.2 MIPS for each mW consumed for 16-bit operations and about 15.8 MIPS for Sub-Word Processings. In an operation point of view, DART may deliver up to 32 MOPS for each mW consumed during the 16-bit operations and up to 55 MOPS for the Sub-Word Processings. It has to be noted that these 9.2 MIPS/mW are obtained in the worst, i.e. when at each cycle, each memory is accessed, 4 addresses are generated, and where the functional units realize an arithmetic operation on 16-bit data. This is however not the typical case since one of the main advantages of this architecture is to allow a large amount of data sharing. Practically, the energy consumption of the implementations described below are between 11.6 and 16.7 MIPS/mW.

## 4. CONCLUSION AND FUTURE WORKS

This paper has described a dynamically reconfigurable architecture for portable multimedia terminals developed within the framework of a project associating the University of Brest and ST Microelectronics, funded by the industry and research French ministry. We have shown, thanks to our implementation examples, that the use of a massively parallel architecture can be compatible with low-energy considerations. The next stage is to provide tool support for the development flow. This tool, which is in

development, is built around a retargetable compiler developed at the IRISA, CALIFE [14], able to supply the software configurations and on a behavioural synthesis framework, GAUT [15], developed at the laboratory which is in charge of generating the Hardware configurations. The codes provided by this tool can then be validate on the DART systemC simulator.

## 5. REFERENCES

- [1] A. Abnous and J. Rabaey. Ultra low-power specific multimedia processors. VLSI Signal Processing IX, pages 459-468, November 1996
- [2] T. Amada, K. Miseki and M. Akamine. CELP speech coding based on an adaptative pulse position codebook. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1999
- [3] Chameleon Systems. Wireless Base Station Design Using Reconfigurable communications Processors. Technical report, 2000.
- [4] D. C. Cronquist, P. Franklin, C. Fisher, M. Figueroa, and C. Ebeling. Archi-tecture Design of Reconfigurable Pipelined Datapath. In Advance Research in VLSI, 1999.
- [5] E. Dinan and B. Jabbari. Spreading Codes for Direct Sequence CDMA and Wideband CDMA Cellular Network. IEEE Communications Magazine, 1998.
- [6] S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. R. Taylor. PipeRench: A Reconfigurable Architecture and Compiler. IEEE Computer, April 2000.
- [7] L. Hanzo and E.-L. Kuan P. Cherriman. Interactive cellular and cordless video telephony : State-of-the-art system design principles and expected performance. Proceedings of the IEEE, 2000.
- [8] R. Hartenstein. A Decade of Reconfigurable Computing : A Visionary retro-spective. In Design Automation and Test in Europe, 2001.
- [9] J. Hauser and J. Wawrynek. GARP:A MIPS processor with a reconfigurable coprocessor. In IEEE Symposium on FPGA-based Custom Computing Machines (FCCM), June 1997.
- [10] T. Ojanpera and R. Prasard. Wideband CDMA For Third Generation Mobile communication. Hartek Publishers, 1998.
- [11] J. Rabaey. Reconfigurable Processing : The Solution To Low-Power Pro-grammable DSP. In ICASSP, April 1997.
- [12] C. Rupp, M. Landguth, T. Graverick, E. Gomersall, and H. Holt. The NAPA Adaptative Processing Architecture. In FCCM, April 1998.
- [13] X. Tang, M. Aalsma, and R. Jou. A compiler directed approach to hiding configuration latency in chameleon processors. In International Conference on Field-Programmable Logic and Applications, April 2000.
- [14] F. Charrot and V. Messé. A Flexible Code Generation Framework for the Design of Application Specific Programmable Processors. In International Symposium on Hardware/Software Co-Design,1999.
- [15] O. Sentieys, J.P. Diguet and J.L. Philippe. GAUT : a High Level Synthesis Tool Dedicated to Real Time Signal Processing Application. EURODAC, September 1995.

# Dynamically Reconfigurable Architectures for Digital Signal Processing Applications

Gilles Sassetelli, Lionel Torres, Pascal Benoit, Gaston Cambon,  
Michel Robert, Jérôme Galy

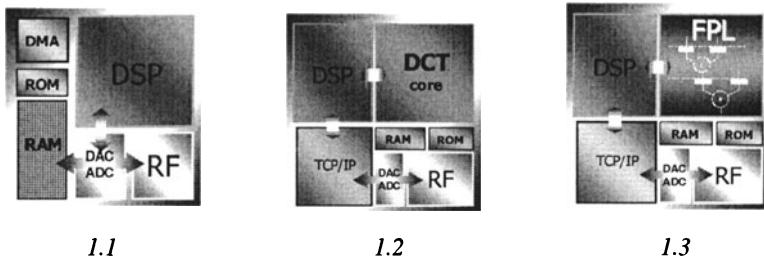
*LIRMM, UMR University of Montpellier II-CNRS C5506,  
161 rue Ada, 34392 Montpellier Cedex 5, France  
{sassate torres diou cambon robert galy}@lirmm.fr*

**Abstract:** Tomorrow's pocket devices will all have Internet-based communication capabilities. The advent of mobile phones, PDAs (Pocket Data Assistant) and pocketPC's joint to the newcomer's third generation wireless networks such as UMTS will soon allow everyone to be connected, everywhere. In this competitive marketplace where many similar products compete for the consumer attention, performances level is a very important criterion. Videoconferencing, digital music broadcast, speech recognition are a few example of the new features allowed by the new third generation networks. This kind of multimedia, data oriented content requires highly efficient architectures; and nowadays mobile system-on-chip solution will no longer be able to deal with the critical constraints like area, power, and data computing efficiency. In this paper we will propose a new dynamically reconfigurable network, dedicated to data oriented applications such as the one targeted on third generation networks. Principles, realisations and comparative results will be exposed for some classical applications, targeted on different architectures.

**Key words:** Reconfigurable computing, Data flow, Digital Signal Processing

## 1. INTRODUCTION

Nowadays pocket devices are mostly based on a SoC (System on Chip) approach (Figure 1). On the same silicon die are grouped heterogeneous IP (Intellectual Property) modules.



*Figure 1. The three main SoC approaches*

There are different ways to face these new problems:

- The easiest, and actual way to deal with this increasing computing requirements is naturally to use a more powerful DSP/ $\mu$ P (figure 1.1) than the ones used today; but it will probably not be feasible for the most demanding applications, as the resulting processor will grow until the size of a Pentium (such as the ones which take place in the most powerful PDA or pocket PCs), with the corresponding area, cost and consumption problems.
- Another way is to try to identify the future application field, and use a dedicated core to compute the common parts of the corresponding algorithms (Figure 1.2). For example, if JPEG and MPEG based applications are targeted, we will make the choice of implementing a wired IDCT (Inverse Discrete Cosine Transform) core, which is known to be the common most time consuming part of both algorithms[7][8]. An interesting, but restrictive solution as the application field is thus not extensible.
- Yet another way is the reconfigurable computing [2][3][10]. For example, integrate a FPGA core[1][3], where, depending on the target application different algorithm/architecture solutions could be synthesized (figure 1.3). Here, if we target JPEG applications, we will choose to synthesize the IDCT core in the FPGA, and also an application dependant part of the algorithm, like Huffman coding, or quantization. But in the other way if we target MPEG[9] applications, we will still make the choice of a wired IDCT, but this time we will also select the motion estimation[6], which is one of the most demanding part of the MPEG.

This kind of approach seems to be quite interesting; we can thus imagine, depending on a given application, a video streaming one for example, that the mobile could directly connect to the vendor's site to download the corresponding applet, which is nothing else than the configuration file of the considered reconfigurable network.

## 2. RECONFIGURABLE SOLUTIONS

A closer look to the kind of tomorrow's mobile applications shows a very data oriented, data intensive trend: the multimedia content needs a very high number of arithmetic operations; which would naturally imply to synthesize numbers of arithmetic operators in case of using fine grained reconfigurable logic (FPGA for example).

Arithmetic operators synthesis is known to be very area costly on fine grained reconfigurable networks. Due to the highly combinational character of adders and multipliers, the resulting functional frequencies are also often very low making FPGA-like architectures bad candidates for arithmetic level data computing.

Coarse grained reconfigurable architectures[2][3] featuring hardwired arithmetic operators are much more adapted to dataflow oriented computations.

## 3. SYSTEM OVERVIEW

Our architecture follows the classical bi-layer FPGA principles. Here are the main characteristics:

- The operative layer is no longer CLB (Configurable Logic Block) based, but use a coarse-grained granularity component: The Dnode (Data node). It is a datapath component, with an ALU and a few registers, as shown in figure 3. This component is configured by a microinstruction code.
- The configuration layer follows the same principle as FPGAs, it's a RAM which contains the configuration of all the component of the operative layer.
- We also use a custom RISC core [5] with a dedicated instruction set; its task is to manage dynamically the configuration of the network and also to control the data transfers between the reconfigurable core and the host CPU.

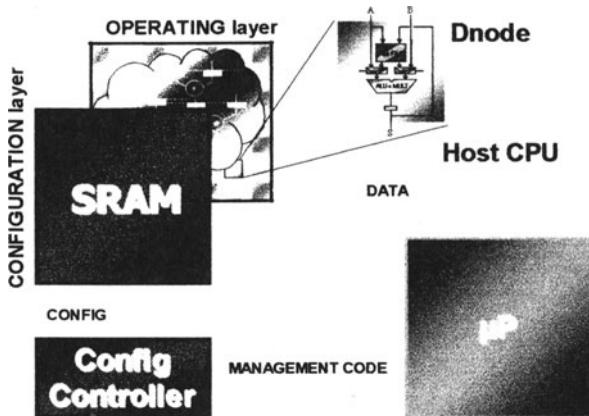


Figure 2. System Overview

This architecture is thus not intended to be a stand-alone solution, rather an IP core for dataflow oriented computing, which would take place in a SoC. Figure 2 shows schematically our system in a SoC context. The  $\mu$ P can thus confide the most demanding part of a given application to our IP core. So it downloads to the RISC memory the corresponding configuration program (which manages the dynamical reconfiguration).

From a functional point of view:

- The host processor first sends the management code to the configuration controller memory (the custom RISC has its own program memory). This is a object code, ready to be executed, and specially designed to manage dynamically the configuration of the network (the content of the RAM thus changes from one cycle to another), as to say, the functionality of the operating layer. Each clock cycle, the configuration controller is able to change up to the entire content of the RAM thanks to its dedicated instruction set.
- Once done, our core is ready to compute. The host processor sends the data to the operating layer via a specific scheme and then get back the computed data. As the configuration is dynamically managed, it is possible to multiplex the sent data, and to compute them by several sequential (hardware multiplexing) or concurrent (static) synthesised datapaths.

#### 4. OPERATING LAYER ARCHITECTURE

In this section we will describe more precisely the operating layer architecture.

## 4.1 4.1 Dnode architecture

It essentially consists in an ALU-Multiplier, able to make all the classical arithmetic and logic operations : addition, multiplication, subtraction, roll, shift and so on. This optimised architecture is able, in the same clock cycle, to make all possible operations, even between two different registers. Its corresponding microinstruction code, the configuration code, comes from a memory location in the configuration layer. As previously said, this code evolves during the computing phase, the functionality can thus be changed from one clock cycle to another, from an addition to a multiplication, load to register, etc.

Each Dnode has in fact two execution modes :

- Global mode (normal mode), already described : the Dnode executes the microinstruction code which comes from the configuration layer, managed by the RISC configuration controller.
- Local mode : The stand-alone mode : Each Dnode has 7 registers, a up to 6-states counter and a 6 to 1 multiplexer forming a small local controller. Each one of the 6 first registers can contain a Dnode microinstruction code, and each clock cycle the counter increases the value on the multiplexer address input, thus sending the content of a register to the datapath part of the Dnode.

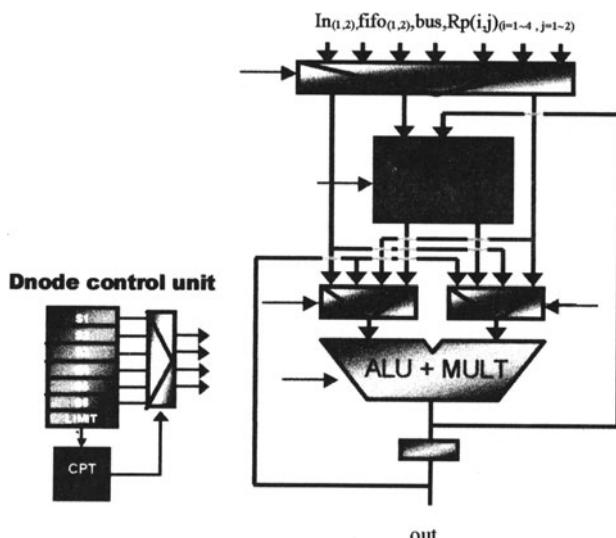


Figure 3. The Dnode Architecture

In this last mode the Dnode is like a basic RISC CPU able to compute various (otherwise control intensive) algorithms like MACs, serial digital filters, FIFO/LIFO emulation. This scheme, joint to a specific input/output Data controller allows very efficient, high bandwidth dataflow oriented computation.

## **4.2 The ring architecture**

Related works[1][2][3] propose mesh, array or crossbar-based operating layer architecture.

Mesh-based architectures[2][3], even very flexible usually suffers from routing problems. Each reconfigurable block must features full routing capabilities with the nearest neighbours for direct communications. Routing over longer distances are achieved by dedicated lines, and with new silicon technologies allowing giant reconfigurable architectures, this requires important routing capabilities, with no-more maintainable propagation delays (general SoC problem, die-long interconnections cause hard timing problems).

Crossbar based arrays[3]. The routing capabilities are again usually quite satisfying, but area costly. The scalability of these architectures is also limited for the same reasons as mesh-based networks; and more specially FPGAs. The largest ones are facing propagation delay problems implying P&R tools to spend lot of time in routing phase.

Linear array-based architectures[3]. Aiming to map pipeline character of datapaths, they are often bi-dimensionals. Feedback operations (opposite dataflow direction, figure 5) of all kinds of digital signal processing like algorithm require additional routing resources and are often area and performance costly thus limiting the scalability for next generations.

Our approach proposes an original linear array like architecture to solve routing relative problems. This one is based on curled bi-datapath structure.

### **4.2.1 Forward : The main Dataflow**

We use a curled, pipelined systolic structure as shown in figure 4. All the Dnodes form a ring, which length (Dnodes layers number) and width (Dnodes per-layer number) can easily be scaled.

We use a curled, pipelined systolic structure as shown in figure 4. All the Dnodes form a ring, which length (Dnodes layers number) and width (Dnodes per-layer number) can easily be scaled.

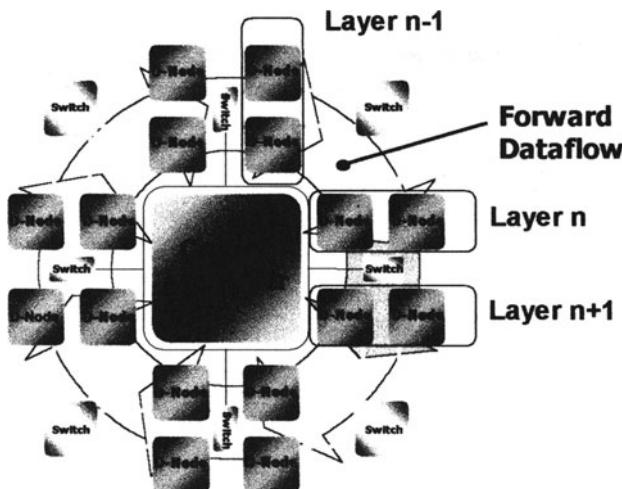


Figure 4. The ring architecture

The Dnodes are organised in layers; a Dnode layer is connected to the two adjacent ones by also dynamically reconfigurable switch components able to make any interconnection between two stages. These switches also manage data transfers with the host by dedicated FIFOs, and optional RISC communications via a shared bus.

In normal mode, each Dnode can be seen as an arithmetic operator of a datapath which computes a data each clock cycle. In stand-alone mode each Dnode can be seen as an autonomous CPU. The structure is also flexible in the way that all Dnodes have not to run in the same mode, allowing the Systolic Ring to compute either in global mode (normal mode), local mode (stand-alone) or hybrid (normal and stand-alone) mode.

#### 4.2.2 Reverse : The secondary flow

The data feedback problem is addressed here: we use special feedback pipelines (figure 5), forming a reverse Dataflow to avoid complex routing structures. The last task that accomplishes each switch is to write unconditionally (no control needed) the computed result of the previous D-node layer in a dedicated pipeline (each switch owns its pipeline), which allows the feedback of each data to the previous stages. These ones can then choose to get these data through the switches, which have direct access to all the pipelines. This technique ensures a good scalability of the architecture, as the routing problem is thus removed.

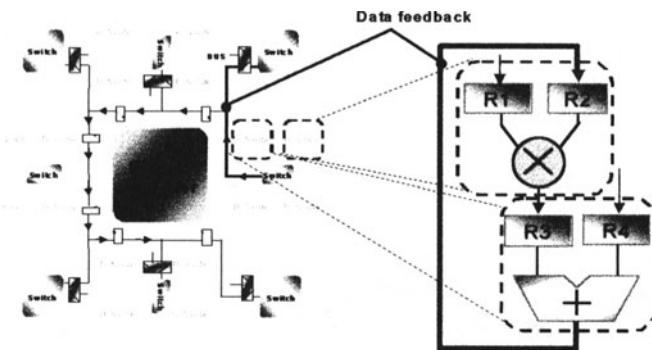


Figure 5. The feedback network

## 5. COMPARISONS & REALISATIONS

### 5.1 Comparative Results

A 8 Dnodes version has a maximal computing power of 1600 MIPS at the typical 200 MHz evaluated functional frequency, quite impressive compared to the 400 MIPS of a Pentium II 450 MHz processor. The theoretical maximum bandwidth of this version of the structure is about 3 Gbytes/s, however often limited by the communication protocol between the host CPU and the core. To program this structure we wrote an assembling tool, which parse both configuration controller level (for the control) and Ring level assembler primitives. It directly generates the machine object code, ready to be executed in the architecture.

#### 5.1.1 Motion estimation algorithm implementation

In the application field targeted by third generation systems we can find lots of video-relative techniques. One of these well known computing intensive algorithm is the motion estimation. Widely used in video compression techniques for broadcasting, storing, and videoconferencing, his task is to remove the temporal redundancy in video streams, as the DCT's is to remove the spatial redundancy.

Block matching and specially Full Search Block Matching (FSBM) algorithm is the most popular implementation, also recommended by several standard committees (MPEG (video) and H.261 (videoconferencing) standards).

The Mean Absolute Difference (MAD) criterion, used to estimate the matching of the current block can be formulated as follows:

$$MAD(m, n) = \sum_{i=1}^N \sum_{j=1}^N |R(i, j) - S(i + m, j + n)|$$

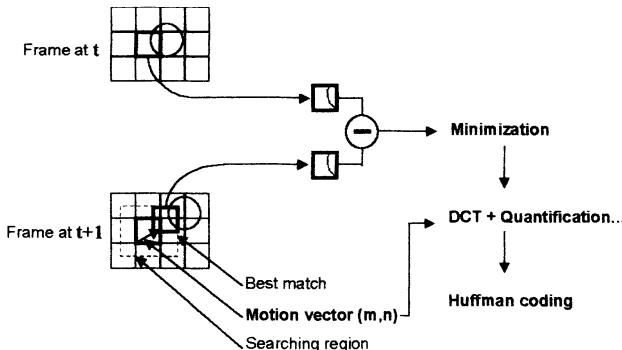


Figure 6. The motion estimation algorithm

$R(i,j)$  is the reference block (figure 6) of size  $N \times N$  and  $S(i+m,j+n)$  the candidate block within the search area determined by  $p$  and  $q$  which are the maximum horizontal and vertical displacements. The size of this area is  $(N+p)(N+q)$  pixels; and the displacement vector represented by  $(m,n)$  is determined by the least  $MAD(m,n)$  among all the  $(p+1)(q+1)$  possible displacement within the search area.

Let's consider the following common specifications: An image size of 352 x 240 pixels at 15 frames/s with a block size of 8 x 8 pixels and a maximum displacement of 8 pixels in horizontal and vertical directions.

For each candidate block the first summation ( $j=1$  to 8) requires  $N$  operations and the accumulation  $N-1$  operations, thus a total of  $2N-1$  operations. The second summation requires to compute  $N$  times the previous one account of operations and again  $N-1$  operations for the accumulation of the partials sums. The total amount of arithmetic operations to compute is so  $2N^2 - 1$ .

The  $(2N-1).N$  first operations can be achieved within  $(2N-1).N / (0,75.Nx)$  clock cycles in a  $Nx$  Nodes version of our structure, as there are no dependencies on these data and one node over four is in wait state (layer  $n$ : 2 nodes computing two  $R() - S()$  operations; layer  $n+1$ : 1 node accumulating of the two previous computed results).

The last N-1 operations (accumulation) are achieved in  $\text{int}(\ln(N))+1$  clock cycles for  $N \leq Nx$ .

In a 16 Nodes version of our structure, and with the previous specified codec ( $N=8$ ) the computation of the MAD for a candidate block requires 13 clock cycles. Each reference block requires the computation of 289 candidate blocks and there are 1320 reference blocks in each frame. The total processing time of an image frame is  $1320 \times 289 \times 13 = 4959240$  cycles. At the 200MHz estimated frequency the computation time would be 24ms, which is two times smaller than the frame period (1/15s).

Table 1 shows the performances of the Systolic Ring compared with the ASIC architecture implemented in [12] and Intel MMX instructions[13] using the criterion of the number of cycles (the three architectures can achieve comparable functional frequencies) needed for matching a 8x8 reference block against its search area of 8 pixels displacement.

| Systolic Ring | ASIC[12] | MMX [13] |
|---------------|----------|----------|
| 3757          | 581      | 28900    |

*Table 1 : Motion Estimation performance comparison (cycles)*

Our structure shows again its efficiency in a such computing intensive context. The ASIC implementation is much faster than our solution at the price of flexibility: The Systolic Ring provides the advantage of hardware reuse and is also almost 8 times faster than a MMX solution.

## 5.2 Synthesis results & future work

The entire architecture (reconfigurable core and configuration controller) has been described in both behavioural and structural VHDL. A 8 Dnodes, 16 bits data width version has been fully simulated, and synthesised in both HCMOS7 and HCMOS8, respectively  $0.25\mu\text{m}$  and  $0.18\mu\text{m}$  ST technology. Table 3 shows the comparative synthesis results in both technologies.

|                     | <b><math>0.25\mu\text{m}</math></b> | <b><math>0.18\mu\text{m}</math></b> |
|---------------------|-------------------------------------|-------------------------------------|
| Dnode area          | $0.06 \text{ mm}^2$                 | $0.04 \text{ mm}^2$                 |
| Ring-8 area         | $0.9 \text{ mm}^2$                  | $0.7 \text{ mm}^2$                  |
| Estimated Frequency | 180 MHz                             | 200 MHz                             |

*Table 3 : Synthesis Results*

The low area of each Dnode, joint to the exposed specific architecture shows that this one could easily be scaled to larger realizations. Figure 7 shows a foreseeable  $.18\mu\text{m}$  technology,  $12 \text{ mm}^2$  die area SoC for high constrained embedded solution. Our specific architecture allows the

integration of a powerful 64 Dnodes version of our core ( $3.4 \text{ mm}^2$  on-die area) with a widely used ARM7 CPU, able to run various operating systems like windows CE, Linux. This kind of solution could provide a great computation power/cost ratio, which combines the flexibility of a CPU / reconfigurable architecture couple with the efficiency of applications dedicated cores.

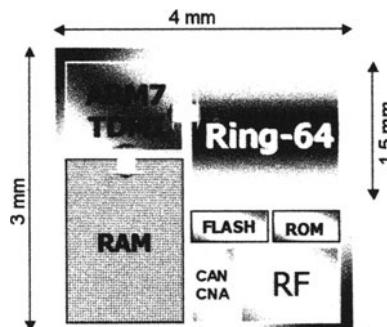


Figure 7. A foreseeable SoC

## 6. CONCLUSION

We have proposed a new coarse grain dynamically reconfigurable architecture which proves its efficiency in data oriented processing. Its scalability shows that its field of applications is not limited to high-constrained embedded applications, but can also make be worth its faculties in other contexts, where high data bandwidth processing remains critical. A small 8-Dnodes version of this structure already provides up to 1600 MIPS of raw power for data dominated applications with a sustained data rate of 3 Gbytes/s at 200 MHz, either in global or local mode.

Our future work takes place in the translation of the structure to floating point and also in the writing of an efficient compiling tool, the key to industrial success for coarse grain reconfigurable architectures.

## 7. REFERENCES

- [1] Stephen Brown and J. Rose, "Architecture of FPGAs and CPLDs: A Tutorial," IEEE Design and Test of Computers, Vol. 13, No. 2, pp. 42-57, 1996.
- [2] Why reconfigurable computing, Department of Computer Science, Computer Structures Group, <http://xputers.informatik.uni-kl.de/>.
- [3] R. Hartenstein, H. Grünbacher (Editors): The Roadmap to Reconfigurable computing Proc.FPL2000, Aug.27-30,2000;LNCS, Springer-Verlag2000.

- [4] J. R. Hauser and J. Wawrzynek, "Garp: A MIPS Processor with a Re-configurable Co-processor," Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines, 1997.
- [5] A. Abnous, C. Christensen, J. Gray, J. Lenell, A. Naylor and N. Bagherzadeh, " Design and Implementation of the Tiny RISC microprocessor," Microprocessors and Microsystems, Vol. 16, No. 4, pp. 187-94, 1992.
- [6] C. Hsieh and T. Lin, " VLSI Architecture For Block-Matching Motion Estimation Algorithm," IEEE Trans. on Circuits and Systems for Video Technology, vol. 2, pp. 169-175, June 1992.
- [7] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete cosine transform," IEEE Trans. On Computers, vol. C-23, pp. 90-93, Jan 1974.
- [8] ISO/IEC JTC1 CD 10918. Digital compression and coding of continuous-tone still images – part 1, requirements and guidelines, ISO, 1993 (JPEG).
- [9] ISO/IEC JTC1 CD 13818. Generic coding of moving pictures and associated audio: video, ISO, 1994 (MPEG-2 standard).
- [10] High Productivity Computing Systems (HPCS), Defense and Advanced Research Projects Agency, <http://www.darpa.mil/ito/research/hpcs/index.html>.
- [11] Xilinx, the Programmable Logic Data Book, 1994
- [12] A.Bugeja and W. Yang, "A Re-configurable VLSI Coprocessing System for the Block Matching Algorithm", IEEE Trans. On VLSI systems, vol. 5, September 1997.
- [13] Intel Application Notes for Pentium MMX, <http://developer.intel.com/>.

# **Reconfigurable Architecture Using High Speed FPGA**

**L. Kessal, R. Bourguiba, D. Demigny, N. Boudouani, M. Karabernou**  
*ETIS – UPRESA CNRS 8051ENSEA - Cergy Pontoise University 6 Av. du Ponceau, 95014  
Cergy Pontoise Cedex, France E-mail: kessal@ensea.fr*

**Abstract:** Here, we discuss about Dynamic Reconfiguration (or Run Time Reconfiguration), a technique based on the reuse of the same device (an FPGA configured on the fly) by scheduling the execution of different algorithms building an application. Our project joins the efforts of ten laboratories working on methods and tools for Adequation Algorithm Architecture. The design of a hardware template with such a concept, will help the emergence of new methods for applications development and the benefits estimation of this approach. In this paper, we present our architecture, ARDOISE, dedicated to real time image processing. Then an analytical model is defined in order to compute the limits and the performances expected in the use of the dynamic reconfiguration scheme.

**Key words:** dynamic reconfiguration, run time reconfiguration, image processing, fpga

## **1. INTRODUCTION**

Ten research French teams study and build a hardware architecture (ARDOISE) dedicated to real time image processing. This architecture uses fast or dynamic reconfiguration provided in new FPGA circuits. This new kind of devices can be totally or partially configured in less than 1ms and

their processing speed is approximately 1/3 of ASIC speed. The basic idea is to chain algorithms used in image segmentation on the same hardware structure, by reconfiguring few devices, several times during the processing of a single image. This is equivalent to assigning the same hardware resource: an FPGA device to execute a sequence of algorithms according to a defined scheduling. Using this concept, the final system allows the implementation of applications on FPGA as fast as on ASIC circuits. However, it authorizes a same flexibility level, formerly, reserved to microprocessors. On the other hand, this project will help in estimating the contribution of the Dynamic Reconfiguration (DR) paradigm to the performances of hard wired systems.

In this paper, we briefly present the ARDOISE architecture, which can be reconfigured entirely or partially. Next, we discuss (section 3) the performances limits of a Dynamically Reconfigured Architecture (DRA) and give an evaluation of the silicon area gain expected, compared to a classical solution based on a set of statically configured FPGA circuits. In the five section, we compare the performances of three possible implementations of a DRA:

1. one large FPGA is reconfigured at run time,
2. two FPGA are alternatively reconfigured at run time, in order to mask the configuration delays,
3. two FPGA are reconfigured at run time, in parallel.

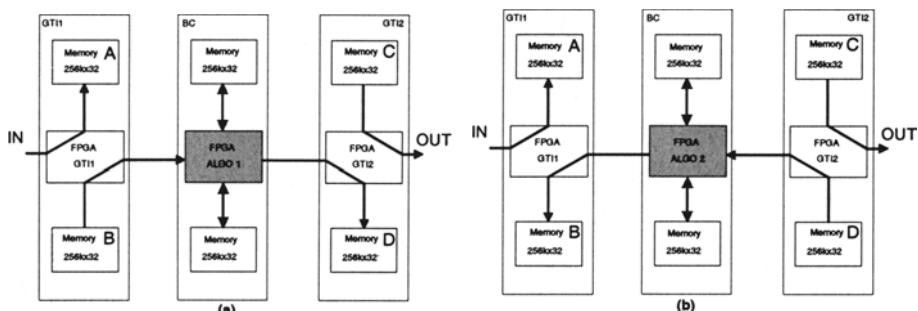


Figure 1. Processing two algorithms on ARDOISE architecture

## 2. THE ARDOISE ARCHITECTURE

The ARDOISE architecture [1,2] is based on three identical modules (figure 1). Each one encloses an FPGA, with a 45K gates count connected to two local memories to store intermediate results.

The GTI1 and GTI2 modules provide an interface with the input and output systems, that allows desynchronization between the central computing module, BC, and the video acquisition system which occurs often at lower frequency than the maximum FPGA abilities. The different treatments are swapped in the central module. Partial results will be stored in the local memories with different memory models, while computing or reconfiguring.

### 3. PERFORMANCES AND LIMITS OF DRA

The main objective of Dynamic Reconfiguration is to allow a system to react during run time and choose the right algorithm at the right time offering performances close to those of the hard-wired systems. Compared to a static solution, DR does not improve the execution speed of an application. However, it reduces and optimizes the use of the silicon area in the FPGA. Bertin suggested [3] the definition of the power  $P_u$  needed by an application in the case of the static architecture as the product of the number of gates  $G_s$  required and of the computing frequency. In the same way, the power of an architecture can be defined as the product of the number of equivalent gates of the architecture and the maximum frequency  $F_t$  that can be used. The application which uses  $G_s$  gates in a static architecture, is splitted in  $C$  parts and then computed on an image with  $C$  configurations of the FPGA. We have already shown [1,2] that there is a limit of the useful application power when using DR:

$$P_{dynamic} = G_d F_t \left(1 - C \frac{G_d}{V_c T}\right)$$

$G_d$  represents equivalent gate count of the dynamically reconfigurable architecture and  $T$  the computing time of a block of data. For an image rate of 25 images/s, the block duration is  $T = 40ms$ .

The limit depends on the configuration speed  $V_c$  (expressed in number of gates configured per second) and of the maximal computing frequency  $F_t$  provided by the technology.

#### 3.1 Maximal power and complexity limits

For a given number of configurations, when  $G_d$  increases, there is a maximum value which can be reached by  $P_u$ :

$$P_{\text{max}} = \frac{F_t G_d}{2} \quad \text{with} \quad G_d = \frac{V_c T}{2C}$$

For each configuration p,  $\beta_p$  data are processed in parallel (parallelism rate). The maximal complexity of an application implemented with DR is then defined:

$$G_s \text{ max} = \frac{V_c T}{2C} \sum_{p=1}^c \frac{1}{\beta_p} \leq \frac{V_c T}{2}$$

This equation shows that there is a limit in the hardware complexity of an application that can be implemented with DR. This limit is exclusively a function of the configuration speed authorized by the technology.

### **3.2 Maximum gate count of a DRA**

If the image size is N pixels, the sampling frequency of the pixel is:

$$F_e = \frac{N}{T}$$

For a given application ( $F_e$ ,  $T$ ,  $G_s$ ), the useful power is:

$$P_{\text{static}} = G_s F_e \leq P_{\text{dynamic}}$$

We obtain:

$$G_d \text{ min} = \frac{\frac{G_s F_e}{G}}{F_t \left(1 - C \frac{d}{V_c T}\right)}$$

The implementation according to the DR concept allows a better silicon area reduction with a parallelism rate of  $\beta_p = 1$ . In this part, we show that the silicon gain is better when the data acquisition frequency is low compared to that of the processing frequency. The rate is defined as:

$$\frac{G_s}{G_d \text{ min}} \approx \frac{F_t}{F_e}$$

Performances on silicon reduction appears when dynamic reconfiguration is used at less than 20% of the limits.

### 3.3 Performances

In order to simplify the expressions and to make the figures analysis easier, we suppose that all configurations use the same data parallelism rate  $\beta$ . Respecting the real time constraints, the following relation has to be considered:

$$C \left( \frac{N}{F_r \cdot \beta} + \frac{G_d}{V_c} \right) \leq T .$$

The following curves, drawn for two different technologies (XILINX 4000 family and ATMEL AT40K family), show the importance of the configuration speed  $V_c$  and of the data parallelism rate  $\beta$ , on the complexity limit as well as on the substantial silicon area gain.

For example, the speed of the configuration  $V_c$  of ATMEL AT40K is 100 times faster than XILINX 4000. The complexity of the application should be 100 time greater.

These curves have been drawn with  $T=40\text{ms}$  (block duration in image processing).

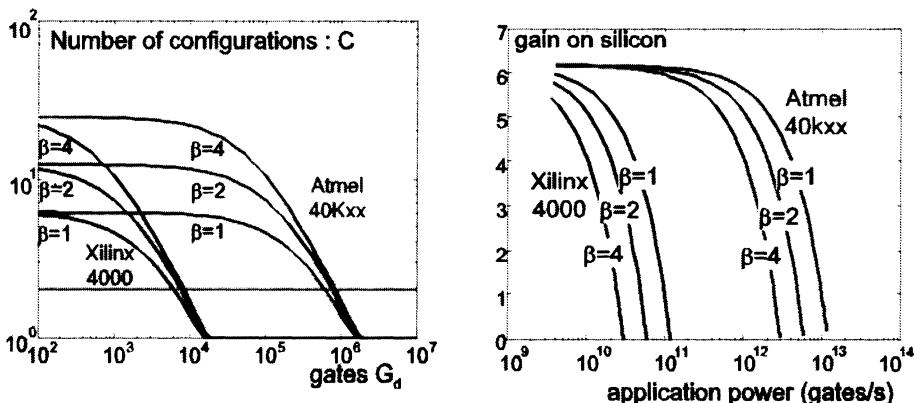


Figure 2. Number of configurations according to the application complexity expressed in terms of gate count and the gain in term of silicon area according to the computing power required by an application

## **4. WHICH TYPE OF FPGA FOR DRA?**

The dynamic reconfiguration is a concept which is not reserved for only FPGA who offer a high configuration speed such as Xilinx XC6200 and Atmel AT40K series. These two FPGA families allow dynamic partial or complete reconfiguration during run-time. This is not currently available in technologies such as the Xilinx 4000 family, because devoid of an interface of fast configuration times. According to the curves shown in §3.3, built architecture with only one FPGA will have significant reductions in efficiency.

The performances can be degraded by the reconfiguration time of the FPGA hardware. However, this does not mean that the use of classical FPGA circuits is excluded in the design of the DRA. There has been much interest in the development of dynamic reconfiguration architecture using reconfigurable FPGA connected in various topology. These last years, a great number of systems are built by teams of research integrating mechanisms aiming to reduce the effect of the configuration times: configuration data caching, bit-stream compression techniques, masking the configuration, ...

The masking of configuration strategy was studied by H. Guermoud in his thesis [4]. In the following of this article, we compare the performances of this approach with the classical DRA (such as ARDOISE).

## **5. DR SYSTEMS WITH OR WITHOUT MASKING ?**

We can make benefit of DR, even with slow configuration speed. One can imagine solutions that use two FPGA, each one with a capacity of  $G_d/2$  gates, instead of a single one with  $G_d$  gates large.

Real-time image processing requires complex: Nagao filter, Edges detector, Boundary closing, Regions labeling, ...). The algorithm implementations [5] include a large variety of hardware models (pipeline, data flow, parallelism, ...) and data arrangement. They usually need very high speed memories and data address generator for an effective data organization (data formatting). For example, some operators require to access to many pixels at the same time. That is why, sufficient memory size and high speed access (bandwidth) are very important for image processing. For processing an image, each step requires large frame storage.

ARDOISE architecture module includes 2Mbytes classic SRAM (for random memory access); the memory bandwidth is 400 Mbytes/s.

Now, we calculate and discuss the performances and the silicon reduction of the following solutions:

1. without masking the reconfiguration time,
2. Masking the reconfiguration time: one device is reconfigured, while the other is computing,
3. Doubling the reconfiguration speed: the two devices are reconfigured at the same time, and then compute together.

The three solutions use the same quantity of hardware resource: total number of gates, memory size and memory bandwidth.

To simplify expressions and figures interpretation, we assume here that the same data parallelism rate  $\beta$  is used for each algorithm.

## 5.1 Solution without masking

In this case, a single FPGA with a capacity of  $G_d$  equivalent gates is used. In order to execute several algorithms, a series of reconfiguration / computation are performed alternatively.

### Number of configurations

$$C = \frac{T}{\frac{N}{F_t} + \frac{G_d}{V_c}} = \frac{1}{\frac{F_e}{F_t} + \frac{G_d}{V_c \cdot T}}$$

### Complexity of the application

$$G_s = C \cdot G_d = \frac{G_d}{\frac{F_e}{F_t} + \frac{G_d}{V_c \cdot T}} < V_c \cdot T$$

Because dynamic reconfiguration has no sense if  $C < 2$ , the maximum value of the equivalent gates of the dynamic architecture is:

$$G_d \max = \frac{V_c \cdot T}{2} \left(1 - 2 \cdot \frac{F_e}{F_t}\right)$$

The expression of  $G_s$  can spell also:

$$G_s = V_c \cdot T \left(1 - C \cdot \frac{F_e}{F_t}\right)$$

The silicon area reduction  $G_g$  according to the complexity of the application is:

$$G_g = \frac{G_s}{G_d} = C = \frac{F_t}{F_e} \left(1 - \frac{G_s}{V_c \cdot T}\right)$$

## 5.2 DR with masking configuration

In this case, a series of reconfiguration / computation is applied in alternation to each of the two FPGA (each provided the equivalent of  $G_d/2$  gates).

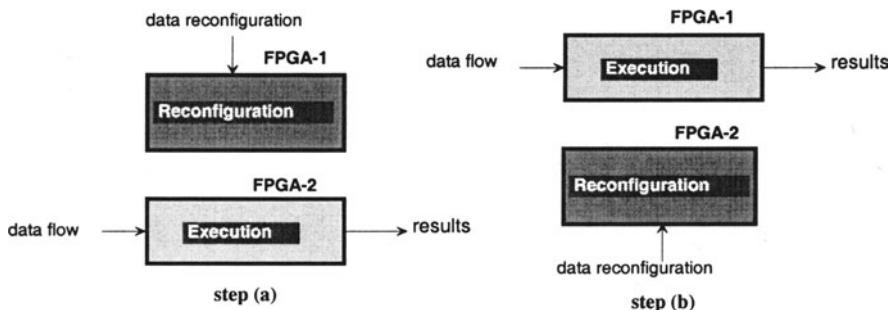


Figure 3. Technique of masking the reconfiguration times

At any step, a phase of reconfiguration of the first FPGA is applied as long as the second FPGA executes a treatment (figure 3a). In the following step, after the reconfiguration time, the first FPGA starts to operate while a configuration is applied to the second one (figure 3b), and so on...

### Number of configurations

Two possible situations should be considered:

1. In the first situation where the reconfiguration time is lower than the computation, the number of configurations can be written as:

$$C = \frac{T}{N \cdot \frac{1}{F_t}} = \frac{F_t}{F_e}$$

2. In the second situation where the time reconfiguration is greater than the computation time, the number of configurations can be expressed as:

$$C = \frac{V_c \cdot T}{G_d} = \frac{2 \cdot V_c \cdot T}{G_d}$$

In this situation the complexity of the application is:  $G_s = V_c \cdot T$ . In both cases, we can deduce the number of configurations by:

$$C = \min\left(\frac{F_e}{F_t}, \frac{2 \cdot V_c \cdot T}{G_d}\right)$$

In practice, it's obvious that partial masking of the configuration time is without interest. For this reason, we continue studying only total masking of the configuration time.

### **Complexity of the application**

$$G_s = C \cdot \frac{G_d}{2} = \frac{G_d \cdot F_t}{2 \cdot F_e} \leq V_c \cdot T$$

Dynamic reconfiguration is interesting beyond 4 configurations, the maximum value of the equivalent gates of the dynamic architecture is:

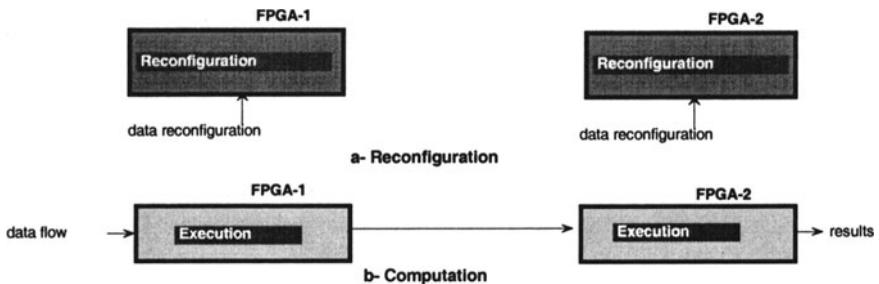
$$G_d \max = 2 \cdot V_c \cdot T \cdot \frac{F_e}{F_t} \leq \frac{V_c \cdot T}{2}$$

The silicon reduction  $G_g$  is:

$$G_g = \frac{G_s}{G_d} = \frac{C}{2} = \frac{F_t}{2 \cdot F_e}$$

### 5.3 DR with double configuration speed

Here, the two FPGA devices are reconfigured simultaneously with different configuration data [6]. After the reconfiguration time, each FPGA starts to execute its own correspondent treatment.



*Figure 4. Technique of doubling of the reconfiguration speed*

The figure 4, shows the functionality of this architecture. One is in the situation of a classical dynamic reconfigurable architecture with a double configuration speed. So the complexity is:

$$G_g = \frac{F_t}{F_e} \left(1 - \frac{G_s}{2.V_c.T}\right)$$

The following table summarizes the expression of the silicon area gain for various solutions:

|  |   |
|--|---|
| Normal architecture                          | $G_{g1} = \frac{F_t}{F_e} \left(1 - \frac{G_s}{V_c.T}\right)$   |
| Architecture masking configuration delays    | $G_{g2} = \frac{F_t}{2.F_e}$                                    |
| Architecture with double configuration speed | $G_{g3} = \frac{F_t}{F_e} \left(1 - \frac{G_s}{2.V_c.T}\right)$ |

The following figure represents the three curves. The silicon gain of the three solutions discussed above are drawn according to the size normalized for a ratio  $F/F_e=10$ .

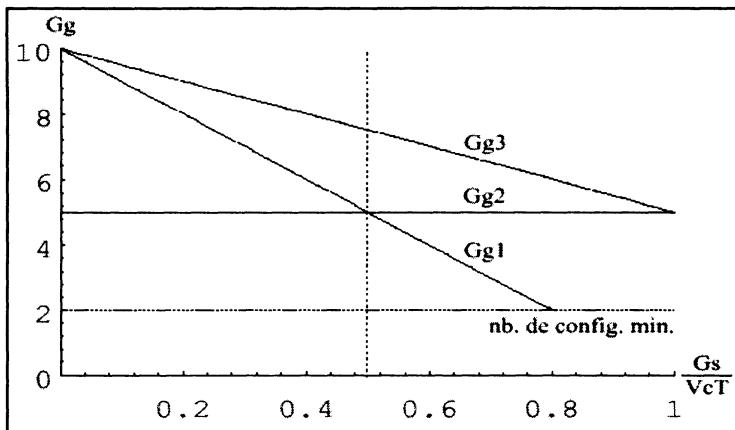


Figure 5. Gain in silicon of the various solutions

Two zones are distinguished. A first zone, in which the masking of configuration time is advised, the silicon gain of this solution is lower than the classical solution. In the second zone, the masking of configuration time becomes interesting. The dynamic reconfiguration is only interesting, if the reconfiguration time of the FPGA is around of 10 % of the block duration  $T$ . That depends on the technology of the FPGA used. It is the case of the circuits AT40K that we use. The technique of masking the configuration becomes interesting if the FPGA has a slower reconfiguration speed. The technique of doubling the speed confirms the importance of the reconfiguration speed. It represents the best solution among the three (less flexible).

This proves that the solution is technological. FPGA circuits should be endowing with mechanisms allowing reconfiguration time reduction (better reconfiguration interface, using configuration data caching, partial reconfiguration, ...). For a given technology, the solution with parallel reconfigurations offers better performances.

The solution based on configuration masking is consequently without great interest. Due to material management in a ping pong manner of the FPGA devices, the system implementation will present some difficulties (complexity in the PCB design).

## **6. CONCLUSION**

In this paper, we presented a new paradigm for designing hardware architecture using dynamically reconfigurable FPGA. We showed the importance of the FPGA reconfiguration time to increase the performances of dynamically reconfigurable systems. We studied and compared the performances with two solutions (masking the reconfiguration delay, doubling reconfiguration speed) using the same hardware resources.

This work indicates that performance increase is possible using doubling reconfiguration speed technique. However, the system hardware realization is more delicate as well as the flexibility is reduced.

The results of this study make our ARDOISE architectural choices are well justified. Indeed, the solution to increase the reconfigurable architecture performances in a significant way consists in enhancement of the FPGA technology: efficient configuration interface, low configuration time, configuration data caching, possibility of partial configuration, ...

Al so, development software tools should contribute to improve reconfigurable architecture performances by including more feature such as configuration data compression, partial and complete configuration management, automatic application partitioning, ...

## **7. REFERENCES**

- [1] Didier Demigny, Lounis Kessal, Riad Bourguiba and Nassima Boudouani. How to use high speed reconfigurable FPGA for real time image processing ? Proc. International Conf. on Computer Architecture for Machine Perception, pages 240-246, Padova, septembre 2000. IEEE Circuit.
- [2] Lounis Kessal, Didier Demigny, Nassima Boudouani and Riad Bourguiba. Reconfigurable hardware for real time image processing. Proc. International Conference on Image Processing, volume 3, pages 159-173, Vancouver, septembre 2000. IEEE ICIP.
- [3] P. Bertin, D. Roncin and J. Vuillemin, Programmable active memories: a performance assessment, In F. Meyer auf der Heide, B. Monien, and A. L. Rosenberg, editors, Parallel Architectures and their efficient use, pp. 119-130, Lecture notes in Computer Science, Springer-Verlag, October 1992.
- [4] H. Guermoud, Architecure reconfigurables dynamiquement dédiées aux traitements en temps réel des signaux vidéo, Thesis, faculty of Nancy I, France 1997.
- [5] Riad Bourguiba, Didier Demigny and Lounis Kessal. Dynamic configuration: a new paradigm applied to real time image analysis. In Proc. The tenth International Conference on Microelectronics, pages 25-28, Monastir, Tunisia, December 1998. IEEE Electron Device Society.
- [6] Riad Bourguiba. Conception d'architecture matérielles reconfigurables dynamiquement dédiées au traitement d'images temps réel. Thèse de doctorat en sciences spécialité traitement d'images, Université de Cergy Pontoise, juillet 2000. Jury: P. Bertin, D. Demigny, L. Kessal, M. Paidavoine, R. Tourki, S. Weber.

# Design Technology for Systems-on-Chip

Raul Camposano and Don MacMillen

*Synopsys, Inc*

**Abstract:** Advancing technology impacts design along several vectors. Interconnect delay became dominant in many designs at  $0.18\mu\text{m}$ ; to obtain timing closure, it was necessary to unify synthesis and placement. Moving forward, increasing degradation of signal integrity will be caused by capacitive cross coupling, by inductive effects and by several other physical effects. This will require the integration of fast and accurate analysis that can drive avoidance and correction of signal integrity problems primarily in routing, as well as in synthesis and placement. Advancing technology also means increasing complexity. Verification is particularly affected by technology as exemplified by the ever-increasing simulation needs. Using hundreds of millions of devices effectively will be possible only by reusing pre-designed intellectual property (IP) effectively and by addressing system-level issues in EDA. This presentation poses EDA solutions to these challenges, gives concrete examples, and argues that complete solutions, rather than point tools, will increasingly and justifiably dominate the EDA field.

## 1. INTRODUCTION

Electronic Design Automation (EDA) is one of the key enablers of the semiconductor industry [1]. No chip is designed without EDA. Conversely, semiconductors drive EDA technology. Throughout the last four decades this has happened mainly along two vectors: technology and complexity.

Technology drives EDA in many ways. Early on, EDA efforts concentrated on capturing and editing artwork. This has led to automatic placement and routing. Eventually synthesis raised the design level to the Register-Transfer Level (RTL). Synthesis, placement and routing are enabled by multiple technology constraints: a logic library, usually in the form of standard cells of the same height and similar size which simplifies

placement and routing, decouples technology from logic and enables synthesis. Furthermore, until recently interconnect was assumed to have negligible delay and digital circuits were modeled without “analog” effects such as cross talk. Power consumption issues were limited to the power grid. All that has changed. Chips today typically are designed using large pre-designed blocks referred to as Intellectual Property (IP) as well as standard cells which may be up to six orders of magnitude smaller [2].

Interconnect can no longer be ignored, since for many designs under  $0.25\mu\text{m}$  interconnect delay actually becomes larger than circuit delay [3]. Capacitive coupling may produce severe cross talk. Power issues are not limited to power grid design: optimizing overall power consumption may be the foremost goal of the design, electro migration and hot spots need to be prevented, and leakage may become a serious power contributor second only to dynamic power. Section 2 examines how technology is driving EDA today. We will show how design technology is radically changing to cope with the changes in semiconductor technology.

The second vector, which has driven EDA relentlessly, is complexity. Moore’s Law epitomizes complexity growth: device density on a chip doubles every 18 months. Design tools need to keep pace, making capacity increases a perennial requirement. Nowhere has this reality been more pronounced than in simulation, particularly in logic simulation. The simulator has to keep up not only with the increasing size of the designs to simulate but also with an exploding numbers of vectors to simulate. Moore’s Law has yielded a hundred times the number of devices per chip in the last 12 years, but the number of vectors needed to simulate a chip to achieve functionality confidence has increased ten thousand times during that same period. A simulator thus has to cope with a complexity that has grown by six orders of magnitude in slightly over a decade. The argument for the development of formal verification techniques, emulators and smart test benches is easy to understand in this context [4]. Section 3 shines more light on how verification techniques are being driven to cope with complexity.

But there is another profound change affecting EDA besides technology. As chips become more complex, they increasingly encompass larger and larger subsystems or complete electronic systems. The electronic design of these “Systems-on-Chip” (SoC) requires system knowledge. Conversely, designing systems becomes increasingly designing chips. Given this increasingly strong correlation, how does system design automation (SDA) relate to EDA? System design has been automated much less than electronic design. Furthermore, SDA tends to be much more domain specific than EDA. Finally, system design today is mostly IP-based. Processors enable the implementation of large parts of functionality in SW. The concept of IP is further extended by defining “platforms,” which are basic architectures

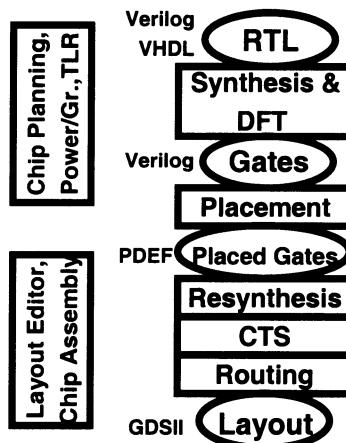
geared towards specific families of applications. In section 4 we examine the influence of system design on EDA, expanding the arguments made above.

The paper concludes summarizing the main trends we see in design technology for systems-on-chip.

## 2. TECHNOLOGY DRIVES EDA

A state-of-the-art design flow consists of some 50 individual design tools. These can be divided into design and verification tools. Technology drives changes in both. In this section we will focus on how technology is influencing design tools and drives EDA towards integrated design flows by migrating “up” in the design tool chain.

Figure 1 shows a simplified design flow for cell-based digital Application Specific Integrated Circuits (ASICs). This kind of methodology was widely used for digital ASICs down to  $0.25\mu\text{m}$  processes. It assumes that you can effectively separate different phases of the design such as logic design, placement, routing, etc., which in turn relies on the fact that physical effects can be “abstracted” (ignored) at higher levels. For example, wire delays can be assumed to be zero or can be modeled statistically during logic design.



*Figure 1.* Simplified Design Flow

As technology progressed below  $0.25\text{mm}$  these assumptions do not longer hold. Wire delay becomes more important than circuit delay in many cases. Figure 2 shows the delay of a  $43\mu\text{m}$  long wire in aggressive copper technology compared with the smallest gate delay.

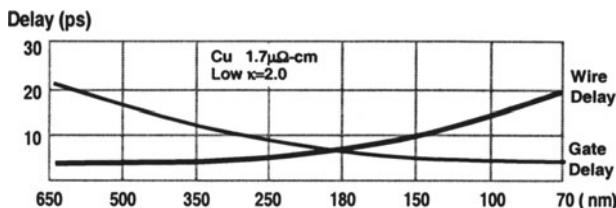


Figure 2. Wire Delay vs. Gate Delay

As a consequence, wire delays can't be ignored in logic design any more, meaning that synthesis needs to have a good estimate for wire delays. The initial solution we adopted long ago was the so-called "wire load model," which is a statistical model of delay dependent on the design size. Clearly this doesn't work well below 0.25μm. Our solution was to do placement and synthesis together so that wire lengths can be accurately estimated during combined synthesis and placement.

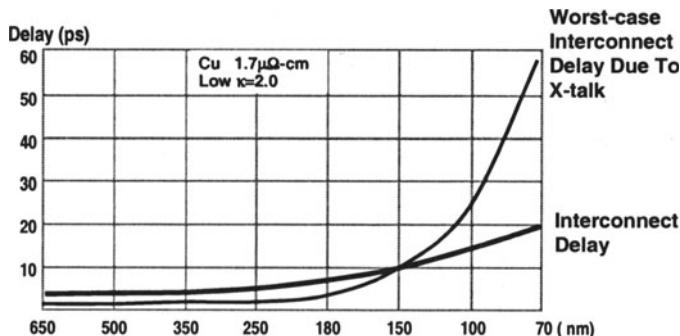


Figure 3. Interconnect Delay due to Crosstalk

Moving forward, as the technology node approaches 0.10μm, other physical effects come into play. Capacitive coupling gives rise to cross talk. Cross talk can be measured as additional delay before a signal stabilizes at the driven value. Worst-case cross talk for a pair of 43μm long wires is illustrated in Figure 3 and compared to interconnect delay in the absence of cross talk.

Cross talk can no longer be ignored in logic design for technology nodes below 0.18μm. Computing cross talk requires knowledge of the exact position of the wires involved. Avoiding cross-talk can be achieved by several mechanisms: signals can be offset so that they switch at different times, wires can be placed further apart, driver strengths can be increased, additional buffers can be inserted, etc. Hence, we expect to see a much tighter integration of synthesis, placement and routing.

In addition to the two examples given above, there are several additional physical effects which will require changes in the design flow. We will only mention some of the most important ones here. Inductivity, typically ignored on chips so far, becomes noticeable for long wires at high speed such as busses of several mm of length operating at more than 1-2GHz [5]. This is already a problem for microprocessor design. The design flow will have to incorporate delays due to inductive effects.

Power dissipation of CMOS circuits so far has been well approximated by exclusively modeling the dynamic power. As we approach 0.10 $\mu$ m, leakage power will become significant. Again, EDA needs to incorporate models for leakage and then minimize it by, for example, using multiple threshold-voltage libraries [6].

Another example involves masks. The physical layout, which defines the patterns for the fabrication process is not used directly for mask manufacturing. To compensate for limited optical resolution, optical proximity correction adds patterns to the masks. Additional resolution enhancements can be achieved using phase-shift masks. In EDA, this resolution enhancement technology is being incorporated into integrated layout verification / mask production tool suites [7].

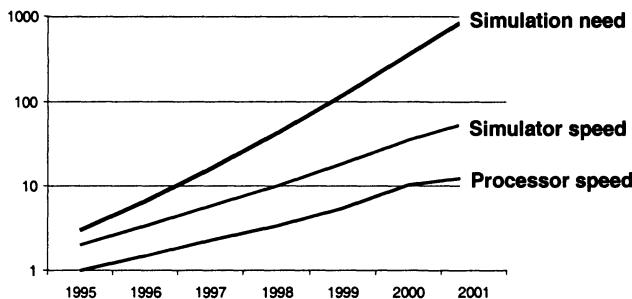
We hope we made our point: technology drives EDA. As we move towards smaller technology nodes, more physical effects need to be taken into account during design. In turn, this drives the integration of design tools, ultimately towards an integrated RTL to layout flow and a layout to mask flow.

### 3. DEALING WITH COMPLEXITY

Nowhere is the effect of complexity felt more than in design verification. Moore's Law has enabled a growth of one order of magnitude (10x) every 6 years in device density and the same growth in design size. The complexity of "system behavior" (the number of input vectors necessary to cover the function of a system) has been increasing by roughly two orders of magnitude over the same period. Functional verification is proportional to the product of the design size and system behavior complexity. For this section, we selected functional verification to show how EDA is dealing with complexity.

Functional verification is based mainly on simulation. How has simulation speed kept up with the thousand-fold increase of simulation need every 6 years (Figure 4)? For a simulator running on a single general-purpose computer, simulation speed increases stem from mainly two sources:

Faster simulators. In our experience, Verilog simulators have provided a speed gain of approximately two times per year over the last six years, resulting in a total speedup of 50 times. This has been achieved through migration from gate to RTL simulation and through optimizations in the simulators algorithms.



*Figure 4. Normalized Simulation Speed 1995=1*

Increasing processor speed. Processor speed is a complex function of many variables such as clock frequency, memory speed, architecture, etc. Over the last 6 years clock speeds have increased by a factor of 10-12. Memory speed has not kept up, while architectural changes have delivered more performance. Overall, processor speed has increased by one order of magnitude in the last six years.

Thus, functional simulation speed has increased by roughly  $10*50=500$  times over the last 6 years, approximately half the rate of the thousand-fold need. Further speedup can be achieved by hardware acceleration or emulation. Emulation can be 4 to 5 orders of magnitude faster than functional simulation. Emulation speed has scaled with the chip (mostly FPGA based) speed increases dictated by Moore's law. Another way of achieving speedup is parallelism. Computer "farms" running thousands of independent simulation runs in parallel provide a cost-effective way of obtaining a 3 order of magnitude simulation speedup.

It needs to be emphasized that simulator speedup has come at the cost of losing the detail that gate-level simulation provides. For example, it is not possible to do an accurate timing simulation at the RTL level. As a consequence, static timing analysis, which scales essentially with the size of a design only, has become adopted as a standard to determine and sign-off timing in SoC design.

Another interesting trend is based on the observation that generating and feeding billions of simulation vectors to a simulator can be as or more time consuming than the simulation itself. Test bench automation aims at

optimizing this process and at effectively monitoring the simulation of a test bench.

A simulation cannot be exhaustive (with the exception of very simple cases). This raises the question of how much simulation is enough or how good is a test bench. To measure this, the concept of simulation coverage is introduced and coverage tools monitor how much of a design has been “covered” by a simulation (a test bench).

But simulation alone, since it can't be exhaustive for large designs, is not enough. Formal techniques have emerged as a powerful aid to functional verification. Equivalence checking asserts whether two net lists (or RTL descriptions) implement the same Boolean function. Although this is an NP-complete problem, in practice equivalence checking works on many large designs up to millions of gates. Equivalence checking allows comparing designs against a “golden” model, which has been extensively simulated and is assumed to be functionally correct. Thus it relieves the designer from the need of having to simulate after changes have been made to a design.

Property checking asserts that a given property holds for a given implementation under all conditions. For example, a protocol may require for a design to output an acknowledge signal before a given number of cycles after receiving a request signal. If this can be verified formally, there is no need to simulate for this property. There is another interesting link between simulation and property checking. Property checking can prove that a set of given states can never be reached. Hence, when measuring the state coverage of a given test bench, those states can be excluded, something simulation alone wouldn't allow.

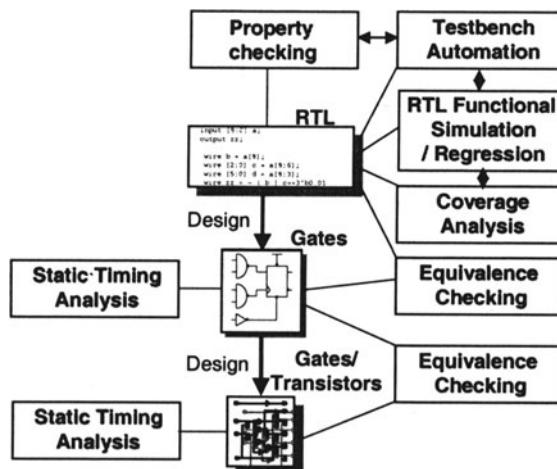


Figure 5. Functional Verification Flow

Figure 5 shows a verification flow using the tools discussed above. Again, we hope we made our point: EDA is dealing with complexity by constantly increasing tool capacity (such as simulator speed), by innovation (e.g., making formal verification practical), and by integrating tools into flows which leverage the combined strength of all tools (such as decreasing the need for simulation by formal verification and measuring coverage).

#### **4. SYSTEM LEVEL DESIGN NEEDS TO BE AUTOMATED**

With the advent of SoCs, the electronic system design and chip design become more tightly integrated. The system designer needs to know what can be done on a chip to deliver good designs and the chip designer needs to understand the systems he/she is designing. Increased communication among system and chip designers can be achieved in many ways, two of which are especially relevant to EDA: system level design (SLD) tools and the use of pre-designed blocks or sub-systems called IP.

SLD tools enable a system designer to enter, verify and possibly implement a system in a way that he/she understands. Several possibilities are shown in Figure 6. Chip designers can use the same SLD tools as a “formal” specification of the design. The SLD tool may be used to execute (simulate) the design and possibly to automatically generate a first implementation, which can then be optimized. EDA technology has addressed multiple levels of abstraction. At the system level however, abstractions tend to be domain specific. For example, data-flow models are used in digital signal processing, hierarchical final state machines model reactive control, and protocols can be described in various languages. EDA has addressed system-level issues in some of these domains, but the SLD market is still embryonic. Recent developments, such as the widespread interest in SystemC and the increasing number of companies offering SLD technology, show that this technology is maturing and is beginning to appeal to a broader audience. System level design automation is a logical next step for EDA.

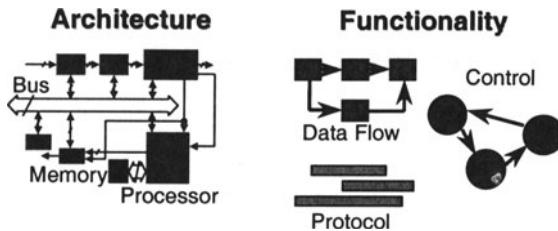


Figure 6. System level Design Models

There are no SoC designs today that don't (re)use IP. Processors, memories, standard interfaces, buses, etc. have become ubiquitous in SoCs. Processors in particular enable the implementation of system functionality in software. System design becomes increasingly influenced by the processor or combination of processors used as a "platform."<sup>[8]</sup> For example, a wireless handset may require a processor and a DSP for application processing and a second DSP and a micro controller for communication processing. System design then consists of integrating some hardware blocks into such a platform and implementing most functionality in software.

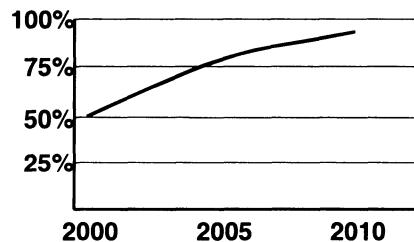


Figure 7. IP Reuse as a % of a SoC

The reuse of IP is also the most effective way of addressing complexity in design (Figure 7). IP is increasingly becoming one of the main differentiators for system and chip design. EDA and SLD need to use the same IP and platforms to enable effective communication between the two levels.

Our final point is that SLD needs to be automated: The introduction of SLD tools and the increasing (re)use of IP are clear signs that this is happening.

## 5. CONCLUSIONS

EDA continues to change to meet the needs of electronic design. In this paper, we showed how the main forces affecting this change are technology, complexity, and the advent of Systems-on-Chip. As a consequence, (1) more technology effects are being taken into account earlier in the design process, (2) individual design tools are being integrated into design and verification flows, (3) tool capacity is keeping pace with complexity, (4) innovation results in smarter tools, (5) system level design is gaining increased attention, and (6) IP reuse and design platforms are becoming pervasive. We expect these trends to continue as long as integrated circuit manufacturing keeps moving to smaller technology nodes.

## 6. REFERENCES

- [1] Don MacMillen, Mike Butts, Raul Camposano, Dwight Hill, and Thomas Williams, "An Industrial View of Electronic Design Automation", *Transactions on CAD, Volume 19, No.21*, IEEE, Dec.2000, pp.1428-1448
- [2] Warren Savage, John Chilton, Raul Camposano, "IP Reuse in the System on a Chip Era", *Proceedings of the 13<sup>th</sup> ISSS*, Madrid, Spain, September 20-22, 2000, pp.2-7
- [3] Jason Cong, "An Interconnect-Centric Design Flow for Nanometer Technologies", *Proceedings of the IEEE*, VOL. 89, No. 4, April 2001, pp. 505-528.
- [4] Bob Bentley, "Validating the Intel Pentium 4 Microprocessor", *Proceedings of the 38<sup>th</sup> Design Automation Conference*, Las Vegas, June 18-22, 2001, pp. 244-248
- [5] Alina Deutsch, Paul Coteus, Gerard Kopcsay, Howard Smith, Christopher Surovic, Byron Krauter, Daniel Edelstein, Phillip Restle, "On-Chip Wiring Design Challenges for Gigahertz Operation", *Proceedings of the IEEE*, VOL. 89, No. 4, April 2001, pp. 529-555
- [6] Luca Benini and Giovanni De Micheli, "Dynamic Power Management, Design Techniques for CAD Tools", Kluwer Academic Publishers, Boston, 1998.
- [7] Andrew Kahng and Y.C. Pati, "Subwavelength Lithography and its Potential Impact on Design and EDA", *Proceedings of the 36<sup>th</sup> Design Automation Conference*, New Orleans, June 21-25, 1999, pp. 799-804.
- [8] K. Keutzer, S. Malik, A.R. Newton, J.M. Rabaey, and A. Sangiovanni-Vincentelli, "System-Level Design: Orthogonalization of Concerns and Platform-Based Design", *Transactions on CAD, Volume 19, No.21*, IEEE, Dec.2000, pp.1523-1543

# Distributed Collaborative Design over Cave2 Framework

Leandro Soares Indrusiak <sup>1,2</sup>, Juergen Becker <sup>2</sup>, Manfred Glesner <sup>2</sup>, Ricardo Augusto da Luz Reis <sup>1</sup>

*1 Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil*

*2 Institute of Microelectronic Systems, Technische Universität Darmstadt, Darmstadt, Germany*

*E-mail: <lsi,becker,glesner>@mes.tu-darmstadt.de, reis@inf.ufrgs.br*

**Abstract:** This paper presents the research and development of mechanisms to allow distributed collaborative design of integrated systems. Those mechanisms were implemented over the software infrastructure developed under the Cave Project, taking advantage on both definitions of the framework concept: classical electronic design frameworks and object-oriented extensible data modeling. The final result is a design environment accessible over Internet-like networks, where groups of designers can work over the design representation in a collaborative way. In order to organize the interaction between the designers, an extension to the Pair Programming collaboration methodology was developed and implemented in a case study.

## 1. INTRODUCTION

Recent advances on the application of system level modeling techniques show that this approach is already mature to be considered for production environments [1,2,3,4]. However, the advantages granted by such approach depends strongly on how those concepts can be understood and adopted by the designers. Such kind of paradigm shift was already imposed to the designers, as a result of the pursuit of productivity boost by raising the abstraction levels of the design specification: physical layout, logic schematic diagrams, HDLs. In most of the cases, they are able to cope with such transitions by relying on new design environments, training sessions and support from the colleagues. And it seems that they are expected to handle the transition we're going through in the same way as they did before. Similar experiences were reported in software engineering, when the transition between the structured and the object-oriented paradigm took place.

Taking into account such transition scenario, new design automation infrastructure is needed to reduce the complexity of the new paradigm, making the transition as smooth as possible for the designers.

The work described in this paper is a part of the ongoing research of a design environment tailored for the needs of a designer without familiarity with system level concepts. The following points are key issues to make such design environment possible:

correlation between system level design concepts and the concepts which are familiar to the designer - e.g. structural or behavioral modeling using HDLs - through the interoperability of their graphical representations;

on-the-tool learning, where the designer can learn the new concepts by using the tool, guided by a remote tutor or pre-recorded usage scenarios;

remote collaboration, allowing beginners to learn from the experienced ones while working in a single project, even if the participants are in different locations.

In order to implement such ambitious proposal, the system is being built over the infrastructure of the Cave Project - a research effort started at UFRGS University in 1996, and which now includes other research groups in South America and Europe.

This paper has its main focus on the third point presented above - remote collaboration - but it will mention briefly some issues related to the other two concepts. It is organized as follows: in the next session, the Cave Project background is briefly presented, and its infrastructure is detailed; Section 3 reviews the constructs used for design modeling at system level; Section 4 details the chosen collaboration models, as well as the software architecture developed to integrate such model in the Cave2 Framework; Section 5 presents a case study for a collaborative design flow from a

system level specification; finally, conclusions, open issues, future development and references are presented.

## 2. CAVE PROJECT BACKGROUND

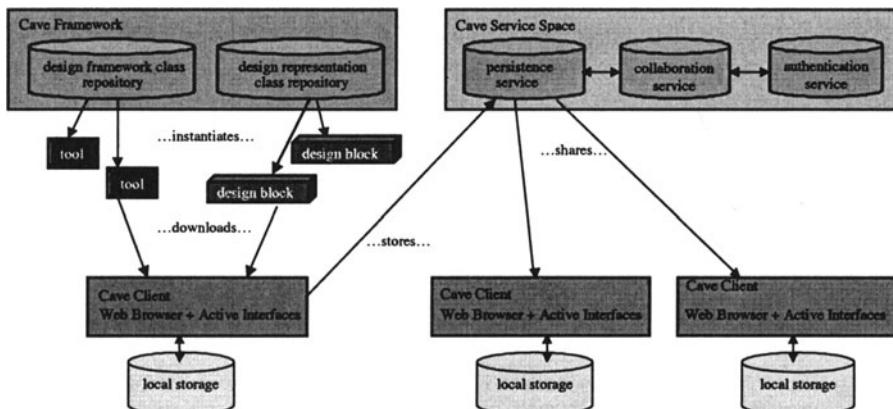
As defined in [5], a CAD Framework is a software infrastructure aimed to support the development, interoperation and usage of design automation tools. The first release of the Cave Project, known as the Cave Framework [6,7], followed that guideline, emphasizing the possibility of the distribution of design automation resources over Internet using WWW and the HTTP protocol. A library of object-oriented primitives for GUI development was created, in order to enhance the functionality of web browsers, which were the main interface between the designer and the framework. A set of modules was also developed, easing the integration of design tools with HTTP servers.

While successful in some of its usage scenarios [8,9], two critical points inherited from the WWW architecture - lack of support for user awareness and file-based data transfer and storage - limited the possibilities of true distributed collaborative design. So, a new architecture was presented in [10] under the name Cave2 Framework, with a different approach to overcome the deficiencies (Figure 1). While still using the Internet as communication infrastructure, it focused more in the library of reusable object-oriented code and on the extensibility of that library through the concept of framework, but this time using the meaning from the software engineering domain. According to [11], a framework is a reusable software design expressed as a set of abstract classes and the way their instances collaborate. Based on such approach, the Cave2 Framework can provide an extensible set of design tool GUI primitives, which are used to assemble dynamically the interface between the user and the design environment according to the tasks he/she wants to perform. Furthermore, the software framework approach also allowed the modeling of design data using objects, in opposition to the file-based model used in the first version. So, relationships among those objects can be defined using patterns [12], keeping the semantic of those relationships on every instance of the object. This is a key issue to allow the correlation between system-level and structural concepts and - as detailed in the following sections - the implementation of richer collaboration models within the design environment.

Besides the organization of the class libraries in a framework, the Cave2 Framework also organized the server-side facilities. Formerly attachments to the HTTP server, now they are implemented also as objects and are

executed in a Service Space, following the architecture proposed in [13]. Current services implemented in such space include persistence, authentication, integration with external tools and, as described in the next session, collaboration.

The Cave2 Framework can be executed over Internet/Intranet, using regular HTTP servers and regular Java-enabled web browsers. Design resources distribution can be easily done, because the Cave2 architecture support the cooperation of several Framework Servers and Service Spaces. In both cases, a non-redundant approach was used. Every design representation or design tool module is stored in only one Framework Server, and a Intra-Server protocol is used to share such resources. Similarly, every service resides in only one Service Space, and the lookup mechanism described in [13] is used for searching for services among all the available spaces.



*Figure 1. Cave2 Framework Architecture*

### 3. DESIGN MODELING CONSTRUCTS

The recent advances on the research of system level modeling techniques is forcing a paradigm shift by the designers. New languages and modeling tools are being introduced, based on such techniques, requiring from the designers a different understanding of the design process.

This scenario was already seen before, as new hardware design paradigms were introduced. If we review the evolution of the integrated circuits design procedure in the last thirty years, we notice that on every step taken to improve productivity - by making the design abstraction levels higher - a new kind of design perception was required by the designers: layout edition, logic schematic capture, FSMs, HDLs, among others.

The transition between paradigms by the designers is always difficult, and it has been supported by the evolution of the design environments.

The use of graphical representations of the design objects in such environments is one of the strategies to reduce the complexity of the design task - for circuit layout, logic schematic or FSM edition it seems obvious, but even HDL-based design environments often take advantage on some diagrammatic representation.

However, the current approaches for system level design environments don't have a common focus. The approaches based on extensions of programming languages - such as SpecC[1], SystemC[2], Forge[3], Ocap[4] - motivate the use of IDE-like tools for source code maintenance without any graphical support other than file management GUI. In other hand, well defined visual modeling tools are also being used, such as UML[14] and SDL[15], requiring complex graphical support from the design environment.

In the Cave2 Framework, graphic and textual edition primitives are available for the designer to assemble his/her design entry tools, allowing both kinds of modeling for system level design, as well as lower abstraction design modeling through HDLs and logic schematic entry.

## **4. DISTRIBUTED COLLABORATIVE DESIGN**

The need for distributed collaborative design can be easily justified:

- increasing complexity of designs, demanding larger teams;
- shortage of engineers, requiring manpower from different parts of the world;
- need for experience sharing among the members of a team, allowing the beginner to learn from the more experienced.

Some approaches for collaborative work support were presented previously in the EDA domain [16, 17], focusing IP core sharing, remote training or task-flow modeling. For the approach presented here, however, a different focus had to be found, because the collaborative model should be generic enough to support collaboration over different kinds of objects: texts and several kinds of diagrams.

The chosen solution used and extended the infrastructure provided by the Cave2 architecture: the support for collaborative work was included in the

abstract classes within the design data representation framework, and the behavior of the collaborative sessions is controlled by a collaboration service in the Cave2 Service Space. The following subsections detail both parts of the solution.

## **4.1 Collaboration-enabled design primitives**

In the Cave2 Framework, the design representation primitives - e.g. logic gates, functional blocks, etc. - are modeled as instances of a concrete class, which inherits behavior from an abstract class. This is common when it comes to object-oriented frameworks, because the abstract classes - while not modeling anything in the application domain - organize the class hierarchy and allow the assignment of common behavior to a particular set of objects. This concept was used to model all the support for the collaborative services: that support was included in the superclass of all the design primitive classes. So, all the design objects in the Cave2 Framework - including the ones which will be integrated in future updates - will inherit such behavior.

Currently, the collaboration platform includes the following features:

- separation of concepts: the design semantic and its graphical/textual representation are modeled by different objects, in order to allow several visualizations - by different designers - from a single design block;
- update/notify mechanism: a 2-way update/notify mechanism was implemented, to grant consistency between the design semantic and its representations.

Using such features, several designers can work concurrently in a design. When two or more designers are working over a single design block, the object representing the block semantic is stored in the Cave Persistence Server, while a view of that block is instantiated for each one of the designers and stored on their GUIs. The interaction between the designer and the design block occurs through that view: when the view is modified in such way the design semantic changes, the view object notifies the block object in the Persistence Service, so it can update its state to reflect the change. Once the state is updated, a notification is done to every other view of that block, making the changes visible to the other users. It is important to note that when the interaction between the designer and the view is not changing the design semantic - e.g. when the designer moves a block in the design sheet without changing its interconnection with the other blocks - the update/notify mechanism is not activated. This approach is called visually decoupled (Figure 2a), because the view for each user can - and probably will - be different from the others.

While allowing rich collaboration through the concurrent access to design blocks, this approach has the disadvantage of making impossible the spatial referencing of design blocks by the designers: one designer would fail to reference a particular block while communicating with another designer if he/she says e.g. "that FIFO in the left side of the encoder", because probably the blocks positioning in the screen of his/her colleague would be different. So, when spatial referencing is desired, we provide a visually coupled approach (Figure 2b), where the update/notify mechanism is slightly different. In this case there is only one view, stored together with the block in the Persistence Server. Every designer would have in his/her GUI a reference to that view, so every update - even the ones which are transparent to the design semantic - would be noticed by every designer.

## 4.2 Collaboration service

Taking into account the 1-to-n cardinality in the relationship of the design blocks with their views, it is easy to realize the need for an data access control mechanism. The possibilities of deadlock in the update/notify mechanism (i.e. when two views try to modify the semantic at the same time) and inconsistencies between the block and its views (i.e. when a view tries an update to the semantic before the reception of the notification of an update done previously) are some of the problems that may occur. The collaboration service performs such control, while providing a common interface for every tool in the Cave2 environment, allowing them to use the design primitives in a collaborative way. Stored in the Cave Service Space and working together with the Authentication and Persistence Services, it uses a transaction-based approach to organize the access to the design objects by the users and to grant ACID properties: atomicity, consistency, isolation and durability.

In the top of such transaction-based mechanism, several collaboration models can be implemented.

## 5. CASE STUDY

### 5.1 Collaboration Methodology

The extensions to the Cave2 Framework described in this paper were implemented, as a case study, and a well known collaboration model was used in the first prototype of the Collaboration Service. The requirements

for such collaboration model were the easy adaptation for the proposed transaction scheme and the suitability for collaboration over heterogeneous data representation. The chosen method for collaboration was the pair programming [18], a methodology mainly used for collaboration in software engineering. According to that method, the quality of the code developed by a programmer increases significantly when it is concurrently reviewed by a

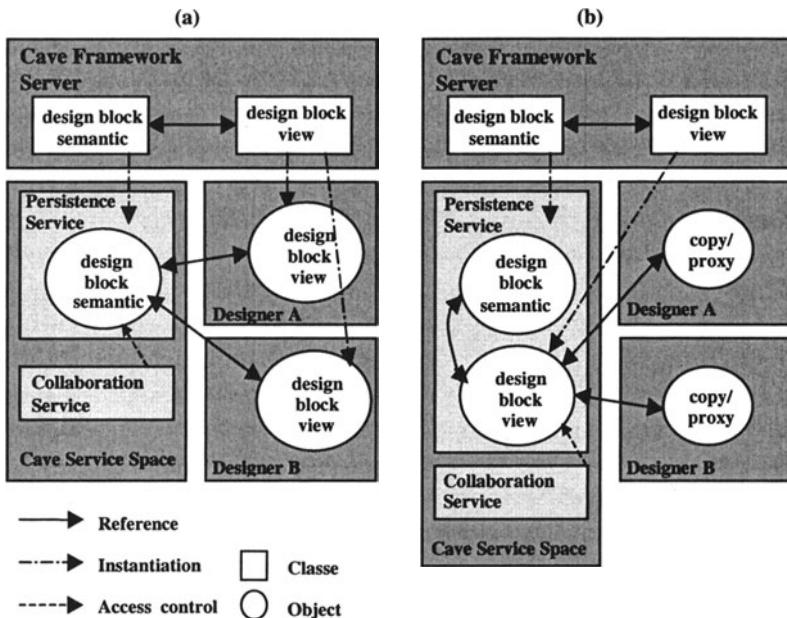


Figure 2. Implemented support for collaboration

colleague. The method is considered synchronous co-located by the time-space taxonomy of collaborative systems, because both developers work at the same time and at the same machine, alternating the control of the keyboard systematically. In the current work, this method was extended and adapted to be used over a network of remote participants, allowing the collaborative work to be done using both text and diagrams. Two major differences were included:

- the participants are not co-located and rely on the network infrastructure to collaborate;
- the collaboration among three or more users is allowed.

For the sake of correctness, the altered methodology was called paar programming, named after the German word *paar*, which designates a collection of elements.

In the implemented prototype, the Collaboration Service is responsible for the management of the users and their collaboration groups, each one named paar. In each paar, only one user can have the "keyboard control", or in this case the permission to update the design block the group is working on. The rest of the group will have read-only access to the design block. Several control mechanisms are being implemented to change the control from one user to another, based on consensus among the users, randomic choices and fixed amounts of time – as used by some of the original pair programming enthusiasts.

The transaction-based model is used to make this approach real. When a paar is formed to collaborative build a design block, every user but the leader opens a read-only transaction to that block, which is stored as an object by the Persistence Service. Then, a transaction is opened to the Persistence Service by the paar leader, granting him/her write privileges while write-locking the design object for the other paar members. Every update done by the leader is broadcast to the other members using the update/notify pattern. When a leader substitution happens, the leader transaction is closed, the lock is released and he/she opens then a read-only transaction. The next leader opens a read/write transaction, locking the design block, and this procedure loops until the end of the design block edition.

The current prototype uses Jini/JavaSpaces [13] and Ozone [19] to implement the service space, hosting the persistency and collaboration services.

To validate the proposed approach, several experiences are being done over the prototyped environment. Both kinds of design description - languages and diagrams - are being used, and a IRC-like text-based communication tool is used for the coordination of the design team. Further work should be done to provide a voice-over-IP channel for real conversation among the designers, enriching even more the collaboration.

About the collaboration experiences observed up to now, it is important to notice that the initial resistance of the designers to work in group using the transaction-based approach is quickly overcome. As the designers get used to the pace of the control change, they start to enjoy the possibility of having from time to time an "outside view" of the design while the colleague has the control of the design block under development. Most of the advantages of the Pair Programming methodology found in the literature could be also noticed, such as:

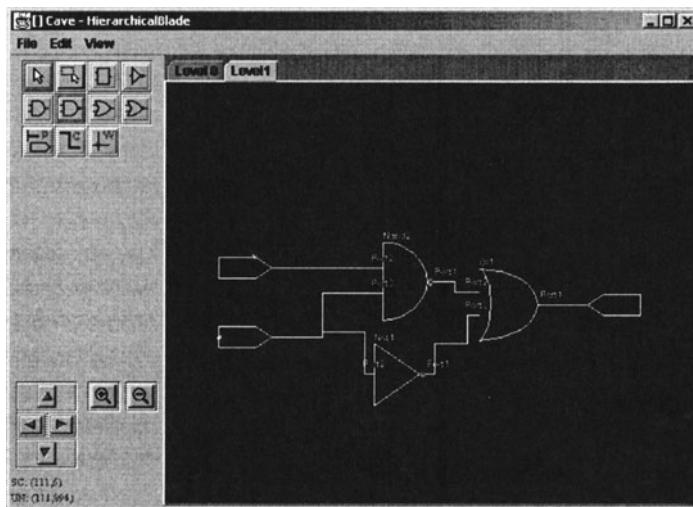
- productivity increase, because one designer is much more focused when are there other colleagues depending on his/her work;
- quality improvement, because the design blocks are reviewed by several people, and usually each one has better perception to a particular detail;

- experience transfer, because the methodology motivates the designer to learn from the expertise of his/her colleagues.

## 5.2 Design Tool Prototypes

In order to make the collaborative system design possible, two design entry tools were prototyped, based on the Cave2 primitives. For diagrammatic entry, a flexible editor called BLADE was created [20]. BLADE allows the collaborative creation and edition of hierarchical block diagrams. The editor supports multiple visual languages, so it can be used in all the cycles of the design, from the functional specification - using UML, for instance - to the structural design - using logic schematics.

The diagrammatic entry is not so effective in the cycles of the design flow where the detailed behavior and structure of the system are input. In those cases, the graphical entry is only performed to define the interface of a particular module with the rest of the system. The actual specification of the module is done using an HDL. To support the collaboration on textual entry, a collaborative text editor named Homero was prototyped [21]. Homero can be executed as a standalone tool, for small projects, or executed as a plug-in to BLADE, when of the programming of the behavior of a particular module of a larger design.



*Figure 3. BLADE Graphical User Interface*

Both the tools work over instances of the design representation primitives, allowing the designer to share the design data using the Persistence service on the Cave Service Space. Due to its peculiar

characteristics, the textual data can be shared synchronously in the visually coupled model only. At this time, the effort to provide separate modeling for the text rendering - indentation, font type, size, color, etc. - in order to allow visually decoupled collaboration over text could not be taken.

## **6. CONCLUSIONS**

This paper presented a design environment which allows the collaborative design over a computer network. The major concern of this paper is on human aspects of the design process, so it focused on the interaction and collaboration among designers, detailing the technological infrastructure developed to make such synergy possible.

The current prototype of the environment is executed over the Internet, using and extending the infrastructure of the Cave2 Design Framework. The major upgrades to the Cave2 infrastructure were on (1) the separation of semantic and visualization on the design data modeling; (2) keeping consistency after the separation using a pattern for state update/notification; and (3) the inclusion of a transaction-based service for controlling the access to the design data by a set of concurrent designers.

Those extensions to the Cave2 Framework are the contributions of this paper to the state of the art in collaborative design automation environments.

An extension to Pair Programming - a widely used collaboration methodology - was also described in this paper. The extension was necessary because of the nature of the collaboration we were looking for (among larger groups of distributed users), changing some of the basics of the Pair Programming methodology. So, for the sake of correctness the new approach received a new name: Paar Programming, once it changed the allowed number of participants in the collaboration, as well as introduced the possibility of remote collaboration. The implementation and the application of such methodology is another contribution of this paper.

The most important contribution, though, may be the experience on the collaborative work between designers. The need of productivity boost in the hardware industry, together with the shortage of hardware engineers, creates a momentum where the levels of abstraction on the design process should be risen and a new profile for engineering teams – able to do collaborative design through a systemic approach - should be followed.

So, this work intends to be the environment to allow such collaboration, as well as the training environment for the next generation of professionals, which will have the opportunity to learn the design process in a systemic level right from the start, by sharing the experiences of those colleagues who had to learn during the transition from the previous paradigm.

## 7. ACKNOWLEDGEMENT

The authors wish to acknowledge and thank Lisane Brisolara, Sandro Sawicki and Emerson Hernandez for their contributions on the design and implementation of the prototypes described in Section 5.2.

## 8. REFERENCES

- [1] D. Gajski et al. "The SpecC Methodology". <http://www.ics.uci.edu/~specc>
- [2] S. Swan et. al. "Functional Specification for SystemC 2.0". <http://www.systemc.org>
- [3] D. Davis et. al. "Forge-J: High Performance Hardware from Java".  
<http://www.xilinx.com/forge/forge.htm>
- [4] DESICS Division. "Ocapi-xl". <http://www.imec.br/ocapi>
- [5] T.J. Barnes et al.. *Electronic CAD Frameworks*, Kluwer Academic Publishers, 1992.196p.
- [6] L.S. Indrusiak and R.A.L. Reis. "A WWW approach for EDA tool integration",  
Proceedings of the X SBCCI, Gramado, 1997.
- [7] L.S. Indrusiak and R.A.L. Reis. "A Case Study for the Cave Project", Proceedings of the  
XI SBCCI, IEEE Computer Society Press, Armacao de Buzios, 1998.
- [8] L.S. Indrusiak and R.A.L. Reis. "3D integrated circuit layout visualization using VRML",  
Future Generation Computer Systems, Elsevier, v.17, n.5, 2001, p.503 - 511.
- [9] J. L. Fragoso, F. Moraes and R. Reis. "WTROPIC: a macro-cell generator on World Wide  
Web", Proceedings of the XIII SBCCI, IEEE Computer Society Press, Manaus, 2000.
- [10] L.S. Indrusiak and R.A.L. Reis. "From a Hyperdocument-Centric to an Object-Oriented  
Approach for the Cave Project", Proceedings of the XIII SBCCI, IEEE Computer Society  
Press, Manaus, 2000.
- [11] R. Johnson and B. Foote. "Designing Reusable Classes", Journal of Object-Oriented  
Programming, Vol 1 (2), 1988, pp. 22-35.
- [12] E. Gamma et al. *Design Patterns - Elements of Reusable Object-Oriented Software*,  
Addison-Wesley, 1995. 395 p.
- [13] E. Freeman, S. Hupfer and K. Arnold. *JavaSpaces: principles, patterns, and practice*,  
Java Series, Addison Wesley, 1999.
- [14] C. Kobryn. "UML 2001: A Standardization Odyssey", Comm. ACM, 42 (10), 1999.
- [15] J. Ellsberger, D. Hogrefe and A. Sarma. *SDL - Formal Object-Oriented Language for  
Communication Systems*, Prentice Hall, 1997, 312 p.
- [16] F. Brglez and H. Lavana. "A Universal Client for Distributed Networked Design and  
Computing". In Proceedings of the 38th Design Automation Conference, Las Vegas, 2001.
- [17] D.R.Kirowski, M. Potkonjak and M. Drinic. "Hypermedia Aided Design". In  
Proceedings of the 38th Design Automation Conference, Las Vegas, June 2001.
- [18] L. Williams, R.R. Kessler. "All I Really Need to Know about Pair Programming I  
Learned In Kindergarten", Comm. of the ACM Vol. 43 No. 5, 2000. P. 108-114.
- [19] G. Mueller and F. Braeutigam. "Tutorial: Getting started with ozone". The Ozone  
Database Project, 2000. <http://www.ozone-db.org>
- [20] L. B. Brisolara, L. S. Indrusiak, R. Reis. "An Hierarchical Schematic Editor to WWW".  
In Proceedings of the I Microelectronics Students Forum, Pirenopolis, 2001.
- [21] É. B. Hernandez, S. Sawicki, L. S. Indrusiak, R. Reis. "Homero - Um editor VHDL  
Cooperativo via Web". In Proceedings of the IWS 2001 - VII Workshop Iberchip,  
Montevideo, 2001.

# High Performance Java Hardware Engine and Software Kernel for Embedded Systems

Morgan Hirosuke Miki, Motoki Kimura, Takao Onoye\*, Isao Shirakawa

*Department of Information Systems Engineering,  
Graduate School of Engineering, Osaka University  
2-1 Yamada-Oka, Suita, Osaka, 565-0871 Japan*

*\*Department of Communications and Computer Engineering,  
Graduate School of Informatics, Kyoto University  
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan  
e-mail: miki@ise.eng.osaka-u.ac.jp, motoki@ise.eng.osaka-u.ac.jp,  
onoye@kuee.kyoto-u.ac.jp, sirakawa@ise.eng.osaka-u.ac.jp*

**Abstract:** This paper describes an effective approach to Java execution through the use of embedded processors. A pair of hardware engine and software kernel are devised for existing embedded systems in order to execute Java applications efficiently, in such a way that 39 instructions are added to the original JVM dedicatedly for the software kernel implementation. The whole embedded system including the hardware engine of 6-stage pipeline with 30K gates can be integrated in a single chip. The proposed approach improves the execution speed by a factor of 5.7 in comparison with J2ME software implementation.

**Key words:** Java, embedded system, hardware engine, software kernel

## 1. INTRODUCTION

Since the introduction of Java [1] in 1995, it has been widely used in innumerable applications, from small systems such as electronic cards to high performance data base servers. Java also receives special interest as the network language not only for personal computers and work stations but also for the growing embedded system applications, due to the main features of (i) platform independence provided by Java Virtual Machine (JVM) [2], (ii) instruction level network security, and (iii) object oriented language.

In general, JVM is implemented by software with the so called *interpreter* or *just-in-time* (JIT) compiler, where it should be pointed out that the software implementation can not suit embedded systems in terms of large memory usage, slow speed, and large power consumption. On the other hand Java specific processors[3-8] intended for efficient execution suffer from enormous overhead and cost of reconstructing operating system as well as intricate interfaces to existing embedded systems.

In order to solve the technical issues stated above, this paper investigates a unified approach to construct an efficient Java execution scheme added to existing embedded systems. This scheme consists mainly of hardware engine, software kernel, and configurable interface to host embedded processor. A part of those Java methods, which are defined in the non-I/O API library, can be invoked from the host embedded processor to execute Java applications.

The proposed architecture employs a 6-stage pipeline including a stack stage customized for the stack-based semantics of JVM. In order to enhance the execution speed of Java applications, the instruction folding and 39 additional instructions to JVM are provided. The architecture has been verified with the use of Altera APEX EP20KE 400 and Tensilica XT1000 emulation system.

As for ASIC implementation, the proposed system has been synthesized with 30K gates and 30K memory bits by using Virtual Silicon 0.18 $\mu$ m library, which can operate at 96MHz of clock rate. This result admits a single chip implementation of the whole Java execution system, including the proposed engine, host processor, and I/O modules.

As a result, Java execution performance has been improved up to a factor of 5.7 on an average, as compared with J2ME[9] for embedded systems.

Section 2 details the proposed Java System, Section 3 describes the implementation results, and Section 4 discusses the performance evaluation. Finally, Section 5 addresses concluding remarks.

## **2. PROPOSED JAVA SYSTEM**

Figure 1 outlines a possible implementation of the overall system of our Java execution scheme. The ‘Java System’ is constructed of the 32-bit ‘Java Engine’, ‘Java Memory’, and ‘Host Interface’, while the ‘Original System’ is composed of the embedded ‘Host Processor’, ‘System Memory’, and ‘I/O Interface’.

JVM defines 32- and 64-bit operation instructions. However, the usage of 64-bit instructions is much less than that of 32-bit ones in embedded systems, and hence in order to reduce the die size, Java Engine is designed for a 32-

bit machine. In addition, Java System can be easily customized for different embedded systems by modifying only Host Interface. For example, Java Memory and System Memory can be either integrated in one memory or separated into two.

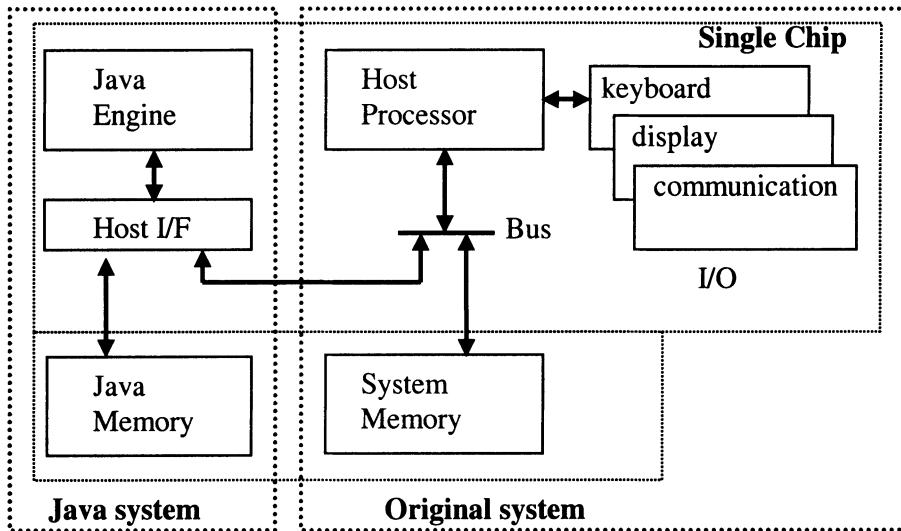


Figure 1. Overall system

Figure 2 outlines a Java application flow, which is achieved cooperatively by Host Processor and Java System. The execution starts with Host Processor invoking a Java method. Then, Java System executes Java bytecode for the method until the termination of execution. Meanwhile, if an I/O method is executed, Java System transfers the process to Host Processor which returns the process to Java Systems after the execution. While Java System executes bytecodes, Host Processor is free to execute any other non Java process in parallel.

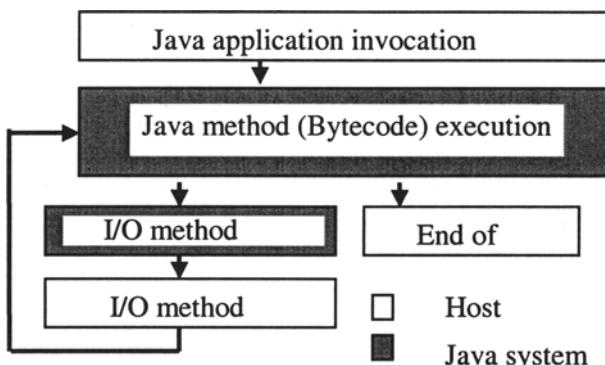
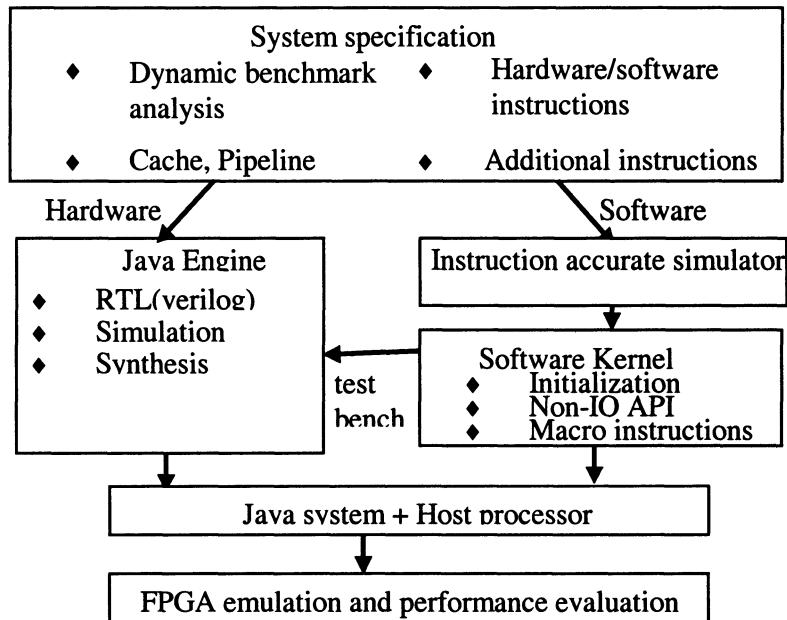


Figure 2. Java application flow

## 2.1 Design flow

Figure 3 shows a design flow of our implementation.

The design flow starts with dynamic analysis of JVM98[10] and CaffeineMark[11] benchmarks in order to determine the system specification features such as hardware/software instructions, additional instructions for software kernel implementation, cache configuration, pipeline, and instruction folding for optimized instruction execution.



*Figure 3. Design flow*

Then, the design flow is split into software and hardware developments. For the efficient software kernel design, an instruction accurate simulator which models both hardware instructions and cache configuration is newly constructed. The software kernel is composed of codes for initialization sequence, non-I/O API library, and software macro instructions.

On the other hand, the hardware design is performed in the register-transfer-level (RTL) through the use of the verilog-HDL language with configurable options for cache size, number of stack registers, and bus width.

Test benches for software kernel implementations are also used for hardware design simulation.

Details of each step of design flow are described in the following.

## 2.2 Instruction set

JVM specifies totally 203 instructions. Most of the instructions, such as those for 32-bit data transfer (`iload`, `iconst_0`, `fload`), 32-bit integer operation (`iadd`, `ior`, `ishl`), and 32-bit branch (`ifeq`, `goto`), are easily implemented in one-cycle hardware pipeline.

However, highly complex instructions, such as those for method invocation (`invokevirtual`, `return`) and object allocation (`new`, `newarray`), are achieved by software macro.

On the other hand, the implementation of such instructions as `swap`, `dup`, 64-bit instructions, and floating point operations must be performed through the careful investigation into the instruction distribution of benchmarks and the complexity of hardware synthesis. As a result, all of 64-bit instructions for data transfer (`lload`, `dload`) and 64-bit integer operation (`ladd`, `lor`) are implemented by multicycle hardware pipeline; while those for `swap`, `dup`, and floating point operation (`fadd`, `dadd`) by software macro.

In order to implement macro software instructions in our Java Engine, 39 instructions are newly added to the JVM instruction set. These are mainly for (i) direct memory address 8/16/32-bit data transfer between the instruction/data memory and the operand stack, (ii) data transfer between special registers and the operand stack, and (iii) replacing/rewriting time-consuming instructions such as `new`, `invokevirtual`, and `getfield`.

## 2.3 Instruction folding

The instruction folding consists in the execution of a set of two or more instructions in one cycle, enhancing the execution performance of Java bytecode. Since it is not suitable to implement all possible patterns, a dynamic analysis is attempted for the benchmarks. Table 1 shows the result of most frequently used patterns, and Table 2 summarizes the performance improvement of equation (1) with the use of these patterns. It can be seen from this table that 15% improvement is achieved on an average.

*Folding performance improvement=*

$$\frac{\text{Number of executed and folded instructions}}{\text{Number of executed instructions}} \quad (1)$$

Table 1. Instruction folding patterns

| pattern  | comment   |
|----------|---|
| LC LV OP | compute memory address for data read and performs the operation, reducing 2 data write in top of stack.                                   |
| LV LC OP | compute memory address for data read and performs the operation, reducing 2 data write in top of stack.                                   |
| LC LC OP | performs an operation with 2 constants, reducing 2 data write in top of stack.  |
| LV OP    | get one data from top of stack, compute memory address for data read, and performs the operation reducing one data write in top of stack. |
| LC OP    | get one data from top of stack, and performs the operation reducing one data write in top of stack.                                       |
| LC MEM   | compute memory address for constant data write, reducing one data write in top of stack.  |

LC: load constant in top of stack

LV: load data from local variable in top of stack

OP: operation instruction

MEM: memory write

Table 2. Instruction folding performance improvement

| benchmark       | execution improvement |
|-----------------|-----------------------|
| JVM98.check     | 25.4%                 |
| JVM98.compress  | 20.1%                 |
| JVM98.jess      | 5.8%                  |
| JVM98.db        | 12.0%                 |
| JVM98.javac     | 6.2%                  |
| JVM98.mpegaudio | 26.2%                 |
| JVM98.mtrt      | 4.4%                  |
| CaffeineMark    | 16.3%                 |
| average         | 14.6%                 |

## 2.4 Java engine architecture

Figure 4 details an architecture overview of Java Engine which is based on *Harvard Architecture* and 6-stage instruction execution pipeline. Since JVM is a stack-based machine, an *operand stack* (STK) stage is included in the pipeline. A *memory read* (MR) stage precedes the *execution* (EX) stage in order to execute efficiently specific patterns of instruction folding.

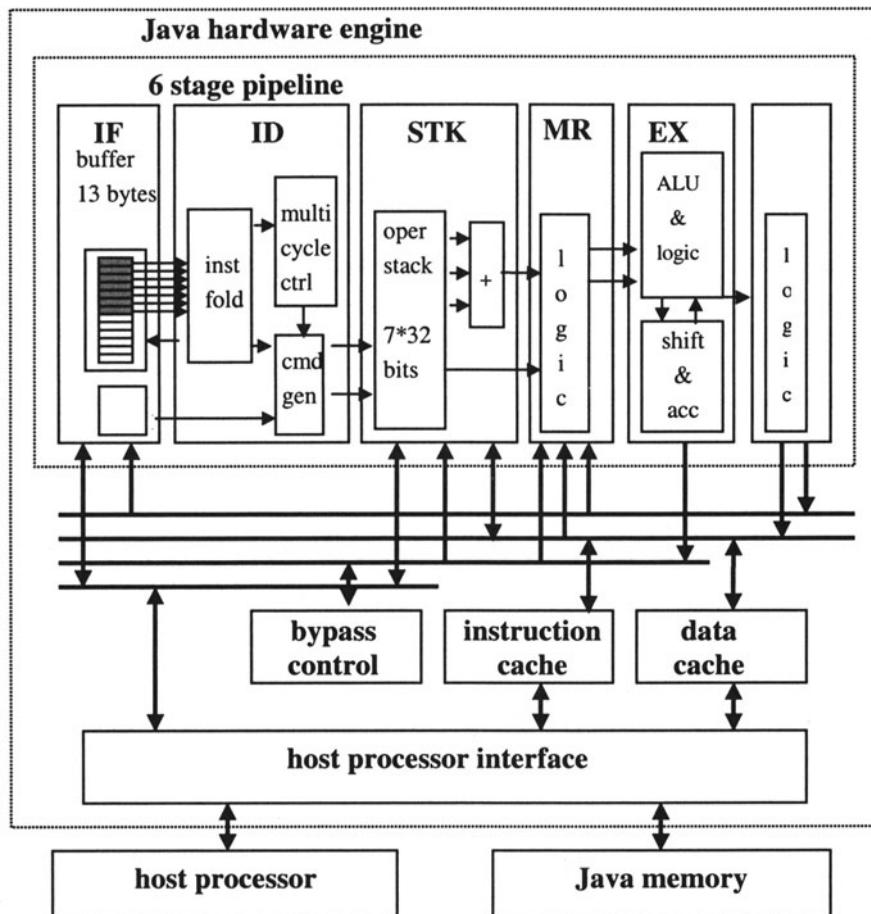


Figure 4. Architecture overview

The following are the details of each module of the architecture.

### Instruction Fetch (IF)

IF is to fetch an instruction code from instruction memory and updates the program counter register. Since Java bytecode length is variable, a 13-byte shift register is used to buffer instructions from the instruction memory. Up to the leading 7 bytes of the shift register can be read by the *instruction decoder* (ID), which returns the number of decoded bytes to update the instruction shift register.

### Instruction Decoder (ID)

ID is constructed by tree units; instruction folding, multicycle control, and command generator.

The input of the instruction folding unit are the 7 bytes from IF, which are interpreted as a single instruction or a pattern of 2 or 3 instructions of instruction folding.

The multicycle control unit is a state machine which generates control signals for the command generator unit for multicycle instructions. The number of cycles is set to 2 for long instructions, 16 for `imul`, and 32 for `idiv`.

The command generator unit gets the output of instruction folding and multicycle control units, and generates the command and data for the following 4 stages.

### **Stack (STK)**

STK consists of an operand stack unit, an adder, and most of special registers of Java Engine.

Top 7x32 bits of the Java operand stack are loaded to the operand stack unit of STK, while the rest is in a segment of data memory. When the operand stack unit underflows or overflows, it transfers data to or from, respectively, the data memory.

The adder is to add the data from ID, operand stack unit, or local variable register, for the memory address calculation or for the comparison of conditional branch instructions, which are used in the next stages of the pipeline.

### **Memory Read (MR)**

MR is to read 8/16/32-bit data from the instruction memory or the data memory.

### **Execution (EX)**

EX is to execute arithmetic/logic and shift operations, where the shift unit is provided with an accumulator which holds the values of multicycle instructions such as multiplication and division.

### **Write Back (WB)**

WB is to write a 8/16/32-bit data into the instruction memory, data memory, operand stack, and program counter.

### **Bypass**

To reduce the number of data hazards of the Java stack engine, the bypass mechanism is indispensable. Whenever possible, one or two 32-bit data output from EX and/or MR are bypassed to the input of EX, MR, or STK, reducing the stalls of pipeline.

### Cache

Instruction Cache is for the direct mapped cache write back, with 16 bytes in each block.

Data Cache is for the 2-way set associative write back with 16 bytes in each block. *Least Recently Used* (LRU) method is used in each set for the cache miss control.

The cache sizes of both Instruction and Data Cache are configurable to 1K, 2K, 4K, 8K, and 16K bytes.

## 2.5 Software kernel

Since the JVM specification does not deal with the implementation details of JVM and its instructions, an optimized software kernel should be constructed to maximize the performance of bytecode execution. The kernel includes the codes for initialization sequences, macro instructions, API library, and all necessary data structures.

During the initialization, the memory is initialized and some classes of API library are loaded.

Macro instructions are the software implemented instructions such as floating operations, method calls, and instructions to deal with objects.

## 3. IMPLEMENTATION RESULTS

The proposed Java System composed of a hardware engine and software kernel implements the total of 203 Java bytecodes plus 39 additional instructions. Table 3 details the implemented instructions.

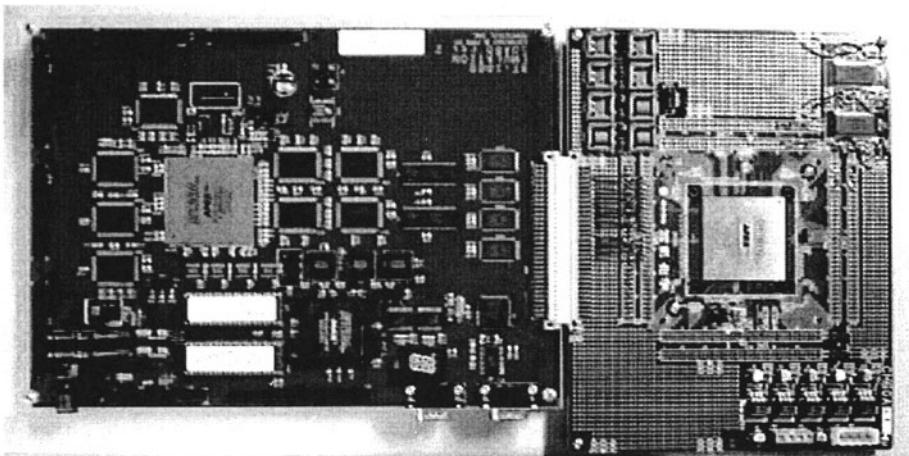
The proposed hardware architecture described in the Verilog-HDL has been verified on Altera APEX EP20KE400 FPGA with the use of Xtensa XT1000 processor emulation module (Figure 5). The architecture has also been synthesized by Virtual Silicon 0.18 $\mu$ m library through the use of Synopsis Design Compiler. Table 4 shows the implementation results. In addition, the cost of instruction folding unit, which improves the performance by 15%, is only 700 of 30K gates.

*Table 3.* Implemented instructions

| instructions            | implementation (num) | detail  |
|-------------------------|----------------------|---|
| JVM instructions        | hardware(140)        | 32 bit data transfer,<br>integer operation, branch<br>method call, object, float<br>operation |
|                         | software(63)         | direct memory access,<br>register data transfer<br>fast instructions                          |
| additional instructions | hardware(24)         |   |
|                         | software(15)         |   |

*Table 4.* Implementation results

|                 |                                |
|-----------------|--------------------------------|
| technology      | 0.18 $\mu$ m                   |
| number of gates | 30k gates                      |
| max. clock rate | 96MHz                          |
| cache           | 30k bits (1kbyte for each set) |

*Figure 5.* Verification board: Xtensa XT1000 (left) and proposed hardware (right)

## 4. PERFORMANCE EVALUATION

Expected performance of the proposed system has been evaluated with the use of the CaffeineMark[11] benchmark. Table 5 shows the performance of the proposed hardware engine and J2ME run on Sun Ultra Sparc (450 MHz). The proposed architecture is 5.7 times faster than J2ME on an average.

Since J2ME is originally devised for PDAs, we have also tested CaffeineMark with the use of Palm III, working on MC68EZ328[12] of 20MHz. However, due to the old CISC architecture, the performance is so slow as to be less than 0.05 cm/MHz.

Table 5. Performance Evaluation on CaffeineMark

| Benchmark           | J2ME on Ultra Sparc(cm/MHz) | ProposedHardware (cm/MHz) |
|---------------------|-----------------------------|---------------------------|
| CaffeineMark.Sieve  | 0.52                        | 9.65                      |
| CaffeineMark.Loop   | 0.50                        | 14.65                     |
| CaffeineMark.Logic  | 0.49                        | 8.35                      |
| CaffeineMark.String | 1.45                        | 2.9                       |
| CaffeineMark.Float  | 0.46                        | 1.05                      |
| CaffeineMark.Method | 0.53                        | 0.45                      |
| Overall             | 0.60                        | 3.4                       |

## 5. CONCLUSION

This paper has described a hardware and software codesign approach to high performance Java execution for embedded systems. The system implements totally 203 JVM bytecode instructions plus 39 additional instructions for software kernel optimization. Among these, 78 instructions are implemented by the software kernel, and 164 instructions by the 6-stage pipeline engine. The proposed system executes all non-IO Java methods allowing the parallel operation of both Java System and Host Processor of the embedded system.

The proposed hardware Java Engine has been coded in the verilog-HDL to be synthesized by Virtual Silicon 0.18 $\mu m$  library, and has been integrated with the use of 30K gates, allowing a single chip implementation of the whole system including Host Processor and I/O modules.

Future work is continuing on the hardware implementation of floating point instructions and the optimization of method invoke/return instructions in order to enhance the performance of String, Float, and Method CaffeineMark tests.

## 6. REFERENCES

- [1] J. Gosling, B. Joy, G. Steele, and G. Bracha, *The Java Language Specification Second Edition*, Addison Wesley, Los Altos, California, April, 2000.
- [2] T. Lindholm and F. Yellin, *The Java Virtual Machine Specification Second Edition*, Addison Wesley, Palo Alto, California, April, 1999.
- [3] J.M. O'Connor and M. Tremblay, "picoJava-I: The Vitrual Machine in Hardware", *IEEE MICRO*, Vol. 17, No. 2, Mar./Apr. 1997, pp. 45-53.
- [4] Sun Microsystems Inc., *picoJava-II Microarchitecture Guide*, Mar. 1999.
- [5] Advancel Logic Corporation Inc., *TinyJ Processor Core Datasheet*, May 1999.
- [6] aJile Systems Inc., *Real-time Low-power Java Processor aJ-100 Datasheet*, Sept. 2000.
- [7] Patriot Scientific Corporation Inc., *PSC1000 Microprocessor*, July 1997.
- [8] S. Kimura, H. Kida, K. Takagi, T. Abematsu, and K. Watanabe, "An application specific Java processor with reconfigurablities", *Proc. Asia and South Pacific Design Automation Conference 2000 (ASP-DAC 2000)*, Jan. 2000.
- [9] Sun Microsystems Inc., *Java 2 Platform, Micro Edition*, June 1999.
- [10] The Standard Performance Evaluation Corporation, *SpecJVM98 VERSION 1.03*, 1998.
- [11] Pentagon Software Evaluation Corporation, *Java CaffeineMark 3.0*, 1999.
- [12] Motorola Inc., *MC68EZ328 – DragonBall EZ Product Brief*

# An Object-Oriented Methodology for Modeling the Precise Behavior of Processor Architectures

João Cláudio Otero and Flávio Rech Wagner

*Instituto de Informática - Universidade Federal do Rio Grande do Sul - Porto Alegre, Brazil  
E-mail: {jcotero,flavio}@inf.ufrgs.br*

**Abstract:** This paper presents SimPL, an object-oriented methodology for modeling processor behavior with precise timing, which may be used as a basis for teaching and design environments. Starting from a specialized class library, SimPL supports a very easy and flexible definition or modification of processor models of different architectural classes.

**Key words:** object-oriented hardware modeling, processor architectures, teaching environments, design environments

## 1. INTRODUCTION

Processors are becoming very complex. Pipelines, memory hierarchies, and superscalar features are implemented in very different ways. A wide range of architectural classes is available (RISC processors, DSPs, VLIW processors, microcontrollers), each of them with their particular solutions. This situation imposes severe requirements to computational environments intended for analyzing or exploring the processor design space.

*Teaching environments* are intended to understand concepts and to experiment alternative architectural solutions. In *design environments*, in turn, processor design space exploration is associated to performance evaluation, as in the design of embedded systems, where the most adequate processor must be found or designed for a given set of design requirements.

These two different goals have been addressed by distinct classes of environments. Teaching environments [1-5] are usually restricted to a given

processor architecture, where a particular configuration may be defined by setting parameter options before the simulation starts. A rich user interface, specialized for the range of processor configurations, is usually supported.

Processor design environments [7,8], in turn, usually offer some specialized language, which gives great flexibility for the definition of the processor architecture. These environments are also aimed at a retargetable generation of supporting tools, such as simulators, compilers, and debuggers.

This paper presents SimPL, an object-oriented methodology for modeling processor architectures that supports features addressing the modeling needs of both teaching and processor design environments. It is based on the SIMOO simulation framework [9], which offers a rich environment for object-oriented modeling of discrete systems and is being applied to the design of electronic embedded systems [10]. SimPL allows a very flexible modeling of various processor architecture classes. The precise timing behavior of the processor instructions may be modeled, if desired, but more abstract models may be also developed. Object-orientation supports an easy reuse, extension, and modification of processor models. The user interface gives specialized support for the modeling methodology.

This paper is organized as follows. Section 2 discusses related work and put SimPL in perspective. The SIMOO framework is briefly introduced in Section 3. The SimPL modeling methodology is presented in Section 4, and its strategy for modeling processor control is discussed in detail in Section 5. Section 6 illustrates the application of the modeling methodology to the well known DLX processor [11]. Section 7 briefly presents the SimPL supporting environment. Finally, Section 8 concludes and discusses future work.

## **2. COMPARISON WITH RELATED WORK**

There are various simulation tools oriented to education in processor architecture, such as PCSpin [1], WinDLX [2], DLXview [3], and ESCAPE [4]. These tools offer a pre-defined set of design alternatives that can be explored by an easy, mostly interactive setting of parameters. Visualization tools allow the user to analyze the impact of these alternatives on instruction execution and processor performance. These tools are dedicated to a specific base processor, which can be modified in a very limited way.

Other simulation tools are more intended for design exploration, offering a richer set of parameters and design alternatives. SimpleScalar [5] is a well known tool of this class. Its parameters allow the design of very different pipelined and superscalar architectures. This feature is enhanced by the fact that the simulator source code is open, so that design alternatives that have not been initially considered can be also implemented. As a drawback, the

great flexibility makes difficult the construction of a specialized built-in user interface for interaction with all possible architectural models.

Tools that allow the specification of processor architectures and organizations are usually based on Architectural Description Languages (ADLs). Many of them automatically generate a retargetable tool suite, containing at least a simulator and a compiler. Some of them, such as ISDL [6], are only intended for the description of the instruction set behavior and cannot generate cycle-accurate simulators for current processors, because they do not support pipeline specifications. Other languages, such as EXPRESSION [7] and LISA [8], describe both behavior and structure.

In EXPRESSION, an RT-level netlist is described, together with the specification of instructions as lists of slots filled with operations corresponding to functional units. The RT-level netlist is composed of primitive components, such as *functional units*, *storage elements*, *ports*, and *buses*. A *pipeline description* specifies components in each pipeline stage, while a *data-transfer path description* specifies valid data transfers in the pipeline. The language has primitives for the detailed description of the memory subsystem. Many language features are intended for the automatic generation of a compiler producing code of high quality.

LISA also supports the description of pipelines. Its definition, however, is mainly intended for the generation of a very fast, cycle-accurate simulator using *compiled code* (as opposed to conventional *interpreted* simulators).

It must be noticed that, although languages such as EXPRESSION and LISA are powerful tools for the development of new processors, for instance in the context of embedded systems design, they may become cumbersome if used in a teaching environment, where the main requirements are easy modification of the processor configuration and good interaction resources.

SimPL is a processor design *methodology* (and not a *language* with some fixed set of constructs), which uses object-orientation as a basic paradigm for a flexible modeling of processor behavior and may serve well both application domains - teaching and processor design environments.

Since SimPL is built on top of a modeling and simulation framework, the processor modeling process automatically results in a simulator with built-in interactive resources for controlling experiments. On the other hand, the methodology does not generate other tools, such as compilers or assemblers, and the simulator code generation is not aimed at high performance.

### 3. THE SIMOO FRAMEWORK

SIMOO [9] is an object-oriented framework for multi-paradigm, multi-threaded, discrete simulation. Simulation entities in SIMOO are mapped to

*autonomous elements*, which are objects with their own execution threads. Each object in a given system may be modeled by a different simulation paradigm - e.g. event or process-oriented behavior, communication by ports or messages. The simulation engine manages communication and synchronization between objects, as well as simulation time advancement.

The model structure is graphically specified by hierarchical class and instance diagrams, while the object behavior is described in C++ and uses a specialized simulation library. From the specified diagrams and entities' behavior, an executable model is automatically generated. It already contains default resources for visualization, interaction, and experiment control.

The class diagram defines various types of relationships between objects, such as *knowledge* (connections for exchanging messages), *aggregation*, and *creation* (so that an object may create a new instance of another class in the model). Knowledge relationships may be dynamically created. The aggregation relationship implicitly defines a knowledge relationship between an aggregating object and all its sub-objects.

Visual interaction facilities are implemented by *interface elements* that belong to a domain that is independent from the model logic. The *monitor* is a special entity that receives copies of all messages sent to an autonomous element *AE* whose behavior is being tracked and redirects them to the interface element that is associated to *AE*. The mapping between autonomous and interface elements is managed by the monitor, so that it may be dynamically modified during the experiments.

## 4. SIMPL MODELING METHODOLOGY

The SimPL modeling methodology explicitly divides a processor into MainControl and Datapath blocks. MainControl aggregates two other types of control elements: InstructionSet and Execution. The datapath is an aggregation of *functional elements*. Furthermore, *information elements* are used for building a bridge between the control and datapath blocks.

The datapath block does not model physical connections between the functional elements, which are not aware of each other. Connections are implicitly defined by means of the processor behavior specified by the control block. This approach allows a more flexible architecture definition, since functional elements may be more easily replaced or introduced.

Each functional element is parameterized and may execute different *micro-operations*, which are defined as methods of the corresponding class. Control elements define the processor behavior by executing these micro-operations. However, they do not directly call methods of the functional

classes. Instead, they communicate only with the aggregating datapath element, which redirects the messages to its aggregated functional elements.

Therefore, the class diagram for the processor does not contain direct connections between a control element and all functional elements it may control. If these connections were necessary, the diagram would be much more complex. The designer, however, must be aware of the operations offered by functional elements. A window in the supporting environment displays all available operations for the definition of a control element.

Micro-operations typically define interactions between functional elements. Suppose the contents of register REG must be transferred to input Inp1 of functional element ALU. A control element would execute the following micro-operation, implemented as one of its local methods:

```
ALU_SET (Inp1, REG_CONTENTS ())
```

As a result, a message is sent to Datapath. ALU\_SET (par1,par2) is a micro-operation that requests Datapath to execute method SET of ALU. Its parameter *par2* is another micro-operation REG\_CONTENTS, requesting Datapath to execute method CONTENTS for reading the contents of REG. These micro-operations implicitly create a path from the register to the ALU.

Figure 1 shows micro-operation REG\_CONTENTS. This method is automatically created by the environment, from the element definition.

```
// Send message ACCESS to Datapath. Its parameters define which functional
// element must be accessed and which of its methods must be called.

int REG_CONTENTS (void) {
    Parameters par;
    MSGEA msg;

    par.Add (0);           // number of parameters
    par.Add ("CONTENTS"); // name of the method
    par.Add ("#REG");     // name of the functional element

    // create a message containing the above parameters
    msg.Set (Id (), InstSet_DatapathBlock, "ACCESS", NORMAL_PR, par);

    // send a message and wait for response in the current simulation time
    SendReceiveNow(&msg);

    return msg.GetDados ().GetParAsInt (0); // return the value
}
```

*Figure 1.* Code for micro-operation REG\_CONTENTS

Decode and Interpret are information elements. Decode can be used to check the contents of the Instruction Register to obtain an instruction opcode, for instance. Interpret may be used to interpret an expression written in an assembly language, for instance in order to calculate a memory address. These information elements are linked to control and functional elements by means of the SIMOO Monitor object, so that they can inspect the contents of other elements without having to send them messages.

## 5. DEFINING PROCESSOR CONTROL

Control elements, from types MainControl, InstructionSet, and Execution, interact with each other to define the overall control behavior.

InstructionSet defines each instruction as a set of *instruction steps*. Each step is a set of micro-operations to be executed in parallel. An instruction step is executed each time the instruction is *fired*. The InstructionSet element, however, does not enforce a timing relationship between the steps.

The complete set of instructions of a processor may be defined within a single InstructionSet element or may be divided into various such elements, for better model management purposes. Instructions may be easily included or removed, because they do not depend on each other.

An InstructionSet element may define complete instructions as well as *instruction fragments*, to be hierarchically used within other instructions or instruction fragments. Fragments model sets of instruction steps that are common to various instructions, such as a fetch operation or a memory access using a particular addressing mode.

Figure 2 illustrates the definition of an InstructionSet element that implements a fragment for a fetch operation in the DLX processor [11]. This fragment is defined as a single instruction step containing 5 micro-operations: read instruction memory position addressed by the program counter (PC); add 4 to the PC; assign PC+4 to the PC; assign PC+4 to the NewPC field of the IF/ID pipeline register; and assign the contents read from memory to the IR field of the IF/ID pipeline register. These micro-operations are implemented as methods that are local to the InstructionSet class, as shown in the methods windows (upper left in Figure 2).

Execution is the element that orders the firing of all instruction steps, according to a selected *execution mode*. Depending on the mode, several *firings* may be necessary to complete the execution of all steps. Each firing may be executed, for instance, at a consecutive clock cycle, but other timing schemes are possible. The following execution modes are available:

- a) *single-cycle* mode - all instruction steps are executed in a single firing;
- b) *multi-stepped* mode - each firing executes a certain number of steps; and

- c) *multi-cycle* mode - each firing executes a single instruction step.

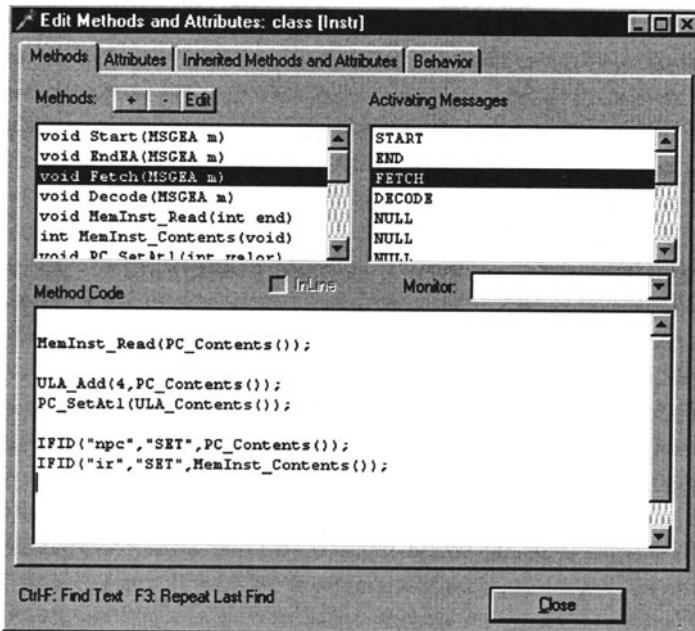


Figure 2. InstructionSet defining a fetch operation

Suppose an instruction composed by 6 micro-operations grouped into 3 steps. Now suppose that an element Execution fires this instruction using a multi-cycle mode. The code shown in Figure 3 is automatically generated for the method that implements the execution of the instruction.

```
// the incoming message specifies the stage to execute
stage = m.GetDados ().GetParAsInt (0)
switch(stage){
    case 1: {   micro-op_1 (); // micro-operations of stage 1
                micro-op_2 ();
            }break;
    case 2: {   micro-op_3 (); // micro-operations of stage 2
                micro-op_4 ();
            }break;
    case 3: {   micro-op_5 (); // micro-operations of stage 3
                micro-op_6 ();
            }break;
}
```

Figure 3. Method for executing an instruction in a multi-cycle mode

The Execution element offers micro-operations, corresponding to the step firings. They have one of the formats below:

- EX (instruction, stage) - for multi-stepped and multi-cycle modes;
- EX (instruction) - for the single-cycle mode.

The execution of instructions or fragments is ordered by MainControl, which is the topmost element of the control block hierarchy and defines the overall processor control, as well as a concrete timing behavior. MainControl uses the micro-operations offered by Execution. In a synchronous processor, firings will be synchronized with the clock cycles. Several cycles, however, may be associated to each firing, so that the latency of a functional element may be easily modeled.

## 6. MODELING THE DLX PROCESSOR

DLX is a 32-bit RISC processor, defined by Patterson and Hennessy [11]. It has four main instruction types: load / store, arithmetic / logical, branches and jumps, and floating point. The processor has a 5-stage pipeline, where stages are identified as Instruction Fetch (IF), Instruction Decode (ID), Execution (EX), Memory Access (MEM), and Write Back to Register or Memory (WB). The first two stages are identical for all instructions. Register values are read from the register file during the second stage.

The topmost level of the DLX model hierarchy contains three elements: MainControl, Datapath, and an information element Decode. MainControl aggregates an Execution element, which in turn instantiates a single InstructionSet element, containing the definition of all processor instructions. InstructionSet is related to Datapath by a *connection* (a *knowledge* relationship, as introduced in Section 3).

The DLX Datapath element aggregates all functional elements and memories. Multiplexers do not need to be included, since connections are not explicitly modeled in SimPL. The basic SimPL library already contains most classes that implement datapath elements of the DLX processor. They need to be only instantiated and parameterized. Datapath elements make a large set of micro-operations available to the control elements, such as:

```
int PC_CONTENTS()
DATA_MEM_WRITE(address, value)
int INST_MEM_READ(address)
EX_REGISTER(subregister, operation)
IF_REGISTER(subregister, operation)
```

The InstructionSet element defines the four DLX instruction types - data transfer, arithmetic and logic, control, floating point - in a hierarchical way, using the modeling features of the SIMOO environment.

For each instruction, the `InstructionSet` element defines three instruction steps. They correspond to the pipeline stages EX, MEM, and WB, which are distinct for each instruction. Each of these stages is defined as a set of micro-operations. The pipeline stages IF and ID, which are identical for all instructions, are modeled by two instruction fragments `InstrFetch` and `InstrDecode` and invoked directly from the `MainControl` element.

The `Execution` element invokes instructions (in fact: invokes micro-operations that are made available by `InstructionSet`) in a multi-cycle mode, where each processor cycle executes a single instruction step. Besides that, the `Execution` element makes micro-operations of the form

`EX (par1, par2)`

available, where `par1` represents an instruction and `par2` an instruction stage. These micro-operations are invoked by `MainControl`.

The overall control behavior and timing is defined within `MainControl`, as shown in Figure 4. Instruction fragments `InstrFetch` and `InstrDecode` are invoked first, to execute micro-operations corresponding to stages IF and ID.

```

instr4 = NOP;    // pipeline is initially empty
instr5 = NOP;
while (true) {
    EX (InstrFetch);
    EX (InstrDecode);
    // execute stage 3 (EX) of instruction decoded in instr3
    EX (instr3 = DECODE(),3);
    // execute stage 4 (MEM) of instruction instr4
    EX (instr4,4);
    // execute stage 5 (WB) of instruction instr5
    EX (instr5,5);
    // instructions advance through the pipeline...
    instr5 = instr4;
    instr4 = instr3;
    // simulation advances to the next clock cycle
    wait(1);
}

```

*Figure 4. MainControl for the DLX processor*

The information element `Decode` monitors all updates to `InstructionRegister`, an element defined within `Datapath`, and returns an instruction `inst` to be executed by micro-operation `EX(instr3=DECODE(),3)`. This information element is necessary because the control block, according to the SimPL approach, does not have a direct association to each functional

element within the datapath block. Decode, using a SIMOO monitor, may obtain the current instruction and pass it to the control block.

The movement of instructions through the pipeline is modeled by auxiliary variables *instr3* thru *instr5* in Figure 4. Variables *instr1* and *instr2* are not necessary because the first two pipeline stages are implemented by instruction fragments InstrFetch and InstrDecode that are identical for all instructions. All 5 pipeline stages are executed in parallel, and instructions move to the next stage also in parallel. The SIMOO function *wait(1)* advances the simulation clock and leads the processor to the next cycle.

Specialized information elements, that monitor the state of the pipeline, inform MainControl about the occurrence of branch and data dependencies. When one of these dependencies is detected, MainControl invokes a method that implements a desired penalty procedure. The invocation of these information elements is not shown in Figure 4.

New instructions may be easily introduced, and existing instructions may be easily modified. This can be done by implementing these changes in the control element InstructionSet. The new micro-operations that are thus created may be used by Execution. If needed, new functional elements may be created within Datapath. The micro-operations that are made available by these new elements may be invoked from InstructionSet.

## 7. THE SUPPORTING ENVIRONMENT

A dedicated environment, built on top of SIMOO, offers resources that give additional support to the SimPL modeling methodology. SIMOO already offers various modeling resources, as introduced in Section 3. It also builds a default user interface for each simulation model, including read and write access to all model variables and full control over the simulation execution.

For helping the modeling process, this environment offers windows for each control element. These windows display all micro-operations that are available for the definition of the corresponding control element. As an example, the large set of micro-operations defined in the datapath block, which are available for the specification of instructions or instruction fragments, is shown in the window corresponding to InstructionSet, so that the programmer is aware of their existence. The same happens for Execution, so that the micro-operations that are made available by InstructionSet can be browsed by the programmer.

As already mentioned in Section 5, the environment automatically generates the code of the method belonging to class InstructionSet, which executes instruction steps according to the execution mode that has been chosen in the control element Execution.

The SimPL supporting environment also gives the user the possibility of accessing and modifying values of parameters that have been defined for the various functional elements.

When a model is built and element parameters are defined for a given processor configuration, SIMOO classes are automatically generated. These classes may be edited by means of all object-oriented features of the basic SIMOO framework. By polymorphism and inheritance, for instance, elements may be specialized, the behavior of methods may be changed, or entirely new elements may be built. If these new classes follow the SimPL modeling paradigm, all resources and advantages of the SimPL environment become available.

## 8. CONCLUSIONS AND FUTURE WORK

This paper presented the SimPL methodology for modeling the precise behavior of processor architectures. SimPL is built on top of SIMOO, a framework for object-oriented modeling and simulation of discrete systems. Benefiting from object orientation, the methodology aims at a very flexible modeling approach. This feature is extremely useful for a fast exploration of the processor architecture design space and also for teaching purposes. The methodology is based on the following main concepts:

- datapath and control are explicitly separated, and each of them is modeled as an hierarchy of elements;
- data paths between functional elements are not explicitly modeled; instead, they are implicitly defined by the control behavior;
- a basic library of highly parameterized functional elements is available;
- micro-operations are offered both by datapath and control elements, corresponding to the functions they may execute;
- the control behavior is modeled by three different control elements - InstructionSet defines micro-operations that must be executed for each instruction; Execution organizes micro-operations of the instructions into steps; and MainControl defines the overall timing by invoking instruction steps and associating them to clock cycles; and
- instructions may be hierarchically defined as combinations of instruction fragments.

The separation of the control behavior into three different types of elements, together with the hierarchical definition of instructions, allows an easy modeling of a large variety of micro-architectures and timing behaviors.

From a given processor model, the methodology makes it extremely easy to obtain a new one, with a different timing behavior, or with a different set

of instructions. Full exploitation of object orientation greatly enhances the capabilities for deriving new processor models from old ones.

As a validation strategy, a complete model of the DLX processor has been developed. Other architectural classes, such as VLIW and DSP, will be modeled later on. The SimPL supporting environment is being implemented. Through this environment, classes and methods that are necessary for the methodology will be automatically generated. SimPL is being linked to CSDSim [12], an environment offering a specialized user interaction with a processor model. CSDSim emulates a classroom, following a client-server approach, where the server runs the processor model and clients implement the interaction of the instructor and students with the model.

## 9. REFERENCES

- [1] J.R.Larus. "SPIM S20: A MIPS R2000 Simulator". Available at  
<ftp://ftp.cs.wisc.edu/pub/spim/> spim\_documentation.ps
- [2] H. Grünbacher. "WinDLX Tutorial - A First Example". Available by e-mail at  
maziar@vlsivie.tuwien.ac.at
- [3] G.Adams. "DLXview - (Preliminary) User's Manual". Available at  
<http://yara.ecn.purdue.edu/~teamaaa/dlxview>
- [4] P.Verplaetse. "ESCAPE v1.1 Manual". Available at <http://www.elis.rug.ac.be/escape>
- [5] D.Burger and T.M.Austin. "The SimpleScalar Tool Set, Version 2.0". Available at  
<http://www.cs.wisc.edu/~mscalar/simplescalar.html>
- [6] G.Hadjiyannis et al. "ISDL: An Instruction Set Description Language for Retargetability". In: 34th Design Automation Conference, 1997.
- [7] A.Halambi et al. "EXPRESSION: A Language for Architecture Exploration through Compiler/Simulator Retargetability". In: DATE'99 - Design, Automation and Test in Europe. Munich, Germany, March 1999.
- [8] S.Pees, A.Hoffmann, V.Zivojnovic, and H.Meyr: "LISA Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures". In: 36th Design Automation Conference. New Orleans, USA, June 1999.
- [9] B.Copstein, F.R.Wagner, and C.E.Pereira. "SIMOO - An Environment for the Object-Oriented Discrete Simulation". In: ESS'97 - 9th European Simulation Symposium. Passau, Germany, October 1997.
- [10] F.R.Wagner, M.Oyamada, L.Carro, and M.Kreutz. "Object-Oriented Modeling and Co-simulation of Embedded Electronic Systems". In: L.M.Silveira, S.Devadas, and R.Reis (eds.), VLSI: Systems on a Chip. Kluwer Academic Publishers, 2000.
- [11] D.A.Patterson and J.L.Hennessy. "Computer Architecture: A Quantitative Approach". Morgan Kaufmann Publishers, Inc, 1996.
- [12] G.Rodrigues, D.M.Becker, and F.R.Wagner. "CSDSim: A Didactic Processor Simulation Environment Based on the Client / Server Architecture". In: 12th Symp. on Computer Architecture and High Performance Computing. São Pedro, Brazil, October 2000.

# Interconnect Capacitance Modelling in a VDSM CMOS Technology

David Bernard<sup>1</sup>, Christian Landrault and Pascal Nouet

*LIRMM, UMR C55060 CNRS / Université Montpellier II*

*161 Rue Ada, 34392 Montpellier Cedex 5, France*

*Tel: (+33) 467.418.527 Fax: (+33) 467.418.500 E-mail: [nouet@lirmm.fr](mailto:nouet@lirmm.fr)*

**Abstract:** This paper introduces a set of analytical formulations for 3D modelling of inter- and intra-layer capacitance. Based on real silicon data, we have developed and validated efficient and accurate analytical models that are an helpful alternative to lookup tables or numerical simulations.

**Key words:** interconnects, capacitance, layout extraction, closed-form models.

## 1. INTRODUCTION

It is now well established that interconnects drastically affect performances of VLSI circuits [1-3]. Even if resistance must be taken into account, capacitance remain the most limiting factor. On the one hand, they affect the total load that must be driven by logical gates. Power consumption is then directly dependent to this capacitive load while propagation delays can only be deduced from the study of distributed RC networks. On the other hand, cross-talk evaluation requires the knowledge of node to node capacitive coupling. Finally, capacitance extraction from a layout must be investigated at different levels:

- at node level, only the total node capacitance is required to evaluate power consumption and delay for short nets,
- at node-to-node level, the mutual capacitance is required to compute cross-talk noise and elementary capacitance of RC networks.

It is then necessary and sufficient to extract node to node mutual capacitance. Even if CAD vendors are proposing several alternatives, this

<sup>1</sup> currently with Atmel, Rousset, France.

problem is traditionally solved using an open tool (e.g. dracula, diva, ...) together with public domain models (like in [4]). The limitations of those models on realistic test patterns, i.e. patterns with dimensions in the range of the minimum feature size, has been pointed out by the authors in 1995 [5]. In this paper, we are proposing a new set of formulas that have been validated in a  $0.25\text{ }\mu\text{m}$  CMOS technology.

In the first section of the paper we discuss previously published models and we select the most complete ones for the purpose of accuracy comparison. After a brief description of the used methodology, the model is extensively presented and discussed in terms of accuracy and efficiency.

## 2. STATE OF THE ART

Numerous analytical models have been published since interconnect capacitance are extracted from a layout. Some of them have been validated in rather old technologies with minimum feature dimensions largely over one micrometer: they only address the capacitance of a metal wire with respect to a plane. Only models proposed during the last fifteen years are mentioned in this study. They are generally based on formulations validated against numerical simulations or large silicon test patterns. They often exhibit a lack in accuracy owing to several assumptions on process such as the shape of conductors or planarity. Moreover, these formulations often require a reference plane to compute capacitance with a good accuracy. In the following, we have studied and evaluated models found in the literature according to three different configurations: i) inter-layer coupling with overlap, ii) inter-layer coupling without overlap and iii) intra-layer coupling.

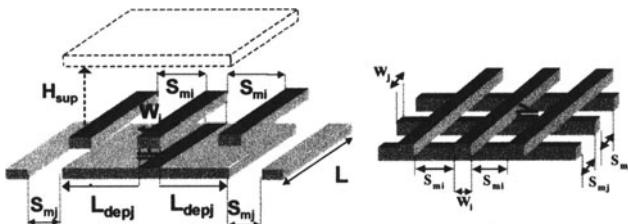


Figure 1. Reference test patterns for inter-layer capacitance with overlap.

Previously proposed models for inter-layer capacitance [6-9] have been evaluated for both 2D and 3D effects according to reference test patterns (Fig. 1). Left-side pattern exhibits only easy-to-model bi-dimensional effects. Chern's model [7] is the only one that takes into account 3D effects (e.g. right side pattern). As accuracy is comparable with other models for 2D

effects we select this model as a reference to evaluate our formulas together with the standard model (deduced from [4]).

Capacitance between two non-overlapping conductors at different interconnect layers has been neglected for years. Arora recently address this phenomenon [8].

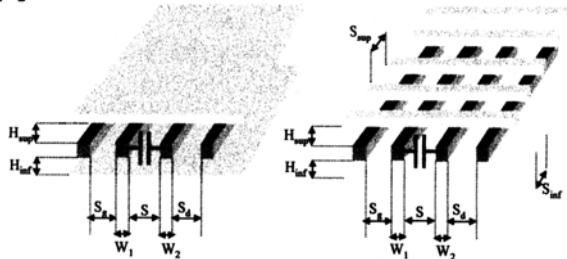


Figure 2. Reference test patterns for intra-layer capacitance.

Concerning Intra-layer couplings, published models [6-10] have been evaluated against test patterns (Fig. 2). Extraction tools generally compute the capacitance using a  $1/S$  model. A  $1/S^2$  behaviour is also mentioned in [10]. Models presented by Chern [7] and Delorme [9] are the most complete and we will evaluate our model against them.

### 3. METHODOLOGY

The process of calibrating a capacitance extraction tool for a given technology is not currently standardised. Each EDA tool vendor is proposing his own methodology that can then be adapted with silicon foundries to their technology. However, several steps are common to most of the proposed methodologies:

- the use of a library of test patterns to identify the behaviour of capacitive couplings with respect to design parameters,
- a 3D finite element simulation engine to calculate accurately each of the reference test pattern,
- the use of analytical or numerical models to speed up the extraction process at the full-chip level.

Our methodology includes the use of on-chip capacitance measurement to obtain numerous silicon data on small silicon test patterns at a reduced silicon cost [11]. It is worth noting that due to their small dimensions, these test patterns allow the identification of small dimension effects. Using these data jointly with technology dependent information, it is then possible to calibrate a numerical simulator. In our case, we have used a 2D finite element simulator that matches silicon data within 5% with a standard deviation of about 2%. A nearly infinite number of test patterns can then be

deduced to study the behaviour of the capacitance with respect to design parameters. To validate our models, we have defined two estimators corresponding respectively to the standard deviation (overall accuracy) and to the maximum deviation (robustness) between data and modelled values.

Our approach consists in the physical decomposition of the total capacitance between two conductors: like in the so-called standard model the mutual coupling is split into physically based elementary effects. Such a decomposition has already demonstrated its efficiency for a 0.8  $\mu\text{m}$  CMOS technology [12].

#### 4. INTER-LAYER CAPACITANCE MODELLING

In this section, we are presenting our model for inter-layer capacitance. The case of two isolated lines is first addressed. We then discuss the impact of lateral environment on the side-wall contribution. Vertical screening and 3D effects are also dealt with. Finally, we describe a practical implementation for layout extraction.

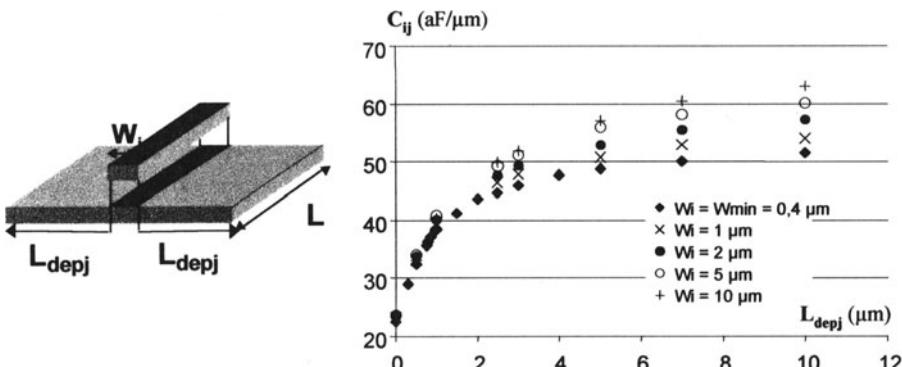


Figure 3. Left side: test pattern for characterisation of  $C_{ij}(L_{depj})$ .

Righ side: side-wall capacitance versus  $W_i$  and  $L_{depj}$ .

##### 4.1 Two isolated lines

Concerning the simple case of two isolated lines, the first test pattern that can be identified concern two conductors that are geometrically identical and perfectly stacked. Capacitance per area unit ( $C_s$ ) and side-wall capacitance per perimeter unit ( $C_{sw}$ ) are then constant parameters already defined in the standard model. In the general case, where both conductors are not perfectly stacked it is necessary to relate the side-wall capacitance with the distance in which a conductor projects beyond the other (Fig. 3). More than fifty

different patterns have been defined respectively to both design parameters  $W_i$  (from 0.4 up to 10  $\mu\text{m}$ ) and  $L_{\text{depj}}$  (from 0 up to 10  $\mu\text{m}$ ).

On the one hand we have identified the dependence of the side-wall capacitance with both design parameters (Fig. 3). This effect is particularly important for the higher value of  $L_{\text{depj}}$ .

|  |   | $W_i = 0.4 \mu\text{m}$                 |       |   |          |   |       |  |
|--|---|---|-------|---|----------|---|-------|--|
| $L_{\text{depj}}$<br>( $\mu\text{m}$ ) | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | LIRMM                                   |       |   | Standard |   | Chern |  |
|  |   | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | Error | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | Error    | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | Error |  |
| 0.5                                    | 85.2                                    | 85.8                                    | 0.7%  |   | 44.4%    |   | 55.7% |  |
| 1                                      | 97.0                                    | 96.4                                    | 0.6%  |   | 26.8%    |   | 36.7% |  |
| 2.5                                    | 109.3                                   | 110                                     | 0.6%  |   | 12.5%    |   | 21.3% |  |
| 3                                      | 111.9                                   | 112.3                                   | 0.4%  | 123                                     | 10.0%    | 132.6                                   | 18.5% |  |
| 5                                      | 117.6                                   | 117.5                                   | 0.1%  |   | 4.6%     |   | 12.7% |  |
| 7                                      | 120.4                                   | 120.1                                   | 0.3%  |   | 2.2%     |   | 10.1% |  |
| 10                                     | 123.0                                   | 122.2                                   | 0.7%  |   | 0.0%     |   | 7.8%  |  |

|  |   | $W_i = 10 \mu\text{m}$                  |       |   |          |   |       |  |
|--|---|---|-------|---|----------|---|-------|--|
| $L_{\text{depj}}$<br>( $\mu\text{m}$ ) | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | LIRMM                                   |       |   | Standard |   | Chern |  |
|  |   | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | Error | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | Error    | $C_{\text{total}}$<br>aF/ $\mu\text{m}$ | Error |  |
| 0.5                                    | 571.3                                   | 570.5                                   | 0.1%  |   | 6.4%     |   | 5.0%  |  |
| 1                                      | 585.6                                   | 581.1                                   | 0.8%  |   | 3.8%     |   | 2.4%  |  |
| 2.5                                    | 604.5                                   | 594.7                                   | 1.6%  |   | 0.5%     |   | 0.8%  |  |
| 3                                      | 608.7                                   | 597.0                                   | 2.8%  | 607.7                                   | 0.2%     | 599.6                                   | 1.5%  |  |
| 5                                      | 618.9                                   | 602.2                                   | 2.7%  |   | 1.8%     |   | 3.1%  |  |
| 7                                      | 625.8                                   | 604.8                                   | 3.4%  |   | 2.9%     |   | 4.2%  |  |
| 10                                     | 630.9                                   | 606.9                                   | 3.8%  |   | 3.7%     |   | 5.0%  |  |

Table 1. Accuracy comparison for modelling two isolated lines.

On the other hand, it is also obvious that when the width of the narrower line increases, the surface capacitance increases much more quickly than the side-wall capacitance. As a consequence the effect of neglecting the dependence of the side-wall capacitance with the conductor width should not impact significantly the total mutual capacitance. The proposed model for the side-wall capacitance is then a simple function of  $L_{\text{depj}}$  that writes:

$$C_{ij}(L_{\text{depj}}) = C_0 + (C_{ij\max} - C_0) \cdot e^{\left(\frac{-A_{ij}}{A_{ij} + L_{\text{depj}}}\right)} \quad (1)$$

where  $C_{ij\max}$  and  $A_{ij}$  are a pair of parameters with a physical meaning and not only fitting parameters.  $C_{ij\max}$  is an asymptotic value representing the maximum side-wall capacitance when  $L_{\text{depj}}$  increases up to an infinite value.  $A_{ij}$  represents a distance that traduces the “speed” of variation between the minimum value of the side-wall capacitance ( $C_{\text{sw}}$  defined with two stacked conductors) and  $C_{ij\max}$ . This parameter is calculated to optimise the average error on a few number of test patterns. Finally,  $C_0$  is calculated as  $C_0 = (C_{\text{sw}} - C_{ij\max} \cdot e^{-1}) / (1 - e^{-1})$ . It is worth noting that the physical origin of all parameters makes  $C_{ij}(L_{\text{depj}})$  robust and reliable: whatever the value of  $L_{\text{depj}}$  is, the boundaries of  $C_{ij}(L_{\text{depj}})$  are guaranteed.

Partial results are given in Table 1 for several version of the test pattern illustrated at Figure 3. It is clearly demonstrated that both standard and Chern's model that are not taking into account  $L_{dep}$  are very inaccurate especially when the conductor width is small, *i.e.* when the side-wall capacitance represents up to 80 % of the total capacitance. The same model has been also validated for the symmetrical test pattern where the upper layer projects beyond the lower level. Finally, on a set of seventy test patterns covering all significant dimensions that can be encountered in a layout, errors have been computed for each selected model (table 2).

| Error   | LIRMM | Standard | Chern |
|---------|-------|----------|-------|
| Average | 2.2 % | 10 %     | 18 %  |
| Maximal | 5.7 % | 57 %     | 72 %  |

Table 2. Accuracy comparison for inter-layer capacitance between two isolated conductors.

## 4.2 Influence of lateral screening

We have first studied the impact of the presence of conductors in the neighbourhood of two stacked conductors. The total mutual capacitance is defined between the two stacked conductors while lateral lines are grounded and separated from the middle conductor by  $S_{mi}$  (Fig. 4). In a 0.25  $\mu\text{m}$  CMOS technology, this distance may range down to 0.6  $\mu\text{m}$ . There is no maximal value, in this case we obtain the behaviour of two isolated conductors. The side-wall capacitance of two stacked conductors can then be defined as a function of the spacing  $S_{mi}$  between the lateral shields and the middle conductor (Fig. 4). Similar behaviour has been obtained for lateral screens placed at the bottom layer. It clearly appears that lateral screens are introducing an attenuation coefficient that should be multiplied to the value of the side-wall capacitance obtained for two isolated stacked lines. However, in the realistic case, screens are present at both top and bottom levels, the total attenuation will be too important if both coefficient are simply multiplied.

This discussion leads to add an amplification coefficient that will be used to amplify the side-wall capacitance in the presence of double lateral screens. Finally, it comes:

$$C_{sw}(S_{mi}, S_{mj}) = C_{sw} \cdot e^{-\left(\frac{S_{ij}}{S_{mi}}\right)} \cdot e^{-\left(\frac{S_{ji}}{S_{mj}}\right)} \cdot e^{\left(\frac{k}{S_{mi} + S_{mj}}\right)} \quad (2)$$

where  $S_{ij}$ ,  $S_{ji}$  and  $k$  are fitting parameters with a physical meaning that traduces the sensitivity (previously called "speed") of the side-wall capacitance to screen proximity.  $S_{ij}$  is first extracted to optimise the error on a set of stacked conductors with lateral screens at the top level. Then  $S_{ji}$  is

calculated for the best fit on test patterns with lateral screens at the bottom level only. Finally,  $k$  is determined when screens are present at both top and bottom levels. It is worth noting that the proposed model can be applied to calculate the side-wall capacitance of two stacked conductors whatever the screening conditions are. Obtained accuracy is reported in table 3.

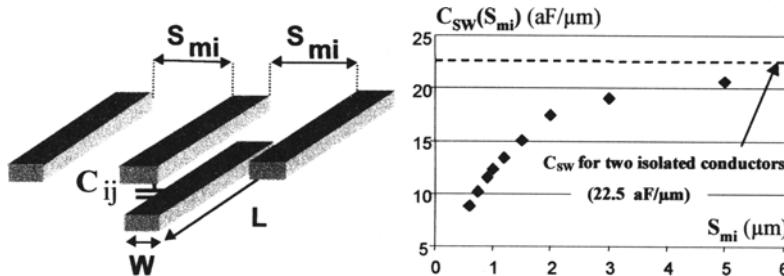


Figure 4. Left side: test pattern for lateral screening evidence. Right side: impact of screening lateral lines on the side-wall capacitance of two stacked conductors.

| Error   | LIRMM | Standard | Chern |
|---------|-------|----------|-------|
| Average | 2.1 % | 50 %     | 42 %  |
| Maximal | 5.6 % | 120 %    | 71 %  |

Table 3. Accuracy of studied models on 36 patterns of stacked conductors with lateral screens.

We then extend the previous model to the case of a conductor that projects beyond the other. The same principle has been applied to reach a general formulation for the previously proposed  $C_{ij}$  and  $C_{ji}$ :

$$C_{ij}(L_{depj}, s_{mi}, s_{mj}) = C_{ij}(L_{depj}) \cdot e^{\left(\frac{-s_{ij}}{s_{mi}}\right)} \cdot e^{\left(\frac{-s_{ji}}{L_{depj} + s_{mj}}\right)} \cdot e^{\left(\frac{k}{s_{mi} + L_{depj} + s_{mj}}\right)}$$

$$C_{ji}(L_{depi}, s_{mi}, s_{mj}) = C_{ji}(L_{depi}) \cdot e^{\left(\frac{-s_{ij}}{L_{depi} + s_{mi}}\right)} \cdot e^{\left(\frac{-s_{ji}}{s_{mj}}\right)} \cdot e^{\left(\frac{k}{L_{depi} + s_{mi} + s_{mj}}\right)} \quad (3)$$

These equations have been validated on more than 300 test patterns with a correct accuracy (Table 4). From previously presented tables, our model always exhibits a better accuracy than the two reference models selected for comparison.

| Error   | LIRMM  | Standard | Chern |
|---------|--------|----------|-------|
| Average | 6.4 %  | 63 %     | 16 %  |
| Maximal | 13.6 % | 205 %    | 92 %  |

Table 4. Accuracy of the studied models for inter-layer capacitance with lateral screens.

### 4.3 Management of vertical profiles

Results presented until now concern the coupling between two conductors from met3 and met4 layers. As no other layers have been used, bottom plane corresponds to substrate and there is no top plane. In the addressed technology, twelve different vertical environments can be defined for the pair met3 and met4. This corresponds to all possible combination of 4 different bottom planes (substrate, polysilicon, met1 or met2) with three different top planes (met5, met6 and none). To represent the impact of the vertical profile, we choose a look-up table approach: in high density layouts, a lot of “possible” vertical profiles are not realistic. As a consequence analytical modelling of vertical profile impact is not required. For each vertical profile that must be covered, a complete set of coefficients must be calculated. We then verify that the proposed model applies whatever the vertical profile is.

| Top plane                        | none      | Met5   |
|----------------------------------|-----------|--------|
| Bottom plane                     | substrate | Met2   |
| $C_s$ (aF/ $\mu\text{m}^2$ )     | 50.49     | 49.8   |
| $C_{sw}$ (aF/ $\mu\text{m}$ )    | 22.5      | 14.33  |
| $C_{ijmax}$ (aF/ $\mu\text{m}$ ) | 53.8      | 33.36  |
| $C_{jimax}$ (aF/ $\mu\text{m}$ ) | 43.72     | 32.55  |
| $A_{ij}$ ( $\mu\text{m}$ )       | 0.616     | 0.304  |
| $A_{ji}$ ( $\mu\text{m}$ )       | 0.326     | 0.293  |
| $s_{ij}$ ( $\mu\text{m}$ )       | 0.574     | 0.453  |
| $s_{ji}$ ( $\mu\text{m}$ )       | 0.542     | 0.442  |
| $k$ ( $\mu\text{m}$ )            | 0.46      | 0.764  |
| Average error                    | 5 %       | 7.4 %  |
| Maximal error                    | 13.6 %    | 14.8 % |

Table 5. Accuracy of the proposed model for inter-layer capacitance for two vertical profiles (2D effects).

Table 5 illustrates two “extreme” cases: i.e. the low density one and the higher density one where each met3-met4 pattern is delimited by two metal planes in met2 and met5. More than five hundred test patterns have been characterised and modelled for each vertical profiles and a similar accuracy has been obtained for all of them.

### 4.4 Modelling of 3D effects

3D effects have been pointed out by the authors in the middle of the 90<sup>th</sup> [5]. A typical test pattern to address 3D effects is illustrated on the right-hand side of Figure 1. Assuming that most of the 2D effects are still present we only added a new term to the 2D model that represents the contribution

of all non overlapping regions of the two central lines. We called this effect the corner effect and the component  $C_c$ , which depends on the vicinity of neighbouring lines, is expressed as follows:

$$C_c(s_{mi}, s_{mj}) = C_{cmin} + (C_{cmax} - C_{cmin}) \cdot \exp\left(-\frac{c_{ij} + c_{ji}}{s_{mi} + s_{mj}}\right) \quad (4)$$

where  $C_{cmax}$  and  $C_{cmin}$  represent respectively the maximum (no neighbours) and minimum (closest neighbours) values of the offset due to one elementary corner.  $c_{ij}$  and  $c_{ji}$  are then determined to fit the variation of the corner effect for various separation between central lines and lateral lines. Validated against nine silicon test patterns, the proposed model leads to a maximal error of 9 % and an average error of 6 %.

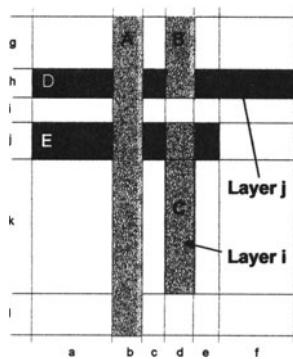


Figure 5. Practical implementation of the proposed model for layout extraction.

#### 4.5 Extraction procedure

This sub-section summarises a practical implementation of the proposed model. The layout represented at Figure 5 is composed of three metal conductors in the vertical routing direction (labelled A, B and C) and two metal conductors in the horizontal routing direction (labelled D and E). The calculated capacitance is the total capacitance between conductor E and conductor C that can be split in 7 independent terms: i) the surface capacitance, ii) four side-wall terms differing in lateral screen configuration and with different projection distance, iii) two corner effects. Finally, the total capacitance writes:

$$\begin{aligned} C_{EC} = & C_s \cdot (d \cdot j) + C_{sw}(s_{mi} = i, s_{mj} = i) \cdot d + C_{ij}(L_{depj} = e, s_{mi} = \infty, s_{mj} = \infty) \cdot j \\ & + C_{ij}(L_{depj} = (a + b + c), s_{mi} = c, s_{mj} = \infty) \cdot j + C_{ji}(L_{depi} = k, s_{mi} = \infty, s_{mj} = \infty) \cdot d \\ & + C_c(s_{mi} = \infty, s_{mj} = \infty) + C_c(s_{mi} = c, s_{mj} = \infty) \end{aligned}$$

## 5. INTRA-LAYER CAPACITANCE MODELLING

We first address the capacitance between two isolated conductors at the same level. To identify each phenomenon independently, we initially used a calibrated 2D numerical simulator to obtain data without any vertical grounded plane (even the substrate was omitted). Two design parameters can then be identified, the conductor width and the spacing between them.

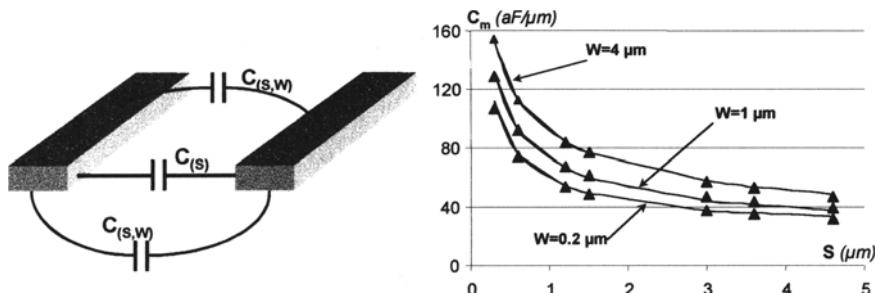


Figure 6. Left side: elementary test pattern for intra-layer capacitance. Right side: characterised vs modelled data for intra-layer capacitance in a 0.12  $\mu\text{m}$  technology.

We have then identified two elementary contributions. One owing to the faced-sides relates to the spacing between conductors. The second owing to both top and bottom sides of the conductors is also a function of conductor width. Finally, the mutual capacitance between two isolated conductors has been validated as  $C_m(S, W) = 2 \cdot C(S, W) + C(S)$  with:

$$C(S, W) = C_0 \cdot \ln\left(\frac{S + 2W}{S}\right) \quad \text{and} \quad C(S) = \frac{C_1}{\ln\left(\frac{S + k_0}{k_0}\right)} \quad (5)$$

where  $C_0$ ,  $C_1$  and  $k_0$  are three constant parameters for a given technology and a given metal layer.  $C_1$  and  $k_0$  are both strongly dependent to the thickness of the conductors. The proposed model has been validated against numerical simulations for technologies down to 0.12  $\mu\text{m}$  (Fig. 6). Discrepancies between simulated and modelled data are always below 5%. It is worth noting that our model can be used for different dielectric constants of insulating materials by changing the coefficients. In order to introduce the effect of vertical environment, we first modelled the case of two isolated conductors in presence of a substrate. From both silicon and simulation data, it appears that previously determined terms are still present but attenuated by a grounded plane. Finally, the following generalised model comes:

$$C_m(S, W) = C(S) \cdot e^{-k_s S} + 2 \cdot C(S, W) \cdot e^{-k_{sw} S} \quad (6)$$

where  $k_s$  and  $k_{sw}$  are constant parameters that depend on the vertical profile.  $C(S)$  and  $C(S, W)$  correspond to previously defined equations where  $C_0$  and

$C_1$  depend on the vertical profile. Finally, only  $k_0$  appears to be constant for a given interconnect thickness. The proposed model has been validated for all possible vertical profiles. Table 6 reports obtained accuracy for two conductors over a substrate. 76 test patterns have been defined for different line width (0.4  $\mu\text{m}$  up to 6  $\mu\text{m}$ ) and different spacing (0.6  $\mu\text{m}$  up to 6  $\mu\text{m}$ ).

| Error   | LIRMM | Standard | Delorme | Chern  |
|---------|-------|----------|---------|--------|
| Average | 1.5 % | 28 %     | 8.3 %   | 5.7 %  |
| Maximal | 7.5 % | 61 %     | 17.6 %  | 15.3 % |

Table 6. Accuracy of the studied models for intra-layer capacitance of two met3 conductors over a substrate.

We have then defined a set of 24 test patterns with two met3 conductors surrounded by two planes in met2 and met4 layers. As capacitance are very low, relative errors can be very important. For this reason, we are only reporting, for each evaluated model, the number of patterns that permits an error lower than the maximal error encountered with our model (Table 7).

| Error   | LIRMM  | Standard | Delorme | Chern  |
|---------|--------|----------|---------|--------|
| Average | 8.3 %  |          |         |        |
| Maximal | 13.2 % | 0 / 24   | 1 / 24  | 8 / 24 |

Table 7. Accuracy of the studied models for intra-layer capacitance in high density designs.

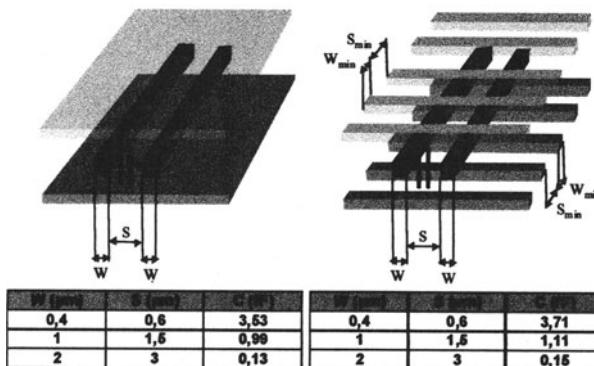


Figure 7. Screening effect of a plane versus a minimum pitch network of conductors.

The complete model [13] also includes the effect of non-symmetrical patterns and the effect of lateral screens through an attenuation factor. Finally, we addressed 3D effects. These effects are present when studying vertical screens. We have identified that a network of conductors routed at the minimum pitch are inducing nearly the same attenuation than a plane (Fig. 7).

## 6. CONCLUSIONS

In this paper we have introduced a set of analytical models for interconnect capacitance extraction in VLSI circuits. These models have been validated in a 0.25  $\mu\text{m}$  CMOS technology. Accuracy has been demonstrated to be within about 10 % of real silicon or 3D numerical simulation. Robustness of these models is guaranteed owing to the self-bounded behaviour of each term. These models are also efficient and it is rather trivial to tune them on a given technology using a minimum set of reference data (real silicon or 3D numerical simulations). The proposed models [13] are also complete as all phenomenon have been covered even if not presented in this paper (i.e. vertical screen due to arbitrarily routed networks on the intra-layer capacitance or modelling of the inter-layer capacitance of non-overlapping conductors).

## 7. REFERENCES

- [1] P. Felix, "Interconnects for ULSI: State of the art and Future trends", Proc. ESSDERC, 1995, pp. 5-14.
- [2] K. Rahmat and al., "A Scaling Scheme for Interconnect in Deep-Submicron Processes", IEDM'95, pp. 245-248.
- [3] M.T. Bohr, "Interconnect Scaling – The Real Limiter to High Performance ULSI", Proc. IEDM'95, pp. 241–244.
- [4] T. Sakurai and K. Tamaru, "Simple Formulas for Two- and Three-Dimensional Capacitances", IEEE Trans. El. Devices, vol. ED-30, pp. 183-185, February 1983.
- [5] P. Nouet and al., "On-Chip measurements: a way for accurate modelling of interconnect capacitances in a CMOS process", PATMOS, October, 1995, pp. 290-301.
- [6] C. Kortekaas, "Interconnect Capacitance Characterization for Mos-IC Process- and Circuit Design", IEEE Proc. on Microelectronic Test Structures, pp. 39-44, Feb. 1988.
- [7] J.H. Chern and al., "Multilevel Metal Capacitance Models for CAD Design Synthesis Systems", IEEE Electron Device Letters, Vol. 13, N°1, pp. 32-34, January 1992.
- [8] N.D. Arora and al., "Modeling and Extraction of Interconnect Capacitances for Multilayer VLSI Circuits", IEEE Trans. on CAD, Vol. 15, N°.1, pp. 58-67, Jan. 1996.
- [9] N. Delorme, Ph.D thesis, University of Grenoble, November 1997.
- [10] D. Sylvester and al., "Investigation of Interconnect Capacitance Characterization Using Charge-Based Capacitance Measurement Technique and 3D Simulation", IEEE Jal of Solid-State Circuits, Vol. 33, N°. 3, pp. 449-453, March 1998.
- [11] P. Nouet and al., "Use of test structures for characterization and modeling of inter- and intra-layer capacitances in a CMOS process", IEEE Trans. On semiconductor manufacturing, vol. 19, May 1997, pp. 233-241.
- [12] A. Toulouse and al., "Efficient 3D Modeling for Extraction of Interconnect Capacitances in Deep Submicron Dense Layouts", DATE'99, March 9-12, 1999, pp. 576-580.
- [13] D. Bernard, PhD thesis, University of Montpellier, November 2000.

# **Abstract Communication Model and Automatic Interface generation for IP integration in Hardware/Software Co-design**

C. Araujo and E. Barros  
*Centro de Informática, UFPE*

**Abstract:** The use of standard languages like VHDL and C for the description of hardware and software IP has became a common practice. Despite this, these languages, specially the hardware description languages lack constructs that allow the IP designer to develop highly re-usable IP blocks. In this paper is described an abstract communication mechanism that uses extensions to the VHDL language, communication library for software and automatic interface generation for the easy integration of IP modules.

## **1. INTRODUCTION**

Current processes for IC fabrication allow the easy integration of millions of transistors in a single chip. Despite this evolution in the fabrication processes, designer are unable to fulfill this enormous capacity respecting a more constrained time-to-market. This phenomena is known as “Design Productivity Gap”. One of the most promising solutions to this problem is the IP reuse. Where known, reliable system modules are integrated in digital designs.

One problem arises here, how to build and integrate these IP modules to designs. Languages currently used for hardware description, like VHDL and Verilog don't have the abstract communication mechanisms needed for an easy “plug and play” integration of IP modules. Use of mixed hardware/software modules is a task even harder.

In this paper is described the mechanisms used in the PISH Co-design system for an easy integration of IP modules. The solution uses a

combination of automatic interface generation, extension of the VHDL language with the introduction of abstract communication constructors for hardware modules and communication library for the software ones.

The rest of this paper is organized as follows: in section 2 is given an overview of the PISH Co-design system. In next section some of the related works in this area are shown. Section 4 describes the proposed system InterfPISH, which allows automatic interface generation as well as code generation for hardware and software synthesis. A case study and results obtained are given in section 5 and finally a conclusion section is given.

## 2. THE PISH CO-DESIGN SYSTEM

An overview of the PISH Co-design system can be seen in *Figure 1*. It can be divided into three main stages: specification and partitioning, co-synthesis and prototyping.

The first stage, specification and partitioning taking an initial specification of the digital system to be implemented and partition it into processes to be implemented in hardware and software components. This system uses occam as specification mechanism [4]. The main reason to use occam is that, being based on CSP [5] occam has a simple and elegant semantics, given in terms of algebraic laws, which allows the partitioning be performed by applying a series of algebraic transformations into the initial occam description in order to preserve the semantics. These transformations change the initial specification and as result new concurrent processes and communication are introduced. Despite being a new description, the transformed one is guaranteed to have the same semantic as the original specification. The set of transformation rules is applied according to the results of a cost analysis obtained by using a Petri net based estimator and clustering techniques [3]. The interface generation depends on the number of concurrent processes of different nature (hardware/software) that communicate, the type of the data being transferred between the processes and also the target architecture taken into account. Most co-design systems considers a very simple architecture composed of one software component. In order to have a pre-defined protocol some systems consider the hardware running as a co-processor, i.e. hardware and software do not execute concurrently [1][2].

In this work, software and hardware can run concurrently and for that device drivers must be generated at the software side, as well as specific hardware to make transparent for the hardware side which processor is being used. The interface between hardware modules must also be synthesized.

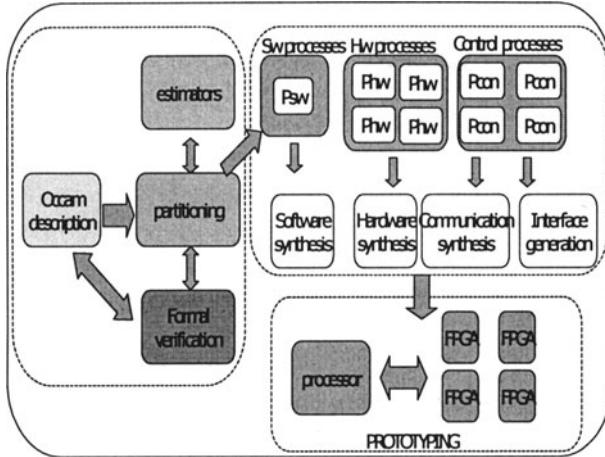


Figure 1: PISH Co-design system

### 3. THE PROPOSED APPROACH

In this section is described an approach for co-synthesis that allows the generation of code for hardware and software synthesis, including the automatic interface generation between the hardware and software parts of the digital system.

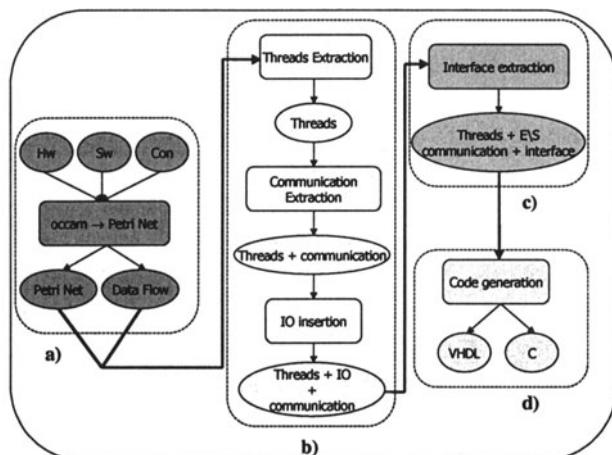
The flow for co-synthesis and interface generation is shown in *Figure 1*. It can be divided in four parts: translation into an internal format representation, threads and communication extraction, interface generation and the last phase code generation. In the first part of the flow, *Figure 1a*, a description of the partitioned system representing the software, hardware and communication processes is given as input.. These processes are described using the occam language. This is the result of the automatic partitioning tool of the PISH system. These descriptions are translated to Petri Net representing the control and a graph representing the data flow.

The second part, *Figure 1b*, performs the capture of the concurrent threads existing in the digital system, the extraction of the communication among the concurrent threads and for the insertion of IO modules in order to allow for interaction with the outside world. Initially are identified in the Petri Net representation all the concurrent threads. These threads can only execute sequential statements. Concurrency in the system is obtained by the simultaneous execution of several threads. Threads can activate others threads in the system and also communicate with other threads through communication channels. After the threads identification, the communication among them is extracted. This information is used for the

implementation of communication channels in hardware and software, depending on the nature of the communicating threads. In the last part of this phase IO elements are inserted in the digital system. These IO elements have a dual mean. They act by one side as an execution thread and can communicate with others threads through the use of communication channels and can also control IO devices, at the other side.

The following part, *Figure 1c*, is responsible for the automatic interface generation between the hardware and software parts of the digital system. In this step the target architecture must be taken into account. The designer can choose a particular target architecture from a library and code for interface between hardware and software is automatically generated. The system may be composed of several concurrent threads and most may want to communicate simultaneously. To handle this situation the generated interface is able to schedule its use of the shared resources by concurrent threads. The data transferred in the communications can also be of any data length. So the interface is also responsible for the transference of data independent of its length.

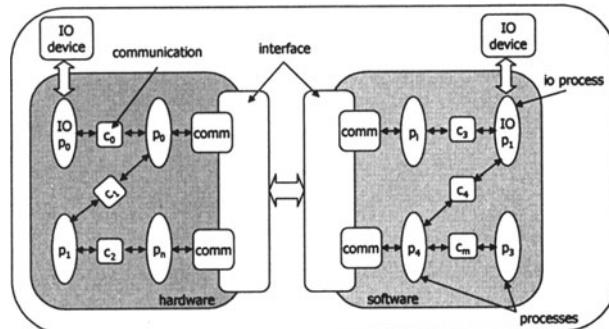
The last step in *Figure 1* is the code generation phase. VHDL code is automatically generated for the hardware parts of the system while standard C code is generated for the software parts of the digital system. Both codes are standard and can be synthesized by most of the VHDL and C tools.



*Figure 1: InterPISH: co-synthesis + interface generation*

### 3.1 System Model

In this approach the defined model for the digital system is seen in *Figure 2*. As can be seen it is completely symmetrical, what means that the software and hardware parts of the system are treated in the same way and have basically the same elements. The digital system is composed of concurrent executing processes or threads that communicate through communication channels. These channels implement the synchronous CSP communication semantics [6]. The IO processes are responsible for the control and transfer of data to/from the IO devices. This way the IO processes or IO threads have a dual behavior, they can communicate through communication channels and control IO devices. Between the hardware and software parts of the digital system is the interface component. This component is also symmetrical, with the same parts implemented both in hardware and software. This component performs the communication among threads of different nature and has some important characteristics: is transparent to the communicating threads, it can transfer different data lengths and can also schedule the use of the shared resources among several concurrent threads communicating through the interface.



*Figure 2: System Model*

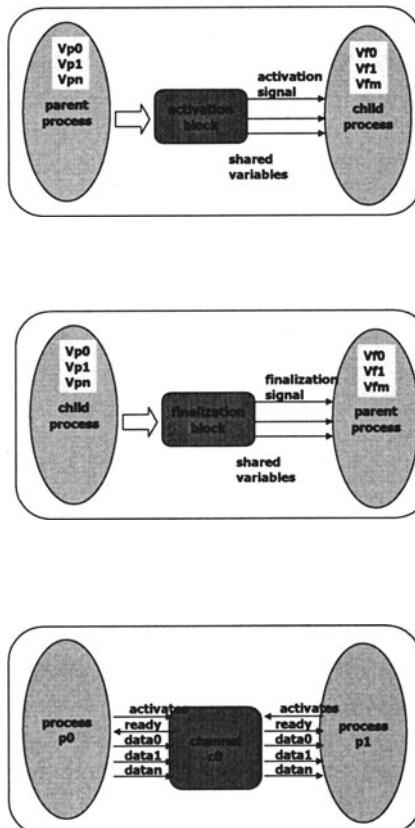
### 3.2 Communication Model

Two communication models are used in the implementation of the digital system. Direct communication and synchronous communication by channel.

Direct communication is used in two occasions. The first is used when one thread activates another one. When this happens data may be need by the activated thread, so there is a direct transfer of data from the parent thread (the one that activates) and the son thread (the one that is activated). The second situation happens when one son thread finishes its execution and then returns to the parent thread the data previously transferred. These data

values are returned updated. In *Figure 3* a parent thread activates a son thread. The set  $\{V_{p0}, \dots, V_{pn}\}$  represents the variables used by the parent thread and  $\{V_{f0}, \dots, V_{fm}\}$  represents the set of variables used by the son thread. The values of the shared variables are transferred from the parent thread to the son thread by using the activation block shown in *Figure 3*. This block is also responsible for sending the activation signal from the parent thread to the son one. The opposite happens with the finalization block. This one, shown in *Figure 4*, is responsible for returning the updated values of shared variables and also to indicate to the parent thread that the son thread has finished its execution.

The second communication type models the synchronous communication by channel. This communication model implements the occam communication semantics for concurrent processes that cannot share variables[7]. This model is shown in *Figure 5* where two concurrent threads  $p_0$  and  $p_1$  transfer data synchronously through the channel  $c_0$ .



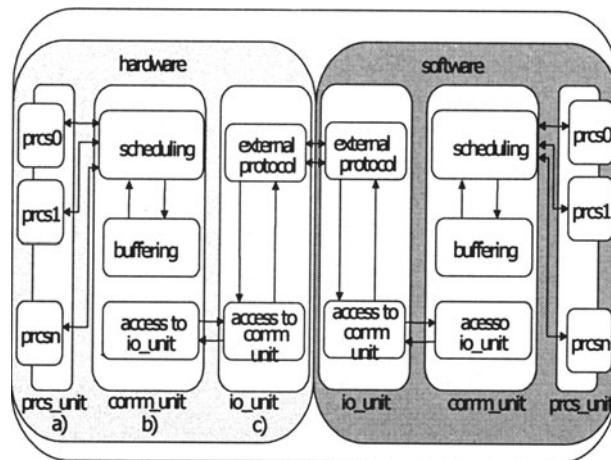
*Figure 5: Synchronous channel communication*

### 3.3 Automatic Interface Generation

This section details the automatic interface generation done in the InterfPISH tool. The communication models shown in the previous section are independent on the nature of the two communicating threads. It doesn't matter whether the two threads are implemented in hardware, software or one is in hardware and the other in software. If the two threads have the same nature the communication component (activation block, finalization block and communication channel) is implemented in the same technology (hardware or software). For instance if the two concurrent threads are in hardware and communicate using communication channel, the channel is implemented as a hardware component. In the software case the channel is implemented as a data structure and functions.

When the threads have different natures an interface is built. The model of the interface can be seen in *Figure 6*. The interface model is completely symmetrical and layered. The interface has three layers: prcs\_unit(*Figure 6a*), comm\_unit(*Figure 6b*) and io\_unit(*Figure 6c*).

The prcs\_unit layer is responsible for implementing the communication components (activation block, finalization block and communication channel). This makes the communication transparent for the threads, once they only communicate through these components. This layer is responsible for rebuilding the data so the threads can used them.



*Figure 6: Interface*

The second layer, comm\_unit, is responsible for the scheduling of the interface and work as a buffer. The scheduling is need because the interface is a limited resource that can be used by several concurrent threads. The buffering function allows that several communications take place while the buffers are not full. This can make the communication faster.

The last layer, io\_unit, connects the software component (processor) to the hardware component (FPGA). This layer is the only one that depends on the target architecture. The tool InterfPISH allows the user for choosing one from several io\_unit stored in a library depending on the target architecture. This layer is not fixed as the previous one.

### **3.4 IO Threads**

The concurrent threads that compose the system may need some data from the outside world. But these threads are not able to access the IO devices directly. The access is performed by the use of special IO threads. These threads are stored in libraries and the designer can choose the IO thread depending on the device to be controlled and on the data type (length) to be transferred. A process, or thread, p0 gets data from the input device using the channel to receive data from the IO thread. It can include three distinct blocks: a device control block, a type composition block and channel control. The device control block is responsible for getting data from/to the IO device. It is able to activate the control and data lines of the IO device when some data is requested or must be sent to the device. The second block, type composition, is responsible for adapting data types with different lengths to be transferred through the channel. Channels are only able to transfer specific data lengths. This means a channel that transfer a 8 bit integer type cannot transfer a 16 bit word type and it must be handled as two 8 bits data. So the composition block is responsible for composing the data coming from the device control, where the data is seen as a stream of bits, to the channel specific type. The last block is responsible for controlling the communication channel. This makes the IO thread to be seen as an ordinary thread by the communication channel.

### **3.5 Hardware/Software Co-synthesis**

As seen before the digital system is modeled as a set of concurrent threads. Each of these threads executes sequentially. Another characteristic of the threads is that they can activate other threads that run concurrently. The threads must also be responsible for informing its parent threads that they have finished their execution. This characteristic is necessary because once a parent thread activates several concurrent son threads it must wait until all the son threads finish execution. This implements the occam semantics for concurrent processes, where one processes can activate concurrent processes and waits until they all complete [7].

In our case each thread is represented as a FSM (Finite State Machine). This representation is consistent with the model for the digital system because the FSM can execute tasks sequentially and can implement control constructs like decision and loops. In this model the state transition can represent a condition or an action.

As said before the thread is able for activating son threads. This is done by a state transition that signals to the sons that they must leave the initial state. An activation transition can activate any number of son threads. After this transition, the FSM goes to a state where it waits for all the sons to indicate their finalization.

The other state transitions can indicate an action. An action can be one of the following: logical operation, arithmetic operation, decision, null action and stop action, which represents a deadlock of the thread that does nothing and stay forever in this state.

A decision allows a changing in the sequence of states depending on conditions. The decisions can be a conditional or loop.

#### 4. CASE STUDY: ATM SWITCH

In this section it is shown some results by applying the proposed methodology and the tool interfPISH to the interface generation of an ATM switch controller proposed in [8] whose partitioning is described in detail in [9]. This ATM switch controller must decide whether a cell must be sent or not based on four policy algorithms. The aim of discarding a cell is to reduce the traffic on an ATM network.

The ATM switch executes four types of policy: high peak, high average, low peak and low average. The peak policies are related to the maximum rate that is established during connection negotiation. The average policies observe the average number of cell arrivals during the connection. The cells can have two types of priority: high and low. A high priority cell is submitted to the high peak and to the high average policies. If approved in both of them the cell is accepted. In the case the cell is rejected in some of the policies it is submitted to the respective low policy. If reproved in any of the low policies the cell is rejected, on the contrary, if approved in both the low policies the cell is accepted and its priority is verified. When the cell has a high priority, this priority is changed to low priority. A low priority cell is submitted to both the low peak and low average policies and is rejected if not approved in both of them.

In the example all the policies are based on the leaky bucket algorithm. The different values for the parameters X, LCT, I and L determine which policy is being used.

In figure 7 can be seen the partitioning result in occam for the ATM switch. It is composed of the protocol and channel declarations that define the communication among the processes and with the outside world. The partitioned system is composed of several concurrent processes that are under the first PAR constructor. In the figure are highlighted the four policies processes that must be implemented in hardware while all the other processes are implemented in software. From this occam representation of the partitioned system is generated a Petri Net representation. This Petri Net is composed of 156 places and 154 transitions.

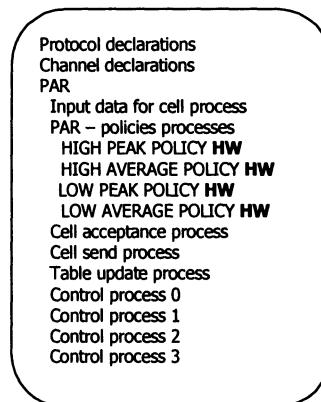


figure 7: Hw/Sw partitioning in occam

The tool extracts the concurrent threads from the Petri Net representation of the partitioned system. In this case 14 concurrent threads are generated and the results are summarised in the *Table 1*. The table gives the number of places and transitions for each thread and also its nature that can be hardware or software thread. The hardware threads are the 4 policy processes and can be noted that they all have the same number of places and transitions. This comes from the fact that only the parameters are different, the policy processes are equal.

| Thread type      | Number |
|------------------|--------|
| Software thread  | 10     |
| Hardware threads | 4      |

*Table 1: Thread results*

In *Table 2* the results for the IO threads selected by the system designer are shown. In this example all the IO threads are implemented in hardware. For each IO thread two files are generated, an VHDL\* file and a VHDL file. The VHDL\* language is an extension of VHDL that has constructs for

synchronous communication. From this extension is generated standard VHDL code for synthesis.

| Number of IO Threads | Type     |
|----------------------|----------|
| 6                    | Hardware |

Table 2: IO Threads

As mentioned before the interface is implemented in layers. The last layer is responsible for implementing the 3 types of communication schemes between threads in hardware and software. One file is generated for each policy thread, resulting in four files. In table 4 are shown the results for the interface in hardware.

| Block name    | Type     | Number of lines |
|---------------|----------|-----------------|
| Communication | Hardware | 1089            |
| Activation    | Hardware | 944             |
| Finalisation  | Hardware | 932             |

Table 3: interface in hardware

The software implementation is simpler than the hardware one. In this case header and C files are generated for the parts of the system to be implemented in software. In *Table 4* are summarized the software results. The first file represents the whole system in software. It contains the main function. The second file, processos.c, implements the threads in software. The next file, comunicacao.c, implements the communication in software. As there are no IO threads to be implemented in software, no files are generated. The last three lines of the table contain the three layers of the interface. As in the case of the hardware interface, the io\_unit.c file is generated based on a description of the target architecture while the others are generated automatically.

| C file          | Lines | H file        | Lines |
|-----------------|-------|---------------|-------|
| atm_protocolo.c | 58    | -             | -     |
| processos.c     | 573   | processos.h   | 11    |
| comunicacao.c   | 1997  | comunicacao.h | 791   |
| e_s.c           | -     | -             | -     |
| io_unit.c       | 40    | io_unit.h     | 2     |
| Comm_unit.c     | 230   | comm_unit.h   | 17    |
| prcs_unit.c     | 808'  | Prcts_unit.h  | 182   |

Table 4: software results

## 5. CONCLUSIONS

In this paper has been described the characteristics and mechanisms of the PISH Co-design system that makes easier the integration of IP modules in a design, independently whether the IP must be implemented in hardware or software. The problem has been approached in two ways. Firstly making easier IO devices integration and secondly through the automatic interface generation. This allows the designer migrate from hardware to software IP's and vice-versa.

This work uses ideas of HDL extension where abstract communication mechanisms are used for the VHDL language. It also allows the reuse of modules stored in library, both for interface and also for IO devices.

The adopted implementation clearly separates the system in concurrent threads, communication, interface and IO components and uses an strategy based on the use of library components, IO threads and inner part of the interface, and automatic generation of code.

The tool can taken into account different architectures, since new target architectures and new IO threads can be added into the library. This way the designer can have more choices for the implementation of the digital system.

## 6. REFERENCES

- [1] Daniel D. Gajski, Rainer Dömer, Jianwen Zhu *IP-centric Methodology and Design with the SpecC Language* Contribution to NATO-ASI workshop on System Level Synthesis, Il Ciocco, Lucca, Italy, August 1998.
- [2] R. Ernst , J. Henkel, T. Benner, *Hardware-Software Co-Synthesis for Microcontrollers*- IEEE Design and Test of Computers, pp. 64-75, December 1993
- [3] E.Barros and W. Rosenstiel A Clustering Approach to Support Hardware/Software Partitioning". In: K. Buchenrieder, and J. Rozenblit (eds.), Computer Aided Software/Hardware Engineering. Chapter 11- IEEE Press, 1994.
- [4] D. Pountain and D. May, *A Tutorial Introduction to OCCAM Programming*. Inmos BSP Professional Books, (1987).
- [5] C. A. R. Hoare, *Communicating Sequential Processes* Prentice-Hall, 1985
- [6] C. A. R. Hoare, *Communicating Sequential Processes* Prentice-Hall, 1985
- [7] Geof Barret *occam 3 reference manual*, INMOS, 1992.
- [8] J.A. G. Lima "Um controlador microprogramado para comutadores ATM". PhD. Thesis, Universidade Federal da Paraiba, Brasil, 1999.
- [9] J. Yioda *PartS – Uma Ferramenta de Suporte ao Particionamento Hardware/Software*. Recife: Universidade Federal de Pernambuco, 1999. Dissertação Mestrado.

# An Evolutionary Approach for Pareto-optimal Configurations in SOC Platforms

Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi

*Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Università di Catania,  
Italy*

**Abstract:** One of the most important problems in SOC platforms design is that of defining strategies for tuning the parameters of a parameterized system so as to obtain the Pareto-optimal set of configurations that provide multi-criteria optimisation. The paper proposes a methodology based on evolutionary techniques for exploration of the range of possible configurations of a parameterized system [1]. A highly parametric system-on-a-chip for digital camera applications will be taken as a case study and a multi-objective genetic algorithm will be used to search for the power-performance trade-off surface. The methodology proposed will be compared with that implemented in Platune [2] in terms of both accuracy and efficiency in relation to the number of simulations performed.

**Key words:** Parameterized systems, system-on-a-chip architectures, design space exploration, genetic algorithms, multi-objective optimization, Pareto-optimal configurations, power/performance-tradeoffs.

## 1. INTRODUCTION

A recent reduction in the time to market has led to the development of a new approach to IP-based design in which a highly parametric pre-designed system-on-a-chip (SOC) is configured according to the application it will have to execute. This new approach called *configure-and-execute* [3] is based on the presence of highly parametric IPs (Intellectual Properties) representing the basic components of a SOC. Once the architecture of a system has been designed, that is, it has been decided which IPs to use, it is necessary to find the optimal configuration for them according to the specific application (or set of applications) that have to be executed. The values

chosen for these parameters (bus sizes, coding techniques, cache parameters, arbitration schemes, etc.) are those that optimise a function which almost always depends on three main variables: area, power and performance.

The greatest problems in this area regard exploration of the range of possible system configurations in search of the optimal configuration for a given application. There are, in fact, a number of parameters involved (bus sizes, cache configurations, software algorithms, etc.), each of which has a great impact on design constraints such as area, power and performance. An exhaustive analysis of all possible configurations is thus computationally unfeasible.

The aim of this paper is to present a general methodology to search for the Pareto-set of configurations of a parameterized system that optimise the system in relation to various objectives. The methodology uses multi-objective optimisation techniques based on genetic algorithms. The results obtained in a case study (a highly parametric SOC for digital camera applications) show the efficiency of the approach in terms of both accuracy and the number of simulations required for the exploration.

The paper is structured as follows. In the next section, we review the related work which focuses on tuning methodologies for parameterised systems. In Section 3 the problem will be stated in formal terms. In Section 4 we will present our approach to design space exploration for parameterized systems. The approach will be validated in Section 5 on a highly parametric architecture for digital camera applications. Finally, Section 6 provides our conclusions and indications as to future developments.

## **2. RELATED WORK**

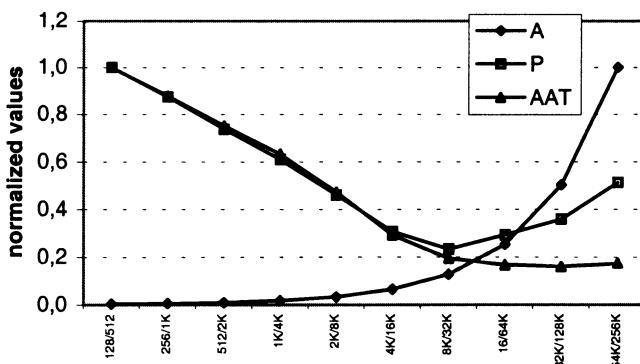
Research in the field of parameterized system design has led to the definition of various approaches to explore the range of configurations. In [4] sensitivity analysis was used to search for the configuration that minimises the power-delay product for a cache memory. In [5] mono-objective genetic algorithms (GAs) were used to search for optimal configurations in terms of area, power and average access time for a memory hierarchy. In [6] a system comprising a CPU, caches and main memory and the interfaces between these cores was analysed to show the power-performance trade-off for various technologies. Of course, any technique used to explore a range of configurations requires tools to evaluate the configurations and thus system-level simulation and estimation techniques. These tools have to be capable of performing system-level simulations in as short a time as possible as well as estimating variables that are typical of a lower level of abstraction (e.g. clock cycles, power consumption) with an

adequate level of accuracy. In [7] the authors used power estimation data obtained from the gate-level for a cores representative input stimuli data, and propagated this data to a higher (object-oriented) system-level model, which is parameterizable and executable. They achieved simulation speedups of over 1000 with accuracies suitable for making reliable power-related system-level design decisions. In [8] the same authors describe a method for speeding up the evaluation further, through the use of instruction traces and trace simulators for every core, not just the microprocessor core.

### 3. STATEMENT OF THE PROBLEM

The problem dealt with in the paper is that of finding the best configurations (on the basis of certain indexes of optimality) for a parameterized system. These aims often clash, in that an improvement in one index may cause degradation in others. This means that in multi-objective optimisation problems reference is not made to an optimal solution (i.e. one that simultaneously optimises all the objectives), but rather to trade-off solutions (or configurations), that is, solutions that provide a trade-off between various objectives.

For example, *Figure 1* shows the normalised trends for area (A), total switching capacitance (P) and average access time (AAT) for a three-level memory hierarchy with various cache sizes. In each configuration the first-level caches are of the same size while the second-level cache is 4 times the size of the first-level ones. It is clearly impossible to find a configuration that optimises all the objectives at the same time.



*Figure 1.* Normalised trends for area, total switching capacitance and average access time for various cache configurations.

Exploration of a range of configurations for a parameterized system can be defined as a set of techniques and strategies to be used to determine *mutually non-dominated* configurations (the concept of dominance will be explained below). The solution to these problems falls into the multi-objective optimisation strategy class. Multiobjective optimisation (also called multicriteria optimisation, multiperformance or vector evaluation) can be defined as the problem of finding [9]: *a vector of decision variables which satisfies constraints and optimises a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term “optimise” means finding a solution which would give values for all the objective functions such as to be acceptable to the designer.*

In formal terms we can define the problem in this way: find the vector  $\underline{x}^* = [\underline{x}_1^*, \underline{x}_2^*, \dots, \underline{x}_n^*]^T$  which will satisfy the  $m$  inequality constraints:

$$g_i(\underline{x}) \geq 0 \quad i = 1, 2, \dots, m \quad (1)$$

the  $p$  equality constraints

$$h_i(\underline{x}) = 0 \quad i = 1, 2, \dots, p \quad (2)$$

and optimizes the vector function

$$\underline{f}(\underline{x}) = [f_1(\underline{x}), f_2(\underline{x}), \dots, f_k(\underline{x})]^T \quad (3)$$

where  $\underline{x} = [\underline{x}_1, \underline{x}_2, \dots, \underline{x}_n]^T$  is the vector of decision variables (i.e. a vector representative of a configuration of a parameterized system).

Let be  $\mathfrak{I}$  the set of vector  $\underline{x}$  that will satisfy (1) and (2). We say that a point  $\underline{x}^* \in \mathfrak{I}$  is Pareto-optimal if for all  $\underline{x} \in \mathfrak{I}$  either

$$f_i(\underline{x}) = f_i(\underline{x}^*) \quad i = 1, 2, \dots, k$$

or there is at least one  $j \in \{1, 2, \dots, k\}$  such that

$$f_j(\underline{x}) > f_j(\underline{x}^*)$$

This definition states that  $\underline{x}^*$  is Pareto-optimal if there exist no vectors  $\underline{x} \in \mathfrak{I}$  that decrease the value of any component of the cost function (assuming that the objective function is a cost function to be minimised) without

increasing the value of another component of the cost function. Unfortunately, the Pareto optimum is not a single one but a set of solutions called *non-inferior* or *non-dominated* solutions.

## 4. METHODOLOGY PROPOSED

In this section we will present an approach to design space exploration (DSE) for a parameterised system based on multi-objective genetic algorithms (GAs) to determine the Pareto-optimal configurations.

### 4.1 Problem Formulation

Current SOC platforms integrate a number of parameterised cores that make it possible to cover a wide range of applications. The space of possible configurations that can be mapped onto a SOC platform is so vast that an exhaustive search for the Pareto-optimal configurations is computationally unfeasible. Evaluation of a configuration requires simulation of the system once configured. Even if a high-level model of the system is available, thus allowing for rapid simulation of a configuration and estimation with a reasonable degree of accuracy of the variables to be optimised, it would be unthinkable to evaluate the whole range of configurations in order to find the Pareto optimal-set.

One possible approach to exploring the range of configurations uses heuristic techniques to limit the range [4] [10]. The main disadvantage of this approach is that it requires accurate analysis of the architecture to identify and discard any Pareto-dominated configurations and thus avoid the need to simulate them. This can be solved by using evolutionary techniques and thus treating the exploration in terms of a problem of global optimisation [5].

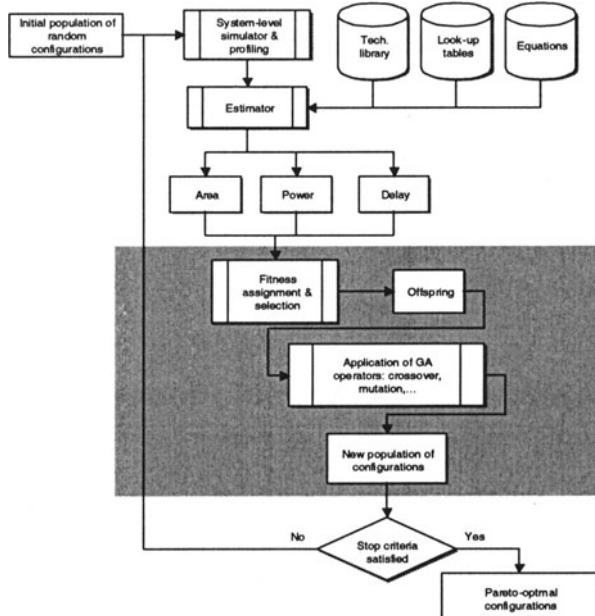
### 4.2 GA-based Approaches to Multi-objective Optimisation

Application of evolutionary algorithms (EAs) in multiobjective optimisation has attracted the attention of researchers from different backgrounds [11]. GA-based approaches to multiobjective optimisation are divided into two classes: those not based on the notion of Pareto optimum and Pareto based ones. The first class includes approaches that use aggregating functions to transform the problem of multiobjective optimisation into one of scalar optimisation [12] [13]. This approach was

used in [5] to search for the configurations that minimised a cost function defined as the aggregate of the area, power and average access time parameters in a memory hierarchy. The main disadvantage of approaches based on aggregation functions is that they do not generate proper Pareto optimal solutions in the presence of non-convex search spaces, which is a serious drawback in most real-world applications. This can be solved by using Pareto-based approaches in which the idea is to find the individuals that are Pareto non-dominated by the rest of the population. These individuals are then assigned the highest rank and eliminated from further contention. Another set of Pareto non-dominated individuals are determined from the remaining population and are assigned the next highest rank. The procedure is repeated until the whole population is suitably ranked.

### 4.3 Our Proposal

In this paper we have considered multiobjective optimisation techniques that use Pareto-based GAs. More specifically, we chose the *Strength Pareto EA* (SPEA) [14] [15] approach, which is very effective in sampling from along the entire Pareto-optimal front and distributing the solutions generated over the trade-off surface. The flow proposed is shown in *Figure 2*.



*Figure 2.* Design flow.

Initially a population of random configurations is generated. Each configuration is mapped onto the SOC platform and the specific application is executed. The information collected (cache misses, memory accesses, transitions on the buses, etc.) is used by an estimation process which, by means of mathematical equations, look-up tables and technological parameters, gives a fairly accurate estimate of the variables to be optimised (e.g. area, power, performance etc.). Each configuration is assigned a fitness value: the higher this value, the “better” the configuration is in relation to the variables to be optimised (objectives to be reached). The configurations with the highest fitness values are selected and genetic operators (crossover and mutation) are applied with a probability proportional to their fitness value. The population thus generated is the input for a new iteration (or generation). This cycle is repeated for a sufficiently large number of generations.

The algorithm was implemented using Galib [16] (a C++ library of genetic algorithm components). A configuration is represented by an individual of the population whose genome defines its parameters. Each gene represents a system parameter (defined by means of an allele) that only codes values defined within the range that is admissible for the parameter involved. Impossible configurations were excluded by using the approach classified in [17] as rejection of unfeasible individuals.

## 5. APPLICATION OF THE PROPOSED METHODOLOGY

In this section the approach described in the previous one will be applied to a highly parametric system for digital camera applications, to obtain the Pareto-optimal configurations that optimise the dissipated power and execution time for certain specific applications.

### 5.1 Reference Architecture

The methodology was applied to the architecture shown in *Figure 3*. It is a highly parametric SOC for digital camera applications developed under the Dalton Project at the University of California at Riverside [18]. The project is an open source one and comprises a parameterized simulation model of a system-on-a-chip composed of an MIPS R3000 processor core, instruction cache (I\$), data cache (D\$), memory, MIPS to instruction cache bus, MIPS to data cache bus, instruction/data cache to memory bus, bus bridge, peripheral bus, uart and codec.

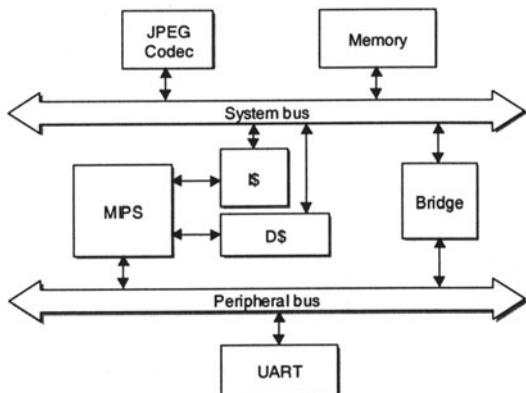


Figure 3. Reference architecture.

Each core is parametric and *Table 1* gives the free parameters and the set of admissible values. For each bus (data bus or address bus) it is possible to configure the number of lines and the encoding scheme to minimise the switching activity. The caches can be configured in size, line size and associativity. For the UART it is possible to define the transmission and reception buffer sizes, and for the JPEG Codec the pixel width can be varied. In all there are 26 separate parameters, giving a total of  $9.7 \times 10^{15}$  possible configurations.

Table 1. Free parameters and the set of admissible values for each core.

| Core            | Parameter          | Parameter space                                      | Config. space         |
|-----------------|--------------------|--|-----------------------|
| I & D Cache     | Size               | 128B, 256B, 512B, ..., 64KB                          | $10 \times 2$         |
|                 | Line               | 4B, 8B, 16B, ..., 128B                               | $6 \times 2$          |
|                 | associativity      | 1, 2, ..., 16B                                       | $5 \times 2$          |
| I\$ & D\$ → CPU | dbus/abus width    | 4, 8, ..., 32  | $4 \times 2 \times 2$ |
|                 | dbus/abus encoding | bin, gray, inv                                       | $3 \times 2 \times 2$ |
| \$ → MEM        | dbus/abus width    | 4, 8, ..., 32  | $4 \times 2$          |
|                 | dbus/abus encoding | bin, gray, inv                                       | $3 \times 2$          |
| Peripheral bus  | dbus/abus width    | 4, 8, 16, 32   | $4 \times 2$          |
|                 | dbus/abus encoding | bin, gray, inv                                       | $3 \times 2$          |
| UART            | TX/RX buf size     | 1,2,4,8,16   | $5 \times 2$          |
| Codec           | pixel width        | 10,12  | 2                     |
| Global          | volt vs. freq      | (1.5,33), (2.6,57), (3.3,72),<br>(4.0,88), (5.0,110) | 5                     |

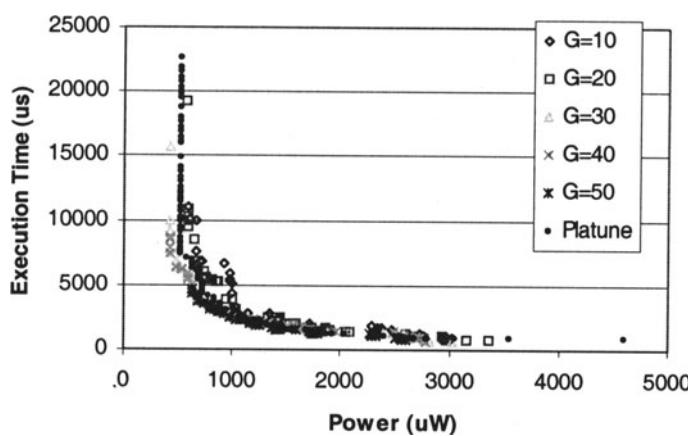
There are two versions of the system: both a synthesisable VHDL version and a high-level model written in C++. With this model it is possible to perform rapid simulations of the system when it is executing an applications, as well as estimating the execution time and power consumption by using the estimation model described in [7].

## 5.2 Experiments

The methodology was tested on three benchmarks typical of embedded applications. *Image* rotates an image by 90 degrees and converts it into a grayscale. The *Matrix* application performs a matrix invert operation on a large matrix. *Key* implements a complex chroma-key algorithm.

The GA-based approach proposed was compared with Platune using two evaluation indexes. The first measures the accuracy (or quality) of the solutions obtained by the two techniques/approaches. The second measures the efficiency of our approach by counting the number of configurations explored (i.e. the number of simulations required) to obtain the Pareto-optimal set.

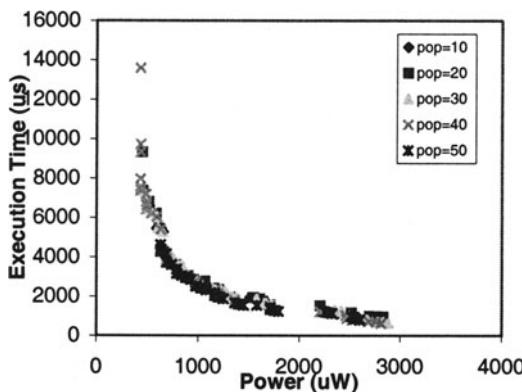
In all cases a crossover probability of 0.9 and a mutation probability of 0.01 were set. Various tests were carried out for each of the three benchmarks, varying the internal population and the number of generations. *Figure 4* compares the results obtained using Platune and those given by our approach, for various numbers of generations. As can be seen, after only 30 generations the solutions found dominate those found by Platune. The results given in the figure refer to the *Image* application, but from a qualitative point of view the same conclusions were reached with the other applications.



*Figure 4.* Comparison between the results obtained using Platune and those given by our approach, for various numbers of generations.

The improvements in terms of the quality of the solutions found do not vary significantly when the size of the internal population changes (see *Figure 5*). When, in fact, the population goes from 10 to 50 individuals after

50 generations the results obtained are almost equivalent. As an increase in the number of individuals making up the population means increasing the number of configurations visited, it is advisable to use small internal populations so as to enhance efficiency without an appreciable deterioration in the results.



*Figure 5.* The improvements in terms of the quality of the solutions found do not vary significantly when the size of the internal population changes.

*Table 2* summarises the results obtained for the three benchmarks, giving the number of simulations performed by Platune and the genetic approach for varying numbers of generations and internal population sizes. In general, the genetic approach allows an 85 to 99% saving in simulations as compared with Platune for all the benchmarks analysed.

*Table 2.* Results obtained for the three benchmarks, giving the number of simulations performed by Platune and the genetic approach for varying numbers of generations and internal population sizes.

| Platune |       |             | GA                       |      |      |      |      |  |
|---------|-------|-------------|--------------------------|------|------|------|------|--|
|         |       |             | Internal population size |      |      |      |      |  |
|         |       |             | 10                       | 20   | 30   | 40   | 50   |  |
| Image   | 36090 | generations | 155                      | 365  | 488  | 601  | 741  |  |
|         |       | 10          | 448                      | 878  | 1195 | 1331 | 1731 |  |
|         |       | 20          | 906                      | 1705 | 1934 | 2445 | 2665 |  |
|         |       | 30          | 1540                     | 2436 | 2904 | 3186 | 3710 |  |
|         |       | 40          | 2294                     | 3104 | 3535 | 4336 | 4581 |  |
|         |       |             | Internal population size |      |      |      |      |  |
|         |       |             | 10                       | 20   | 30   | 40   | 50   |  |
| Key     | 41410 | generations | 214                      | 310  | 436  | 585  | 707  |  |
|         |       | 10          | 515                      | 997  | 1305 | 1271 | 1456 |  |
|         |       | 20          | 1242                     | 1736 | 1851 | 2514 | 2925 |  |
|         |       | 30          | 1741                     | 2268 | 2601 | 3422 | 3751 |  |
|         |       | 40          | 2130                     | 3219 | 3684 | 4278 | 4587 |  |

|        |       | GA          |                          |      |      |      |      |
|--------|-------|-------------|--------------------------|------|------|------|------|
|        |       |             | Internal population size |      |      |      |      |
|        |       | generations | 10                       | 20   | 30   | 40   | 50   |
| Matrix | 34690 | 10          | 211                      | 358  | 471  | 575  | 703  |
|        |       | 20          | 575                      | 710  | 993  | 1344 | 1454 |
|        |       | 30          | 901                      | 1392 | 1969 | 2390 | 2564 |
|        |       | 40          | 1213                     | 2234 | 2638 | 3076 | 3650 |
|        |       | 50          | 1969                     | 3007 | 3788 | 4090 | 4683 |

## 6. CONCLUSIONS

In this paper we have proposed a GA-based methodology for multiobjective exploration of the range of configurations for a parameterized system. The methodology was applied to a highly parametric SOC for digital camera applications. The approach was evaluated in terms of both accuracy and the CPU time required to explore the range of configurations and compared with the approach used in Platune [2]. The results obtained show that unlike others this approach solves the problem by successive refinement: the more the algorithm is made to evolve, the closer the Pareto-set found is to the Pareto-optimal set. This would appear to be a very useful feature, as the user can choose the accuracy/CPU time trade-off. In terms of CPU time to search for the trade-off front, the approach requires on average 90% fewer simulations than the Platune approach.

Future developments will address definition of a mixed genetic/heuristic approach using evolutionary techniques to achieve global optimisation together with efficient local optimisation techniques.

## 7. REFERENCES

- [1] T. D. Givargis and F. Vahid. Parametrized system design. In 8th International Workshop on Hardware/Software Codesign, 2000.
- [2] The UCR Dalton Project IP-Based Embedded System Design. <http://www.cs.ucr.edu/~dalton/>.
- [3] F. Vahid and T. Givargis. The case for a configure-and-execute paradigm. In International Workshop on Hardware/Software Codesign (CODES), pages 59--63, May 1999.
- [4] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria. A design framework to efficiently explore energy-delay tradeoffs. 9th. International Symposium on Hardware/Software Co-Design, pages 260--265, Copenhagen, Denmark, Apr. 25--27 2001.
- [5] G. Ascia, V. Catania, and M. Palesi. Parameterized system design based on genetic algorithms. 9th. International Symposium on Hardware/Software Co-Design, pages 177--182, Copenhagen, Denmark, Apr. 25--27 2001.

- [6] T. D. Givargis, J. Henkel, and F. Vahid. Interface and cache power exploration for core-based embedded system design. In International Conference on Computer-Aided Design (ICCAD), pages 270--273, Nov. 1999.
- [7] T. D. Givargis, F. Vahid, and J. Henkel. A hybrid approach for core-based system-level power modeling. In Asia and South Pacific Design Automation Conference, 2000.
- [8] T. D. Givargis, F. Vahid, and J. Henkel. Trace-driven system-level power evaluation of system-on-a-chip peripheral cores. In Asia South-Pacific Design Automation Conference (ASP-DAC), Jan. 2001.
- [9] C. A. C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269--308, Aug. 1999.
- [10] F. Vahid and T. Givargis. Platform tuning for embedded systems design. *IEEE Computer*, 34(3):112--114, Mar. 2001.
- [11] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1--16, 1995.
- [12] D. A. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125--147, 2000.
- [13] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In L. Erlbaum, editor, *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93--100, 1985.
- [14] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 4(3):257--271, Nov. 1999.
- [15] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173--195, 2000.
- [16] M. Wall. GALib: A C++ Library of Genetic Algorithm Components. Mechanical Engineering Department, Massachusetts Institute of Technology, Aug. 1996.
- [17] C. A. C. Coello. Treating constraints as objectives for single-objective evolutionary optimization. Technical report, Laboratorio Nacional de Informatica Avanzada, Rebsamen 80, Xalapa, Veracruz 91090, Mexico, 2000.
- [18] The UCR Dalton Project IP-Based Embedded System Design.  
<http://www.cs.ucr.edu/~dalton/>.

# **Design of a Branch-Based Carry-Select Adder IP Portable in 0.25 $\mu\text{m}$ Bulk and Silicon-On-Insulator CMOS Technologies**

Amaury Nève and Denis Flandre

*Laboratoire de Microélectronique, Université Catholique de Louvain, Place du Levant 3,  
B-1348 Louvain-la-Neuve, Belgium*

**Abstract:** By reducing the parasitic node capacitances, the Branch-Based Logic design style can increase the performances of digital circuits. In order to benefit from the full potential of the design style and to be able to port it to different technologies, it is important to take into account the specific features of each technology. We investigate the case of three advanced 0.25  $\mu\text{m}$  CMOS technologies: bulk, Partially-Depleted SOI and Fully-Depleted SOI. The design of a 16-bit carry-select Branch-Based adder IP is discussed. The Branch-Based adder shows lower power consumption compared to an implementation with conventional CMOS logic gates.

**Key words:** Digital design, SOI Technology, Low Voltage Low Power design

## **1. INTRODUCTION**

As the integration density and operating frequencies of digital IP's steadily rise, the dynamic power dissipation becomes a serious concern. Most of the usual techniques used to decrease the power consumption of logic cells imply major disadvantages. Decreasing the supply voltage  $V_{dd}$  leads to higher delays or requires a lower threshold voltage  $V_{TH}$  in order to maintain the speed performances. This in turn increases the off-state leakage currents  $I_{off}$  through the devices, and thus leads to higher static power dissipation. Decreasing the operating frequency of the IP core is not a good solution if high-speed performance goals must be achieved. There is a need for low-power techniques that do not result in excessive speed degradation.

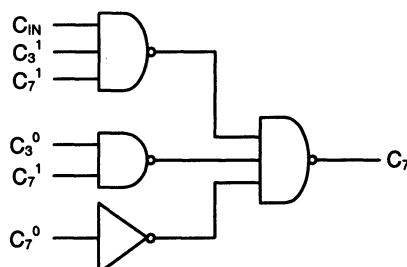
This paper describes the comparison between Branch-Based Logic (BBL) and conventional static CMOS cells, for three advanced  $0.25\text{ }\mu\text{m}$  CMOS processes: bulk, Partially-Depleted Silicon-On-Insulator (PD SOI) and Fully-Depleted (FD) SOI. We demonstrate that BBL circuits have lower dynamic power dissipation than conventional CMOS with a negligible speed loss.

## 2. THE BRANCH-BASED LOGIC DESIGN STYLE

In Branch-Based Logic (BBL), a function is implemented with branches, instead of standard logic gates or pass-gates [1]. Each branch is made of a chain of NMOS or PMOS devices between supply and the output node. Basically, the design methodology consists in writing the logic equation as a sum of products. Each product represents a branch in the cell. It can be shown that any logic function can be written as a sum of products.

Figure 1 shows a Carry-Select (CS) circuit implemented in conventional CMOS logic, with NAND and NOR gates. Figure 2 shows the same function implemented in BBL. All the branches are connected to the output node, without any other connection between the branches.

It has been reported that the main advantage of BBL is the reduction of the parasitic capacitances [2], resulting from two effects. Firstly, the node capacitances in one logic cell are reduced, thanks to the absence of wiring between two branches. Secondly, as some cells can be designed in one delay stage instead of a cascade of two stages, the number of internal nodes in the cell is reduced. All this results in a lower parasitic capacitance to be switched in one clock period and contributes to better power and speed performances.



*Figure 1.* Conventional CMOS implementation of CS-C1.

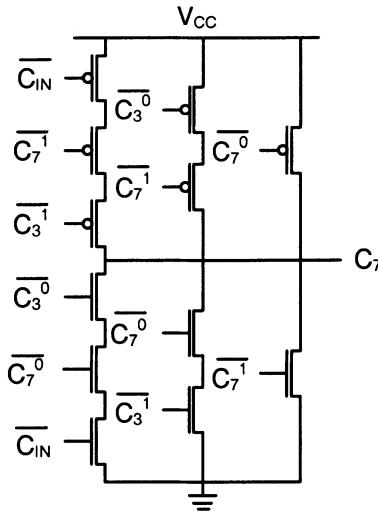


Figure 2. BBL implementation of CS-C1.

### 3. DESIGN OF THE 16-BIT ADDER

For comparison purposes, we designed two 16-bit carry-select adders [3], the first with conventional CMOS logic gates, the second using BBL cells. Thanks to the independent carry network, the carry-select architecture is very fast. Our adder is composed of four parts, each one featuring two 4-bit adders, the first with the carry-in at “0”, the second with the carry-in at “1”. A multiplexer, whose control signal is generated by the carry network, selects the right output. The great advantage of this structure is that the carry signals coming from the 4-bit adders, referenced as  $C_{\#}^0$  and  $C_{\#}^1$  (with  $\# = 3, 7, 11$  or  $15$ ) on figure 4, are computed in parallel and arrive approximately at the same time. They are then fed into the Carry-Select (CS) boxes, which compute the control signals for the multiplexers and the final carry-out. No time is lost waiting for a carry signal to ripple through all the adder cells. The logic equations of the CS boxes are represented in figure 3. The 16-bit adder architecture is depicted in figure 4.

$$\begin{aligned}
 C_3 &= C_3^1 C_{IN} + C_3^0 \\
 C_7 &= C_7^1 C_3^1 C_{IN} + C_7^1 C_3^0 + C_7^0 \\
 C_{11} &= C_{11}^1 C_7^1 C_3^1 C_{IN} + C_{11}^1 C_7^1 C_3^0 + C_{11}^1 C_7^0 + C_{11}^0 \\
 C_{15} &= C_{15}^1 C_{11}^1 C_7^1 C_3^1 C_{IN} + C_{15}^1 C_{11}^1 C_7^1 C_3^0 \\
 &\quad + C_{15}^1 C_{11}^1 C_7^0 + C_{15}^1 C_{11}^0 + C_{15}^0
 \end{aligned}$$

Figure 3. Logic equations of the carry-select boxes.

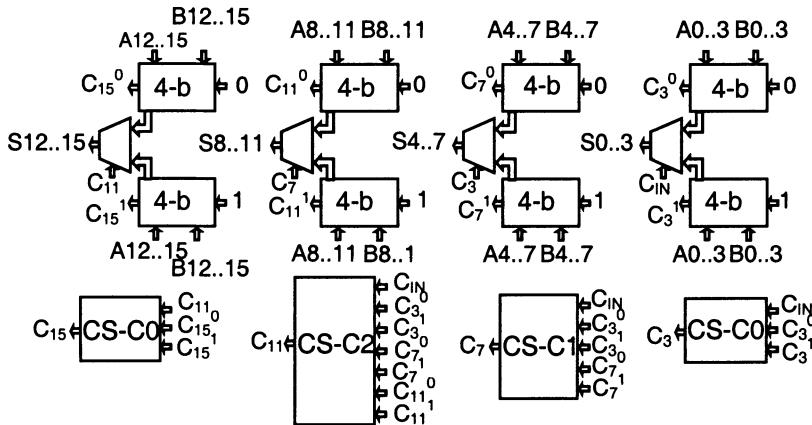


Figure 4. Architecture of the 16-bit carry-select adder.

CS-C0 and CS-C1 are implemented in one stage in BBL (figures 2 and 5), while the conventional CMOS implementation requires two successive stages. CS-C2 has two stages linked by an inverter (figure 6), while the CMOS equivalent is composed of four successive stages.

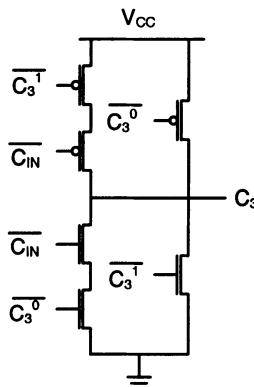


Figure 5. BBL Implementation of CS-C0

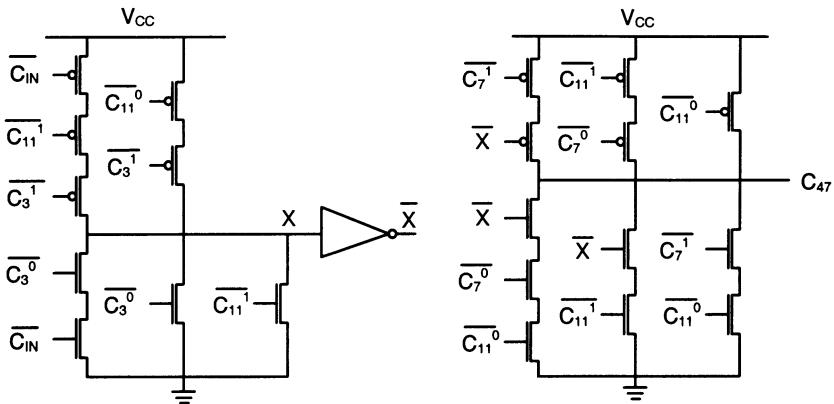


Figure 6. BBL Implementation of CS-C2.

The 4-bit adders also use a carry-select architecture, with 1-bit adders and a carry-network featuring a similar structure as at the 16-bit level. The BBL 1-bit half adders have a very compact implementation in one stage only, while the CMOS half-adder requires two successive stages.

#### 4. DESIGN IN THREE $0.25 \mu\text{m}$ CMOS TECHNOLOGIES

##### 4.1 Figures of merit

The simulations of the previously described cells were carried out using ELDO, a SPICE-like circuit simulator, using the parameters of three different CMOS processes: BULK silicon  $0.25 \mu\text{m}$ , Partially-Depleted (PD) SOI  $0.25 \mu\text{m}$  and Fully-Depleted (FD) SOI  $0.25 \mu\text{m}$ . In order to understand our results, figures of merit for each technology are proposed in tables 1 and 2. Except for the threshold voltage, all the other values are normalized to bulk-Si data, e.g.  $I_{DSAT} = 1$  for bulk NMOS. From the threshold voltage data, it can be observed that the bulk process has been optimised for speed, on the contrary of the two SOI processes, whose parameters are rather conservative. But thanks to the lower body effect and the better sub-threshold slope, the FD SOI transistors show a high current in the ON-state, and a low leakage current in the OFF-state. The rather high relative value of the OFF-current for SOI PD NMOS devices is associated with the parasitic floating-body effect at high drain voltages. The figure of merit we use for the capacitance combines the gate and the source/drain capacitances. The values are given

for 25°C as well as for 80°C, since the latter is a typical operating temperature for high-performance circuits.

*Table 1.* Threshold voltage ( $V_{TH}$ ) values and figures of merit for the NMOS devices, normalized to bulk (data corresponding to bulk = 1), for the three considered technologies at 25°C and at 80°C (value between brackets): 0.25  $\mu\text{m}$  bulk Si, 0.25  $\mu\text{m}$  PD SOI and 0.25  $\mu\text{m}$  FD SOI. Nominal channel length is 0.25  $\mu\text{m}$ .  $V_{TH}$  = Threshold voltage;  $ID_{sat}$  = Saturation Drain-to-Source Current;  $I_{off}$  = OFF-state leakage current when the gate voltage  $VG=0$  V;  $C_{ox}$  = gate oxide capacitance per unit area;  $C_j$  = Source/Drain-to-substrate capacitance per unit area; LS,D=Source/Drain contact region extensions.

| NMOS    | $V_{TH}$ | $ID_{sat}@VD=VG=1.5V$ | $I_{off}@VD=1.5V$ | $2/3.C_{ox}.L+C_j.L_{s,p}$ |
|---------|----------|-----------------------|-------------------|----------------------------|
| Bulk-Si | 0.57 V   | 1 (1)                 | 1 (1)             | 1                          |
| PD SOI  | 0.55 V   | 1.02 (0.95)           | 5.48 (2.27)       | 0.77                       |
| FD SOI  | 0.50 V   | 1.21(1.09)            | 0.04 (0.03)       | 0.77                       |

*Table 2.* Threshold voltage values and figures of merit for the PMOS devices, normalized to bulk (data corresponding to bulk = 1), for the three considered technologies at 25°C and at 80°C (value between brackets): 0.25  $\mu\text{m}$  bulk Si, 0.25  $\mu\text{m}$  PD SOI and 0.25  $\mu\text{m}$  FD SOI.

| PMOS    | $V_{TH}$ | $ID_{sat}@VD=VG=1.5V$ | $I_{off}@VD=1.5V$ | $2/3.C_{ox}.L+C_j.L_{s,p}$ |
|---------|----------|-----------------------|-------------------|----------------------------|
| Bulk-Si | -0.61 V  | 1 (1)                 | 1 (1)             | 1                          |
| PD SOI  | -0.60 V  | 0.72 (0.65)           | 0.02 (0.02)       | 0.85                       |
| FD SOI  | -0.52 V  | 0.97 (0.83)           | 10E-4 (10E-4)     | 0.85                       |

## 4.2 Methodology

Each cell has been optimised for speed in the three technologies. For the optimisation process, each cell is loaded with two CMOS inverters, which represents a charge similar to what the cell will have in the complete adder. The nominal gate length is chosen at the minimum allowed drawing dimension, i.e.  $L = 0.25 \mu\text{m}$ . In order to determine the W/L-ratio of the transistors, the input pattern that activates the top transistor of the critical branch in one particular cell is applied at the input. Inside one branch, the W/L-ratio is increased when going from the device closest to the output to the device closest to the supply rails, with a ratio of 1.5 between each successive transistor in the stack. The W/L-ratio of the other branches is further tuned to lower the capacitance of the output node, to which all the branches are connected.

## 4.3 Bulk 0.25 $\mu\text{m}$

BBL design implies the use of stacks of series transistors which are particularly affected by the large substrate effect of bulk CMOS. The impact on the delay can be estimated by considering the difference between the case where the substrate of the transistors is connected to the supply rails and the case where the substrate is connected to the source. Except for the half-adder

(referenced as ha\_cin0), the substrate effect causes more delay increase for the cells implemented in BBL, due to the activation of a stack with two or three PMOS transistors in the worst case (Table 3).

*Table 3.* Percentage delay increase due to the bulk CMOS substrate effect in BBL and in conventional CMOS cells. To cancel out the effect of the substrate-source potential, we connected source and substrate for each transistor in the circuit. Vdd=1.5V; T=80 °C.

|         | Conventional CMOS | BBL   |
|---------|-------------------|-------|
| CS-C0   | 6.6%              | 12.9% |
| CS-C1   | 7.0%              | 24.5% |
| CS-C2   | 10.1%             | 12.4% |
| Ha_cin0 | 8.8%              | 5.0%  |

*Table 4.* Influence of the threshold voltage non-uniformity on the delay of the BBL and conventional CMOS cells in bulk-Si. The numbers represent the relative variation between worst case and best case. Vdd=1.5 V; T=80°C ;  $\sigma_{V_{TH,NMOS}}=10$  mV;  $\sigma_{V_{TH,PMOS}}=15$  mV.

|         | Conventional CMOS | BBL  |
|---------|-------------------|------|
| CS-C0   | 0.11              | 0.09 |
| CS-C1   | 0.13              | 0.05 |
| CS-C2   | 0.12              | 0.10 |
| Ha_cin0 | 0.11              | 0.09 |

In order to evaluate the impact of the threshold voltage non-uniformity, we considered a maximum threshold voltage variation  $\Delta V_{TH} = \pm 3\sigma_{V_{TH}}$  around the nominal  $V_{TH}$  value. In bulk-Si CMOS, we obtain a delay difference of about 10 % between the best case and the worst case, the variation being slightly lower in BBL (Table 4). This is intuitively interpreted as follows. In BBL, one branch is activated in the worst case, while in CMOS, at least two cascaded sub-cells must switch, each one contributing to the increase of the impact of  $V_{TH}$  variations on the delay.

Comparing the absolute delay values, including the body effect, the two small cells (ha\_cin0 and CS-C0) are faster in BBL, whereas CS-C1 and CS-C2 are slower in BBL (Table 5). For these two cells, this can be associated with the stacks of three PMOS transistors that are activated in the worst case. In the conventional CMOS cells, a stack of three NMOS devices is activated in the worst case. Moreover, the high number of branches connected at the output nodes in BBL result in a higher parasitic capacitance at this node, even if the total cell capacitance is lower.

All the BBL cells consume between 10 % and 58 % less dynamic power than the conventional CMOS cells (Table 6). This is associated with a reduction of the short-circuit current and with the lower internal node capacitances.

**Table 5.** Speed increase for the BBL vs. the conventional CMOS implementation of the basic building blocks of the 16-bit adder for BULK, SOI-PD and SOI-FD technologies. Vdd=1.5 V; T=80°C.

|         | Bulk   | SOI-PD | SOI-FD |
|---------|--------|--------|--------|
| CS-C0   | 24.3%  | 24.4%  | 25.6%  |
| CS-C1   | -20.2% | -8.7%  | -12.6% |
| CS-C2   | -11.9% | -12.6% | -16.4% |
| Ha-cin0 | 29.9%  | 31.2%  | 32.5%  |

**Table 6.** Dynamic power reduction for the BBL vs. the conventional CMOS implementation. Vdd=1.5 V; T=80°C.

|         | Bulk  | SOI-PD | SOI-FD |
|---------|-------|--------|--------|
| CS-C0   | 58.5% | 22.7%  | 21.4%  |
| CS-C1   | 32.3% | -4.8%  | 28.9%  |
| CS-C2   | 27.2% | -16.8% | 7.7%   |
| Ha-cin0 | 10.6% | 32.0%  | 16.3%  |

#### 4.4 Partially-Depleted SOI 0.25 μm

The floating-body in PD SOI transistors is responsible for parasitic behaviors such as kink and hysteresis effects [4]. The latter represents the dependence of the threshold voltage on the initial conditions or on the history of the body charge. Table 7 presents the delay variations resulting from initial input conditions which result in different body potentials. Except for CS-C0, the delay variation is slightly higher in BBL due to the higher stacks, but this effect remains small, even when compared to the impact of  $V_{TH}$  variations.

The floating body is also known to result in the discharge of internal nodes due to the parasitic bipolar effect. The use of BBL, which is a static design style, minimizes the risk of erroneous states, on the contrary to dynamic logic [5].

Concerning the delay, in PD SOI, only the small cells perform better in BBL than in conventional CMOS logic, as in bulk (Table 5). Concerning power, on the contrary to bulk, the PD SOI large BBL cells consume more than the conventional CMOS equivalents due to floating body effects (Table 6).

**Table 7.** Delay variation due to different initial conditions in BBL and conventional CMOS cells in PD SOI. The input signal is set at an initial value which is “Low” or “High”. Vdd=1.5 V; T=80°C.

|         | Conventional CMOS | BBL  |
|---------|-------------------|------|
| CS-C0   | 2.4%              | 0.8% |
| CS-C1   | 3.5%              | 5.1% |
| CS-C2   | 0.1%              | 5.0% |
| Ha_cin0 | 1.6%              | 6.7% |

## 4.5 Fully-Depleted SOI 0.25 μm

FD SOI transistors show better on/off performances and much lesser floating body effects than PD SOI devices thanks to the complete depletion of the body. Moreover, the rise of the threshold voltage with increasing source-to-substrate voltage is lower in FD SOI than in bulk. Figure 7 shows the evolution of the delay when adding transistors to the stack. The FD SOI technology is thus particularly appropriate when stacks of transistors are used, such as in BBL. The lower substrate effect enables us to use one more transistor in the stack for the same delay as in bulk-Si.

Despite the superior performances, the concern about higher threshold voltage non-uniformity in FD SOI, when compared to bulk and PD SOI, has delayed the acceptance of FD SOI in the semiconductor industry. When including a  $\Delta V_{TH} = \pm 3\sigma_{VTH}$  variation around the nominal  $V_{TH}$  value in FD SOI, a difference of about 20 % is obtained for the delay between the best case and the worst case for both design styles (Table 8), which is indeed larger than the 10 % we obtained in bulk. By a better control of the uniformity of the silicon film thickness and related fabrication steps, this effect can be reduced as shown in [6]. Moreover, FD SOI cells do not show additional delay variations due to the floating-body effects as in PD SOI and remain much faster than bulk cells, even in the worst case.

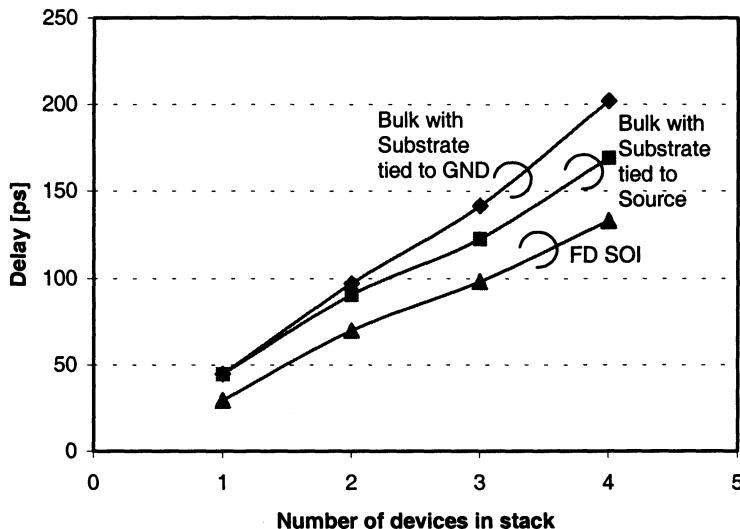
The two small cells are faster in BBL than in conventional CMOS logic for FD SOI (Table 5). As in bulk, the dynamic power consumption is reduced when using the Branch-Based design style (Table 6).

*Table 8.* Influence of the threshold voltage non-uniformity on the delay of the BBL and conventional CMOS cells in FD SOI. The numbers represent the relative variation between worst case and best case.  $Vdd=1.5$  V,  $T=80^\circ\text{C}$ .  $\sigma_{VTH,NMOS}=15$  mV,  $\sigma_{VTH,PMOS}=30$  mV.

|         | Conventional CMOS | BBL  |
|---------|-------------------|------|
| CS-C0   | 0.19              | 0.23 |
| CS-C1   | 0.22              | 0.17 |
| CS-C2   | 0.20              | 0.18 |
| Ha-cin0 | 0.15              | 0.18 |

## 5. RESULTS FOR THE COMPLETE ADDER

The logic cells described above were used to implement two 16-bit carry-select adders, the first with conventional CMOS logic gates, the second with BBL. Figure 8 compares the power-delay products of the two versions of the adder in each technology. The data points representing the BBL adder are all shifted to the left compared to the points related to the adder in conventional



*Figure 7.* Delay in function of the number of devices in a stack of NMOS transistors with  $W/L=(2.5/0.25)$  for bulk-Si and FD SOI. The increase of the delay is the smallest for FD SOI thanks to the lower bulk substrate effect.  $V_{dd}=1.5$  V;  $T=80^{\circ}\text{C}$ .

CMOS logic, which means that the former consumes less dynamic power for a similar delay. Indeed, in bulk-Si and in FD SOI, the dynamic power is reduced by resp. 13 % and 16 % when comparing the BBL adder to the conventional CMOS adder. The delay is increased by resp. 4 % and 2 %, which is negligible. In PD SOI, the power reduction is the highest, reaching 36 %, with a delay increase less than 2 %.

From the point of view of the static power, the branch-based design style is also beneficial. Thanks to the lower number of leakage paths between  $V_{dd}$  and ground, the BBL 16-bit adder achieves a reduction of static power consumption of resp. 23 %, 18 % and 34 % in bulk-Si, in PD-SOI and in FD-SOI when compared to the conventional CMOS design.

If it is kept in mind that the two SOI processes are experimental processes, not optimised for speed, some trends can be identified in the comparison with bulk (Table 9). Even if the delay is larger in PD SOI than in bulk, the PD SOI 16-bit adder is able to achieve a 20 % power-delay product improvement over bulk. When moving the design to the FD SOI process, a reduction of 20 % delay and 35% dynamic power is obtained at 1.5 V, resulting in a power-delay improvement of nearly 50 %. When lowering the supply voltage down to 1 V, the dynamic power consumption is reduced by 40 % and the delay is still 20% lower than in bulk.

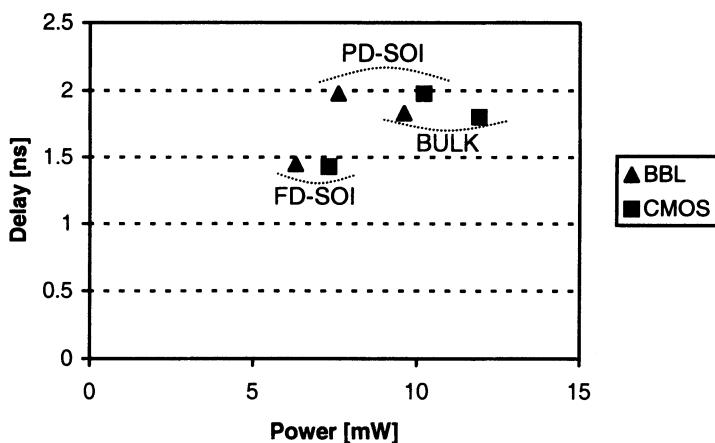
The static power consumption is also significantly lower in FD-SOI than in PD-SOI and bulk, thanks to the better sub-threshold slope, which allows a

reduction of the threshold voltage without increasing the OFF-state leakage current.

The total active area is very similar for the adder in the three technologies. The bad performances of the SOI PMOS devices in our study lead to an optimization point where the widths of the PMOS devices in the critical branches are larger than for the equivalent bulk-Si devices. This is compensated by the use of smaller widths for the SOI NMOS devices compared to bulk-Si. But thanks to the better layout efficiency of SOI, NMOS and PMOS devices can be abutted and easily interconnected, thus saving total die area. On another hand, the use of larger PMOS sizes in FD SOI would improve the matching between devices and reduce the delay variations associated with the threshold voltage non-uniformities.

**Table 9.** Delay, power consumption, power-delay product and active area of the 16-bit BBL adder for the three technologies.  $V_{dd} = 1.5$  V;  $T = 80^\circ\text{C}$ ;  $f = 200$  MHz.

|                     | BULK                 | PD-SOI               | FD-SOI               |
|---------------------|----------------------|----------------------|----------------------|
| Delay               | 1.833 ns             | 1.978 ns             | 1.455 ns             |
| Static Power        | 1.747 $\mu\text{W}$  | 656 nW               | 6.527 nW             |
| Dynamic Power       | 9.7 mW               | 7.6 mW               | 6.3 mW               |
| Power-Delay Product | 17.8 pJ              | 15.0 pJ              | 9.2 pJ               |
| Active Area         | 3428 $\mu\text{m}^2$ | 3410 $\mu\text{m}^2$ | 3408 $\mu\text{m}^2$ |



**Figure 8.** Delay vs. dynamic power for the BBL and the conventional CMOS 16-bit carry-select adder.  $V_{dd} = 1.5$  V;  $F = 200$  MHz;  $T = 80^\circ\text{C}$ .

## 6. CONCLUSION

Our results investigate a methodology to reliably port digital IP cores from conventional CMOS to BBL design style, and from bulk to SOI processes. Specifically we studied the potential of Branch-Based Logic design for high-performance IP cores. The analysis was done using the parameters of three advanced  $0.25\text{ }\mu\text{m}$  CMOS processes: bulk, PD SOI and FD SOI. Each of these technologies has its specific features which have to be taken into account during the design: the influence of the body effect on the stacks in bulk, the floating-body effects in PD-SOI and the spread of the threshold voltage in FD-SOI. A comparison between Branch-Based design and conventional CMOS logic design reveals similar trends for the different processes: the delay is lower in small BBL cells only, but all the BBL cells have less dynamic power consumption in bulk and FD SOI. The complete 16-bit Branch-Based carry-select adder shows a lower dynamic power dissipation for a similar delay compared to the conventional CMOS version. Moreover, we showed that the FD-SOI performances are already sufficiently better than bulk-Si in order to maintain a comfortable power-delay product improvement, even though much larger  $V_{TH}$  non-uniformities than in bulk would have to be accommodated. Finally, delay variations with  $V_{TH}$  do not appear higher in FD SOI than the sum of the delay variations with  $V_{TH}$  and those associated with the floating-body effects in PD SOI.

## 7. REFERENCES

- [1] Masgonty J.M., Arm C. and Piguet C., "Technology- and Power Supply-Independent Cell Library", in proceedings of the IEEE Custom Integrated Circuits Conference, 1991, pp. 25.5.1-25.5.4.
- [2] Masgonty J.-M., Mosch P. and Piguet C., "Branch-Based Digital Cell Libraries", in proceedings of EURO ASIC 1991, pp. 27-31.
- [3] Hwang K., "Computer Arithmetic: Principles, Architecture and Design", Wiley, New York, 1979.
- [4] Fossum J.G., "Designing reliable SOI CMOS Circuits with Floating-Body Effects", in Proceedings of the European Solid-State Device Research Conference 1998, pp. 34-41.
- [5] Canada M. et al., "A 580 MHz RISC Microprocessor in SOI", in proceedings of the IEEE International Solid-State Circuit Conference, 1999, pp. 430-431.
- [6] Vanmackelberg M. et al., "0.25  $\mu\text{m}$  fully-depleted SOI MOSFET's for RF mixed analog-digital circuits, including a comparison with partially-depleted devices with relation to high frequency noise parameters", in Solid-State Electronics, vol. 46, n°3, 2002, pp. 379-386.

# A Standardized Co-simulation Backbone

Braulio Adriano de Mello<sup>1,2</sup> and Flávio Rech Wagner<sup>1</sup>

<sup>1</sup>*Instituto de Informática – UFRGS – Porto Alegre – Brazil;* <sup>2</sup>*URI – Brazil*

**Abstract:** In the field of co-simulation, the construction of a bridge between different simulators and the solution of problems like synchronization and data translation are some of the main challenges. This paper discusses the advantages of the HLA (High Level Architecture) standard to solve these problems and presents a generic architecture to support environments for geographically distributed co-simulation, called Distributed Co-simulation Backbone (DCB), which is based on the HLA. This architecture is very flexible and does not enforce code modifications to the simulators to be integrated into the environment.

**Key words:** distributed co-simulation, simulation backbone, cooperation, HLA

## 1. INTRODUCTION

Co-simulation is used to make experiments and get information on the behavior of heterogeneous systems aiming at the validation of their design or at the evaluation of performance. Heterogeneous systems are characterized by a combination of hardware and software parts or by descriptions in different languages and/or at different abstraction levels [1]. In order to validate the design of embedded electronic systems, research in co-simulation mainly emphasizes the cooperative simulation of hardware and software parts.

Therefore, one of the major challenges in co-simulation is the construction of a mechanism for a consistent cooperative simulation involving those parts. This challenge has been increased by the evolution of technologies for communication and distributed processing [2].

Current techniques, environments, and tools for co-simulation have shown that one of the main bottlenecks is the communication interface. The

large variety of technologies and their continuous evolution make it difficult to conceive an adaptive (or generic) mechanism, which promotes the cooperation between heterogeneous simulators without imposing restrictions regarding their data formats or behavior.

This paper presents an architecture for a distributed co-simulation backbone, called DCB, which is based on the High Level Architecture (HLA) [3]. HLA has its roots on defense programs and has been recognized as a standard by the IEEE in 2000. It proposes rules and mechanisms for the interoperability of distributed, heterogeneous simulators. In particular, the Run Time Infrastructure, which is part of the standard, offers facilities that give an important contribution for the construction of a generic mechanism supporting distributed co-simulation environments.

The DCB architecture presents three definite advantages over other co-simulation environments: it is based on a public standard; it is far more flexible than current approaches, allowing an easier definition of particular environments; and it allows the integration of existing simulators without imposing modifications on their internal structures.

This paper is organized as follows. An overview of co-simulation is presented in Section 2. Section 3 discusses related work and introduces the contribution of DCB. The HLA standard, and its Run Time Infrastructure in particular, are introduced in Section 4. Section 5 thus presents DCB, which is a generic architecture to support the cooperation between heterogeneous simulators in distributed environments. Finally, Section 6 gives final remarks and discusses future work.

## **2. AN OVERVIEW OF CO-SIMULATION**

The complexity of current embedded systems have raised so much that their design cannot be performed with a single design language nor at a single abstraction level. These systems usually aggregate hardware and software cooperating parts. Their designs ask for models combining descriptions at different abstraction levels, and multiple specification languages are needed for representing those parts and/or abstraction levels. Moreover, these parts are usually developed and validated by means of separate design processes. For this, the validation of the whole system must take communication and cooperation aspects into account, becoming extremely complex because of the system heterogeneity.

Aimed at the validation and performance evaluation of heterogeneous systems, the co-simulation approach has been the target of an increasing number of research groups. Its fundamental principle is the cooperative execution of different simulators [1]. Each simulator is responsible for a

different system part. The simulators may be executed in a single machine or in multiple machines (in LANs or WANs) [4]. Research is lately emphasizing co-simulation in wide area networks, due to potential benefits such as the management of intellectual property and cooperative design. In this scenario, each simulator may send and receive data through a co-simulation interface that must handle communication, synchronization, and data format conversions. The literature presents the construction of this bridge between heterogeneous simulators as one of the main challenges of co-simulation [5].

### 3. RELATED WORK

WESE (Web-based Environment for Systems Engineering) [6] is a collaborative and distributed environment for systems engineering. It supports distributed simulation with the cooperative execution of remotely located components. Components are stored in repositories called *factories* and are specified by the SSL language, which is specialized for web-based design. The environment is also suited for handling IP components.

IPCHINOOK [7] is a component-based design tool for distributed, embedded systems. Its main feature is a set of design abstractions to raise the level at which the designer interacts with the design environment. Communication and synchronization details are synthesized by the tool. The designer, however, must follow some modeling and synthesis rules.

The JavaCAD [8] tool supports the web-based reuse of IP components, guaranteeing the confidentiality of information for suppliers and clients. JavaCAD is implemented as a simulation backplane that supports the evaluation of IPs during the design process. For this, the user may instantiate IP components from multiple remote suppliers and simulate them transparently with proprietary blocks. For performing a simulation, the user must initially specify his/her design through a JavaCAD client, thus somehow restricting reuse flexibility.

Multilanguage approaches are used for the design of systems consisting of heterogeneous components, by offering environments that support multiple (but limited) languages. An example is the MCI tool [1]. Starting from an abstract description of a communication interface among subsystems, it automatically generates a specialized co-simulation environment. MCI allows a geographically distributed co-simulation of subsystems.

The Pia co-simulation tools [4] supply a specification mechanism that allows the interconnection of nodes through a geographically distributed network. Nodes are connected by sockets to the tool, so that other simulators, compilers, or even physical devices may be linked together.

A co-simulation approach based on the Ptolemy project [9] is presented in [10]. Following this approach, software and hardware simulators communicate through a master process (a backplane, in fact). This process manages communication between simulators, offering a standard interface that is based on a set of rules.

In general, existing approaches and tools are adequate for solving a set of predefined problems. Frequently, however, unexpected problems must be handled, regarding the integration of new tools. In this case, solutions that are proprietary or that are based on a given set of restrictions may impose an excessive burden, in terms of redesign of tools. Examples of restrictive conditions may be found in all approaches: particular techniques for model specification and synthesis, as in IPCHINOOK; use of predefined functions or structures, as in JavaCAD [8] and Pia sockets; limitations on the number or types of languages, as in multilanguage tools; and proprietary communication interfaces, as in some solutions based on backplanes.

The fundamental idea behind the architecture of DCB – Distributed Co-simulation Backbone is to remove the restrictions that are imposed upon independent simulators, considering aspects of interface, cooperation, and synchronization, thus adding great flexibility to co-simulation environments. Preserving the integrity of each simulator, DCB more easily handles unexpected situations, without loosing fidelity and precision. Besides that, the explicit use of non-proprietary, distributed solutions has not been addressed as a fundamental issue of other co-simulation approaches.

## 4. HLA AND CO-SIMULATION

### 4.1 Introducing HLA

HLA - High Level Architecture was initially conceived within the community of “distributed interactive simulation”, considering special needs that were observed in military training [11]. Its development process involved government, academia, and industry, and its main concepts have been defined in 1995.

HLA offers a common architecture for the cooperative and distributed execution of individual simulations, also on WANs, as well as a structure for reusing simulations in existing or new applications. In order to enhance reuse and interoperability, HLA proposes the reduction of design restrictions and the implementation of fairly independent simulations, by using the object-oriented paradigm and introducing the idea of a simulation *federation* [3]. This concept introduces a high flexibility in the modeling process and eases the definition of interfaces, functionality, and cooperation [11]. HLA

is not a standard with a fixed set of rules or restrictions. Instead, it may be considered as a meta-standard, which defines meta-rules for building particular distributed simulation environments. HLA is specified by three main parts:

- the interface specification;
- the Object Model Template (OMT) [3], which defines a common and structured format for *federates* (the individual simulators) and *federations* (environments built from cooperating federates); and
- a set of rules for the definition and management of the behavior of federation and federates.

The interface specification defines the communication mechanism between the federates and includes the RTI – Run Time Infrastructure. The task of RTI is to offer services for communication and cooperation between federates, since they cannot communicate directly with each other. For establishing communication between federates, both RTI and the federates must offer communication methods, and the *ambassador's* paradigm may be used [12], as will be explained later on. A federate may represent a simulation model in a computer, a dedicated physical simulator, or an interface to a physical object or human.

The union of the federates with RTI is defined by a Federation Object Model (FOM). In turn, for each federate a Simulation Object Model (SOM) is defined. The FOM specifies a contract between federates, regarding the types of objects that will be visible among them and the interactions they will perform.

Although there is an expressive, inherent flexibility in HLA, it still imposes some restrictions:

- federates must implement methods for communication with RTI;
- federates and federation must follow a given set of rules (which may be, however, specialized for each federation);
- RTI may not accept a given communication protocol.

These restrictions apply mainly when new models or simulators have to be added to a certain federation and they have not been developed considering this federation. This problem is also found in co-simulation, where the heterogeneity of simulators must be handled in order that interoperability is achieved. These restrictive aspects in the construction of co-simulation environments have motivated research looking for more flexible approaches [12]. Nevertheless, the HLA standard has three definite advantages as a basis for co-simulation environments:

- it is a *standard*, so that we may see in near future commercial simulators that follow its guidelines and have an intrinsic interoperability;
- it is far more flexible than current approaches, allowing an easier definition of particular federations; and

- it allows the integration of existing simulators without imposing severe modifications on their internal structures, provided that they support some basic features.

## 4.2 Supporting co-simulation with the RTI

RTI specifies services to be offered to the federates, as well as services to be offered by the federates. These services can be grouped into three categories: federation management, data management, and time management. The federation management services control the dynamic inclusion and removal of federates and implement other global operations for the federation that are similar to those found in distributed co-simulation environments.

The main goals of the data management services are: control the data to be communicated between the federates, establishing origin – destination relationships; assign ownership of simulation objects to federates; offer support for the creation, removal and identification of data; and implement data routing between federates. Time management services control the time progression in the federates. These services must be also offered by mechanisms supporting distributed co-simulation environments, even if specified or implemented with alternative (usually proprietary) approaches.

The specification of RTI includes APIs that define how these services are accessed. The use of APIs, however, suggests adapting the code of a new simulator to be integrated into a federation. As an alternative to maintain transparency and to avoid the need of severe modifications in the simulator implementation, *gateways* between the federates and RTI may be introduced [12], as explained later on.

## 5. DISTRIBUTED CO-SIMULATION BACKBONE

DCB may be seen as a simulation-specific coordination layer for supporting distributed co-simulation. Its main goal is to offer a generic mechanism supporting communication and cooperation services between heterogeneous federates. The word “federate” refers to any computational sub-model, simulation sub-model, physical device, or even human actor.

DCB has been defined in the scope of the SIMOO project. SIMOO is a general-purpose, object-oriented simulator for discrete systems, which is being applied to the design of electronic embedded systems [13,14]. A non-distributed co-simulation tool, integrating SIMOO and VHDL, has been already implemented [15]. An adaption of SIMOO to the HLA standard is now underway [16]. The DCB implementation must support a transparent cooperation between models generated by SIMOO.

As strongly based on the HLA standard, DCB considers its definitions, especially those related to RTI, for building a co-simulation backbone. DCB has three main functions that are closely related to the RTI functionality: interface specification, synchronization management, and management of data exchanges between federates. In order to give a better support to co-simulation, however, DCB implements special strategies in these three parts, extending the RTI functionality and also removing some of its features that would inhibit flexibility.

## 5.1 Overall architecture

In order to handle the complexity of communication interfaces between heterogeneous federates and maintain flexibility, DCB uses the concept of *gateway* [12]. Gateways translate data formats, according to the destination of the data sent through DCB. Gateways are not implemented directly within the simulators or DCB, but as part of *ambassadors* [3]. Figure 1 shows the DCB architecture.

According to the DCB architecture, the simulators don't need to invoke communication primitives of DCB, as needed when using the RTI, for sending data. The simulator transfers output data to its ambassador. This, in turn, makes these data available at its interface, and the DCB ambassador will check for the data. In the opposite direction, the same tasks are performed by the two ambassadors.

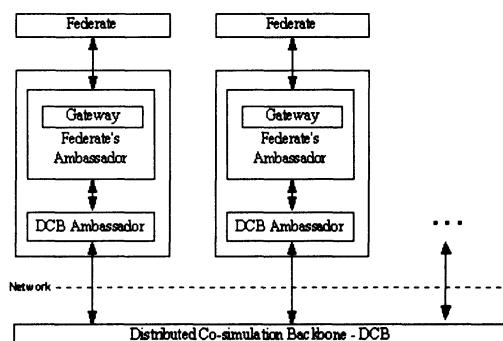


Figure 1. Architecture of the DCB

Because of this architecture, two ambassadors (the simulator and the DCB ones) must be developed when a new simulator is integrated into a co-simulation environment. They are specific for a particular application model, because they depend on the particular data to be exchanged through the sub-model interfaces. It is thus much more appropriate to talk about the

integration of a new *federate* (or sub-model) into an environment, and not of a new *simulator*.

If a federation includes N federates, even if they are implemented by the same simulator type (e.g. a VHDL simulator), 2N different ambassadors will need to be developed. These 2N ambassadors, however, are very similar, following the same control patterns, and differing only by the attributes they must control.

Although there may be some implementation effort in the development of ambassadors, they avoid modifications both in the communication and synchronization infrastructure (the DCB kernel) and in the simulator. These modifications would be also very costly, if compared to the construction of ambassadors.

The DCB infrastructure is general-purpose and is not affected by particular federates to be integrated into a federation. Furthermore, a simulator can be easily integrated into a DCB co-simulation environment because its code doesn't need to be reprogrammed at all. There are, however, certain requirements that must be met by federates, as discussed in the following sub-sections, in order to integrate them.

## 5.2 Interface specification

When a new federate is integrated into the co-simulation environment, the environment designer must explicitly declare the *attributes* that will be used for data exchanging and for synchronization at the sub-model interface. These attributes are used by DCB and the federate (in fact through their ambassadors) as structures for sending and receiving data.

In the attribute specification (a process called *interface configuration*), the designer must include the following attributes:

- the simulator execution mode (synchronous or asynchronous - a constant value);
- TLS – time of the last state saved by the federate;
- LVT – local virtual time of the federate;
- set of output variables of the federate; and
- set of input variables of the federate.

The execution mode and the LVT are standard attributes for all federates. The TLS is also mandatory for asynchronous simulators. The remaining attributes may vary in quantity and purpose and depend on the particular application model.

Therefore, according to the DCB approach, a simulator may be integrated into a co-simulation environment only if it can make both TLS and LVT available at its interface. When a federate is included in the environment, the designer must also indicate the other federates with which

the new one must interact and through which attributes the cooperation will take place. With this information, DCB implements the data exchanges, guaranteeing consistency regarding synchronization and interfaces.

Attributes are handled by the ambassadors, which are responsible for their management. Ambassadors must keep track of attribute values, by monitoring their changes, translating them to other data formats, and sending their contents to the respective destinations.

A new federate that is added to a co-simulation environment receives a unique identifier from DCB. This feature doesn't allow the recognition of two identical (replicated) federates in the same federation. In DCB, each federate (or simulator instance) is unique.

### 5.3 Data management

Data management in DCB reflects the basic principles prescribed by the HLA standard but is adapted to the purposes of generality and heterogeneity that are fundamental to DCB.

RTI offers six different classes of services [3], and four of them are related to data management: Declaration Management; Data Distribution Management; Object Management; and Ownership Management. Since some of these services define requirements that affect the implementation of the federates, their use in DCB would be restrictive, considering that code modifications in the simulators to be included in a federation are not desirable. Therefore, these particular services are not implemented by DCB.

The *ownership management* service, in turn, is very important. As RTI, DCB defines and manages through this service a responsibility over the attributes that it shares with all federates. Only a federate that *owns* a given attribute may update its value, and the ownership of the attribute may vary along the time. It is thus possible for DCB to implement a mutual exclusion policy regarding the use of attributes by the federates, so avoiding undue attribute value updates, performed either by the local or by a remote federate. Ownership management is essential also for synchronization purposes, as explained next.

### 5.4 Synchronization

According to the HLA definition, while each federate has its own local virtual time, RTI manages the global simulation time. RTI offers pre-defined control functions that must be used by the federates in order to receive a grant for advancing their local times. This approach, however, makes the integration of new heterogeneous simulators into a federation a more complex task, since each simulator must be coded so that it calls those

control functions. Therefore, although using the HLA global mechanism for time advancement, DCB does not follow RTI's approach of control functions.

As prescribed by the HLA standard, DCB supports an hybrid synchronization (synchronous or asynchronous time advancement) [11]. In distributed environments, the asynchronous approach usually presents less overhead. In order to implement an asynchronous mode, however, a simulator must support rollback to a previous safe state [17], because of events it receives with past time stamps. The simulation performance may be degraded if federates must execute rollbacks very often, but this is highly dependent on the application.

In the synchronous mode, DCB uses the ownership management service together with the LVT of the federates in order to guarantee that only safe events are executed. Suppose two federates A and B, and A sends a message that modifies a value of an attribute X of B. DCB will grant the ownership of X to A, thus allowing A to modify the value of B. Besides that, each federate may only increment its LVT if the ownership of the LVT is granted to it by DCB. From this strategy, it becomes clear that a synchronous simulator may be integrated into a DCB federation only if it may declare the LVT at its interface and wait for a grant signal from the ambassador to advance the LVT's value.

In order to optimize performance in a synchronous operation, DCB supports the look-ahead mechanism [18], both static and dynamic. If L is the look-ahead value, then the update of an attribute X of a federate B, by another federate A, is safe if  $(LVT(A)+L) \geq (LVT(B))$ .

The same ownership management mechanism is used for time advancement in the asynchronous mode. The only difference is that, in this case, DCB supports unsafe events and implements a rollback mechanism. For this, federates need to store safe states. DCB, on the other side, must store messages sent by the federates with future time stamps, if compared to the global time, and send anti-messages [17] in case of causality errors.

## **6. CONCLUSIONS AND FUTURE WORK**

One of the most relevant challenges in co-simulation is the construction of an adequate mechanism for the cooperation between heterogeneous simulators. This paper presented a generic architecture, called DCB, which is based on the HLA standard and supports distributed environments consisting of heterogeneous simulators. DCB presents flexible strategies for the management of data interactions and synchronization between the simulators.

Generality and flexibility have been the main requirements in the development of the DCB. New simulators may be easily introduced into an heterogeneous and cooperative environment, without regard to their implementation technologies or languages and without needing any modification of their internal code.

The simulators, however, must show a few fundamental features, which have been discussed in the paper, in order to be able to participate in a federation. Nevertheless, these features are very simple and may be considered as unavoidable, if one wants to integrate a simulator into a cooperative environment.

Because of its generality regarding the management of the integration of heterogeneous simulators, DCB may be used as a basis for the implementation of distributed simulation environments in a wide range of application domains.

Different domains, however, may have distinct performance requirements, and synchronization requirements and temporal restrictions may vary heavily. Consider for instance the distinctions between a pure software co-simulation of electronic designs, a co-simulation environment with physical mechatronic components-in-the-loop, a co-simulation environment with physical electronic components-in-the-loop, and a training simulation environment with human-in-the-loop. These very distinct situations considering real-time requirements may enforce different implementations of the DCB architecture.

Using DCB, the tool designers must interact just with the ambassadors, because DCB's main goal is to provide and support cooperation among any existent simulators. On the other side, with commercial tools, such as from Cadence, Coware, Cossap, and Synthesia, the tool designers must interact with a more complex design environment and are usually restricted to a given number of description languages, such as C++, VHDL, or Verilog.

For the validation of the DCB architecture, a particular federation for the co-simulation of electronic embedded systems is being implemented. In parallel, a supporting environment for the DCB development methodology is being built. It will offer services and resources for the configuration of interfaces of federates and for the semi-automatic generation of the ambassadors. The object-oriented features of the SIMOO modeling and simulation framework [13] will be basic for this supporting environment.

## 7. REFERENCES

- [1] F.Hessel, P.L.Marrec, C.Valderrama, M.Romdhani, and A.A.Jerraya, "MCI: Multilanguage Distributed Cosimulation Tool". In: F.J.Rammig (ed.), *Distributed and*

- Parallel Embedded Systems*. Kluwer Academic Publishers, 1999. (Proceedings of DIPES'98)
- [2] G.Borriello and R.Want. "Embedded Computation Meets the World Wide Web", *Communications of the ACM*, Volume 43, n. 5, May 2000.
  - [3] F.Kuhl, R.Weatherly, and J.Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, MITRE Corporation, 2000.
  - [4] K.Hines and G.Borriello. "A Geographically Distributed Framework for Embedded System Design and Validation". In: *Proceedings of the 35<sup>th</sup> Design Automation Conference*, June 1998.
  - [5] K.Kim, Y.Kim, Y.Shin, and K.Chi. "An Integrated Hardware-Software Cosimulation Environment with Automated Interface Generation". In: *7<sup>th</sup> International Workshop on Rapid Systems Prototyping*, June 1996.
  - [6] R. Dhananjai, V.Chernyakhovsky, and P. A.Wilsey, "WESE: A Web-based Environment for Systems Engineering". In: *Proceedings of the 2000 International Conference On Web-Based Modelling & Simulation (WEBSIM 2000)*. January 2000.
  - [7] P.Chou, R.Ortega, K.Hines, K.Partridge, and G.Borriello. "IPChinook: An Integrated IP-based Design Framework for Distributed Embedded Systems". In: *Proceedings of the 36<sup>th</sup> Design Automation Conference*, New Orleans, June 1999.
  - [8] M.Dalpasso, A.Bogliolo, and L.Benini. "Virtual Simulation of Distributed IP-based Designs", *Proceedings of the 36<sup>th</sup> Design Automation Conference*, New Orleans, June 1999.
  - [9] E.A.Lee et al. "Overview of the Ptolemy Project". *Technical Memorandum UCB/ERL M01/11*. University of California, Berkeley, 2001.
  - [10] W.Sung and S.Ha. "Hardware Software Cosimulation Backplane with Automatic Interface Generation". *Proceedings of the ASPDAC'98*, 1998.
  - [11] J.S.Dahmann. "High Level Architecture for Simulation". *Proc... of the 1<sup>st</sup> International Workshop on Distributed Interactive Simulation and Real-Time Applications*, 1997.
  - [12] S.Strassburger, T.Schulze, U.Klein, and J.Henriksen. "Internet-based Simulation Using Off-the-shelf Simulation Tools and HLA". *Proceedings of the Winter Simulation Conference*. Washington, USA, December, 1998.
  - [13] F.R.Wagner, M.Oyamada, L.Carro, and M.Kreutz. "Object-Oriented Modeling and Co-simulation of Embedded Electronic Systems". In: L.M.Silveira, S.Devadas, and R.Reis (eds.), *VLSI: Systems on a Chip*. Kluwer Academic Publishers, 2000.
  - [14] L.Carro, M.Kreutz, F.R.Wagner, and M.Oyamada. "A Design Methodology for Embedded Systems based on Multiple Processors". In: B.Kleinjohann (ed.), *Architecture and Design of Distributed Embedded Systems*. Kluwer Academic Publishers, 2001. (Proceedings of DIPES'2000)
  - [15] M.Oyamada and F.R.Wagner. "Co-simulation of Embedded Electronic Systems". In: *Proceedings of 12<sup>th</sup> European Simulation Symposium*. Hamburg, Germany, October 2000.
  - [16] D.Wildt and F.R.Wagner. "Adapting Simulation Environments to HLA: Discussion and Case Study". In: *Proceedings of the European Simulation Multiconference*, Prague, Czech Republic, June 2001.
  - [17] M.Elnozahy et al. "A Survey of Rollback-Recovery Protocols in Message-Passing Systems". *Technical Report CMUCS99148, Department of Computer Science*. Carnegie Mellon University, September 1999.
  - [18] R.M.Fujimoto. "Parallel Discrete Event Simulation". In: *Communications of the ACM*, Vol. 33, n. 10, October 1990.

# **Automatic Code-Transformation and Architecture Refinement for Application-Specific Multiprocessor SoCs with Shared Memory**

**Samy Meftali, Ferid Gharsalli, Frédéric Rousseau and Ahmed. A. Jerraya**  
*TIMA laboratory, 46 av. Felix Viallet 38031 Grenoble cedex (France)*

**Abstract:** Memory represents a major bottleneck in embedded systems. For multimedia applications bulky of data in these embedded systems require shared memory. But the integration of this kind of memory implies some architectural modifications and code transformations. And no automatic tool exists allowing designers to integrate shared memory in the SoC design flow. In this work, we present a systematic approach for the design of shared memory architectures for application-specific multiprocessor systems-on-chip. This work focuses on the code-transformations related to the integration of a shared memory.

**Key words:** Shared memory, code transformation, architecture refinement.

## **1. INTRODUCTION**

The design of modern digital systems is influenced by several technological and market trends, including the ability to manufacture ever more complex chips but with increasingly shorter time-to-market.

The choice of a shared memory architecture for a given SoC implies the integration of some new modules in the application description (memory and controllers) and many code transformations at several abstraction levels in the design process.

The goal of transformations and code generation in the case of multiprocessor SoCs with a shared memory is to adapt the code of the application to a such memory architecture. In fact, we imperatively need to replace the simple shared data accesses at a high abstraction level by explicit requests to the shared memory block.

Unfortunately, nowadays, there is not a complete and automatic method allowing designers to integrate all these memory types (particularly the shared memory) in the SoC from a high abstraction level. Our objective is to provide designers with a global and fast method and tools to design such a systems in order to satisfy the time-to-market constraints. We focus in this work on the code-transformations due to the integration of the shared memory into the SoC and on the automatic code generation.

Our approach is easily automatisable and allows a completely automatic generation of an architecture level specification of the application. Now, multiprocessor SoC integrate more and more elements, and the description of such a system at the architecture level can reach 200k lines of code (SystemC, C, VHDL), which makes this work very beneficial to the designers from the time-to-market point of view.

This work is organized as follows: in Section 2, we give an overview of related work on code transformations on SoC with shared memory. In Section 3, we present our multiprocessor SoC design methodology, then our three abstraction levels and the memory representations in section 4. Section 5 describes the code transformations. These transformations are illustrated by an application in Section 6. We conclude this work in Section 7.

## **2. RELATED WORK**

In this work we are only concerned with SoCs. These system architectures are different from classic general purpose architectures [3] because they target a specific application. This makes the memory architecture and the communication network specific to the application and then simpler. For instance, in most of these applications data regularity is quite trivial or non existing and thus no sophisticated data cache is required.

In the literature three kinds of code transformations exist depending on the level where they are performed.

The high level transformations concern the application code at the system level. Their goal is to improve the code quality. In fact they consist mainly in modifying loop structures in order to reduce the number of memory accesses [2]. This kind of code-transformations does not take into account the specificity of the memory architecture chosen for the application, and these transformations are not generally needed when the code is written by an experimented designer. Many research groups [13] [8] [4] work on these high level transformations.

The low level transformations are generally very related to the low level (RTL) characteristics of the architecture. Some of them are due to memory characteristics. For example in [10] [11] they use fast access modes (read

and write page mode) on a typical DRAM to improve memory cache performances. Using these access modes, different data can quickly be read or write in the same page. Moreover, a good scheduling of elementary instructions (generated by compiler) which access to the memory allows to obtain better performances. Unfortunately the low level transformations appear at the end of the design flow. So, the integration of a shared memory in the system is done manually, which is very time consuming.

Some code transformations in the literature concern more than one abstraction level in the design flow. IMEC [2] works on the generation of the optimized memory part for embedded systems (single process). The main idea is to study the application and generate memory architecture, for single process applications with a parallelizing compiler.

The contribution of our work is the proposal of a full systematic approach allowing a multi level automatic code transformations and generation for application-specific multiprocessor SoCs with a shared memory.

### 3. MP SOCS DESIGN WITH SHARED MEMORY

The methodology and tools we developed to design multiprocessor SoC are described in [1]. It starts a system level specification (Figure 1).

Processors and communication components are allocated and system behavior and communication (ports and channels) are mapped/scheduled on processors and communication components of the architecture template (by the designer). After the allocation/mapping step, an architecture level specification is obtained.

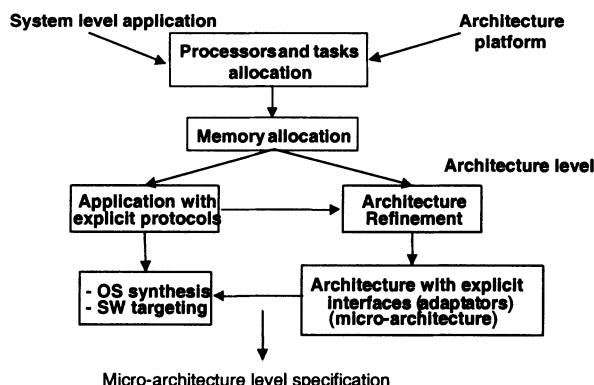


Figure 1. Our design flow

For each processor, the software code (Operating System and application code of tasks) is assembled (from libraries). Communication interfaces between processors and the communication network are also generated. We obtain the micro-architecture of the system.

The choice of the processor was based on availability (only ARM7 and MC68K processors). The communication network is a point-to-point network. Designers can modify some parameters such as the number of CPUs, the memory size and I/O ports for each processor, interconnection between processors, the communication protocols and the external connections (peripherals).

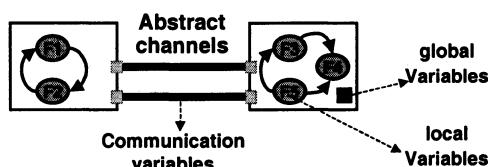
One of the most important steps in our design flow is the memory allocation. During this step, we try to choose an optimal memory architecture for the SoC. Therefore, we use an integer linear programming model generated from the system specification of the application. It allows us to choose the best memory architecture for the design [9].

## 4. ABSTRACTION LEVELS IN OUR DESIGN FLOW

In this section, we define the three abstraction levels used in the design flow [12]. As our objective is to integrate the shared memory into the SoC from a high abstraction level, one will focus in the remainder of this work primarily on the first two levels.

### 4.1 System level

At this level (*Figure 2*), modules communicate through abstract channels. No assumption about communication implementation is made. Hence, the abstract channels ensure independent protocol communication of concrete generic data types by providing abstract level communication primitives (c.g. send/receive). Such primitives encapsulate all the communication details. Basic module behavior are described by tasks communicating by sending and receiving messages. SDL is a typical system level language.



*Figure 2. System level*

## 4.2 Architecture level

At this level (Figure 3), modules correspond to the architecture blocks. Communication is modeled by logical interconnections encapsulating architecture level protocols (e.g. handshake or finite FIFO). The communication primitives on the module ports are read/write fixed data types in conformity with a certain protocol (e.g. read-handshake or write-handshake). SystemC1.0 is an example of languages that describe systems at this abstraction level.

In our design flow, the architecture level description is automatically generated. It is obtained after the memory blocks allocation step. In fact, if we decide for example to integrate a shared memory into the SoC, we have to insert a block into the application description. This block will mainly contain two modules:

- The memory matrix: it is a generic code describing the memory block. It is independent of the application which permits an easy automation. At this level, this module is connected only to the controller by the address channel, data read and data write channels. So it does not depend at all on the number of processors, accessing the shared data.
- The memory controller: it is built of two parts connected to the memory matrix: one input and one output controller. The input controller is also connected to all the processors writing data in the shared memory and the output controller is connected to those which read data from the shared memory. These controllers will be the memory adaptator at the micro-architecture level.

The fact of separating the shared memory into 3 modules (matrix, input and output controllers) at this level gives a better modularity to the SoC. For example, if we decide to reuse the SoC for an other application with an additional processor accessing to the shared data for writing, we do have to modify neither the description of the matrix nor the output controller.

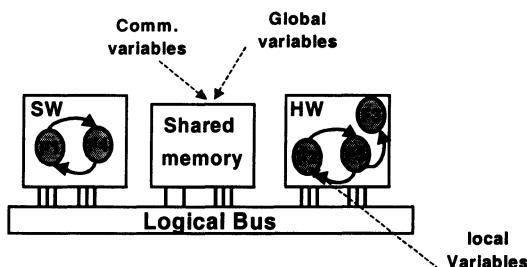


Figure 3. Macro-architecture level

### **4.3 Micro-architecture level**

At the micro-architecture level, the modules are physical blocks (DSP, CPU, IP ...) Communication is modeled by physical signals and communication primitives are consequently set/reset of signals [1]. Communication time is based on the clock cycle. VHDL and Verilog are languages permitting to describe systems at the micro-architecture level.

At this level, memories are physical (SDRAMs). In order to connect the shared memory to the communication network, we insert between them one memory adaptor which adapts the access protocol of the memory to that of the network. The memory interface is independent of the processors, which increases the flexibility of the target architecture. It depends only on the communication network and the memory. The memory interface is assembled using basic components in our libraries [5].

## **5. APPLICATION CODE-TRANSFORMATIONS**

In this section, specific code-transformations are presented. Some of them are related to memory accesses, when a shared memory is added, and some others concern memory controllers refinement.

### **5.1 Architecture level transformations**

At the system level we have only processors communicating by messages passing, and we do not find any shared memory or any memory controller blocks in the application description. The data exchange between the blocks at this level is made by simple Send/Receive primitives.

After deciding which data would be in the shared memory block at the memory allocation step, we have to deal with two kinds of shared data. In fact, we distinguish global variables and communication data. Each one of these types needs an appropriate code-transformation in order to generate a new application specification taking into account the shared memory.

#### **5.1.1 Synchronization signals (binary type)**

We assume in our applications that the data consistency problems are entirely resolved by the system level designer, by synchronization. So, we choose in our tools to never modify such signals. In fact, synchronization signals are boolean so, we do not put any binary variable/signal in a shared memory. This does not decrease the performances of our SoC due to the small size of such a type of data.

### 5.1.2 Non-binary communication variables

When we decide in the allocation step to insert a shared memory in the SoC, we have to modify all accesses to the variables that we decide to put in such a memory by explicit accesses to the shared memory. We insert the shared memory into the application, and we generate an abstract allocation table that contains for each shared variable, in which memory it will be placed. So, at this level each data in the shared memory must be characterized by a logical address (index in the memory matrix) and a name in the abstract allocation table.

Suppose that at the system level “X” is a shared data between the processors P1 and P2. The system level code in the two processors corresponding to the exchange of “X” will be as in *Figure 4*

|  |  |
|--|--|
| <b>Processor_sender_P2</b><br>{<br>..<br>Send(X,P1);<br>Send(synch_signal,P1);<br>Wait();<br>..<br>} | <b>Processor_receiver-P1</b><br>{<br>..<br>Wait for synch_signal_P2<br>Receive(X);<br>..         } |
|--|--|

*Figure 4.* System level communication

In order to send “X” to P1, the processor P2 has just to send the variable value then a synchronization signal informing P1. This later waits on the synchronization signal from P1, then receives “X” through the channel connecting it to P2.

At the architecture level, if we decide to insert a shared memory into the system, the shared data “X” will be into this memory and not in the P1 and/or P2 local memories.

#### 5.1.2.1 Sending data

The primitive `Send(X)` in P2 behavior code will be transformed in a writing request to the input shared memory controller as shown in the following code (*Figure 5*).

|  |   |
|--|---|
| <b>Processor_sender_P2</b><br>{<br>..<br>write_SM_input_ctr("X",X);<br>write(synch_signal);<br>wait();<br>.. | <b>Input_Controller</b><br>{<br>..<br>ind = allocation_table ("X");<br>write_value(ind,X);<br>wait();<br>.. |
|--|---|

Figure 5. Sending data at macro-architecture level

After receiving a such request, the input memory controller takes the data index in the memory matrix from the abstract allocation table. Then writes the data value in the corresponding memory cell.

### 5.1.2.2 Receiving data

As in the case of sending a data, the primitive Receive(X) in P1 behavior code will be transformed as in *Figure 6*

|   |   |
|---|---|
| <b>Processor_receiver_P1</b><br>{<br>..<br>Wait for synch_signal;<br>Ask_for_data_in_shared_memory ("X");<br>Wait();<br>Read_data(X);<br>.. | <b>Output_Controller</b><br>{<br>..<br>Ind = allocation_table ("X");<br>X = Read_value(ind);<br>Write_data_2_P1(X);<br>Wait();<br>... |
|---|---|

Figure 6. Receiving data at macro-architecture level

#### Notes:

- For the architecture level transformation of the Send primitive, the processor sender must give the variable value (X) and its label ("X").
- All the instructions corresponding to a write or read operation in the memory controller code are executed in one clock cycle.

### 5.1.3 Global variables

In the system level application code, we find some accesses to the global data in some expressions/assignments in the behavioral part of processes. In fact, if "X" is a global variable, we can find in a process in the system level description an expression as:  $Y = X + 2$ , or  $X = Y/2$ ;

The first instruction correspond to a read access and the second one to a write access to "X".

If "X" is in the shared memory, we must generate explicit accesses to this variable in the new application code. So, in the two cases (read and write) we replace the occurrence of "X" in the code as in *Figure 7*.

The instruction (1) consists in sending a signal through the channel connecting process to the shared memory output controller, to ask for the variable “X”. After receiving a such signal (after the synchronization instruction (2)), the controller finds the index corresponding to “X” in the abstract allocation table then reads the variable in the memory matrix, and sends it to the processor. In the code, this value is read and copied in a local variable “var” (3). Then the expression is performed in (4).

For the second instruction (write) “ $X = Y/2$ ”, the instruction (5) consists in sending a writing message from “A” to the input shared memory controller. This message contain two parameters: the name of the shared variable “X” and its new value ( $Y/2$ ).

| <b>“<math>Y = X + 2</math>”</b>  | <b>“<math>X = Y / 2</math>”</b>  |
|--|--|
| <pre>{ ... Sig_to_read_shared_mem(X); --- (1) wait(); --- (2) var = read_shared_mem(X); --- (3) Y = var + 2; --- (4) ... }</pre> | <pre>{ ... write_shared_mem("X", Y/2); --- (5) wait(); --- (6) ... }</pre> |

Figure 7. Code transformation for global variables

## 5.2 Architecture level application-code generation

One of the main contributions of our work is the systematic and completely automatic generation of the architecture level description of the application. This step consists in inserting the high level memory matrix block in the application specification, then generating the application specific code of the input and output memory controllers in order to connect the memory with other modules of the SoC. After that we perform the necessary code-transformations described in this work in order to adapt the accesses to the shared memory architecture.

## 5.3 Micro-architecture level transformations

At this level, read and write operations in the shared memory become very explicit and dependent of the shared memory characteristics.

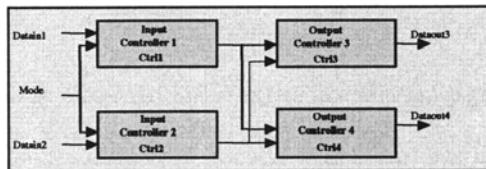
The input and output controllers of the architecture level are refined to be a communication adapter between the memory and the communication network. In our case we consider that the memory adaptator is a slave processors adaptors. The adaptator receives an access request containing the address and the data in the case of a writing request. If there is any

competitor access, the controller performs the address decoding while using an allocation table more detailed than that of the previous level, and updates the memory signals (in read or write modes). After a certain memory latency, the adaptor sends an acknowledge to the processors adaptor allowing it to ask for new accesses.

In the case of several simultaneous accesses, the memory adaptor does the same thing while respecting mutual accesses exclusion to the memory.

## 6. APPLICATION

In order to illustrate the efficiency of the proposed code-transformations and code generation methodology, we detail in this section the flow steps on a packet routing switch. It constitutes a powerful solution for large-frame or cell-switching systems [7]. The version we present here consists of two input controllers and two output controllers. Each of the controllers handles one communication channel. The communication links between input and output controllers are configured by an external signal to be direct or switched. *Figure 8* shows the block diagram of the packet routing switch.



*Figure 8.* Block diagram of the packet routing switch

### 6.1 Architecture level code-transformations

In this application, after the memory allocation step, we decide to put two variables in the shared memory block. This stage modifies the application code by taking into account the shared memory architecture.

At this step a shared memory module, an input and an output memory controllers are generated and integrated to the application as shown in the *Figure 10*. The memory input controller is connected to controller1 and controller2 because these later access memory to write data. The memory output controller is connected to controller3 and controller4 as they access the memory to read the shared data. We generate also an abstract allocation table which contain label, type, index in memory of each shared data.

|  |   |
|--|---|
| <pre>// Implementation of Ctrl1//</pre> <pre>void ctrl1 ::entry() {     ...     if(ordre.read() == true)     {         dataCtrl1_mem_in.write(x);         dataCtrl1_mem_in.write("x");         signalCtrl1_C3.write(true);         Wait();     ... }</pre> | <pre>/Implementation of Ctrl 3 //</pre> <pre>void ctrl3 ::entry() {     ...     if(sig_Ctrl1_C3.read() == true)     {         Signal_mem_out.wrtite("X");         Wait();         y = data_mem3_out.read();         output3.write(y);     ... }</pre> |
|--|---|

Figure 9. Communication between Ctrl1 and Ctrl2 using a global shared memory

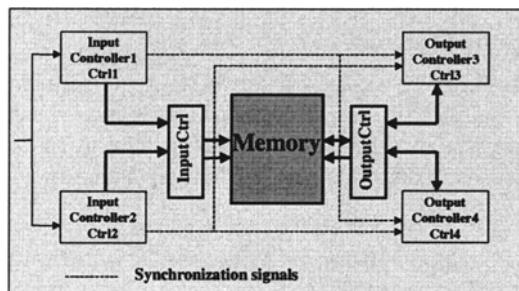


Figure 10. Packet routing switch description at macro-architecture level

## 6.2 Analysis

This application was described at the functional level mainly in 4 interface files and 4 implementation files. Automatic refinement adds 4 files to the specification (2 interfaces and 2 implementations) corresponding to the memory body and to the memory controller (200 lines at the functional level). The interfaces of the 4 processors were modified automatically in order to connect them to the global shared memory, and all the accesses to the data resident in this memory were modified (Figure 9). We obtained the application code at the architecture level with a shared memory architecture in a complete automatic way.

The automatic code-transformation and generation is surely very profitable to the designer in time-to-market point of view because of the huge size of the SoC descriptions especially at micro-architecture level.

## 7. CONCLUSION

In this work, we presented an automatic architecture refinement and code-transformations flow for application-specific SoC with a shared memory, starting from a parallel system-level description of a given application. We focus on the shared memory representation at different abstraction levels and the code-transformations. The proposed methodology permits a systematic code-transformation and the generation of a generic memory architecture for multiprocessor embedded SoC, from a high abstraction level distributed specification of the application. We have seen the effectiveness of our approach on an example.

## 8. REFERENCES

- [1] A. Baghdadi, D. Lyonnard, N-E. Zergainoh, A.A. Jerraya, "An Efficient Architecture Model for Systematic Design of Application-Specific Multiprocessor SoC", DATE'2001.
- [2] F. Cathoor & al, Custom Memory Management Methodology, Kluwer Academic Publishers, 1998.
- [3] D. Culler, J.P. Singh, A. Gupta, "Parallel computer architecture: A Hardware/Software approach", Maурган Kauffman publishers, August 1998.
- [4] A. Fraboulet, G. Huard, A. Mignotte, "Loop Alignment for Memory Accesses Optimization", Proc. of ISSS 1999.
- [5] F. Gharsalli, S. Meftali, , F. Rousseau, A.A. Jerraya, " Automatic Generation of Embedded Memory Wrapper", Proc. of DAC 2002.
- [6] J. Hennessy , M. Heinrich, A. Gupta, "Cache-Coherent Distributed Shared Memory: Perspectives on Its Development and Future Challenges", Special issue on distributed Shared-Memory Systems, Match 1999.
- [7] IBM, Inc. "28.4G Packet Routing Switch", Networking Technology Data sheets, [http://www.chii\)ibm.com/techlib/products/commun/datasheets.html](http://www.chii)ibm.com/techlib/products/commun/datasheets.html)
- [8] D. Kulkami, M. Sturm, "Linear loop transformations in optimizing compilers for parallel machines" in The australian computer journal, pp.41-50, May 1995.
- [9] S. Meftali, F. Gharsalli, F. Rousseau, A.A. Jerraya, "An Optimal Memory Allocation for Application-Specific Multiprocessor System-on-Chip", Proc. of ISSS 2001.
- [10] P. R. Panda, N. Dutt, A. Nicolau, Memory Issues in Embedded Systems-on-chip: Optimization and exploration, Kluwer Academic Publishers, 1999.
- [11] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, J. D. Owens, "Memory Access Scheduling", Proc. of ISCA 2000.
- [12] K. Svarstad, G. Nicolescu, A. A. Jerraya, "A Model for Describing Communication between Aggregate Objects in the Specification and Design of Embedded Systems", DATE'2001.
- [13] M. Wolf, "improving locality and parallelism in nested loops", Ph.D dissertation, Stanford University, USA, August 92.

# **Modeling Power Dynamics for an Embedded DSP Processor Core**

## *An Empirical Model*

C.H.Gebotys and R.Muresan

*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L3G1*

**Abstract:** A new model for dynamic current analysis and simulation is presented for power and energy analysis of a complex VLIW DSP processor core, targeting secure wireless communications. Unlike other research, an instruction level RC based model, whose input parameters can be extracted from the DSP core's assembly level program, is introduced for power simulation. Experimental results utilizing several benchmark cryptographic applications show that the model can accurately simulate power variation of a program. For the first time results are verified with real power traces of a DSP processor core in a VLSI chip running cryptographic applications with an average error in energy estimation of 7%. This research is important for analyzing the impact of software on power and the design of embedded cryptographic VLSI systems that are safe from power attacks.

**Key words:** Power, DSP, Current, Model, Energy

## **1. INTRODUCTION**

Power dissipation has a large impact on cost, reliability and security of an embedded system. Previous literature has discussed the first two issues at length, however only recently with the widespread use of wireless communications has interest in security of embedded system designs increased. Typically embedded systems involve the design of low power, low cost, high performance algorithms supporting wireless communication

systems with audio and video types of data. The encryption of audio and video data before transmission is very important. Not only must encryption algorithms be high performance and low power, but more importantly they must be secure or safe from power attacks. For example, power attacks involve utilizing power traces (or dynamic power measurements over time) of a device to determine the users secret key. The dynamic power trace must be simulated during the design of the encryption algorithm to make sure the embedded system is secure. Unfortunately most power simulation is performed at the transistor or module level which is too slow for complex processors. This paper for the first time introduces a high level instruction based dynamic power simulation model which is targeted towards encryption algorithms, yet can be also applied to any general DSP application to simulate power.

DSP processors are utilized in many embedded systems in particular for wireless communication applications, due to their low cost, low power dissipation and high performance. Recently advances in DSP cores have provided highly parallel VLIW (very long instruction word) based processor cores, such as the SC140 Star Core processor developed by Motorola and Lucent Technology [1]. This processor core has four functional units, two address arithmetic units, and one bit mask unit [1]. The SC140 core processor is based on a Variable Length Execution Set model [1]. Like VLIW processors, a group of instructions, called an execution set, can be executed in parallel. An atomic operation within an execution set is encoded by an individual instruction. An eight-word instruction set, called a fetch set, is fetched from memory every clock cycle and the processor detects the portion of this set that can be executed in parallel [1]. Based on grouping constraints, an execution set may vary from one to six atomic instructions [1]. The register file blocks within the SC140 DSP core are divided into two banks each containing eight 40bit registers. Based on the instruction types and the usage of the upper bank registers within an execution set the instructions are serially grouped or prefix grouped by the assembler [1]. Unlike other DSP processor cores, the SC140 can load or store eight sixteen bit words per cycle (providing a high data memory bandwidth[2]). In this paper we introduce a methodology for developing an empirical instruction-based model which is capable of estimating dynamic current, power or energy consumption for a DSP processor core. The methodology for creating the dynamic power model generally can be used with most processors, however it is illustrated in this paper with the SC140 DSP processor core.

## 2. PREVIOUS RESEARCH AND PROBLEM DEFINITION

A critical problem in wireless communication applications is the battery life time. The complexity of the targeted applications for the DSP processors makes low power designs very difficult. Transistor and module level power models have been developed [3] as well as software techniques [6]. Recent research work is directed towards finding power consumption modeling techniques at the software level which will benefit the compilers for these DSP processors [5,7,11]. Power modeling at the instruction level for processors were investigated in [4,7]. However dynamic power simulation is not supported and only static single average power values are returned for a given software program running on a processor. Furthermore few researchers have examined complex VLIW processors where more than one instruction execute in parallel. Dynamic power traces have been used in power-attacks of cryptographic devices. In particular the analysis of the variation of power, and computations on a number of power traces can be used to detect data and algorithmic dependencies. For example this information could be used to detect the secret key of a smart card, thus performing a power-attack. Currently power attacks of cryptographic devices, have been analyzed using slower clock frequency, non-VLIW processors in non-time critical applications, such as smart cards [8]. Future wireless communications devices such as encryption of voice and video will be time-critical and require VLIW processor implementation. Power attacks of more sophisticated processors with parallel instruction execution have not been reported in the literature. Thus RC circuit type models of power at the instruction level are new for the VLIW DSP processors. This paper will present a methodology for modeling dynamic power simulation for a complex VLIW DSP processor core, the SC140. An instruction level model based on RCs is developed and verified with real current measurements[9] of the DSP hardware VLSI core in a chip.

Given a complex VLIW DSP processor, the following questions are important for the design and analysis of power efficient applications:

1. Is there a relationship between the activity involved by a certain type of instruction, data handled by the instruction and the current consumption of the DSP processor?
2. Can we predict at any given time the current consumption behavior of an application based on the information received at assembly language level (or higher level) without knowing the detailed structure of the DSP processor and all the computational data handled by the application?

3. Is there a possibility of finding a general standard procedure of modeling the current behavior for an application at software level?

From extensive analysis of current dynamics of individual assembly language instructions and assembly language programs for the SC140 DSP processor[9,12] this paper will show that these questions can be answered positively. The methodology used to study these questions can also in general be applied to many other DSP processors.

### **3. METHODOLOGY**

The following section will outline the empirical methodology for developing a dynamic power model for a processor core. The SC140 core processor is a pipelined processor with five stages and parallel execution capabilities [1]. The parallel capabilities are given by the core's hardware configuration and by the type of instructions executed. The maximum number of atomic instructions executed in parallel in an execution set is six. In Fig.1 an RC model is shown, representing the execution set level of the processor. Each atomic instruction present in an execution set has a parallel RC module component in the total current draw for the execution set. For example if two type 1 instructions [1] are present in an execution set then two RC modules have their contacts closed. The instantaneous current for an execution set is given by the sum of the instantaneous currents of the individual atomic instructions that are part of the execution set. In this model there is an RC module activated by the presence of a high bank register in any of the atomic instructions of the execution set. Waveform analysis performed in this research indicated that this component is mainly dependent on the internal activity involved by using a high bank register and not on the number of high bank registers used in the execution set or on the fact that the execution set has a 2-word prefix. In a processor at any given time there is intense switching activity at the transistor level. The number of transistors involved in the switching activity is proportional with the amount of activity required by the instruction that is executing. The SC140 core pipeline has a pre-fetch stage, fetch stage, dispatch stage, address generation stage and execution stage [1].

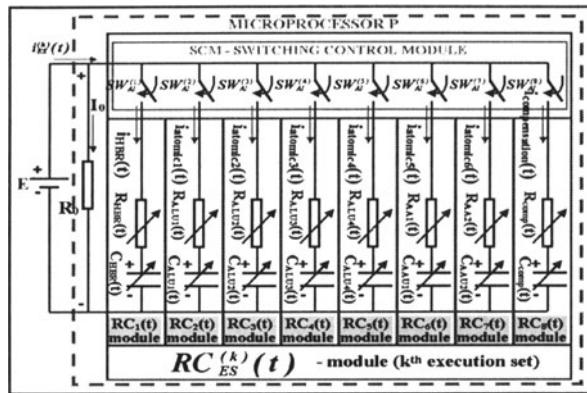


Figure 1. RC Model at the execution set level

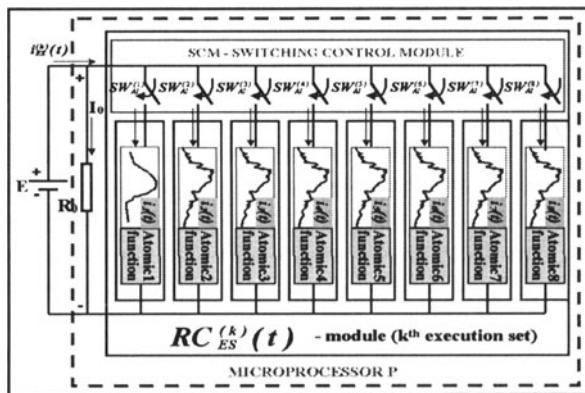


Figure 2. Program level RC model

Typically the most active stage of an instruction in a pipelined processor is the execution stage. As a result in each clock cycle the current draw of the processor will have a predominant component described by the executing instruction at that time. Considering that the total current draw at any given time is proportional with the number of transistors activated, then each RC module of Fig.1 can be considered to be formed by a large number of RC elements that work in parallel, and each RC element models one transistor. The fact that the total number of transistors activated at any given time is variable gives the variable component for the R and C elements of Fig. 1.

Based on extensive current measurements performed at the instruction level on different instructions executing on different data and it was deduced that

the number of transistors active during the instruction's life time could be modeled by a distribution known as the gamma function[10], in Fig.3 :

$$g(x; n, \lambda) = \frac{\lambda^n (\lambda x)^n}{n!} e^{-\lambda x}; x \geq 0, \lambda > 0$$

In the equation above,  $n$  is the order of the gamma function and characterizes the shape of the waveform. We found that the instantaneous current value of each instruction for the SC140 core processor can be approximated by:  $i(t) = K * g(x; n, \lambda)$  where  $\lambda$ , is dependent only on the clock frequency and  $K$  depends mainly on the type of the instruction executed (and could also characterize the data handled by the instruction).

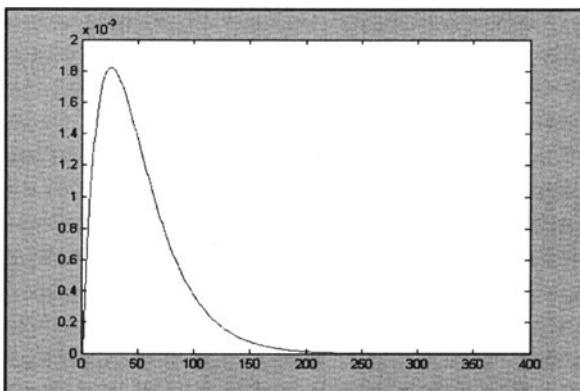


Figure 3. The gamma function

An important observation which made possible the RC model development at the software level is that the waveform shape is not dependent on the data handled by the instructions and during the execution time of any execution set the instantaneous current waveform can be approximated using the linear superposition principle by the following sum:

$$RC_{es} = \sum_{i=1}^8 k_i g_i(t - t_{start} - t_{delay}^{(i)}; n, \lambda)$$

where  $t_{start}$  is the time when the execution set is pre-fetched, and  $t_{delay}^{(i)}$  is a delay time which is present for *move* type instructions due to the fact that these instructions do not fully complete their operation in the last stage of the pipeline. Also  $g_i()$  is the gamma function for the  $i$ 'th RC module of Fig.2 ( $i$ 'th instruction in the execution set,  $es$ ) and  $k_i$  is the  $K$  coefficient for the RC module  $i$ . Good instruction and program approximations have been obtained for values of  $n$  equal with 1 and 2. If one of the modules of Fig.2 is not active then the corresponding  $K$  coefficient is zero. For most of the instructions in an execution set  $t_{start}$  is identical and  $t_{delay}^{(i)}$  can be considered zero. Now let us consider an assembly language program that starts to

execute its first execution set at time  $t_0$ . At run time we assume that the program executes a total number of execution sets equal with  $(n+1)$  at a rate of one execution set per clock cycle. Also let us consider that  $t_{delay} = 0$  for all the instructions and the clock cycle period is  $T$ . The equation for the  $m$ 's execution set becomes:

$$\begin{aligned} RC_{es}(t) &= g(t - (t_{start} + mT); n, \lambda) \sum_{i=1}^8 k_i \\ &= K_m * RC_p^{(m)}(t) \end{aligned}$$

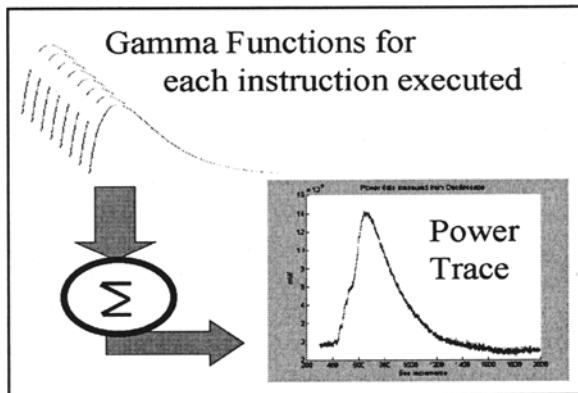


Figure 4. Gamma functions summed to model for power trace.

The general RC circuit for the above program is represented in Fig.3 , where the contact for each execution set  $ES_m$  ( $m \in \{0,1,2,\dots,n\}$ ) is closed at time  $t_0+(m-1)T$  and remains closed until the execution stops. Evaluating the total current draw for the RC circuit of Fig.3 as a sum of the individual currents per execution set, we can develop the following general formula, given below and illustrated in Fig.4, for calculating the instantaneous current draw at the software level:

$$i(t) = I_0 + \sum_{m=0}^n K_m * RC_m(t - t_0 - mT)$$

where

$$RC_p^{(m)}(t) = \begin{cases} 0; & \forall t < t_0 + mT \\ K_m * 0.621 * g(x; 1, 0.0041); & x = t - (t_0 + mT); \end{cases}$$

## 4. EXPERIMENTAL RESULTS

The experimental setup for dynamic power measurements and results of the gamma model will be outlined in this section. The models were generated using MATLAB. A number of cryptographic applications were used to illustrate the gamma model and energy dissipation accuracy was quantized. The following procedure was used to acquire the instruction dynamic power models: Dynamic current waveform measurements were performed for each individual instruction operating on a mixed data values using experimental setup shown in Fig.5[9] using synchronized clock and interrupt scheme shown in Fig.6[9]. The instantaneous current waveforms were captured for a block of 8 identical execution sets (where each execution set was the instruction under measurement, or *IUM*). The peak to peak variation of the measured current was recorded as  $P_k(IUM)$ . Using the  $P_k(IUM)$  value we generated the individual  $k_i$  coefficients for each instruction of the SC140 assembly language as  $k_i = P_k(IUM)/8$ . These coefficients were then used in the RC model for SC140 assembly language applications. The dynamic current waveforms measured and their models will be outlined next, followed by complete models and measured current for some cryptographic application programs.

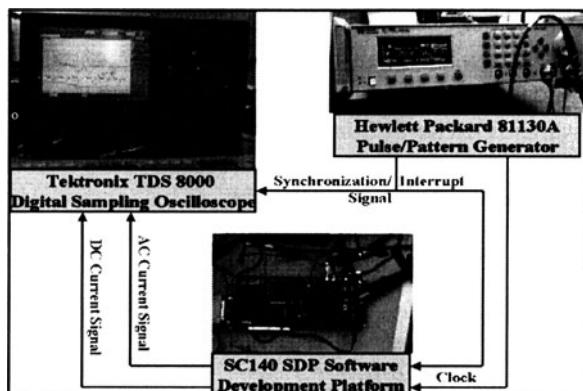


Figure 5. Experimental Set up.

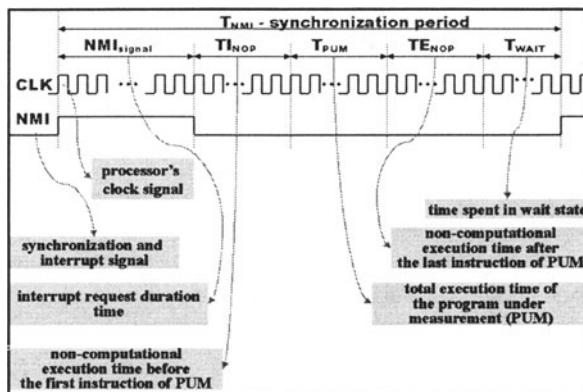


Figure 6. Synchronization of clock and interrupts

Fig. 7 shows three waveforms representing from highest to lowest amplitudes the three instructions: *CLR Dn* (sets a register to zero), *DOSETUPn label* (sets up for a loop), and *DOENn #u6* (loads control register with a number). The measured power trace (thick 'noisy' waveform) is superimposed with the gamma function (thinner line waveform, specifically  $RC_p^{(m)}(t)$ ) in the figure for each instruction.

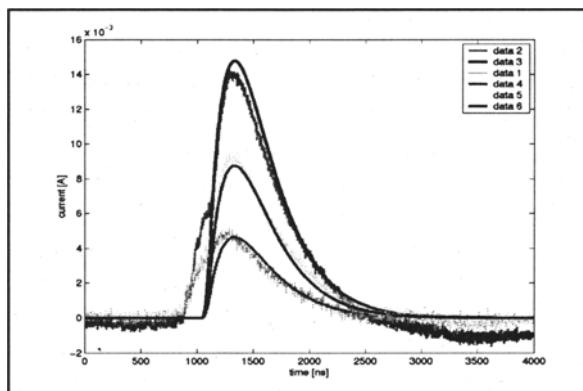


Figure 7. Power traces and gamma functions for three instructions.

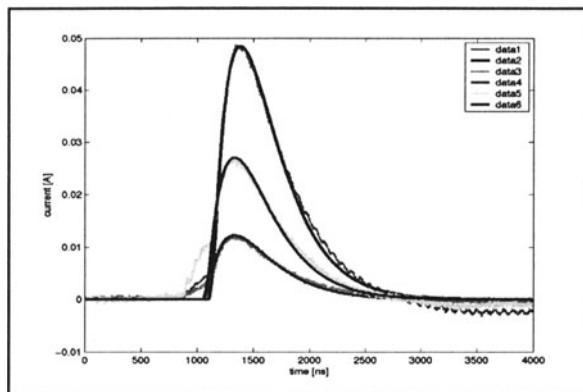


Figure 8. Power traces and gamma functions for load and exclusive or instructions.

Figure 8 shows the measured power traces and superimposed gamma models for 3 separate types of instructions: *MOVE.2L (EA),Da:Db*, *MOVE.L #s32,C4*, and *EOR Da,Dn* in order of highest to lowest amplitudes. The first instruction loads two 32bit words into two data registers. The second instruction loads an absolute 32bit value into a control register. The lowest current draw was obtained from the exclusive or on two registers. For plotting purposes in Fig. 8 the amplitudes for the exclusive or instruction (*EOR*) were multiplied by a factor of 2.

Fig. 9 presents the measured current and superimposed gamma model for a SC140 DSP cryptography application which runs for 160 clock cycles. The correlation of the measured current and the gamma model is 0.989 indicating the model is very good. The estimated energy of the gamma model has an error of 7%. Fig. 10 presents the superimposed waveforms for the same functional program but with a current variation generated by inserting a block of 40 NOPs (or no operation instructions), one per execution set. The purpose of this insertion was to verify that the gamma model could reproduce the current variation generated by the low current consumption instructions (NOPs). Both Fig. 9 and Fig. 10 were created using MATLAB. The clock frequency of the SC140 processor during the measurements was stable at 100MHz. As it can be seen the approximate current waveform is very close to the real current waveform and the current variation is faithfully reproduced by the model.

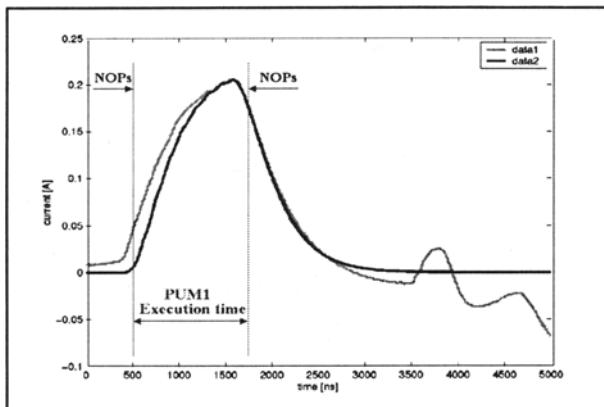


Figure 9. Program 1 power trace versus model.

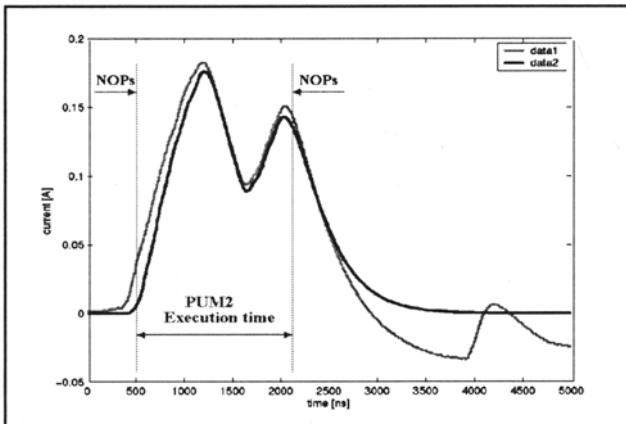


Figure 10. Program 2 power trace versus model.

## 5. DISCUSSION AND CONCLUSIONS

This study presents for the first time an instruction level model for dynamic power simulation of a complex VLIW DSP processor core. Unlike previous research, power traces have been verified with real hardware VLSI chip power measurements. Energy estimates are accurate to 7%. This working modeling technique for the SC140 DSP processor can in general be applied to many other DSP processors. The importance of this modeling technique is that it can provide instantaneous current, power or energy information at the

software level without performing an actual power measurement for the application. As a result, analysis of power dynamics is supported with optimization goals or with goals of preventing cryptographic power attacks, thus supporting a new dimension to optimization, namely security. This research is crucial for supporting a methodology for designing software that is not only optimized for performance, power and cost, but also optimized for security, and supporting algorithmic modifications such that the desired current waveform shape or amplitude is generated. The  $k_i$  coefficients can be calculated one time for the whole processor's instruction set. These coefficients can be later used for analyzing the power consumption of any application regardless of the program's memory storage location, data handled by the individual instructions or the registers used by the individual instructions. Future research intends to apply these modeling techniques to develop optimized and secure communication applications targeting DSP processors. This research was supported in part by grants from NSERC, Motorola, and CITO.

## 6. REFERENCES

- [1] Alliance Group, Motorola and Lucent Technologies, SC140 DSP Core Reference Manual, <http://www.starcore-dsp.com/docs>, 1999.
- [2] C.H.Gebotys, "A Network Flow Approach to Memory Bandwidth Utilization in Embedded DSP core Processors", to appear IEEE Transactions on VLSI, 2002.
- [3] A. Chandrakasan, R. Broderson, Low Power Digital CMOS Design, Kluwer, 1995.
- [4] V.Tiwari, S.Malik, A.Wolfe,"Power Analysis of Embedded Software:A First Step Towards Software Power Minimization" , IEEE Transactions on VLSI, vol2, 4, pp.437-445, Dec1994.
- [5] V.Tiwari, S.Malik, A.Wolfe, "Compilation Techniques for Low Energy", Int'l Symp. On Low Power Electr. Des.,1994.
- [6] V.Tiwari,D.Singh,S.Rajgopal G.Mehta R.Patel F.Baez, "Reducing Power in High-performance Microprocessors",35th Design Automation Conference, June 1998.
- [7] G.Qu, N. Kawabe, K. Usami M. Potkjonak,"Function-level power estimation methodology for Microprocessors", Proc. of Design Automation Conf., 2000, pp810-813.
- [8] P.Kocher, J.Jaffe,B.Jun, "Differential power analysis", LNCS, vol 1666, Springer-Verlag, 1999, pp.388-397.
- [9] R.Muresan, C.Gebotys,"Current Consumption Dynamics at Instruction and Program Level for a VLES DSP Processor", Proc. of 14<sup>th</sup> Int'l Symp. On Sys. Level Synthesis, Montrl, Can, 2001, pp.130-135.
- [10] G. Hadley, "Introduction to Probability and Statistical Decision Theory", HOLDEN-DAY, Inc., 1967.
- [11] C.H.Gebotys, R.Gebotys, "Statistically Based Prediction of Power Dissipation for Complex Embedded DSP Processors", Microprocessors and Microsystems, Vol23, May1999, pp135-144.
- [12] R.Muresan, C.Gebotys "Dynamic Power Simulation Model for VLIW DSP Processor VLSI Cores with Secure Applications", IFIP Proceedings of 11<sup>th</sup> In'tl Conf. On VLSI SOC, Montpellier, Fr., 2001,pp67.-72.

# Power Consumption Model for the DSP OAK Processor

Guitton-Ouhamou Patricia, Belleudy Cécile, Auguin Michel

*Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis,  
Les Algorithmes batiment. Euclide, 2000, route des Lucioles-BP 121,  
06903 Sophia-Antipolis Cedex,  
belleudy@i3s.unice.fr, guitton@i3s.unice.fr, auguin@i3s.unice.fr*

**Abstract:** The remarkable growth of personal computing devices, like portable desktops, audio and video products, and wireless communication equipments require high speed processor with low power consumption. In order to reduce power consumption, it is necessary to quantify first this value for current processors. Methods working at logic level require numerous computations to evaluate consumptions of each processor's instruction [6]. For complete and complex applications, these methods are not really applicable. The novelty of our approach is to estimate the power consumption in the same time as the designer describes his application. The objective would be to define a power consumption estimator from the C language description of the application and a library of processors. In this sense, we propose, as first work, to define a power consumption estimation model at assembly level. Our estimation technique is based on an operator level model of the data path and is compared with measurements on a test board. In this paper, this technique is applied to the OAK DSP™ processor. To conclude, we propose some writing rules of the assembly code to optimise the power consumption.

**Key words :** DSP processor, power consumption model.

## 1. INTRODUCTION

Previous IC designs constraints were area, performance and cost. Due to the growth of portable devices requiring greater autonomy, power becomes an important factor. In order to get power savings, it is necessary to derive first a power consumption model. Working at gate level requires many hours of simulation for small designs. Applications are generally written in C language, thus an estimator working at this level allows rapid evaluation in the

early design steps. Since debugging tools including a C-cross compiler are common for DSP processors, a first approach could be to extract a model at the assembly level. Unfortunately C compilers for DSPs are ineffective [1] due to specific architectures of DSP processors. Our approach is based on a two level estimator:

- The C program may be compiled in an intermediate representation based on the Register Transfer Language (RTL) [8]. This representation gives the list of operations and their precedences to be executed. The advantage of the RTL representation is that it is independent of the target processor. Unfortunately, this representation doesn't take into account parallel instructions and addressing modes of DSPs. So we consider the VESTIM tool [1]fig1., developed with the support of Phillips semiconductor Sophia. This tool uses the modified RTL representation by grouping operations forming parallel instructions, including addressing modes. The goal of this tool is to estimate the execution time of an application from a C description.

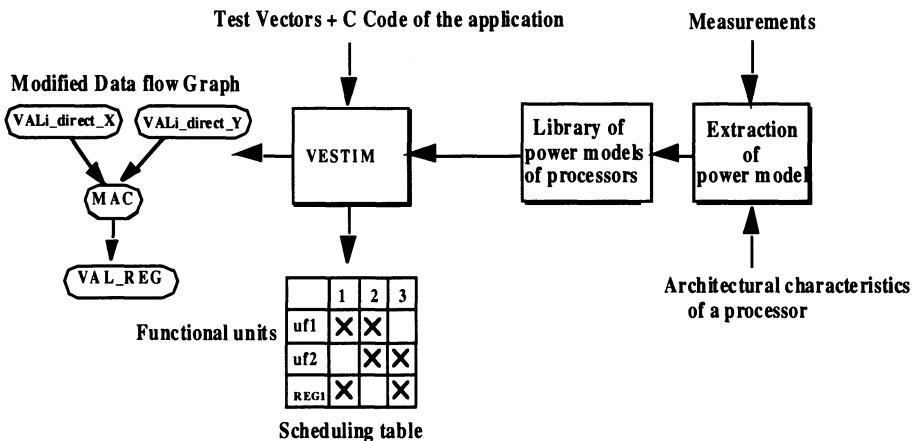


Figure 1. VESTIM tool

- From individual evaluation of power consumption of these operations and the associated addressing modes, the total power for the application can be estimated. In order to know the consumption of one operation, we consider measurements at the assembly level [11][6]. This approach is easy to implement and a power consumption model of the DSP can be deduced. Furthermore, this model could be embedded in current debugging tools, providing a power consumption profiling of the application.

In order to extract this model, we took into account micro-architectural details of the processor. Since the power cost is an increased problem in mobile designs, we choose the OAK DSP core that is embedded in numerous

designs. Furthermore, this approach (at assembly level) was considered for processors, like SPARClite, MB86934, intel i80486DX, ARM7TDMI [6][11]. As detailed in the sequel, models for RISC cannot be applied for DSPs which are data dependent.

The paper is organised as follows. First, we give the architecture characteristics of the OAK DSP processor and we describe the power measurement method. We then present preliminary results and the consumption model. The consumption due to the base cost of one instruction and the inter-instruction cost due to data path changes between two consecutive instructions. Results for test applications (FFT, FIR) are given to evaluate the accuracy of the estimator. We illustrate that writing rules of the assembly code permit a significant reduction of the power consumption. We conclude with future work.

## 2. THE OAK DSP PROCESSOR AND ITS POWER MEASUREMENTS

The OAK DSP processor [12] is a CMOS 16 bits fixed point DSP processor (figure 2).

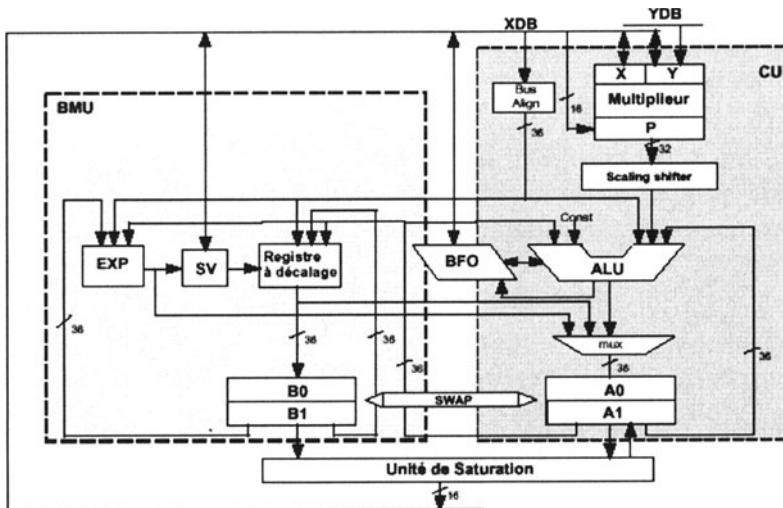


Figure 2. The data path of the OAK DSP core.

It consists briefly of the MPU, the ALU, the barrel shifter, the exponent logic unit, the data address arithmetic unit, four 36-bit accumulators, six data/address registers r0...r5 (16 bits), and two internal memories X and Y. Busses connecting the registers ri are preloaded to 1. Measurements are done

at 40 Mhz with a supply voltage of 3.3V for the board and 2.5V for the processor core.

In order to evaluate the power consumption, we use the board-based measurement method developed in [11][6]. The average power is given by  $P=I \cdot V_{dd}$  where  $I$  is the average current and  $V_{dd}$  the supply voltage. The energy  $E$  is given by:  $E=P \cdot T$  where  $T$  is the execution time of the program. The basic idea is to measure the consumed current by inserting an ammeter between the power supply and the CPU (the power supply connection to the CPU must be isolated from the remainder of the system).

Evaluation of the energy consumption of an assembly program, requires to differentiate two types of consumptions:

- the base cost of an instruction provided by the consumption of the execution of one instruction,

**Measurement principles:** for each instruction, we create a test program that contains initialisation of the system and a loop including same instruction repeated 200 times. This number has been determined so that the loop instructions does not modify the measured current value.

- the inter-instruction (or overhead) cost due to the changes in the data path activated by two successive instructions.

**Measurement principles:** The test program contains 200 pairs of instructions.

In a first approach, these measurements must be realised for each instruction, each value and each addressing mode of the operands. Since this work is too exhaustive, classes of instructions are formed. Two instructions in a class activate the same data path in the DSP. For example, operations executed by the ALU and using the same operand addressing belong to the same class.

The evaluation of the power consumption of a program is calculated by [6][11]:  $E = \sum_i (R_i \cdot N_i) + \sum_{i,j} (O_{(i,j)} \cdot N_{(i,j)}) + \sum_k E_k$

Where:  $O_{(i,j)}$  : is the overhead cost due to execution of instruction i followed by instruction j;

$R_i$  : the base cost of instruction i;

$N_i$  : number of cycles to execute instruction i.

$E_k$  represents the consumption due to lost cycles, for example, cache and pipeline stalls.

### 3. BASE COST OF A SINGLE INSTRUCTION

In [6][11], the authors approximate the base power consumption of the DSP Fujitsu 3.3V, 0.5 m, 40 MHz by an average value for each class of instructions. The range of variation inside a class is small, less than 10%.

For the OAK DSP processor, the current values inside a class can vary up to 300%. Its approximation by an average constant value would lead to make an error of 100%. It is necessary to develop a more accurate model that takes into account the architectural characteristics of the processor. This requires to identify the power consumption sources.

#### 3.1 The consumption sources of the DSP

Considering the architectural characteristics, two sources of variation were identified:

- a) **the preload of the busses:** the bits equal to 0 consume more than those equal to 1.

This preload allows a faster clock rate. Indeed, in CMOS technology forcing busses to the logical level 0 is faster than forcing them to the logical level 1.

The preload of the busses is activated only when operands use the data/address registers of the DSP.

To evaluate the effect of the preload of busses, the current values of an instruction are measured first using accumulator input operands and next using data/address register input operands. The difference between these two values represents the consumption due to the preload bus activity. We notice that in the worst case, the current can **double** when data/address register operands are used. The preload of busses is an important factor of consumption for this processor.

In this study, we consider different numbers and positions of bits equals to 0 in the input operands.

- b) **The switching activity:**

Another part of the consumption is due to the transitions of the bits, varying at each cycle from 0 to 1 and conversely.

This switching activity results from the charges and discharges of the capacities of transistors, that all put together corresponds to the switching capacity C of the circuit, giving a power consumption proportional to:  $CV^2$  where  $V_{dd}$  is the supply voltage.

Measurements lead to conclude that the consumed power of the DSP depends mainly on the switching activity of the signals. Let  $\alpha$  be the average

number of transitions per cycle. The power  $P_{\text{unit}}$  consumed by a unit of the processor with a clock  $f_{\text{clk}}$  is [3][10]:  $P_{\text{unit}} = \alpha * f_{\text{clk}} * (C * V_{\text{dd}}^2)$ .

The consumption of energy generated by the execution of an application for all the units of the processor [3], [10] is:

$$P_{\text{switching}} = \sum_{0 \text{ to } n} \alpha_i * f_{\text{clk}} * (C_i * V_{\text{dd}}^2).$$

with:

- $n$ : number of units of the processor;
- $\alpha_i$ : average number of bit transitions per cycle of the  $i$ th unit;
- $C_i$ : output capacity of the considered unit.

However the output capacity of each unit is unknown. By measuring the current consumed by an instruction, we can estimate the power induced by the switching activity by changing the number of switching bits of the input operands. The influence of the positions of the switching bits was also tested.

In the worst case, we notice that the current may be multiplied by 1.5 when all bits switch compared to the current with no bit changes (to avoid effect of the preload of busses, operands are accumulators). In order to take into account the preloaded busses and switching activity, consumption sources, the following instruction current model is:

$$I = I_{\text{cst}} + I_{\text{preloaded bus}} + I_{\text{switching activity}} \quad (1) \text{ where } I_{\text{cst}} \text{ is the current measured with no switching activity.}$$

In the next section, we give the models for common instructions. For more details refer to [4].

### 3.2 Consumption model for the instructions relative to the ALU

We refine the previous formula (1). The measurements of the current due to the preloading of busses show that all the bits have the same influence on the consumption with the exception of the sign bit. The model for the pre-loaded bus is:

$$I_{\text{preloaded bus}} = N * I_{\text{bus unit}} + I_{\text{sign extension}}$$

where:  $N$  is the number of bits equal to 0 in the input data.

$I_{\text{bus unit}}$ : elementary current (by bit) due the preload of the busses.

$I_{\text{extension sign}}$ : if the 15th bit of the input operand is equal to 0, the extension of the sign entails the following 20 bits to 0. If the 15th bit is equal to 1,  $I_{\text{sign extension}} = 0$ .

**These basic currents are measured as follows:**

- $I_{bus\ unit} = (I_{and\ r1=0,a0} - I_{and\ r1=7FFF,a0})/15$ ,  
second operand is equal to 0 so that there is no switching activity.
- $I_{extension\ sign}$ : If the 15<sup>th</sup> bit is equal to 0,  $E_{extension\ sign} = (I_{and\ r1=7FFF,a0} - I_{and\ r1=FFFF7,a0})$  Else  $I_{extension\ sign} = 0$ .
- $I_{switching\ activity} = (I_{xor\ a1=1,a0} - I_{xor\ a1=0,a0})$ .

This model is applicable for arithmetic and logic instructions executed by the ALU such as add, sub, cmp, and, or, xor.... (instructions of this class are labelled *ALU*). It concerns the switching activity of outputs of the functional units.

#### *Accuracy of the model:*

For these instructions, the estimated values of the consumption are compared with real values:

- for logic instructions (and, or, xor), the worst case reveals an error of 2.5%.
- for arithmetic instructions, the error is 6% in the worst case.

### 3.3 Consumption model for the instructions invoking the multiplier (MPU)

We consider the basic model:  $I = I_{cst} + I_{preloaded\ bus} + I_{switching\ activity}$ .

- a) **Preload of busses:** the busses are preloaded when the second operand is a data/address register. As performed for the ALU unit, we changed in the operand data the number of zeros and their position while avoiding switching activity in the multiplier by setting to zero the second operand. In this case, we observe that the number of zeros has a different impact on the power consumption according to the considered slice of 4 bits.

Table 1.

| Number of bits to 0 | $I_{preloaded\ bus\ (mA)}$ |
|---------------------|----------------------------|
| Slice 0 to 3        | I1                         |
| Slice 4 to 7        | I2 (<I1)                   |
| Slice 8 to 11       | I3 (<I2)                   |
| Slice 12 to 15      | I4 (<I3)                   |

- b) **The switching activity:** the MPU in the OAK is a Booth multiplier, thus only the second operand contributes to the consumed current.

To estimate the switching activity, we consider a pair of instructions that changes the result on the output bits of the functional unit at each cycle. This measured value is subtracted from the average value of the base costs of each instruction.

**Example:** mpy y=1,r1=0 and mpy y=1,r2=000F

At each cycle, four bits change, giving an elementary weight per bit. The method is the following:

$$I_{\text{mpy}, r1=0-\text{mpy}, r2=F} = I_{\text{cst}} + I_{\text{preloaded bus}} + I_{\text{switching activity for the 4 LSB bits}}$$

$$I_{\text{mpy}, r1=0-\text{mpy}, r2=0} = I_{\text{cst}} + I_{\text{preloaded bus}}$$

The difference between these two currents provides the switching activity for the four LSB bits. The switching activity for the other groups of 4 bits can be measured with the same approach.

We notice that a transition on the signed bit consumed 5 times the consumption due to each other bits.

This model is applicable for instructions executed by the MPU such as mpy, sqr, mac (instructions of this class are labelled *MPU*)... Notice that the mac instruction has the same model as the instruction mpy and does not result from the addition of the currents consumed by the ALU and the MPU. So parallel instructions lead to reduce the consumption.

**Accuracy of the model:** Compared to actual measurements, the maximum error is 8%.

### 3.4 The mov class

Let us apply the same formula 1.

**Preload of busses:** the preload of busses works as for the ALU. The position of the zeros in the operand has no effect on the consumed current.

**The switching activity:** the switching activity behaves as for the MPU with a different weight on the sign bit. We notice also that memory writings consume more than memory readings.

**Accuracy of the model:** the maximum error is 2.8%.

### 3.5 Impact of internal memory accesses

The consumed current analysed in the previous sections is the current consumed by the processor core. The consumption due to memory accesses are measured with an ammeter placed between the memory and the supply voltage. In the executed instructions, some memory accesses are realised.

Let  $I_{mem}$  be the base current of the memory when no accesses are performed by the core, for example, with a nop instruction. The currents generated by memory accesses are:

- memory accesses in XRAM, lead to a current 6% higher than  $I_{mem}$ .
- memory accesses in YRAM, lead to a current 14.4% higher than  $I_{mem}$ .

The difference of consumption between the XRAM and the YRAM memories is due to the interconnection in the data path of the processor. This results are obtained with a size of 8 Kbytes for the X and Y memory banks. They illustrate that internal memory accesses do not have a great influence on the total consumption. The model developed in this section is the base cost per instruction, we focus now on the inter-instruction cost.

## 4. THE INTER-INSTRUCTION COST

### 4.1 Power model of the inter-instruction cost

The inter-instruction cost (overhead) is derived from the difference between the current measured on the execution of a couple of different instructions and the sum of the currents associated with each instruction:

$$I_{ov} = I_{instr1,instr2} - (I_{instr1} + I_{instr2})/2.$$

We notice that  $I_{ov}$  has a range of variation from 1 to 42.

- For pairs of instructions belonging to the same class,  $I_{ov}$  is small, excepted for some addressing modes of the mov instruction.
- For pairs of instructions belonging to different classes,  $I_{ov}$  is significant. To take into account this behaviour we introduce two factors in the model:
  - a base current depend on the pair of the consecutive instructions:  $I_{cstov}$ ,
  - a cost dependent on the data value where the bits equal to 1 consume more than the 0s.

The inter-instruction cost is:  $I_{ov} = I_{cstov} + M*I_{unit ov}$  (2).

where: M is the number of bits equal to 1 in the operand data.

$I_{unit ov}$ : elementary current (by bit) consumed by the 1 logic in the data.

The current of  $M*I_{bus unit ov}$  is relatively small compared with the bus preload value in the base cost of instructions. If this expression is

approximated by a constant, the maximum error is 15%. So, to simplify the model, the formula (2) becomes:  $I_{ov} = I_{cstov} + I_{cst \text{ preloaded bus}}$ .

On the contrary to the base cost of instructions,  $I_{cstov}$  has a great variation according to the considered pair of instructions and the addressing modes. The value  $I_{cst \text{ preloaded bus}}$  is constant for all pairs of instructions.

**Accuracy of the model:** with this model, the maximum error is 16%.

Using this complete instruction level model, we can derive some writing rules to reduce the consumed current.

## 5. OPTIMISATION THROUGH WRITING RULES

### 5.1 Optimisation of the base cost of instructions

Addressing modes have a significant impact on the current consumed by an instruction. Therefore, it is preferable to use the following resources:

- **For the ALU :** the source operands and addressing modes that lead to reduce the power consumption are: accumulators and registers (a0, a1, p), short direct, and indirect addressing modes.
- **For the multiplication unit :** even if the model is not obvious, we can deduce the following writing rule: it is better to use the indirect addressing mode for op2 and/or for op1. As it is the number of 1 in the second operand that determines the consumption, more power swings are observed if the value with maximum of 1s is considered for this operand.
- **For mov instructions :** the current is reduced if the operands involved in the mov instructions are the accumulators a0, a1, and the register p.

### 5.2 Optimisation including the overhead

In a general approach, higher power savings are obtained for successions of instructions belonging to the same class than instructions belonging to different classes. However, these savings are achieved if successive instructions have the same addressing mode. Particularly:

- some pairs of instructions have a low power consumption: mov-mpy, mov-add.
- In contrast, it would be preferable to avoid: mpy-add.

## 6. RESULTS

On two basic DSP functions (FIR and FFT issued from the DSP library of the OAK processor), we have estimated the consumed current, by adding the base costs of instructions and the associated overheads. This value is then compared with the measurement of the real current on the test board. We obtain:

- the FIR example, the error is 5.6%,
- the FFT radix2 example, the error is 8.8%.

These results show the power consumption behaviour of the OAK DSP core is well captured in our estimation model and accurate energy consumption can be evaluated for complete program executions. By applying the writing rules described in the previous section, let us try to reduce the consumption of these two DSP functions. The order of assembly instructions is modified in order to group the instructions belonging to the same class. On the two previous examples, the FIR filter and the FFT, a power saving of 14% and 12.5% is achieved respectively with the same executing times.

## 7. CONCLUSION

The power estimation model developed in this paper fits well the power behaviour of the OAK DSP core. As introduced in [6], the model is based on a base cost of instructions and on an overhead. Major characteristics of the DSP require to extend the model in [6]: busses of the DSP are preloaded to 1 and for some instructions, the sign bit has a great impact on the switching activity. The model is accurate and gives a maximum error of 8.8%. Although the power consumption is basically evaluated at the data path level it is really an instruction level power consumption estimator. Thus, it can be integrated easily in a assembler/debugger tool allowing the design to get power consumption estimates for its application and for the code section of the application that have the highest contributions to energy.

However, integration of this model in a performance estimator like VESTIM is more challenging since real values of data are not considered in the performance extraction process of VESTIM. One solution could consist in using the Dual Bit Type model [5] to evaluate statistically the ratio of zero bits in the data in order to estimate the switching activity. To extract the

characteristics needed in the DBT, it is necessary to study statistically some test vectors. We project also to insert this model in a codesign tool (CODEF [2]) to explore hardware/software designs to optimise silicon area, power consumption and performances.

## 8. BIBLIOGRAPHY

- [1] Thesis, A. Pegatoquet, "Méthodes d'estimation de performance logicielle: application au développement rapide de code optimisé pour une classe de processeur DSP", Octobre 1999
- [2] L. Bianco, M. Auguin, G. Gresset, A. Pegatoquet, "A system prototyping tool for efficient system architecture exploration ", ICSPAT 99, Orlando, novembre 1999.
- [3] Chingwei Yeh, Min-Cheng Chang, Shih-Chieh Chang, Wen-Bone Jone,"Gate-level Design exploiting Dual supply voltages for power-driven applications", pp68-71 DAC 1999, New Orleans, LA, USA
- [4] Rapport de DEA de patricia Guitton-Ouhamou,"Modélisation de la consommation d'un processeur type DSP", université de Nice-Sophia Antipolis, juin 2000.
- [5] Paul E.Landman and Jean M. Rabaey, "Architecture power analysis: The Dual Bit Type Method" , VLSI Systems, vol.3 n.2, June 1995.
- [6] M. Tien-Chien Lee, V. Tiwari, S. Malik and M. Fujita, "Power analysis and minimization techniques for embedded DSP software", IEEE Transactions on VLSI Systems, Mars 1997.
- [7] Massoud PedramT, "High level Power modeling, estimation and optimization", Tutorial of DAC 1997.
- [8] A. Pegatoquet, M. Auguin, L. Bianco, E. gresset, "Rapid Development of optimized DSP Code from a High Level Description through Software Estimations", 36th Design Automation Conference, June 19-24, New Orleans, Louisiana, USA, 1999.
- [9] J. Rabaey and Pedram, "Low Power Design Methodologies", Kluwer Academic Publishers, Boston, MA, 1996.
- [10] Jan M. Rabaey, "System-level power estimation and optimization-challenges and perspectives", International Symposium on Low-Power Design, pp. 158-160.
- [11] V. Tiwari, S. Malik and A. Wolfe, «*Instruction level optimisation software*», Journal of VLSI Signal Processing Systems, Avril 1996, p.139 à 154.
- [12] VVF3500 DSP Core User Manual, VLSI technology, Inc.
- [13] C.Brandolese, W.Fornaciari, F.Salice, D.Sciuto, *An instruction-level Functionality-based Energy Estimation Model for 32-bits Microprocessors*. 37th Design Automation Conference June 5 - 9, 2000 Los Angeles, CA, proceedings p. 346-351.

## 9. ACKNOWLEDGEMENTS

We would like to thank Emmanuel GRESSET from Philips Semiconductors Sophia for his help to work out this project and for the fruitful discussion we had with him. This work is partially sponsored by the ADEME France and the Region PACA.

# **Integration of Robustness in the Design of a Cell**

**J.M. Dutertre, F.M. Roche, G. Cathebras**

*LIRMM, Université Montpellier II, 161 rue Ada, 34392 Montpellier cedex 05, France*

**Abstract:** When exposed to an harsh environment in space, high atmosphere or even on earth, Integrated Circuits undergo soft errors. Among these events the most worrying is an electrical upset, so called Single Event Upset (SEU) evidenced in latches. We present here the circuit architecture of a new SEU hardened latch. The hardening is based on an integrated redundancy of the information and a high impedance state switching. The design prevents perturbation to propagate inside the latch and saves an uncorrupted information source for recovery mechanisms. Post layout circuit simulations are used to verify the hardness assurance of this design; we also compare it to usual techniques and report significant improvements for its use in SoC.

**Key words:** Robustness, Radiation Hardened Design, Reliability, Single Event Upset, Data Latch

## **1. INTRODUCTION**

Integrated Circuits ( ICs ) are adversely affected by environment [1]. Two main kinds of effects occur : the Total Ionizing Dose effects ( TID ) due to charge trapping in oxides which can cause threshold voltage shift and current leakage and the Single Event Effects ( SEE ) induced by a single ionized particle hitting on a sensitive area. Among these effects, the Single Event Latch-Up ( SEL ) and the Single Event Upset ( SEU ) are the most significant. The SEL is a destructive effect through the activation of a parasitic thyristor while the SEU results in a flipping of the information bits stored by latches. The SEL problem has now its solutions. We will here address on SEU effects. SEU originates many soft errors in ICs operating in

space or high atmosphere as previously reported in [2-4], but also at ground level in terrestrial computer electronics as evidenced by IBM [1] or Normand [5]. The consequences of this effect are enhanced by scaling and the emergence of blocks embedded in System-on-Chip (SoC), difficult to test. As a consequence the robustness of functional blocks is becoming a prior concern in integrated architectures, making of circuit hardening to the environmental constraints a new challenge.

Our approach puts aside special technological solutions. It consists in the implementation of rad-hard designs at the circuit level using available commercial technology. The process is unvaried but the design solutions are easily transferable.

In the following, we shall first examine the phenomenon originating the upset and its consequence on the data latch. Then the new tolerant structure is presented with the principles involved in hardening and its efficiency is evidenced. We conclude by a comparison using an equivalent redundant structure to analyze the performances as far as speed and area are concerned.

## **2. SEU EFFECTS ON DATA LATCHES**

### **2.1 Upset phenomenon at transistor level**

We will first briefly describe the Single Event Upset mechanisms and present the way to simulate its effects on CMOS. More details will be found in [6-8].

Three basic concepts explain SEU : energy loss, charge collection and transient upset. When an energetic ion passes through any material it loses energy through interaction with the semiconductor bound electrons. It creates a dense track of hole-electron pairs, which may recombine with no effect. But in the presence of electric fields, typically in reverse biased P-N junction, electrons and holes are drifted in opposite directions and collected at device junctions. The main consequence is a transient current pulse. Such an upset drives the drain voltage of PMOS or NMOS respectively to the high or low level. Thus, ICs performed with CMOS technology, have several sensitive areas localized under contact diffusions. It is necessary to check off all the SEU sensitive areas to predict SEU effects. Moreover the electric field dependence implies a variation of their location according to the logic state of the node. Summarizing, a voltage difference on a junction induces an electric field and so a sensitive area. In case of a hit on this area, contact voltage is driven to the corresponding substrate voltage ( high voltage for PMOS or low voltage for NMOS).

The physical evolution of this phenomenon is relatively complex, yet electrical simulations of the upset can be done in a simple way with a pulse current source located at the sensitive node giving a good representation of the SEU effects observed on ICs [7]. The current pulse is fixed at a 10 mA amplitude, 50 ps rise and fall times and a programmable duration to represent worst case conditions. Usually designers consider an upset duration between two or three hundred picoseconds.

Fortunately one another point is the random of the phenomenon joined to a low occurrence of the hits. So, we shall assume that just one upset (a single event upset) can happen at the time on the same sensitive area, excluding the case of multiple upsets.

These rules are used to identify and simulate the SEU sensitivity of a data latch in the next section.

## 2.2 Data latches and SEU

SEU effects on latches is a classical issue. The concern is the sensitivity of a standard data latch circuit to identify its level of vulnerability. This level is high.

If we consider a standard latch in static mode ( $H=0$ ,  $Hb=1$ ), the Fig. 1 evidences the cell sensitive areas, state 1 corresponding to  $Q=G=1$  and  $Qb=0$  and state 0 to  $Q=G=0$  and  $Qb=1$ . The sensitivity of the latch is characterized by logic state dependence.

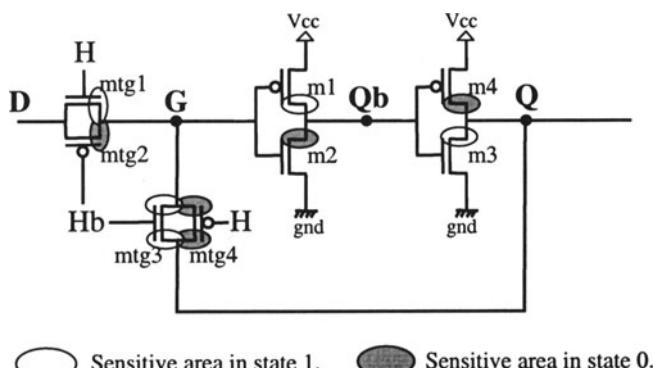
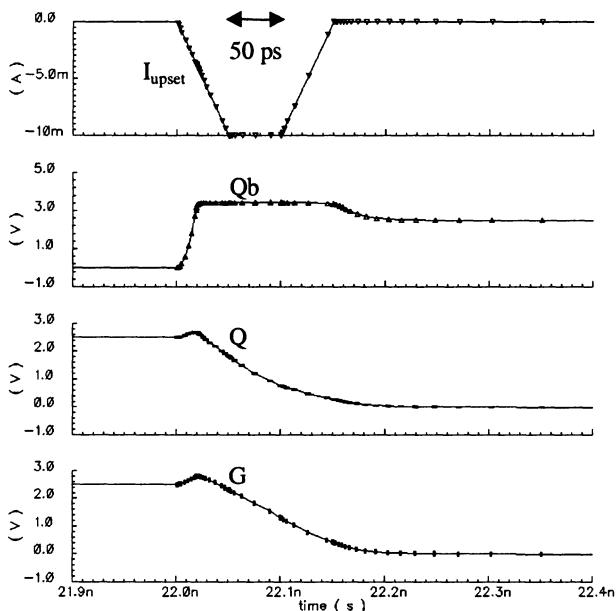


Figure 1. Latch design and its SEU sensitive areas

Simulations show, as expected, this high level of SEU sensitivity for the nodes Q, Qb and G. Figure 2 illustrates the results on node Qb with a 0.25  $\mu$ m CMOS process and a post layout description, including parasitic capacitance extraction. The SEU simulations are achieved with a 50 ps current upset on node Qb and a latch in the logic state 1.

The current upset drives node Q<sub>b</sub> voltage to high level, turning the second inverter output Q and as a consequence G to low level. Thus, Q<sub>b</sub> is forced to high level through the first inverter, even after the current pulse has ceased. Feedback dynamics of data latches provoke a permanent bit flip in case of a transient upset.

Regarding the huge amount of data cells of the majority of ICs, designers in charge of radiation hardness have to cope with the SEU issue. We propose in the next section a way to improve the tolerance to the upset by revising the cell design.



*Figure 2. Upset on node Q<sub>b</sub>*

### 3. THE SEU HARDENED HZLATCH

A few hardened CMOS inverters and cells have been presented in the past few years [9-14]. Some are limited by their lack of robustness to the longest upsets, others present prohibitive areas. Our approach consists on providing SEU immunity by restructuring the data latch at the gate level.

### 3.1 The HZ inverter design

To attain SEU hardening assurance, we conceive a new inverter that we use to complete latches. This inverter is itself hardened against SEU through the use of two techniques: an information redundancy and a tristate double-output ( high impedance state gives the inverter name : HZ inverter ). Figure 3 details its design and Table 1 its truth table.

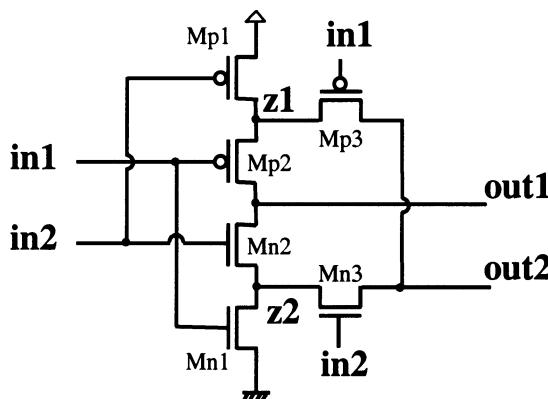


Figure 3. HZ inverter design

| In 1 | In 2 | Out 1 | Out 2 |
|------|------|-------|-------|
| 0    | 0    | 1     | 1     |
| 0    | 1    | HZ    | HZ    |
| 1    | 0    | HZ    | HZ    |
| 1    | 1    | 0     | 0     |

Table 1. HZ inverter truth table

When the two inputs ( In1 and In2 ) are settled in the same logic state the circuit works like an inverter, with outputs at the same logical level ( Out1=Out2 ) a complementary one of the inputs, as the truth table shows. Now, if an error occurs above in the IC and inverts the logic state of one of the inputs, the outputs are tristating ; thus their logic states remain uncorrupted and the error propagation is stopped.

We take advantage of these properties in the new proposed SEU hardened HZlatch.

### 3.2 The HZlatch design

The way we design the HZlatch is similar to any basic data latch design comprising two cross-coupled inverters. Figure 4 shows how to put together the basic cells.

Two HZ inverters are used in conjunction with pass gates to avoid feedback during write operation.

Redundancy and high impedance state provide both static and dynamic SEU hardening by cutting error propagation due to the feedback. A SEU is said to be static when it arises during hold state and dynamic during write operation.

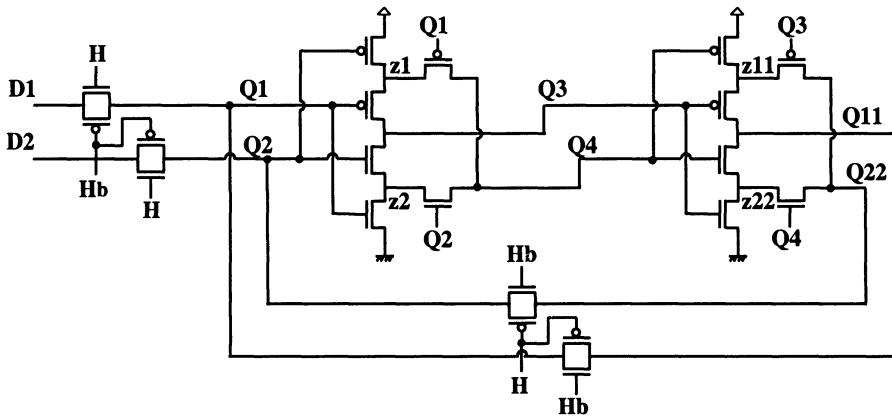


Figure 4. HZlatch design

Considering that the hardened latch of figure 4 is in hold state ( $H=1$  and  $Hb=0$ ) and stores data so that nodes Q1 and Q2 exhibit a low state. As a consequence data nodes Q3 and Q4 are established in a high state, and data nodes Q11 and Q22 are low. For example if a hit occurs at one of node Q1 sensitive area, Q1 momentarily goes high. Then the outputs of the first HZ inverter, Q3 and Q4, are tristated. Thus a source of uncorrupted data remains because Q3 and Q4 are still high. These nodes drive the second HZ inverter, which assigns low state to its outputs Q11 and Q22. A recovery mechanism of node Q1 through the cell feedback is provided, thereby hardening the latch against single particle induced upsets. Similar dynamics are at work to harden the latch when an opposite data state is initially stored and when upset occurs at any of its sensitive nodes.

Dynamic hardening mechanism is similar. After the latch stored the right data, the design prevents any false information to be rewritten. For example

any error on D1 has the same effect than any upset on Q1 ( respectively on D2 and Q2). The sole limitation is appearing at high clock frequencies. In this case an upset could last long enough to prevent a new state to be written, by tristating the first HZ inverter outputs during the whole write cycle.

Next section reports SPICE simulations for the purpose to confirm this approach.

### 3.3 HZlatch simulations

A layout of the HZlatch was completed using  $0.25\mu\text{m}$  CMOS process and SPICE simulations were conducted after parasitic capacitance extraction.

We report here response curves during upset on node Q1. The results are similar for the other nodes.

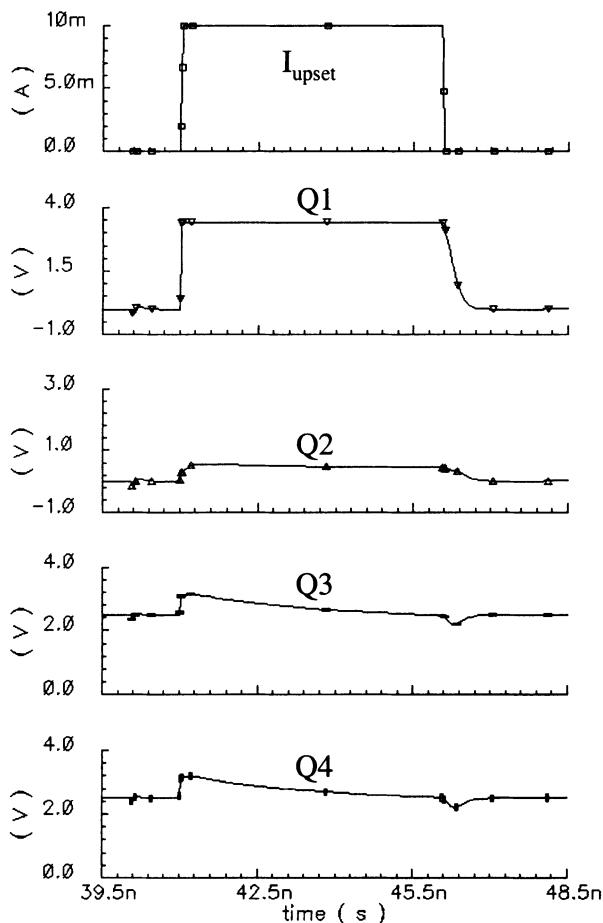
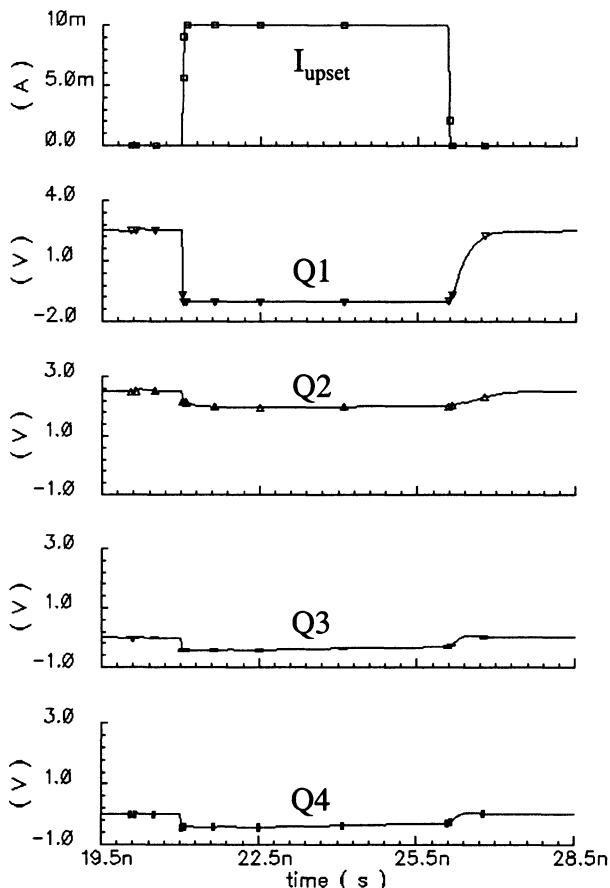


Figure 5. Upset ( 10 mA, 5000 ps ) on node Q1

The upset is simulated on node Q1, using a current pulse of 10 mA amplitude with a duration up to 5000 ps. the purpose is to force its voltage to high level. The extreme case is illustrated in Figure 6 and confirms the intervention of hardness mechanisms (part 3.2.)

Indeed node Q2 state is unchanged and due to high impedance effect nodes Q3 and Q4 hold on the high level; thus we verify that the low level of Q1 is restored as the upset current ceases, owing to recovery mechanism.

A simulation of the complementary upset on the same node Q1 ( see Fig. 7 ) confirms the hardness assurance.



*Figure 6. Upset ( 10 mA, 5000 ps ) on node Q1*

Exhaustive simulations were run on all the HZlatch nodes for a 10 mA amplitude and 5000 ps duration current; in all cases the cell was not flipped. These upsets are longer than upset durations generally reported in scientific

literature ( 1000 ps for the longest ) showing that HZ latch is safe even in utmost conditions.

#### 4. COMPARISON WITH USUAL DESIGNS

In order to evaluate the performances in terms of area and speed, the HZlatch must now be compared with an equivalent tolerant architecture based on a standard latch circuit. A general method to achieve an hardness efficiency of systems is the use of a Triple Modular Redundancy ( TMR ). Figure 8 presents the structure used : three standard latches are connected in parallel to the clock and data inputs while their outputs feed a voting gate.

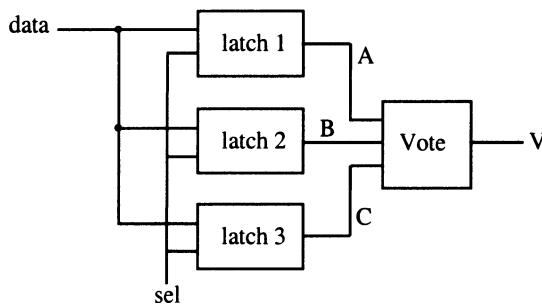


Figure 7. Tripple Modular Redundancy Structure

Any SEU arising in the latches will be corrected by the majority voting logic, implemented with a MUX structure as shown in Fig. 9.

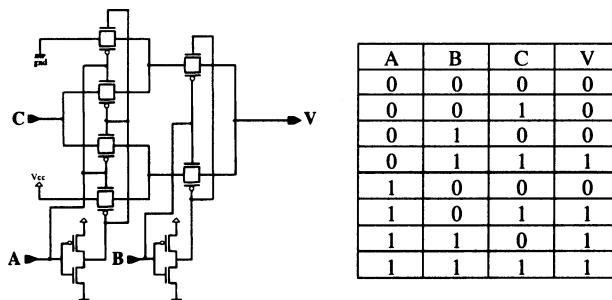


Figure 8. Majority vote module

Such a design is not totally SEU proof. Indeed an upset at the voter level is always possible, even if it will be corrected rapidly when the upset ceases. Another problem may appear if the latches are not continuously clocked; in

that case errors can "accumulate" in the TMR triplet ruining its functionality. This eventuality is not coming in sight with the HZ latch which is continuously correcting any upset.

Now we evaluate the worst delays to invert the data stored by the standard latch, the redundant circuit with propagation time through the voter and the HZ latch. These delays are estimated from mid-point of the inputs change to mid-point of the outputs voltage swing. The whole SPICE simulations are run with post layout extracted view of a 0.25  $\mu\text{m}$  CMOS process. The simulation results are listed in Table 2 with the respective cell areas.

|                          | Delay ( ps ) | Area ( $\mu\text{m}^2$ ) |
|--------------------------|--------------|--------------------------|
| Std latch                | 184          | 58.5                     |
| Vote module              | 238          | 110                      |
| <b>Redundant circuit</b> | <b>500</b>   | <b>285</b>               |
| <b>HZlatch</b>           | <b>455</b>   | <b>126</b>               |

*Table 2. Delay and area comparisons*

This comparison shows that to achieve a similar SEU hardness, the HZ latch presents a delay improvement of 10% and an area reduced by more than half compared to an equivalent TMR design. Besides the HZ latch has an improved reliability.

## 5. CONCLUSION

We have proposed a new data latch circuit ( the HZlatch ) devoted to work in an harsh environment by restructuring the design at the transistor level. The hardening method stops error propagation inside the cell by using high impedance state and provides an "integrated redundancy" in the shape of a recovery mechanism. The HZlatch can be advantageously compared to the full traditional TMR structure; this make the HZlatch a good candidate for its use in embedded systems like SoC with specific reliability constraints.

## 6. REFERENCES

- [1] J.F. Ziegler et al, "IBM experiments in soft fails in computer electronics (1978-1994)", IBM Journal of Research and Development, Vol.40, No.1, p.3-18, 1996.
- [2] D. Binder et al., "Satellite Anomalies from Galactic Cosmic Rays", *IEEE Trans. Nuc. Sci.*, December 1975, NS-22,6.
- [3] "Analysis of Environmentally Induced Spacecraft Anomalies", Journal of Spacecrafts and Rockets, Vol. 31, 2, March 1994.
- [4] P.M. O'Neil, G.D. Badhwar, "Single event upsets for space shuttle flights of new general purpose computer memories devices, *IEEE Trans. Nuc. Sci.*, NS-41,5, October 1994, p. 1755-1764.
- [5] E. Normand, "Single event upset at ground level", *IEEE Trans. Nuc. Sci.*, NS-43,6, p. 2742-2750, December 1996.
- [6] E.L. Peterson, "Single Event Analysis and Prediction", IEEE NSREC Short Course Text, 1997.
- [7] L. Massengill, "SEU Modeling and Prediction Techniques", IEEE NSREC Short Course Text, 1993.
- [8] F.W. Sexton, "Measurement of Single-Event Phenomena in Devices and ICs", IEEE NSREC Short Course Text, 1992.
- [9] J. Canaris, "An SEU Immune Logic Family", 3<sup>rd</sup> NASA Symposium on LVI Design 1991, 2.3.1-2.3.11.
- [10] D. Wiseman, J.A. Canaris, S.R. Whitaker, J. Venbrux, K. Cameron, K. Arave, L. Arave, M.N. Liu and K. Liu, "Design and Testing of SEU/SEL Immune Memory and Logic Circuits in a Commercial CMOS Process", IEEE Radiations Effects Data Workshop, Record, pp. 51-55, 1993.
- [11] R. Velazco, D. Bessot, S. Duzellier, R. Ecoffet, R. Koga, "Two CMOS memory cells suitable for the design of SEU tolerant VLSI circuits", *IEEE Transactions nuclear science*, Vol. 41, 6, 1994
- [12] L. Salager, "Conception en vue du Durcissement des Circuits Intégrés numériques aux effets radiatifs", Thèse de doctorat, Université de Montpellier,1999.
- [13] T. Monnier, F.M. Roche, G. Cathébras, "Flip-flop Hardening for Space Applications", IEEE International Workshop on Memory Technology, Design, and Testing, San Jose, CA, USA, pp104-107, 1998.
- [14] F.M. Roche, T. Monnier, A design methodology to secure memories in Space environnement, IEEE Electronic Circuits and Systems Conference, Bratislava, p. 89, 1999.

# **Impact of Technology Spreading on MEMS design Robustness**

V. Berouille<sup>1</sup>, L. Latorre<sup>1</sup>, M. Dardalhon<sup>2,3</sup>, C. Oudea<sup>3</sup>, G. Perez<sup>2</sup>,  
F. Pressecq<sup>2</sup>, P. Nouet<sup>1</sup>

*1 Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, France*

*2 Centre National d'Études Spatiales, Toulouse, France*

*3 European Aeronautic Defense and Space Company, Launch Vehicle Dpt., Paris, Franc*

**Abstract:** This paper addresses the characterization of MEMS structures, industrially manufactured using front-side bulk micro-machining post-process techniques on CMOS dies. The systematic characterization of mechanical parameters, such as stiffness or mass, on a set of 100 cantilever devices provides us with ground knowledge concerning process parameter variations. Taking into account the foundry-given process parameter spreading such as the layer thickness uncertainties characterization results are compared with simulation results obtained using Monte-Carlo analysis in standard CAD environment. This study can be considered as a first step in the development of a global monolithic MEMS design methodology.

**Key words:** MEMS, Characterization, Process Scatterings

## **1. INTRODUCTION**

An increasing interest is given to microsystems that monolithically integrate mechanical sensor parts with dedicated signal processing circuits [1,2]. The fabrication process usually relies on an industrial CMOS foundry followed by post-process releasing treatments for the fabrication of the mechanical elements. As a consequence, moving parts are made of CMOS materials such as silicon, oxides and aluminum. Since CMOS foundries neither characterize nor monitor mechanical properties of materials such as density or Young's modulus, particular design approaches have to be considered.

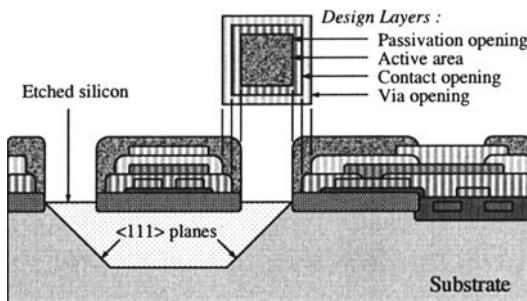
The goal of this study is to characterize the scattering of mechanical parameters for basic cantilever structures made by an industrial FSBM (Front Side Bulk Micromachining) post-process on CMOS dies. The paper is organized as follows: the first part introduces the test chip used for the experimental study. The characterization protocol is presented in the second part. The third part summarizes and comments experimental results. Part four addresses design issues. AHDL modeling of mechanical structures is first stated. Then Monte-Carlo analysis is performed on mechanical devices using the CMOS layer thickness distribution given by the foundry. The so-obtained statistical data is finally compared with experimental data.

## **2. THE TEST CHIP**

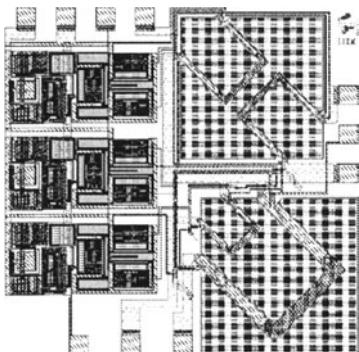
In order to get experimental access to specific parameters, test samples have been designed and fabricated. The following sections introduce the so called U-Shape cantilever test device. Fabrication approach, and basic actuation principles (static and dynamic) are therefore developed.

The Front Side Bulk Micromachining (FSBM) post-process allows the fabrication of micrometric mechanical structures using a silicon wafer issued from a standard CMOS industrial process. This MEMS technology can be easily addressed in Europe through Multi-Project Wafer services of CMP [3]. CMOS is currently a  $0.8\mu\text{m}$  or  $0.6\mu\text{m}$  standard process with two or three metal layers from Austria Mikro Systems [4] while post-process is performed by IBS [5]. Since we have been using this fabrication facility for over 6 years we were able to observe a high process repeatability level.

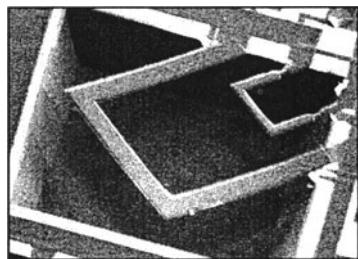
Figure 1 represents a cross-sectional schematic of a CMOS process where polysilicon, metal and oxide layers are deposited on a substrate. According to designer requirements several etching masks can be superimposed to leave the silicon bulk uncovered at the end of the standard CMOS process. Then, the post-process operates as an anisotropic silicon etching that uses the various oxide layers as self-aligned masks.  $\langle 100 \rangle$  substrate planes can then be etched leaving the  $\langle 111 \rangle$  planes of silicon substrate unaltered. In that way, suspended structures can be obtained as a heterogeneous stacking of various materials (namely Silicon Oxides, Polysilicon, Aluminum, Silicon nitrides). A set of 25 chips has been fabricated using AMS  $0.8\mu\text{m}$  CMOS technology. As shown on figure 2, each chip includes 4 U-Shaped cantilever devices (namely C,D,E1,E2) and the corresponding readout circuitry. Figure 2 shows an example of such a structure after fabrication. Some important structure dimensions are reported in table 1 for further reference.



*Figure 1.* CMOS manufacturing process.



*Figure 2.* Layout of the whole test chip.



*Figure 3.* Example of U-Shape cantilever devices used for characterization

| Device                          | Lc  | Wc  | Wb |
|---------------------------------|-----|-----|----|
| C                               | 520 | 520 | 80 |
| D                               | 210 | 200 | 40 |
| E <sub>1</sub> & E <sub>2</sub> | 320 | 320 | 40 |

*Table 1.* Test structure dimensions in micrometers.

Our characterization technique is based on the U-shape cantilever beam that offers different actuation modes. Using a CMOS aluminum metal path in association with an external magnetic field, the device can be actuated by

means of the Lorentz force (figure 4). In that case, the calibrated force can be static, using a constant electrical current, or dynamic.

In order to apply a known vertical displacement, a test probe driven by a micromanipulator is used. For each actuation mode, the device bending is electrically monitored via embedded polysilicon strain gauges and on-chip signal conditioning.

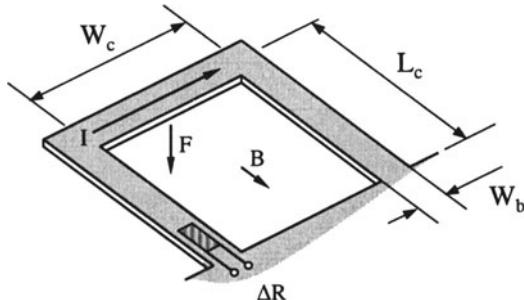


Figure 4. Magnetic actuation principle and reference dimensions.

### 3. CHARACTERIZATION PROTOCOL

#### 3.1 Static mode characterization

The purpose of static characterization is to measure the cantilever stiffness. Since the stiffness relates to the force and the displacement, the cantilever is deflected (i) by applying a known displacement by means of a micromanipulator (figure 5) and (ii) by applying a known Lorentz force (figure 6).

While using the micromanipulator, it is very difficult to precisely determine when the contact between the probe and the structure occurs. In the extraction process, only the slope of the characteristic is considered so that this offset has no effect on the result.

In both cases, a linear response of the structure is observed as expected. From the measured slopes A and S, we can calculate the stiffness k as follow:

$$k = \frac{F}{z} = \frac{A}{S} \quad (1)$$

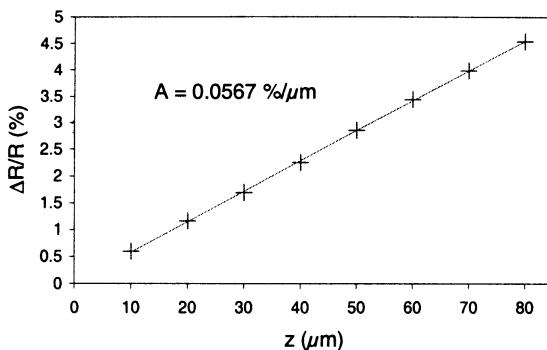


Figure 5. Static response versus displacement

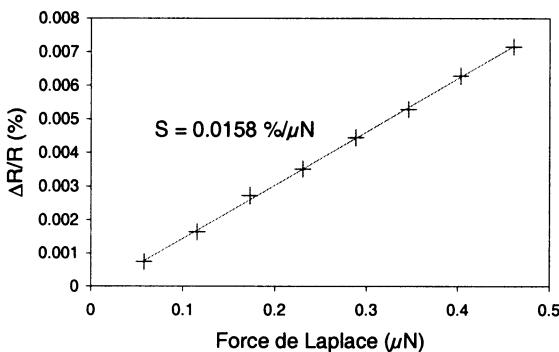


Figure 6. Static response versus force

### 3.2 Dynamic mode characterization

Some important dynamic parameters are the cantilever mass  $M$  and the damping factor  $D$ . Those parameters can be calculated from the resonant frequency and the quality factor that are easily extracted on the frequency response. The frequency response is measured using a sinusoidal force current in association with a dc calibrated magnetic field. Experimentally, on-chip signal conditioning and an oscilloscope are used to get the output signal amplitude. However, results (figure 7) are presented here in term of peak-to-peak relative resistance variation to be consistent with previous results.

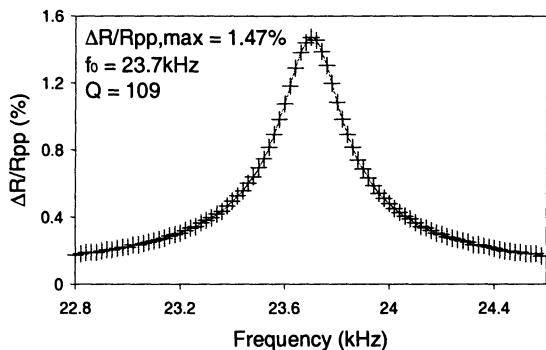


Figure 7. Frequency response

The mass can be calculated from the previously determined stiffness using the observed resonant frequency:

$$M = \frac{k}{(2\pi f_0)^2} \quad (2)$$

The quality factor is then calculated using the definition of the -3dB bandwidth:

$$Q = \frac{f_0}{\Delta f_{-3dB}} \quad (3)$$

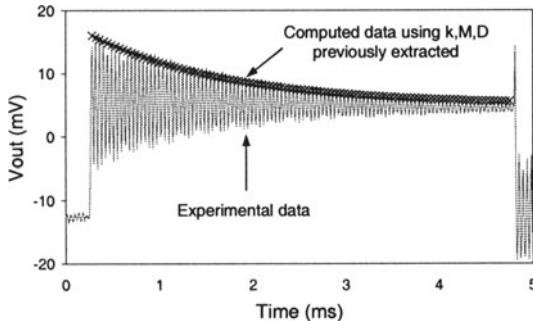
Finally, the damping factor is calculated from both  $k$ ,  $M$ ,  $f_0$  and  $Q$  as follows:

$$D = \frac{k}{2\pi f_0 Q} \quad (4)$$

### 3.3 Mechanical parameter validation

Comparing actual digitized step response of the U-Shape cantilever with the calculated overshoot envelope validates both static and dynamic parameters. For low damped second order mechanical systems, the nth order overshoot amplitude  $O_n$  is deduced from the following relation:

$$\ln \frac{O_1}{O_n} \approx (n-1) \frac{D}{2Mf_0} \quad (5)$$



*Figure 8.* Comparison between experimental step response and calculated overshoot envelope.

Taking apart the first measurement involving human control of the test probe, the whole characterization protocol has been implemented onto an automatic test bench in order to reduce experimental condition spreading. The test bench is composed of standard test equipment (oscilloscope, supplies, function generator) controlled by a workstation using a standard IEEE488 instrumentation interface bus.

#### 4. CHARACTERIZATION RESULTS

Table 2 summarizes the average value and standard deviation found for each of the extracted parameters. Since beams are made of a CMOS layer stack, reported deviations concerning Mass and Stiffness can be correlated with a known foundry uncertainty of about 20% given on each CMOS layer thickness.

For the beam stiffness, uncertainties of about 10 percents have been observed on the same fabrication run. One should keep in mind that from one run to another, standard deviations should be greater. Also, it is worth noting that standard deviation becomes generally smaller when the mechanical device size increases (the lowest deviations are obtained on the C cantilever). This result is not surprising given that it is a rule of thumb in the analog IC design field to increase dimensions in order to minimize process parameter distribution.

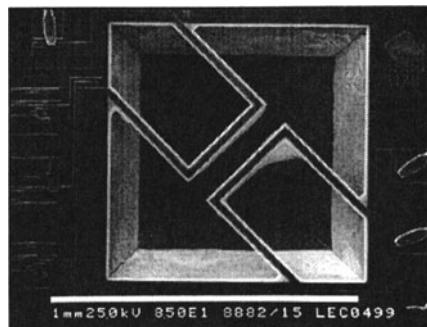
|  | <b>C</b>      |                | <b>E</b>      |                | <b>D</b>      |                |
|--|---------------|----------------|---------------|----------------|---------------|----------------|
|  | <b>Avg.</b>   | <b>Std (%)</b> | <b>Avg.</b>   | <b>Std (%)</b> | <b>Avg.</b>   | <b>Std (%)</b> |
| <i>A</i> (%/ $\mu\text{m}$ )                     | <b>0.025</b>  | 3.1            | <b>0.059</b>  | 4.5            | <b>0.161</b>  | 7.5            |
| <i>S</i> (%/ $\mu\text{N}$ )                     | <b>0.011</b>  | 7.2            | <b>0.015</b>  | 8.6            | <b>0.017</b>  | 14             |
| <i>K</i> (N.m <sup>-1</sup> )                    | <b>2.4</b>    | 10.3           | <b>3.9</b>    | 13.1           | <b>9.5</b>    | 14.1           |
| <i>f<sub>0</sub></i> (kHz)                       | <b>8.97</b>   | 2.6            | <b>23.06</b>  | 6.7            | <b>37.84</b>  | 7.3            |
| <i>Q</i>   | <b>59.07</b>  | 4.7            | <b>110.02</b> | 7.0            | <b>159.12</b> | 19.7           |
| <i>M</i> (10 <sup>-10</sup> kg)                  | <b>7.53</b>   | 6.9            | <b>1.86</b>   | 4.8            | <b>1.67</b>   | 7.7            |
| <i>D</i> (10 <sup>-7</sup> N.m.s <sup>-1</sup> ) | <b>7.19</b>   | 7.4            | <b>2.46</b>   | 10.1           | <b>2.4</b>    | 10.8           |
| <i>T<sub>Gr</sub></i> (%/ $\mu\text{N}$ )        | <b>0.0032</b> | 7.2            | <b>0.0038</b> | 8.6            | <b>0.0033</b> | 28             |

Table 2. Characterization results

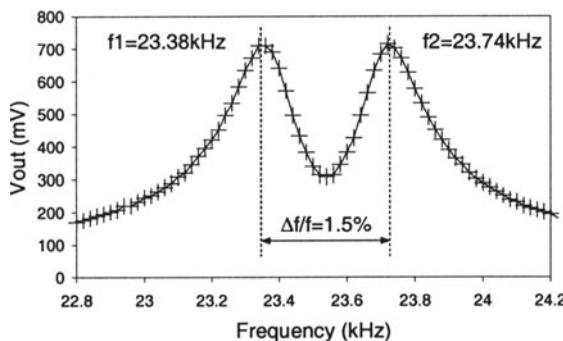
Robust design may only be achieved by taking into account parameter spreading. For illustration, an example of mechanical parameter scattering subsequent failure that has been observed with the device presented on figure 9. The matched cantilevers are designed in order to improve the device sensitivity to magnetic field by a factor two at resonant frequency. Cantilevers are then actuated with phase opposition and the four gauges are integrated into a single Wheatstone bridge for full sensitivity. In this particular non-robust design example, the expected improvement has never been reached on a set of 15 devices due to systematic mechanical parameter mismatch between the two structures. On figure 9, residual oxide on the corner of the lower U-Shape cantilever is undoubtedly the cause of a mass mismatch that produced the frequency response shown on figure 10.

## 5. SIMULATIONS

The way to improve design robustness is to take into account the process parameters variation at the design level. In the microelectronic design environment, this is achieved by performing Monte Carlo analysis, using the parameter distributions given by the foundry. Unfortunately, in the case of monolithic MEMS fabricated using industrial CMOS facility, no mechanical parameter distribution has been so far characterized.



*Figure 9:* Residual oxides on matched cantilevers



*Figure 10.* Frequency response on a pair of matched cantilevers : two different resonant frequencies are identified due to mass mismatch

The only information we were able to find in the foundry process parameter datasheet is the maximum, the average, and the minimum thickness for the various CMOS layers. Based on this information, a normal distribution for each layer thickness has been assumed and calibrated by taking one third of minimum (or maximum) thickness for the standard deviation. The so-obtained statistical data has been used in Monte Carlo analysis in association with an analog HDL model of the U-Shape device. It is worth noting that a special model has been developed for this purpose [6]. This model relates low-level parameters such as material properties and dimensions to the high-level behavior of the structure. An example of Monte Carlo analysis result is shown on figure 11. The resonant frequency  $f_0$  has been computed 1000 times, taking into account the layer thickness distribution.

For C, E, and D cantilevers the same  $f_0$  standard deviation of 1.9% has been computed when deviations from 2.6% to 7.3% have been

experimentally observed (table 2). This result is not surprising since simulation does neither consider material properties distribution (e.g. density or Young's modulus) nor post-process related uncertainties (e.g. remaining oxides).

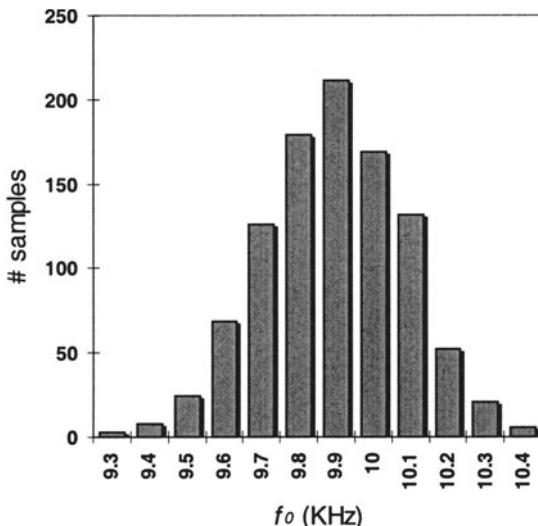


Figure 11: Monte Carlo analysis. The resonant frequency of the C cantilever has been computed 1000 times taking into account CMOS layer thickness distribution.

Realistic simulations would consider all the process parameter statistical distribution. This can be achieved by persuading the CMOS manufacturers to characterize and monitor several mechanical parameters. Obviously, such approach is not reasonable until CMOS microsystems rise significantly on the market. Another way is to use characterization data with some assumptions in order to get matched simulation results. Such approach will be developed as a further work.

## 6. CONCLUSION

In this paper, the robust design of MEMS monolithic devices based on CMOS process is addressed. In particular, a characterization methodology has been detailed in order to extract data that are not provided by traditional microelectronic manufacturer. This methodology involves a specific test structure, the U-Shape cantilever and does not require expensive equipments. Mechanical parameter such as stiffness, mass, damping factor or resonant

frequency have been characterized on a set of about 400 structures, thus providing statistical evaluation of parameters spreading. The paper finally introduces the use of Monte Carlo analysis for MEMS design improvement. So far, only the layer thickness distribution has been considered in simulations. On going work concerns the use of experimental data to calibrate Monte Carlo analysis.

## 7. REFERENCES

- [1] H. Xie, G.K. Fedder, "A CMOS-MEMS Lateral-axis Gyroscope", proc. MEMS 2001, pp. 162,165.
- [2] "Single- and dual-axis micromachined accelerometers (ADXL150, ADXL250)", Analog Dialogue Journal, Vol. 30, n°4, 1996.
- [3] Circuits Multi-projets (CMP) is a broker in ICs, MCMs and MEMS. See <http://cmp.imag.fr/>
- [4] See <http://www.ams.co.at>
- [5] Ion Beam Services, ZI Peynier Rousset, Rue Gaston Imbert Prolongée, 13790 Peynier – France – Email : ion-beam-services@wanadoo.fr
- [6] L. Latorre, P. Nouet, Y. Bertrand, P. Hazard, F. Pressecq, "Characterization and modeling of a CMOS-compatible MEMS technology", Sensors and Actuators, Vol. 74, 1999, pp. 143-147.

# A New Efficient VLSI Architecture for Full Search Block Matching Motion Estimation

Nuno Roma and Leonel Sousa

*Instituto Superior Técnico / INESC-ID, Lisboa, Portugal*

**Abstract:** A new efficient *type I* architecture for motion estimation in video sequences based on the Full-Search Block-Matching (FSBM) algorithm is proposed in this paper. This architecture presents minimum latency, maximum throughput and full utilization of the hardware resources, combining both pipelining and parallel processing techniques. The implementation of an array processor for motion estimation in a single-chip using 0.25 µm CMOS technology is presented. Experimental results show that this processor is able to estimate motion vectors in 4CIF video sequences at a rate of 16 frames/s.

**Key words:** Motion Estimation, Block Matching, Array Architectures, Specialized Processors.

## 1. INTRODUCTION

In the last few years, video coding systems have been assuming an increasingly important role in several application areas tied in with digital television, videophone and video-conference, video-surveillance and with the storage of video data. Several video compression standards have been established for these different applications [1], exploiting both spatial and temporal redundancies of video sequences to achieve the required compression rates. Among these techniques, motion-compensation has proved to be a fundamental technique to improve interframe prediction in video coding.

Motion estimation requires a huge amount of computations. Consequently, a great research effort has been made to develop efficient dedicated structures and specialized processors [6]. Due to their regular processing scheme and simple control structures, FSBM algorithms have been the most widely used in VLSI implementations, providing optimal

estimation results and leading to fast and efficient processing structures. Moreover, the sum of absolute differences (SAD) matching criteria has been extensively applied in these processors, due to its simplicity and satisfactory results.

Several different structures have been proposed over the last few years [4][9][3][2]. Most of them are 2D or 1D arrays derived from the Dependence Graph (DG) of the FSBM algorithm. However, a comparative analysis of these architectures shows that none of them provides a maximum and constant throughput or a full utilization of the hardware resources.

The main goal of the research presented in this paper is the analysis and development of a new and efficient array architecture for motion estimation in video sequences based on the FSBM algorithm. This new architecture uses the *AB2 type* architecture proposed by Vos [9] and its peculiar processing scheme as the basis for the present research. In fact, it will be shown that Vos' architecture can be significantly improved in what concerns both the latency and the hardware requirements. The amount of memory used to store the search area data can be substantially reduced through a full utilization of the hardware resources. Moreover, the time wasted to fill the processor pipeline whenever a new reference block and search area are needed can be avoided, by introducing in the architecture an extra layer with pre-fetch registers.

The proposed architecture was described using fully parameterizable IEEE-VHDL code and its functionality was thoroughly tested. Fast arithmetic units for addition and absolute difference computation were also designed based on prefix-adder and binary adder-tree structures. An integrated circuit for motion estimation was developed, by making use of the proposed architecture and using a standard cell library of a CMOS - 0.25  $\mu\text{m}$  technology process. Experimental results show that the implemented circuit is able to estimate motion vectors in 4CIF video sequences at a rate of 16 frames/s.

## 2. FSBM ARCHITECTURES

In this section, the efficiency of the main systolic architectures proposed over the last few years for the FSBM algorithm is compared. Those architectures can be regarded as regular arrays of Processor Elements (PEs), where each PE computes the SAD similarity measure. The number of PEs that composes the array defines the concurrency level of the estimation process, which is usually dependent on the used performance versus circuit area trade-off.

|   |   |
|---|---|
| $(x,y) \leftarrow (0,0)$                    | {motion vector initialization}          |
| $SAD(x,y) \leftarrow \infty$                |   |
| <b>for</b> $c = -p$ to $p$ <b>do</b>        | $\{(2p+1) \times (2p+1)$ search area}   |
| <b>for</b> $l = -p$ to $p$ <b>do</b>        |   |
| $SAD(c,l) \leftarrow 0$                     | {SAD similarity measure initialization} |
| <b>for</b> $u = 0$ to $N$ <b>do</b>         | $\{N \times N$ reference macroblock}    |
| <b>for</b> $v = 0$ to $N$ <b>do</b>         |   |
| $SAD(c,l) +=  R(u,v) - S(c+u,l+v) $         |   |
| <b>end for</b>                              |   |
| <b>end for</b>                              |   |
| <b>if</b> $SAD(c,l) < SAD(x,y)$ <b>then</b> |   |
| $(x,y) = (c,l)$ ; $SAD(x,y) = SAD(c,l)$     |   |
| <b>end if</b>                               |   |
| <b>end for</b>                              |   |
| <b>end for</b>                              |   |
| <b>return</b> $(x,y)$                       | {motion vector = $(x,y)$ }              |

Figure 0. FSBM algorithm using the SAD similarity function.

The FSBM algorithm can be described using the four nested loops presented in *Figure 0*: the inner  $(u,v)$  loops perform the matching calculation for a given candidate macroblock, while the outer  $(c,l)$  loops are responsible for the displacements inside the search area, to test all the considered candidate macroblocks. The specific characteristics of a given FSBM architecture are defined by the considered configuration and by the set of loops that are executed in parallel. Assuming, for example, that the index  $l$  in the algorithm of *Figure 0* is set to a fixed value and that each PE performs the primitive operation  $SAD$ , a 3D DG can be derived [5]. Systolic structures can be derived by applying the usual operations to project the DG and obtain array structures defined in lower dimensional spaces: index projection, time scheduling and graph folding [5]. Architectures are usually classified according to the set of projections performed, giving rise to 1D structures if multi-projection techniques are applied. Their execution time is dependent on the specific arrangement of data supply and on the number of projections performed in the retiming procedure.

One of the first discussions about FSBM architectures classification was presented by Komarek and Pirsch [4]. They discussed the characteristics of a set of 2D and 1D arrays, obtained by reducing the dimension of the original DG using traditional index projection, time scheduling and graph folding techniques [5]. The main difference between these arrays is the exploited processing concurrency, implying the usage of different structures and different number of PEs (#PE). The so called *type I - AB2* bidimensional structure requires  $\#PE=N^2$ , while the *AS2 type* array uses  $\#PE=N \times (2p+1)$ . By projecting the DG twice, one-dimensional arrays are obtained, such as the *AB1* structure, with  $\#PE=N$ , and the *AS1* structure, with  $\#PE=2p+1$ .

Vos and Stegherr [9] proposed an improved version of the *type I - AB2* two-dimensional structure, which presents some significant advantages in

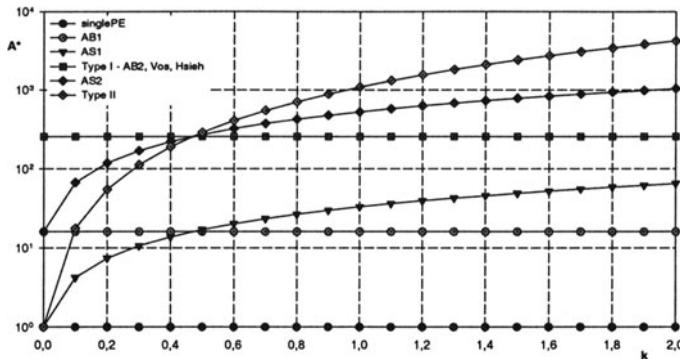
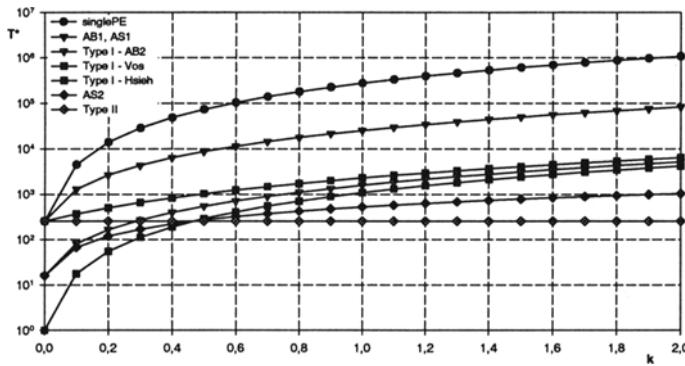
what concerns the processing time. They also proposed another structure that was obtained by reversing the processing order of the four loops of the FSBM algorithm: by indexing the inner loops with the variable pair  $(c,l)$ , all candidate macroblocks are considered whenever a different pixel of the reference macroblock is read from the frame memory at a given clock cycle. Another *type I* architecture was proposed by Hsieh [3], with some improvements in what concerns the transfer of data into the processing circuit. In his proposal, the candidate macroblock pixels are supplied as a series of one-dimensional data through a set of delay elements. Chang [2] proposed an alternative notation for the DG representation in order to improve the four loop based model: instead of nodes and links, he repeatedly allocated a two dimension  $(u,v)$  projection (slice) in the  $(c,l)$  two-dimensional space (tiling). Under certain particular conditions, Chang's model provides a hardware utilization rate very close to the optimal (100%). However, this is only possible if multiple data input lines are used.

Array processors can be classified according to their performance level, which is related to the required number of clock cycles ( $T$ ) to estimate the motion vectors. This last measure is usually the most important figure of merit used to compare architectures intended to work in real time. The values of #PE and  $T$  of the referred architectures are presented in *Table 1*. For comparison purposes, it was also considered the limit situation corresponding to a processor array with a single PE, which was designated by *SinglePE* architecture. It is worth noting that, in practice, the real values of  $T$  can be significantly greater than those presented in *Table 1*. Frequently, extra clock cycles are necessary to fill the pipeline and dummy results are often computed to preserve a regular data flow.

The circuit area ( $A^*$ ) and the processing time ( $T^*$ ) of the considered architectures were estimated by parameterizing the set of expressions presented in *Table 1* in terms of  $k = p/N$ . The obtained results are presented in *Figure 1* and *Figure 2*, respectively, using logarithmic scales to accommodate the large range of values.

*Table 1.* FSBM systolic structures.

| Architecture   | #PE                    | T                               |
|----------------|------------------------|---------------------------------|
| SinglePE       | 1                      | $N^2 \times (2p+1)^2$           |
| AB1            | $N$                    | $N \times (2p+1) \times (2p+N)$ |
| AS1            | $2p+1$                 | $N \times (2p+N) \times (2p+1)$ |
| type I - AB2   | $N \times N$           | $(2p+1) \times (2p+N)$          |
| type I - Vos   | $N \times N$           | $(2p+1)^2$                      |
| type I - Hsieh | $N \times N$           | $(2p+N)^2$                      |
| AS2            | $N \times (2p+1)$      | $N \times (2p+1)$               |
| type II        | $(2p+1) \times (2p+1)$ | $N^2$                           |

Figure 1. Circuit area ( $A^*$ ) in function of  $k=p/N$  ( $N=16$ ).Figure 2. Processing time ( $T^*$ ) in function of  $k=p/N$  ( $N=16$ ).

In *AB1* and *type I* architectures the circuit area is independent of the search window size ( $N$  and  $N^2$  processing elements, respectively), while in *AS1*, *AS2* and *type II* structures it increases significantly with the dimension of this window. Therefore, these last structures are usually advantageous for small sized search windows ( $p \leq N/2$ ), while the formers offer advantages for greater search areas. In what concerns the processing time, while for most architectures it increases with the search window size, it remains constant for the *type II* structure. This result was already expected, since one PE is used to compute the similarity measure of each candidate macroblock.

Among all these array structures, the *type I* architecture proposed by Vos and Stegherr [9] was recognized as being one of the most efficient structures [7]. Its main advantages are the short processing time and the limited amount of required hardware resources, when compared with the other bidimensional structures. However, this architecture still has some non-exploited features, which can be used to significantly improve its efficiency in terms of hardware requirements and parallelism level. In the next section, a new efficient array architecture is proposed.

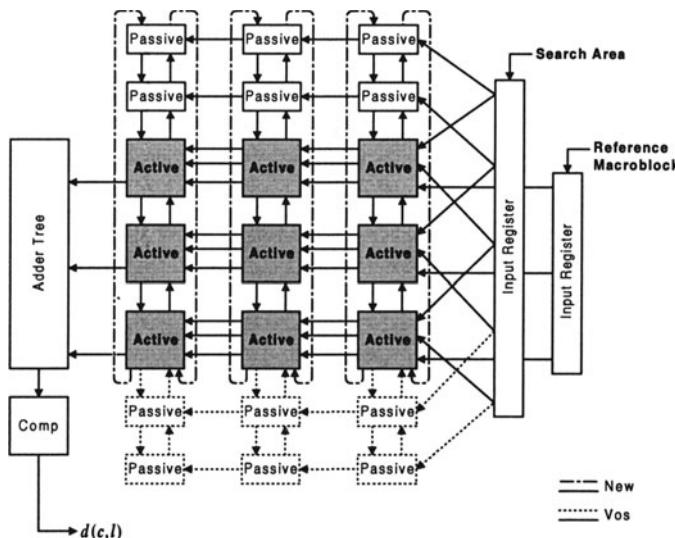
### 3. A NEW ARRAY ARCHITECTURE

The proposed architecture, designated by “*New-AB2*”, is based on *Vos* architecture but presents some significant improvements in two different aspects: *i*) processor structure; *ii*) data transfer. Due to the similarities between the processing schemes of these two architectures, the description of the proposed structure is done by contrasting its optimized characteristics with those presented by *Vos* and *Stegherr* [9]. Therefore, references to *Vos* architecture will be done whenever it shows to be convenient.

#### 3.1 Processor Structure

The diagram shown in *Figure 3* illustrates the main differences between the architecture proposed by *Vos*, represented using solid and dotted style lines (—), and the *New-AB2* architecture, represented with solid and dot-dashed style lines (—).

Like other *type I* bidimensional structures, each pixel of the reference macroblock is assigned to one of the  $N^2$  PEs that compute the SAD similarity function (designated by *active PEs*). Besides this *active block*, the processor proposed by *Vos* is also composed by two *passive blocks* with  $2p \times N$  *passive PEs*, which are appended to each side of the active block (see *Figure 3*). Each passive PE is composed by running-data registers for the displacement and storage of search area pixels. Both the reference macroblock and the search area pixels are transferred into the processor



*Figure 3.* Type I processor array for FSBM motion estimation, considering each reference macroblock with  $N \times N$  pixels ( $N = 3$ ) and the search area composed by  $(N+2p) \times (N+2p)$  pixels ( $p = 1$ ).

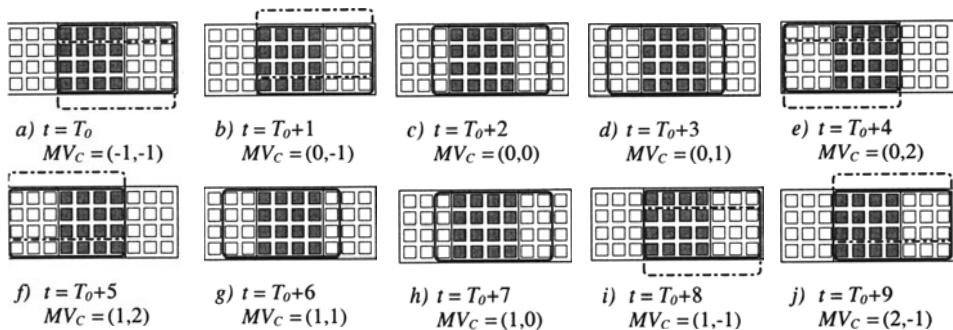
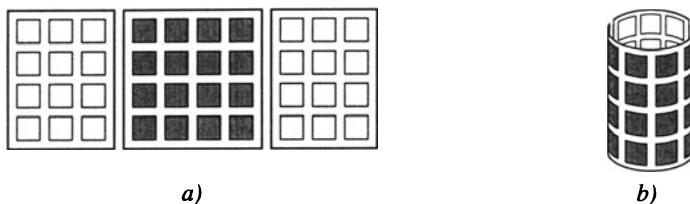


Figure 4. Zig-zag data flow of search area pixels in *Vos* architecture ( $N=4$ ,  $p=2$ ).

through two vertical input register chains, with length  $N$  and  $2p+N$ , respectively.

Within the PE array, search area pixels can be displaced in three directions: upwards, downwards and to the left. If at a given clock cycle one column with  $2p+N$  pixels of the search area is fed into the structure through the set of  $2p+N$  upper inputs, all search area pixels within the PE array are simultaneously shifted one position to the left. During the next  $2p+1$  clock cycles, search area data is shifted downwards one position per cycle. Meanwhile, the pixels corresponding to different candidate macroblocks are transferred through the several active PEs, which provide one SAD similarity value at each clock cycle. After  $2p+1$  shift-down operations, another left shift of the search area is performed and a new column of pixels is fed in the right side of the array. However, this column is now loaded through the  $2p+N$  lower inputs. This alternation of input positions in the input register chain is repeated along the search process. During the next  $2p+1$  clock cycles, search area data is shifted upwards in a similar manner as described above, being shifted to the left after  $2p+1$  clock cycles. This zig-zag processing scheme provides fast processing capabilities, preventing the need for dummy clock cycles between two adjacent lines of the search area. These extra cycles are often required by other architectures to displace search area data inside the array [4][3].

The processing scheme of *Vos* architecture can be represented in a simplified way by the sequence of states shown in Figure 4. The fraction of the search area being processed by the structure at a given clock cycle was represented using a solid-line rectangle, whereas those leaving or entering the processor were represented using a dashed-line rectangle. The bottom dashed-line rectangles represent search area fractions entering the processor in the next clock cycle, while the top dashed-line ones represent search fractions leaving the array, corresponding to the start of the search procedure in a new row of candidate macroblocks.

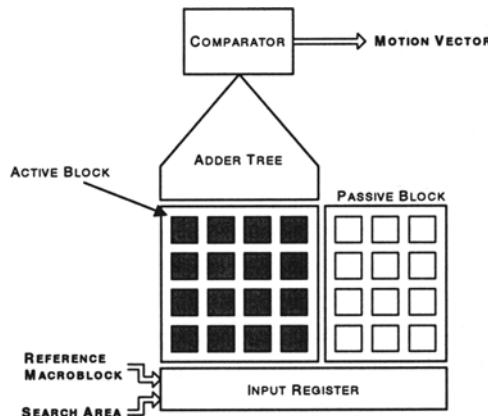


*Figure 5.* Rearrangement of the processor array: (a) - planar processor; (b) - the pair of passive blocks is superimposed by disposing the processor over a cylindrical surface.

From *Figure 4* it is possible to realize that in an array composed by  $N^2$  active PEs and by  $2 \times N(2p-1)$  passive PEs, used to process search fractions with  $N \times (N+2p-1)$  pixels, half of the total amount of passive PEs,  $N \times (2p-1)$ , are not being used. However, these passive PEs are required whenever search area pixels are displaced into their registers. The proposed solution to overcome this drawback consists in disposing the *Vos* planar structure over a cylindrical surface, as it is shown in *Figure 5*. By doing so, since the pair of passive blocks is superimposed, one can naturally discard one of them, using the other to displace the search area pixels. Nevertheless, it is worth noting that the zig-zag processing scheme can still be applied to this modified structure, preserving the properties of *Vos* architecture but keeping all PEs busy at any instant.

A simplified block diagram of the proposed *new-AB2* structure is presented in *Figure 6*. The cylindrical structure of *Figure 5(b)* is obtained by connecting the passive PEs located on the right margin of the passive block with the active PEs of the left margin of the active block, as it was shown in *Figure 3*. The processing scheme of this architecture is shown in *Figure 7* for the same setup of *Figure 4*.

Contrasting with the architecture proposed by *Vos*, this structure does not require the usage of passive PEs not carrying useful data at some clock



*Figure 6.* Simplified diagram of the proposed *new-AB2* structure.

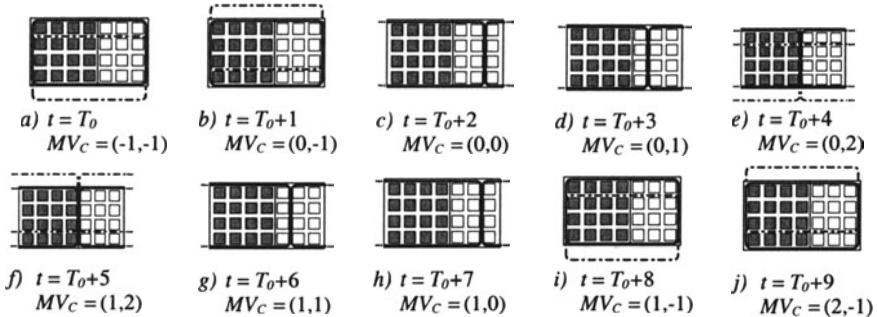


Figure 7. Zig-zag data flow of search area pixels in the proposed *new-AB2* architecture ( $N=4$ ,  $p=2$ ).

cycles. Moreover, the zig-zag processing scheme of *Vos* architecture is preserved, thus maintaining its recognized efficiency properties. However, while *Vos* architecture requires  $[N+2 \times (2p-1)] \times N$  registers, in the proposed architecture only  $[N+(2p-1)] \times N$  registers are necessary. The chart presented in Figure 8 shows the variation of the number of registers required by both structures to perform the displacement of search area pixels, by considering  $N=16$  and  $k=p/N$ . The line-chart represented with the  $\square$  marks shows the relation between the number of registers required by both architectures. This relation is about 60% for  $k=1$  ( $p=N$ ) and 55% for  $k=2$  ( $p=2N$ ).

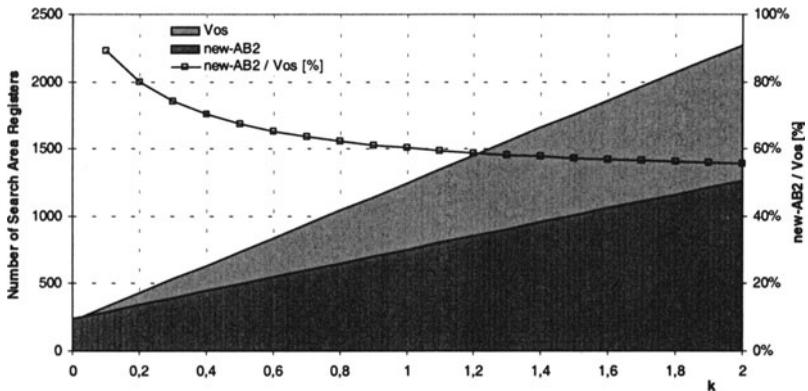


Figure 8. Relation between the required number of displacement registers in *Vos* architecture and in the *new-AB2* architecture.

### 3.2 Data Transfer

Many motion estimation architectures require extra clock cycles between the processing of adjacent lines of the search area to displace the search data inside the array [4][3]. Moreover, additional clock cycles are often spared in many architectures between the processing of consecutive reference

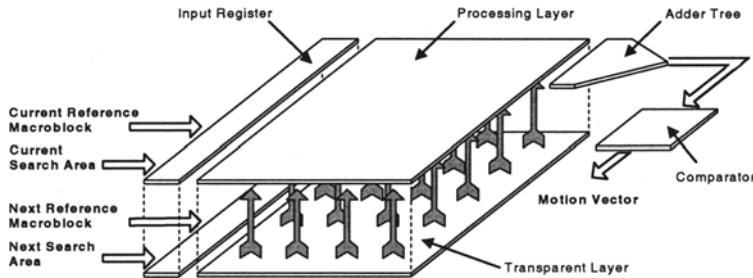


Figure 9. Prefetch layer used to load the internal registers in transparent mode.

macroblocks, to insert and remove unused or already processed data from the array [9][3]. In both situations, these extra clock cycles often lead to the loss of a significant amount of time.

Although the zig-zag processing scheme proposed by Vos provides the means to avoid the time loss associated with the displacement of search data inside the array, it does not prevent from sparing extra clock cycles between the processing of consecutive reference macroblocks. To attenuate this problem, Vos proposed the usage of running-data registers to store the pixels corresponding to the next reference macroblock while the current reference macroblock was being processed in the so called standing-data registers. When this processing is concluded, the next reference macroblock can then be instantaneously transferred from the running-data registers to the standing-data registers.

However, this transfer mechanism is not sufficient to eliminate the need for extra clock cycles, since search area data still has to be loaded into the array. To circumvent this limitation, a new data transfer method based on the usage of an additional pre-fetch layer (see *Figure 9*) is now proposed. With such a structure, it is now possible to pre-load both the reference and part of the search area data corresponding to the next reference macroblock, while the current macroblock is being processed. Data stored in the so-called transparent layer is transferred to the processing layer as soon as the last candidate macroblock is processed. Therefore, not only does this structure preload the reference macroblock data like *Vos* structure does (by using the so-called running-data registers), but it also enables a simultaneous prefetching of the search area data, making it possible to compute a new SAD similarity value in every clock cycle.

It is worth noting that this improved transfer scheme does not imply any increase of the data input bandwidth. In fact, during the processing of a given reference macroblock, it is now necessary to load  $(N+2p-1) \times (2p-1)$  pixels corresponding to the current search area,  $(N+2p-1) \times N$  pixels corresponding to the next search area and  $N^2$  pixels of the next reference macroblock, which is exactly the same amount of data that would be

required if no transparent layer was used. However, this efficiency improvement implies the usage of some more registers: with the proposed *new-AB2* structure  $2 \times [N+(2p-1)] \times N$  registers are required, while with the original *Vos* architecture  $[N+2 \times (2p-1)] \times N$  registers are used, thus leading to an increase of  $N^2$  registers. In a typical implementation, with  $N=16$  and  $k=1$  ( $p=N=16$ ), it represents an increase of only 20.5%, which can be easily tolerated if the achieved performance gains are taken into account.

## 4. EXPERIMENTAL RESULTS

The desired efficiency level of the proposed processor can only be achieved if its implementation is carried out in conjunction with a careful study of the blocks that more significantly affect its overall performance, trying to minimize its processing critical path. The most time consuming operations performed by the processor are those involved in the computation of the SAD similarity measure: addition, subtraction and absolute value computation. Consequently, gate level optimizations must be considered in order to obtain the shortest critical path as possible [8].

The description of the several blocks that compose the processor was carried out using IEEE-VHDL description language. To achieve the required characteristics in what concerns the processor configurability, this description was focused on easily obtaining fully parameterizable VHDL code, by making extensive use of 'generic' type configuration inputs. Furthermore, in order to achieve the required optimization levels of the several processing structures, a fully structural description of the main blocks of the processor was carried out by using the most elementary logic operations provided by the implementation library.

A FSBM chip based on the proposed architecture was designed with Synopsys synthesis tools and Cadence design tools, using a standard cell library based on a  $0.25\text{ }\mu\text{m}$  CMOS technology process. The implemented processor is composed by  $16 \times 16$  active PEs ( $N=16$ ), with a search range from -15 to +16 pixels ( $p=16$ ). Since the active PE is the most important module, a careful optimization procedure was carried out in terms of area and speed. The total area of the implemented chip is about  $16.07\text{ mm}^2$ , with a total pin-count of 56. The main characteristics of the processor are presented in *Table 2* and *Table 3*.

The chip is able to deliver 28.6 GOPs at 36.5 MHz over typical voltage and temperature ranges, giving rise to a total of  $1.78\text{ GOPs / mm}^2$ . In each clock cycle, each of the  $N^2$  active PEs computes one difference, one absolute value and one accumulation operation; each of the  $(2^{\log_2 N} - 1)$  adder-tree PEs computes one addition; and the comparator unit computes one comparison.

**Table 2.** Algorithm characteristics.

|                  |                  |
|------------------|------------------|
| Algorithm        | FSBM             |
| Block size*      | 16 × 16          |
| Search range*    | -15, +16         |
| Max. resolution* | 4CIF 16 frames/s |
| * - configurable |                  |

**Table 3.** Chip characteristics.

|                |                          |
|----------------|--------------------------|
| Process        | 0.25 µm CMOS – 1P5M      |
| Supply voltage | 2.5V / 3.3V              |
| Die size       | 4 × 4 mm                 |
| Active PE area | 32,184.2 µm <sup>2</sup> |
| Max. frequency | 36.5 MHz                 |
| Pin count      | 56                       |

## 5. CONCLUSIONS

A new efficient *type I* architecture for motion estimation in video sequences was proposed in this paper. This architecture presents minimum latency, maximum throughput and full utilization of the hardware resources. These optimized characteristics were achieved through the development of a new processing scheme for the processor array and through the introduction of an extra pre-fetch layer to avoid the need for extra clock cycles to transfer the data between the processor and the video coding system. Experimental results proved that this architecture can be used to implement the existing ITU-T H.26x and ISO MPEG video coding standards, with configurable search ranges and video quality tradeoffs. The implemented processor is able to estimate motion vectors in 4CIF video sequences at a rate of 16 frames/s.

## 6. REFERENCES

- [1] V. Bhaskaran and K. Konstantinides., *Image and Video Compression Standards: Algorithms and Architectures*, Kluwer Academic Publishers, 2nd edition, June 1997.
- [2] S. Chang, J. H. Hwang, and C. W. Jen., Scalable Array Architecture Design for Full Search Block Matching, *IEEE Transactions on Circuits and Systems for Video Technology*, 5(4):332-343, August. 1995.
- [3] C. H. Hsieh and T. P. Lin., VLSI Architecture for Block Matching Motion Estimation Algorithm, *IEEE Transactions on Circuits and Systems for Video Technology*, 2(2):169-175, June 1992.
- [4] T. Komarek and P. Pirsch, Array Architectures for Block Matching Algorithms, *IEEE Transactions on Circuits and Systems*, 36(10):1301-1308, October. 1989.
- [5] S. Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.
- [6] Y. Ooi, *Motion Estimation System Design*, in K. Parhi and T. Nishitani (eds.): *Digital Signal Processing for Multimedia Systems*, Marcel Dekker Inc., chap. 12: 299-327, 1999.
- [7] K. K. Parhi and T. Nishitani, editors, *Digital Signal Processing for Multimedia Systems*, Marcel Dekker, Inc., 1999.
- [8] N. Roma, L. Sousa, Implementation Aspects of MESA Processor, Technical Report, INESC-ID, RT/001/2001, January, 2001.
- [9] L. Vos and M. Stegherr, Parameterizable VLSI Architectures for the Full-Search Block-Matching Algorithm, *IEEE Transactions on Circuits and Systems*, 36(10):1309-1316, October 1989.

# **Design Considerations of a Low-Complexity, Low-Power Integer Turbo Decoder**

**Stephen M. Pisuk and Peter H. Wu**

*MIT Lincoln Laboratory, Lexington MA 02420*

**Abstract:** In this paper we present an efficient Turbo decoder implementation with minimal performance loss. Based on the non-optimal Max-Log-MAP algorithm, we carefully manipulate the algorithm, through a-priori scaling to improve BER within only 0.25dB loss of  $E_b/N_0$  of the optimal decoder. All computations are performed as integers avoiding the complexity and logic requirements of fixed and floating-point arithmetic. By combining integer computations with a design based on a single “sliding-window” decoder and a zero-delay interleaver, the final FPGA implementation achieves 1Mbps while consuming less than 600mW.

## **1. INTRODUCTION**

Turbo coding has shown great promise with near channel capacity achieved with long block sizes and numerous decoding iterations [1]. However the enormous computational complexity required for the decoder has thus far limited the application of these codes. Nevertheless, several upcoming communication systems (e.g. IMT-200 and MILSTAR Advanced EHF) have chosen to utilize the coding gain of Turbo codes.

In order to truly utilize Turbo codes, one must find an efficient decoder implementation in terms of both BER and computational resources. This is particularly true in wireless systems as well as any portable application where size and power are key limitations. The goal of reduced size and power require simplification of the decoding algorithm as well as a careful and compact hardware implementation.

This work is sponsored by the Department of the Air Force under contract F19628-00-C-0002. Opinions, interpretations, conclusions are those of the author and not necessarily endorsed by the United States Government.

The optimal Turbo decoding algorithm is Log-MAP[2]. Its complexity arises from a modification of the traditional Viterbi Add-Compare-Select (ACS) operation. In Log-Map, the Compare-Select has been replaced by  $\max^*(x,y) = \ln(e^x + e^y)$ . Through some manipulation, the  $\max^*(x,y)$  function can be expressed as  $\max^*(x,y) = \max(x,y) + \ln(1+e^{-|x-y|})$ . From this alternate form of  $\max^*$ , two forms of simplification arise. The first form of simplification is to pre-compute the natural logarithm term and store the results in a look-up table[2]. However many tables are required for the design to be robust with a range of SNR. This is less complex than the straightforward Log-Map algorithm, but still too computationally intense for a low-power design. Along the same lines is the Linear-Log-Map algorithm[3]. Here the  $\ln(1+e^{-|x-y|})$  is approximated by a linear function. Linear-Log-Map accounts for the fact that the entire term is approximately 0 when  $|x-y|$  is greater than 3 or 4. This achieves near optimal performance and reduced complexity. In order to truly simplify the Turbo decoding algorithm, we come to the second class of simplifications to the  $\max^*$  function. Algorithms such as Max-Log-MAP[2] approximate the  $\max^*(x,y)$  with  $\max(x,y)$ . Although this results in approximately 0.5dB of performance loss, the computational requirements are greatly reduced. In addition, by slightly modifying the Max-Log-MAP algorithm, we are able to gain back a portion of the 0.5dB loss while still realizing a tremendous reduction in computational complexity.

Although many fixed and floating-point decoder implementations have been presented[3][4][5][6], these implementations appear to be too complex. Although less complex than floating-point, fixed-point arithmetic still requires normalization. We constrain ourselves to 2's compliment integer representations for all calculations and metric storage. While we still must consider the dynamic range of our numbers, integer normalization does not require multiplication or division.

In addition, we have further simplified our decoding process by implementing a single decoding engine rather than a traditional two-decoder design. Although this will be further explored in Section 2.4, this simplification allows to the effectively reduce the hardware resources by a factor of two.

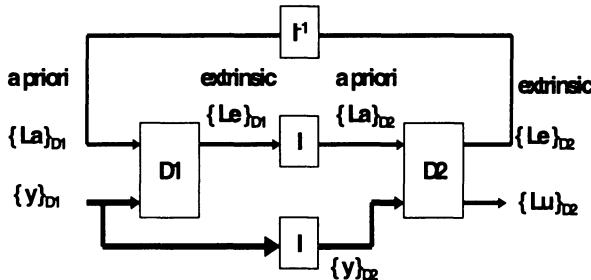
For reference, the specifications of the coding scheme presented are as follows:

- Code Rate  $\frac{1}{2}$
- Constituent code  $(23,35)$  octal
- Odd/Even Puncturing
- Block Size 640 with no trellis termination
- S=18 S-Random Interleaver

## 2. IMPLEMENTATION SPECIFICS

### 2.1 Turbo Decoder Basics

Traditional Turbo decoders utilize two decoding engines separated by a set of interleavers and de-interleavers. Each decoder inputs a set of noisy codewords  $\{y_k^1, \dots, y_k^{1/R}\}$  ( $R$  is the code-rate) as well as an a-priori value  $\{La_k\}$ . After some processing interval, the decoder produces soft-decisions,  $\{Lu_k\}$ , and extrinsic information,  $\{Le_k\}$ . The extrinsic information output from one decoder is passed through an interleaver or de-interleaver and becomes the a-priori input to the next decoder. As shown in *Figure 1*,  $\{Le\}_{D1}$ , the extrinsic output from decoder 1, is passed through an interleaver and becomes  $\{La\}_{D2}$  the a-priori input to decoder 2. Similarly,  $\{Le\}_{D2}$  is de-interleaved and becomes  $\{La\}_{D1}$  for the next decoding iteration.



*Figure 1.* Traditional Decoder Design

This process of exchanging information is what produces the coding gain present in Turbo coding. Because each decoder has a different view of the data, it can offer insight to the other decoder not otherwise possible. This exchange of information continues for a number of iterations at which point the soft decisions are limited to form the final hard decision. More generally, this process is known as iterative decoding.

For a more complete theoretical explanation of the max-Log-MAP algorithm please refer to [2] or [9].

## 2.2 Modified Max-Log-MAP

In our modified Max-Log-Map decoder, we carefully examine the computation of the branch metric. The equation is as follows:

$$c_k(s', s) = \left( \frac{1}{2} \right) \left\{ \sum_{j=1}^n y_k^{(j)} \cdot x_k^{(j)}(s', s) + La_k \cdot u_k(s', s) \right\}$$

In the first part of the equation, the incoming noisy codewords  $y_k$  are correlated against the  $j$  possible codewords,  $x_k$ , generated by the encoder. The second part of the equation represents the a-priori knowledge about this transition from all previous decoders. For the first decoder of the first iteration, the a-priori is set to 0, as we do not have any a-priori knowledge. Essentially we are combining past decoding results with the current decoding results to form a new result.

By adding a scaling constant in front of either term, we can control the strength of the current result against the past result. For example, if we were to multiply the codeword correlation by a number greater than 1, intuitively we are saying that the result from the previous decoders are less important than the current decoder's results. Similarly, if we multiply by a number less than one, we are stating that the previous decoder's result should be valued more than the current result. Hence we arrive at the following equation:

$$c_k(s', s) = \left( \frac{1}{2} \right) \left\{ \sum_{j=1}^n y_k^{(j)} \cdot x_k^{(j)}(s', s) + R \cdot La_k \cdot u_k(s', s) \right\}$$

We have added a scaling constant  $R$  to the a-priori information. As verified in [7], if SNR is constant and  $R$  is swept from 0 to 2, thus varying the importance of the a-priori information, a wide range of BER is observed.

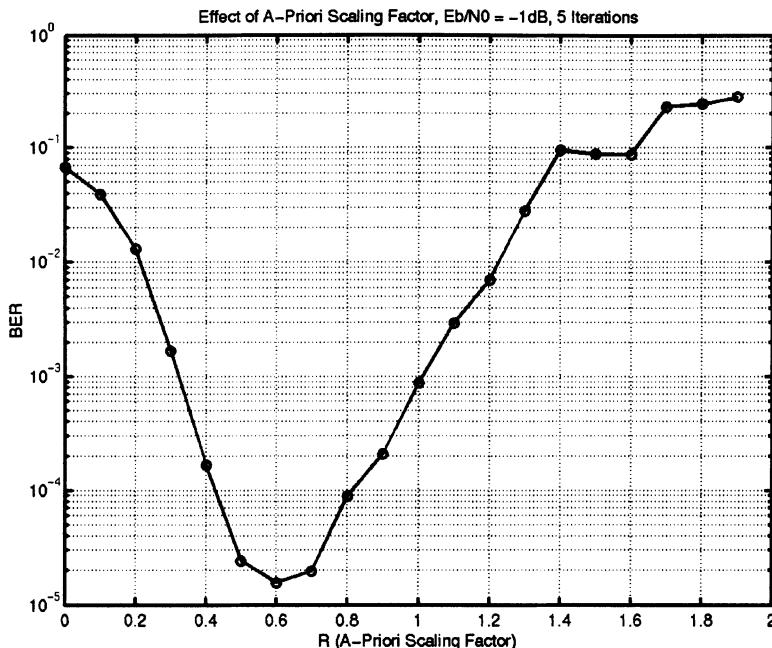


Figure 2. BER vs. A-Priori Scaling Constant

As Figure 2 shows, when  $0.45 \leq R \leq 0.6$  the BER is improved by an order of magnitude. By setting  $R = 0.5$ , we are able to gain back 0.35dB of the approximate 0.5dB loss incurred by using Max-Log-MAP instead of the optimal Log-MAP.

Although the absolute reason for the BER improvement cannot be totally explained, [7][9] suggest that  $R$  mitigates the propagation of errors from iteration to iteration. Normally a cluster of errors passed to the next decoder via extrinsic information dominates the branch metric calculations in the decoder. More generally, the codeword correlation is not strong enough to right an extreme error in the a-priori information serving to propagate errors.

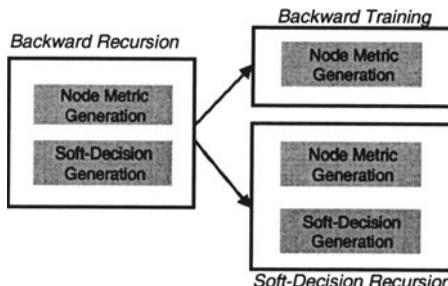
## 2.3 Sliding-Window Decoding

As explained in [1][2][9], Log-MAP decoding is broken into two main components, forward recursion and backward recursion. Forward recursion steps through the trellis from bits 0 to  $N-1$  producing node metrics representing the logarithmic probabilities that the encoder was in a given state. Backward recursion has two tasks. First it calculates node metrics like

forward recursion, but steps its way through the trellis from bit N-1 to bit 0. Second, it combines the probability of arriving at a node via forward recursion with the probability of arriving at the next node via backward recursion into a complete path metric. The complete path metrics for a bit may be combined to form a soft decision of the transmitted bit.

Unfortunately, traditional Log-MAP-based decoders require large amounts of memory because the decoder must store all forward node metrics as well as all branch metrics for use in backward recursion. For the decoder presented here: 8 bits per forward node metric, 16 node metrics per bit, 640 bits per decoder block, or 81,920 bits. For branch metrics, 6 bits per branch metric, 8 branch metrics per bit, 640 bits per decoder, or 30,720 bits. This is a prohibitively large memory requirement for a single FPGA implementation. To combat this problem, sliding-window or sub-block processing was proposed in [8].

With sliding-window processing, the backward recursion unit is split into a backward training metric generator, and a backward soft-decision generator (shown in *Figure 3*).

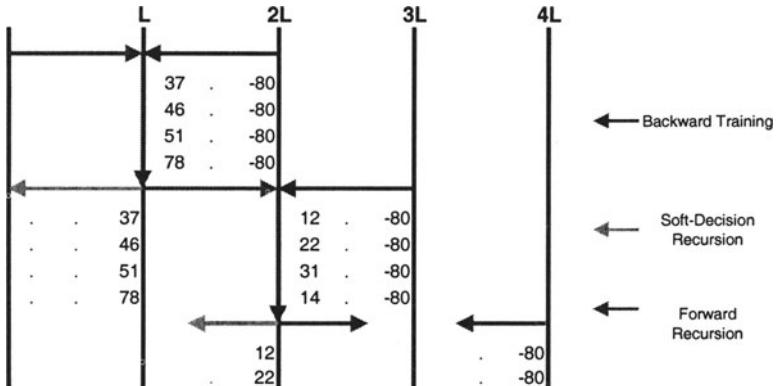


*Figure 3. Sliding-Window Modifications*

According to [8], after 5 to 6 times the constraint length, backward metrics are considered reliable. This allows us to decode the code block in segments rather than as a whole. Conveniently, the sliding-window block was chosen to be 32 bits long for constraint length 5. This result is that the decoder only needs to store 64 bits worth of forward node metrics and branch metrics rather than 640 bits worth. This reduces the memory requirements by a factor of 10.

In sliding-window processing, the backward soft-decision recursion unit is exactly the same as the backward recursion unit in traditional Log-MAP processing. The backward training metric generator is a stripped down version of the backward soft-decision generator, as it only contains the node-metric generation logic.

*Figure 4* shows a brief example of how sliding-window processing works.  $L$  is equal to 5 to 6 times the constraint length. As the forward recursion unit works from  $L$  to  $2L$ , the backward training metric generator works from  $3L$  back to  $2L$ .



*Figure 4.* Sliding-Window Processing

When the two units meet, the backward training metric generator passes its node metrics to the soft-decision generator. The soft-decision generator then continues from  $2L$  back to  $L$  producing soft-decisions while the backward training metric generator processes  $4L$  down to  $3L$ . This process is repeated until the entire decoder block is finished.

## 2.4 Single Decoding Engine

As shown in *Figure 1*, the traditional Turbo decoder utilizes two decoding engines separated by interleavers and de-interleavers. Because most Turbo codes utilize some sort of random interleavers for performance reasons, we cannot make any assumption about when bits will be ready. For example, if block or convolutional interleaver were used, we might be able to say that after D1 runs for  $N$  bits we can start D2. This is a nearly impossible assumption to make with random interleavers.

As a result, we simplify the decoder in *Figure 1* to the decoder shown in *Figure 5*.

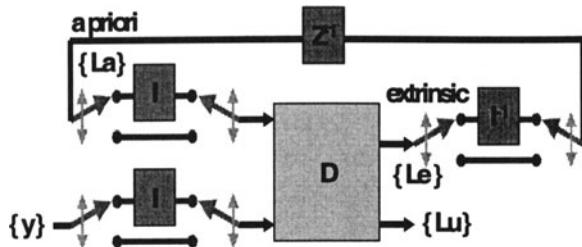


Figure 5. Single Decoder Design

Decoders D1 and D2 have been simplified into a single decoder D. The interleavers and de-interleavers remain, but multiplexers have been added to allow them to be bypassed. For example, if we are using D1, the interleaver and de-interleaver are not used so the multiplexers are set to the bypass position. However before beginning what was D2, the multiplexers are switched to pass the data into the interleavers and de-interleavers. In essence, we have traded an entire decoding engine for three multiplexers. This simplification results in an approximate 45% savings in area.

## 2.5 Zero-Delay Interleaving

Another aspect of the decoder design is its zero-delay interleaving. Turbo decoding requires interleavers on the inputs of D2 and a de-interleaver on the output of D2 in order to give D2 an alternate view of the data stream.

One way to accomplish this is to have separate interleaving and de-interleaving units that require some processing time to complete. With this approach, upon the completion of D1, its output would be interleaved and then fed into D2. Similarly when D2 was complete, its output would be de-interleaved and fed back into D1. However this approach would consume at least N additional clock cycles for interleaving and an additional N clock cycles for de-interleaving. Because additional processing time relates to increased power consumption, it is desirable to limit the time required by these functions.

The time required to interleave and de-interleave can effectively be reduced to zero if this process is performed by address manipulation. This is illustrated in *Figure 6*.

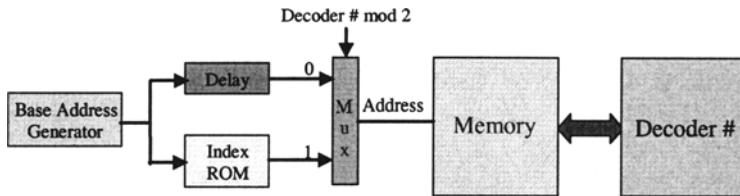


Figure 6. Address-based Interleaving

In the address manipulation approach, the base address generator simply counts from 0 to  $N-1$ , where  $N$  is the block size of the code. If we are in the non-permuted decoder,  $D_1$ , this count is simply delayed by a cycle and then used to read data out of input memory. However if we are in  $D_2$ , the permuted decoder, the count is used to read an address from an Index ROM containing the interleaver sequence. The result from the ROM is then used to read from the input memory. As for de-interleaving, we simply pass the multiplexed address to the backwards soft-decision unit where the address is used to store the result.

*Figure 7* illustrates the two interleaving schemes. In the first illustration, we see the traditional approach with distinct interleaving and de-interleaving stages. Although the implementation may be efficient, it will still require some finite processing interval. However when we examine the second illustration representing interleaving by address manipulation, we do not see any distinct interleaving blocks.

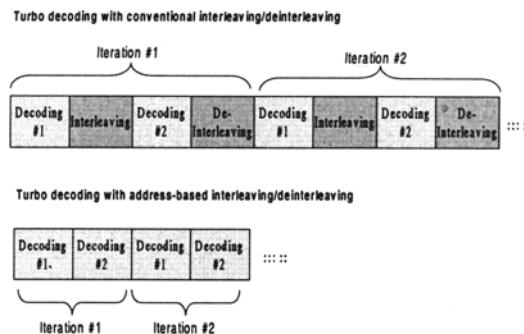


Figure 7. Various Interleaving Schemes

Because the interleaving / de-interleaving process has been combined with the normal process of retrieving data from memory, we are able to replace at least  $N$  cycles of interleaving delay by a clock cycle delay.

At first this modification may seem minor. However let us consider the case where interleaving and de-interleaving both require  $N$  clock cycles. If the decoder runs for  $I$  iterations, the traditional approach spends  $I*N$  cycles interleaving and  $I*N$  cycles de-interleaving. Whereas the address manipulation scheme spends  $2*I$  cycles interleaving and zero cycles de-interleaving. For the decoder presented here with 5 iterations, this is a savings of approximately 6,390 clock cycles.

## 2.5. Metric Normalization

As shown in [2] and [3], forward/backward node metrics accumulate as recursion progresses. Because finite precision is used, the node metrics can easily overflow or underflow. This is especially catastrophic with 2's compliment integer representations where an overflow results in a negative number and an underflow results in a positive number. The overflow/underflow problem can be solved by metric normalization. For each decoded bit in both forward and backward recursion, all node metrics are compared with a threshold  $T$ . If the absolute value of any node metric is greater than  $T$ , then all node metrics are shifted towards the center, as shown in *Figure 8*.



*Figure 8. Forward/Backward Metric Normalization*

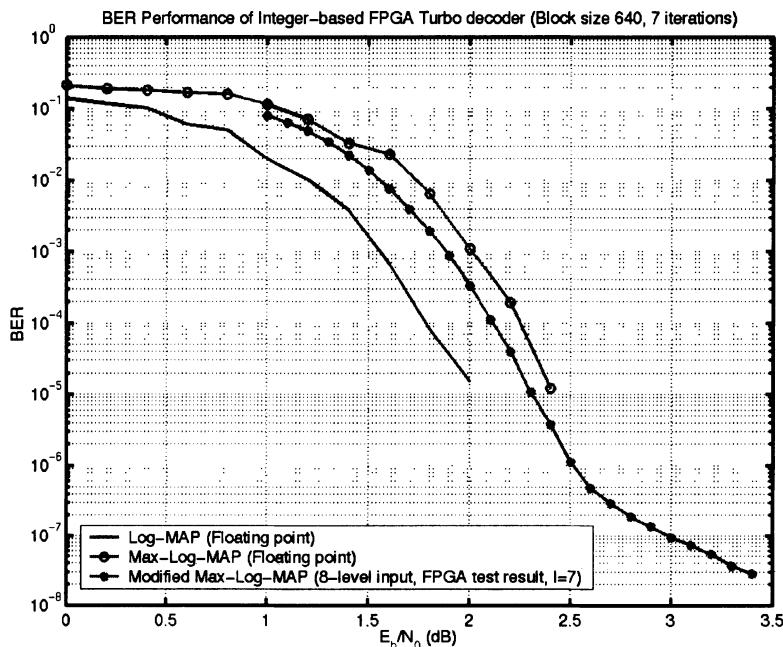
Note that the shifting is done for all node metrics so that soft-decision values [10] are not affected. The threshold value is chosen such that the update (the node metric plus a branch metric) does not cause overflow and the logic required for comparison is minimized. In our implementation, the thresholds are set to be -95 and 96. With normalization the node metrics range from -128 to 127, which can be represented by 8 bits.

## 3. RESULTS

The Turbo Decoder was designed as a Verilog RTL description and implemented on a XILINX XCV600E-7 FPGA. The design utilized approximately 2900 out of 15550 Logic Elements (LEs) and approximately 24 out of 72 RAM blocks. The interleaver ROM is included and there are no external storage or processing components required.

The design requires 1350 clock cycles per iteration with a maximum clock rate of approximately 25MHz. When run at its normal operating point of 7 decoding iterations, this equates to approximately 1.7 megabits per second. At 1Mbps, the decoder consumes less than 600mW.

*Figure 9* shows the BER performance of our all-integer FPGA Turbo decoder. Our 8-level integer-based Turbo decoder even outperforms floating point Max-Log-MAP, thanks to the a-priori scaling factor. Compared to the optimal (floating-point Log-MAP), the performance loss is less than 0.25 dB at  $\text{BER}=10^{-5}$ .



*Figure 9.* BER Performance of the integer FPGA Turbo Decoder (Block side=640, S=18 Interleaver, A-priori factor 0.5)

## 4. CONCLUSION

This paper presented the implementation of an integer Turbo decoder with minor performance loss. The efficient implementation comes from algorithm modification, integer arithmetic, and hardware management. Based on the Max-Log-MAP decoding algorithm, we modify the branch metric by weighting a-priori value, resulting in a significant BER improvement.

By performing all computations as integers, the complexity of fixed and floating-point arithmetic was avoided. By analyzing the data flow through the interleavers we were able to simplify the traditional two-decoder design to a single decoder realizing a 45% savings in area. Interleaving was accomplished through address manipulation eliminating latency and resulting in higher throughput and lower power consumption. Finally, a simple metric normalization scheme was implemented to eliminate the possibility of overflow or underflow.

In summary, FPGA design utilizes approximately 2900 XILINX Logic Cells and consumes approximately 600mW to achieve throughput of more than 1 Mbps. With 3 bits (8-level) input, our integer-based Turbo decoder outperforms floating-point Max-Log-MAP, and is only 0.25 dB away from the optimal floating-point Log-MAP.

## 5. REFERENCES

- [1] C. Berrou, A. Glavieux, P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo Codes," Proceedings of the 1993 International Conference on Communications, pp. 1064-1070, 1993.
- [2] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and suboptimal MAP decoding algorithms operating in the Log domain," Proceedings of IEEE ICC '95, pp.1009-1013, June 1995.
- [3] J.-F. Cheng and T. Ottosson, "Linearly Approximated Log-MAP Algorithms for Turbo Decoding," Proceedings of IEEE Vehicle Technology Conference 2000 Spring, pp. 2252-2256, May 2000.
- [4] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," Proceedings of IEEE Globecom '00, Nov. 2000.
- [5] G. Park, S. Yoon, C. Kang, D. Hong, "An implementation method of a Turbo-code decoder using a block-wise MAP algorithm," Proceedings of IEEE VTC '00 Fall, Sept. 2000.
- [6] A. Wiesler, O. Muller, F. Jondral, "MAP-algorithm with fixed-point representation for software radios," IEEE VTC '00 Fall, Sept. 2000.
- [7] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder ,," IEE Electronics Letters, Vol. 36, No. 23, Nov. 2000, pp.1937-1939.
- [8] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," IEEE Journal on Selected Areas in Communications, VOL. 16, NO. 2, Feb. 1998, pp.260-264.
- [9] P.H. Wu and S. Pisuk, "Implementation of a Low Complexity, Low-Power, Integer-based Turbo Decoder," Proceedings of IEEE Globecom ' 01, Nov. 2001.
- [10] G. Battail, "Ponderation des symboles decodes par l'algorithme de Viterbi," Ann., Telecommun., vol. 42, pp. 31-38, Jan./Feb. 1987.

# Low-Voltage Embedded-RAM Technology: Present and Future

Kiyoo Itoh, and Hiroyuki Mizuno

*Central Research Laboratory, Hitachi, Ltd. Kokubunji, Tokyo 185-8601, Japan*

**Abstract:** First, key issues for low-voltage (<1V) embedded RAMs are summarized in terms of stable operation, suppression of leakage (gate-tunneling/subthreshold) currents, and speed variation of memory cells and peripheral logic circuits. Next, DRAM and SRAM cells to cope with the above issues, the circuit design focusing on subthreshold-current issue, and suppression of or compensation for design-parameter variations to reduce the speed variations are discussed. Voltage converters and power management for low-power and low-voltage operation are also explained. Finally, based on the above discussions, a perspective is given with emphasis on needs for simple/high signal-to-noise ratio memory cells (such as gain cells) with a pure logic compatible process, high-speed subthreshold-current reduction focusing on active mode, and memory-rich SoC architectures.

**Key words:** subthreshold current, gate tunneling current, DRAM, SRAM, peripheral circuits, gate-source backbias, dynamic  $V_T$ , multi-static  $V_T$ , power management,  $I_{DDQ}$  test, gain cells, memory-rich architectures.

## 1. INTRODUCTION

Low-voltage embedded RAM (eRAM) is vital for low-power system-on-a-chip (SoC) technology [1]. To take advantage of device miniaturization, however, the power-supply ( $V_{DD}$ ) of eRAMs must keep up with the rapid lowering of the  $V_{DD}$  of MPUs, as discussed later. As  $V_{DD}$  makes the transition to sub- $V$  levels there are three major challenges [1]: stable operation of memory cells against reduced signal charges; suppression of the ever-increasing leakage (subthreshold and tunneling) currents of the MOSTs

in the peripheral logic circuits and memory cells; and compensation for the increased variations of speed that are seen at lower levels of  $V_{DD}$ .

In this paper, low-voltage eRAM circuits are discussed with emphasis on subthreshold-leakage current issue: First, low-voltage trends are discussed, and then recent developments to do with the above issues for DRAM and SRAM cells, and peripheral logic circuits are investigated. Other design issues for low- $V_{DD}$  operation, such as on-chip supply-voltage converters, power management, and testing methodology to cope with the subthreshold current are also described. A position is then taken that emphasizes needs for gain cells, subthreshold-current reduction circuits for use in active mode, and memory-rich SoC architectures.

## 2. CURRENT GENERATION OF LOW-VOLTAGE eRAM TECHNOLOGY

### 2.1 Trends in Gate-Oxide Thickness

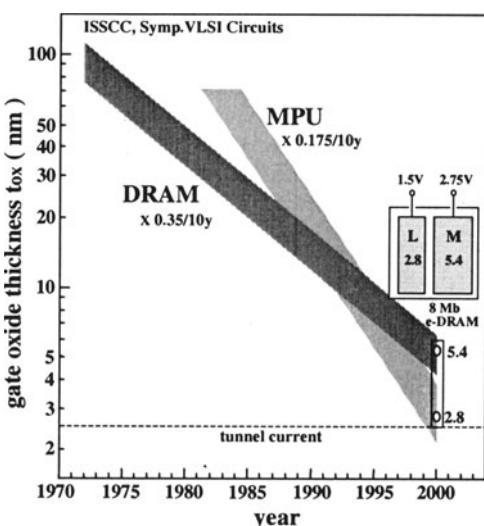


Figure 1. Trends in gate-oxide thickness for MPUs and standard DRAMs. L, M; logic core and memory array.

Device miniaturization for the core logic and embedded SRAM cache (i.e., eSRAM) of high-end MPUs has recently been accelerated by the use of an MOST structure with a thinner gate oxide ( $\text{to}_x$ ), and thus a lower  $V_{DD}$  and a lower threshold voltage ( $V_T$ ) for lower power and higher speeds (Fig. 1) [2]. Here, the dual- $V_{DD}$  and dual  $\text{to}_x$ -device approach has been maintained with an MOST that has a thicker- $\text{to}_x$  and higher- $V_T$  for the higher- $V_{DD}$  I/O circuitry. As a result, 1- $V$   $V_{DD}$  and 2-3-nm  $\text{to}_x$  have become popular for use in the core and eSRAM.

For standard (stand-alone) DRAMs, operation with a single external  $V_{DD}$  and an on-chip voltage-down converter (i.e., series regulator) has been used to realize power-supply standardization despite the internally dual- $V_{DD}$  operation. In addition, the single thick  $\text{to}_x$  (necessary for word-bootstrapping of the cells) has been used

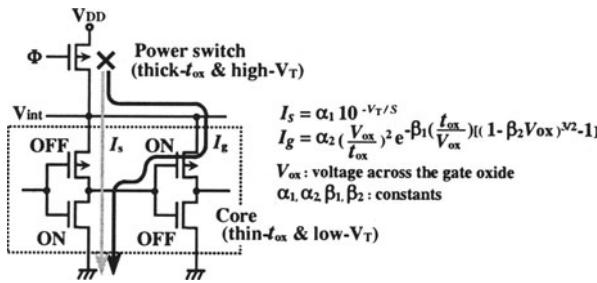


Figure 2. Two kinds of leakage currents; subthreshold current ( $I_s$ ) and gate-tunneling current ( $I_g$ ), and a power switch for the reductions.

throughout the chip for low cost. Recently, however, a dual- $V_{DD}$  and dual  $t_{ox}$ -device approach, similar to that taken with MPUs, has been adopted to achieve higher speeds for eDRAMs. One example is a recent 8-Mb eDRAM with 3.7-ns access (Fig. 1) [3].

In any event, the above three issues arise with a rapid lowering of  $V_{DD}$ . In particular, leakage-current reduction is essential for all the LSIs of the future. There are two major leakage currents; the subthreshold current and the gate-oxide tunneling current of the MOSTs (Fig. 2). A tunneling-current reduction circuit with a power switch, in which an MOST with a thick  $t_{ox}$  and high  $V_T$  is used, has recently been proposed (Fig. 2) [4]. However, continued reduction in  $t_{ox}$  will eventually come to depend on the developments of an MOST with a gate insulator of some high- $k$  material, because even the circuit mentioned above would not be able to manage the exponential increases in current that are a result of lower and lower values for  $t_{ox}$ .

## 2.2 Memory Cells and Relevant Circuits

**DRAM:** For the low-voltage one-transistor one-capacitor DRAM (1-T) cell, using vertical capacitors and high-permittivity thin-films to maintain sufficient signal charge, and adjusting the potential profile of the storage node to suppress the *pn*-leakage current are critical ways of maintaining the cell's signal voltage, soft-error immunity and retention times even as  $V_{DD}$  is lowered.

Low-voltage circuits are also important. The negative word-line (NWL) scheme (Fig. 3) [5] cuts the subthreshold current from the storage node to the data line despite the low-actual  $V_T$  with a gate-source back-bias of  $\delta$  during the non-selected period. This makes it possible to reduce the word-line voltage in a full- $V_{DD}$  write operation by  $\delta$ , which allows the use of an MOST with a thinner  $t_{ox}$  for a given device reliability. The resultant small subthreshold swing (i.e.  $S$ -factor) enables low-voltage operation. The Multi-divided data-line scheme, in which data-line capacitance ( $C_D$ ) and line delay

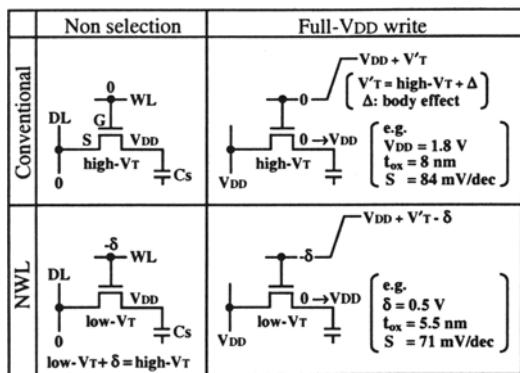


Figure 3. Conventional DRAM cell and negative word-line scheme.

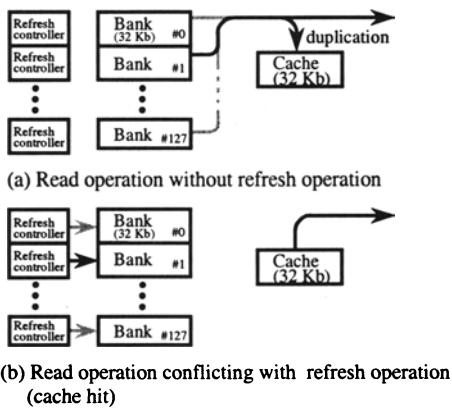


Figure 4. 1-T SRAM™ using 1-T DRAM cells and operations of multi banks.

a conflict between a normal access and a refresh operation at the same bank, a cache hit occurs while permitting a refresh cycle for that bank since all of the data in that bank have been copied to the cache.

Low-voltage high-speed sensing is also essential for the well-known mid-point sensing by which the data-line power is halved without using a dummy cell, since using the half- $V_{DD}$  data-line voltage makes the sense-amplifier (SA) operation quite slow. In the overdrive sensing scheme [7,8,9] this problem is solved by applying a higher voltage solely to the SA-inputs with isolating the data line from SA or with capacitive coupling. The use of additional capacitors may be acceptable in eDRAMs for which area is not a concern.

are reduced, is another way of obtaining attractive eDRAMs. If the storage capacitance ( $C_S$ ) is sufficient, this allows ultra-low voltage operation by maintaining cell-signal voltages ( $\sim C_S V_{DD} / 2 C_D$ ) with the help of a small  $C_D$ . Moreover, if  $V_{DD}$  is raised even a small  $C_S$  using a simple fabrication process that is essential for eDRAMs is acceptable. A good example is the so-called 1-T SRAM™ (Fig. 4) [6], in which a 1-T DRAM cell with a  $C_S$  as small as less than 10 fF is realized by a single poly-Si planar capacitor, and an extensive multi-bank scheme with 128 banks (32 Kb in each) that are simultaneously operable is used. A greater than 300-MHz row-access frequency was achieved for a 0.18- $\mu$ m 1.8-V 2-Mb eDRAM. A DRAM cache with the same capacity as a single bank is used to hide the refresh operation: Even when there is

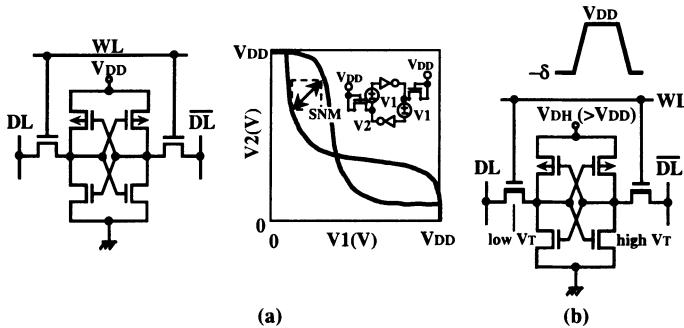


Figure 5. 6-T SRAM cell and the voltage margin (a), and improved cell (b).

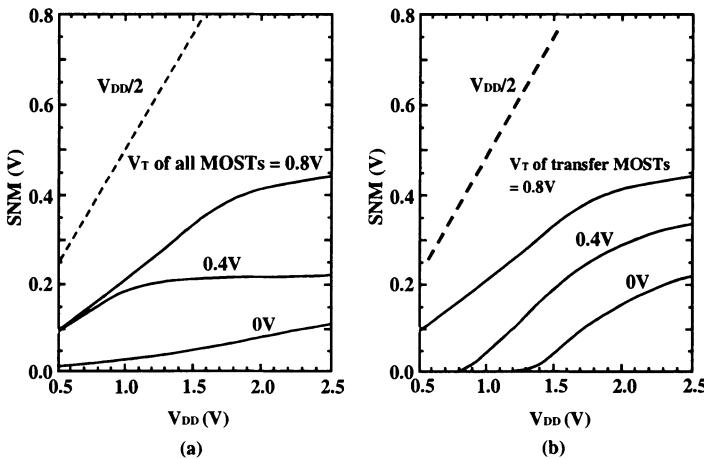


Figure 6. Static noise margin, SNM, of the 6-T SRAM cell.  $V_T$ s of all MOSTs are varied at the same time (a) while  $V_T$  of transfer MOST is varied and  $V_T$  of cross-coupled MOSTs is 0.8V (b).

**SRAM:** Of the many proposed designs [1], the 6-T full-CMOS cell is most suitable despite the fact that it is large. This is because of the simple process and the ease of design provided by the cell's wide-voltage margin. Even for this cell, subthreshold-currents must be reduced and voltage margin must be widened as  $V_T$  is lowered by lowering  $V_{DD}$ . Note that the subthreshold current for  $V_T = 0$  V would lead to a retention current of as much as 10 A in a 1-Mb SRAM array [1]. This places a strict limit on the reduction of  $V_T$ . The worsening of voltage (noise) margins as  $V_{DD}$  and  $V_T$  are lowered (Figs. 5 and 6) creates a demand for a decrease in the ratio of transconductance of the transfer MOST to that of the driver MOST. The voltage margin is also worsened by variations in  $V_T$  and mismatches of  $V_T$  between paired MOSTs, both of which increase by device miniaturization [1]. In addition, for a low- $V_T$  transfer MOST, the margin is further worsened by the data pattern of non-selected cells in the column, as shown in Fig. 7. The total of the

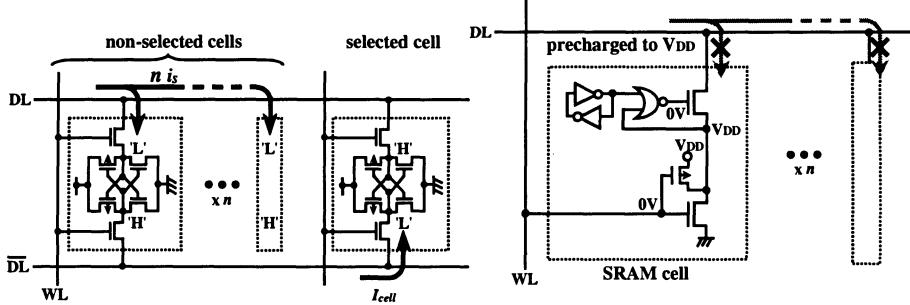


Figure 7. Leakage currents of low- $V_T$  transfer cell.

Figure 8. SRAM cell using a gate-source offset driving.

subthreshold leakage currents from the non-selected cells may be greater than the selected cell's current. This causes a read failure. A hierarchical data-line scheme provides a partial solution to this problem by limiting the number of memory cells connected to the data line [10,11]. Using a data equalizer for current compensation [12] is also effective, despite the penalty in terms of area. Applying a gate-source offset driving scheme (Fig. 8) to the memory cell allows the use of a low- $V_T$  transfer MOST for high speed and a negligible data-line current [13], although eleven transistors are required for each cell.

The loadless CMOS 4-T SRAM (Fig. 9) [14] is attractive because the cell size is only 56% of that of a conventional 6-T cell. Transfer MOSTs of the non-selected cells at  $V_{DD}$ -word line (WL) voltage work as load elements, and the load current to keep the storage node high is supplied from a data line that has been precharged to  $V_{DD}$ . The WL voltage, however, must be precisely controlled, as shown in the figure, so as to keep the load current (i.e., the off-current of the transfer MOST) to more than a hundred times that of the off current of the driver MOST. The data-pattern problem outlined above also arises in this approach.

Almost all of the above problems may be solved by using high- $V_T$  cross-coupled MOSTs combined with a boosted power supply, as shown in Fig. 5(b) [15]. High  $V_T$  for the paired MOSTs is necessary to obtain a low subthreshold current. The raised power-supply voltage increases the transconductance of the driver MOST and compensates for variations in

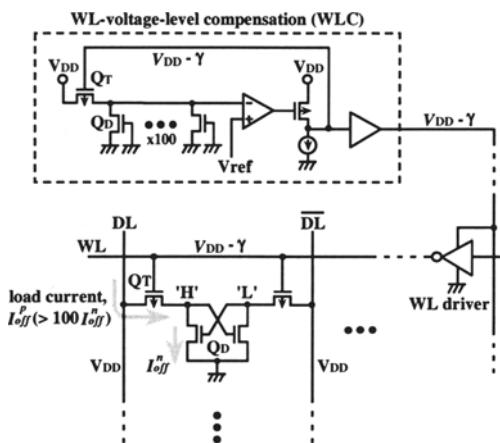


Figure 9. 4-T SRAM cell and control of the nonselected WL level.

$V_T$  and mismatches of  $V_T$ . Even a low- $V_T$  transfer MOST for high-speed is then acceptable, due to the increased transconductance of the driver MOST. Negative word-line biasing reduces the leakage currents from cells in the column. Using a high- $V_T$  transfer MOST with word bootstrapping is an alternative approach that achieves the same result.

## 2.3 Peripheral Logic Circuits

**Subthreshold Current:** Many attempts [1] have been made to reduce subthreshold currents in standby mode. Gate-source back-biasing [1] is effective most for memory applications. For example, applying a back-bias of as little as 0.25 V to a word-driver block reduces the standby subthreshold-current by 2-3 decades without inflicting penalties in terms of speed and area. In combination with a multiply divided power-line scheme [1], this approach is able to confine subthreshold currents to a single sub-power line. Applying dual- $V_T$  [16] to the critical paths of logic circuits is quite effective, reducing the standby subthreshold current to one-fifth of its value for a single low  $V_T$ . In another form of multiple- $V_T$  design called random modulation [17], four kinds of  $V_T$ s are applied to further reduce the current in sub-1V circuitry despite additional cost and complexity of design. Applying the well-known dynamic substrate back-bias ( $V_{BB}$ ) reduces the current by 1-2 decades. Even so, the required  $V_{BB}$  swing is as large as 1-3 V and this approach becomes less effective with device scaling [18]. The smaller body constant, enhanced short-channel effects, and increases in other leakage currents such as the gate-induced drain-lowering (GIDL) current [18] are responsible for the drawback. Even if the subthreshold current is still a small fraction of the active current because  $V_T$  is still high, the subthreshold current in the active mode makes the operation of dynamic circuits such as dynamic NAND for the decoder unstable, discharging the floating nodes. The level keeper proposed for logic circuits (Fig. 10) [19] would also be effective in memory design, although its effects would be fatal if subthreshold current is increased to the extent that the keeper cannot manage.

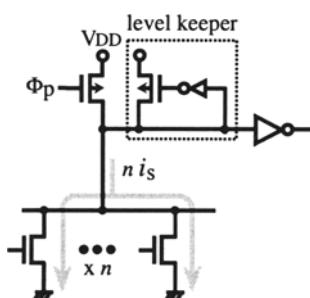


Figure 10. A level keeper in a dynamic NOR against precharge-level degradation by leakage current.

**Speed Variation:** The amount of speed variation for a given variation in design-parameters is increased by lowering  $V_{DD}$ . Control of  $V_{BB}$  and internal  $V_{DD}$  [20,21] in accordance with the parameter variations reduces the speed variation. Controlling a forward  $V_{BB}$  is more effective in reducing speed variations [22,23] than controlling a reverse  $V_{BB}$ . This is because the  $V_T$ - $V_{BB}$  characteristics is more sensitive to  $V_{BB}$

[1]. Forward  $V_{BB}$  in [23], for example, reduced the variation of logic circuits and improved operation speed by 10%. If a forward  $V_{BB}$  is used, however, requirements for noise suppression become more stringent, i.e., there is a call for the uniform distribution of the forward  $V_{BB}$  throughout the chip. However, having a quasi-floating  $V_{BB}$  at each MOST induces instabilities such as history effects, as in SOI, resulting in intra-die fluctuations in  $V_T$  [24]. This is especially so for high-speed LSIs. When the clock cycle is smaller than the noise decay time in the substrate, which may be more than 10 ns for a substrate capacitance of 1 nF, substrate noise may accumulate and cause a variation in  $V_T$ . In particular, eDRAMs may couple spike noises to the floating  $V_{BB}$  when many data lines are simultaneously charged and discharged. This is so unless a triple-well substrate is used. Additional current consumption, in the form of bipolar current induced by the forward  $V_{BB}$ , is another matter that must be considered.

## 2.4 Other Design Issues

**On-Chip Supply-Voltage Converters:** On-chip voltage conversion is essential for the stable operation of memory cells, reduction of the subthreshold current, and suppression of variations in speed with fewer external supply voltages. A high efficiency of voltage conversion, high degree of accuracy in the output voltage and low cost of implementation are the key issues. Using voltage-up converter with a charge pump suffers from the poor current-driving ability of this approach, which means that subthreshold currents at the load must be strictly limited to maintain the boosted voltage level. Voltage-down converters have two types; the series regulator and the switching regulator. The series regulator, widely used in modern commercial DRAMs, provides a small output current and a poor efficiency ( $\eta$ ) of around 50% when the difference between input and output voltages is large. A regulator of this type, however, offers a highly accurate output voltage and it is possible to implement all of the required components on a single chip. The switching regulator has a good efficiency, even in the high output-current region. However, a regulator of this type inherently suffers from a large ripple in the output voltage and a poor transient response for large changes in the current. In general, it also requires a large NMOS and PMOS, and external parts such as an inductor and capacitor. There have been many attempts to solve these problems. A combination of a charge pump and series regulator achieves a high efficiency and precise output voltages [25], and is applicable to the NWL scheme described above. In the hybrid converter [26], consisting of both a series regulator and switching regulator, the switching regulator is turned on and off according

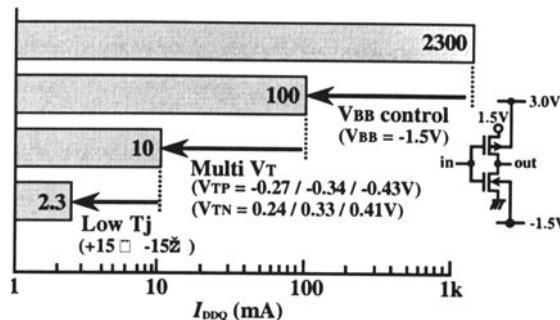


Figure 11.  $I_{DDQ}$  testing by reducing subthreshold current.

the load current to maintain a high efficiency across a wide range of load currents.

**Power Management:** Here are some view points of memory designers on the methods of power management for SoC technology that have been proposed by logic designers. Although it incurs a long recovery time at heavily capacitive internal power-lines, power switching with a high- $V_T$  MOST completely cuts the leakage currents of the internal core circuits. Dynamic voltage scaling (DVS) [27,28], in which the clock frequency and  $V_{DD}$  are dynamically varied in response to the computational load, provides a reduced energy consumption per process during the periods of little computational, while still providing peak performance when this is required. However, this approach becomes less effective in the low- $V_{DD}$  era since the range across which it is possible to vary  $V_{DD}$  becomes narrow. In addition, successful operation over a wide range of  $V_{DD}$  requires the accurate tracking of all circuit delays. Furthermore, applying DVS makes dynamic circuits (e.g. eDRAMs) unstable. This is due to the trapping of charge at floating nodes when voltage bumps are applied. DVS does not reduce subthreshold currents. However, in elastic- $V_T$  CMOS [29], where the clock frequency,  $V_{DD}$ , and  $V_{BB}$  are all dynamically varied, these currents are reduced. The cost is the added complexity of design. Recently, low-power designs for which it is possible to specify the maximum level of power have been proposed. In ChipOS [30], for example, each circuit-block's power is managed by controlling the gated clock and power switch to achieve a given power budget. The confinement of junction temperature and load current that this allows may permit the use of temperature-sensitive eRAM, or the suppression of fluctuations in the voltage output of on-chip voltage converters.

**Testing:** A large subthreshold current makes the discrimination of defective and non-defective  $I_{DDQ}$  currents difficult, and thereby poses a problem for the  $I_{DDQ}$  testing of low-voltage CMOS circuits.  $I_{DDQ}$  testing with

the application of a reverse  $V_{BB}$  (Fig. 11) [31] is effective when low-temperature measurement and multi- $V_T$  design are combined.

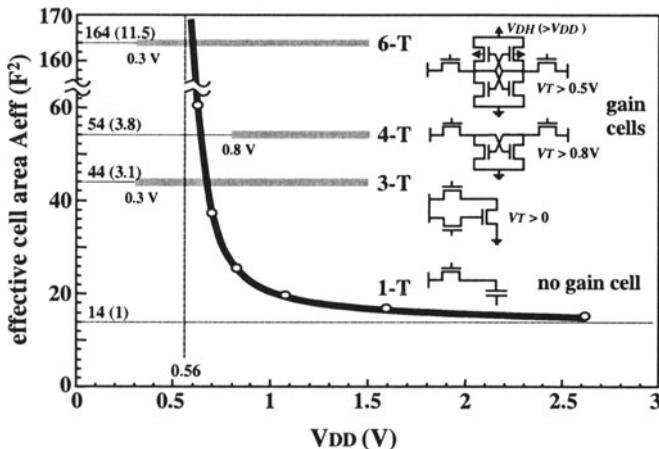


Figure 12. Effective cell area of various RAMs vs. power-supply voltage.

### 3. FUTURE TRENDS

The 1-T DRAM cell is not suitable for low-voltage operation because the reduction of its signal voltage is not acceptable. More division of the data-line, aiming at a smaller  $C_D$ , to offset the reduction in cell-signal levels at a lower  $V_{DD}$  sharply increases the effective cell area, as shown in Fig. 12. This area is defined as the sum of the actual cell area and the total area of the additional cell-related circuits (such as sense amp) in each division. Using gain cells would be one solution for the problem, because they generate a high enough signal voltage without requiring any data-line division even at low values of  $V_{DD}$ , and thus provide a fixed effective cell area that is independent of  $V_{DD}$ . For eSRAM, the advanced 6-T cell (Fig. 5(b)) may continue to be used.

The issue of leakage current presents a real challenge in the design and testing of a low-voltage high-speed SoC. The developments of innovative MOSTs that have new gate materials and subthreshold-current reduction circuits are the keys to tackling this challenge. In particular, with a further reduction in  $V_T$ , subthreshold-current reduction circuits for use in high-speed active mode will be indispensable, even for static logic circuits. This is because the subthreshold current comes to exceed the capacitive current and eventually dominates the total active current of the chip [1]. In this case, the low-power advantage of CMOS circuits is lost. If this problem is not solved,

we can envision a scenario in which even CMOS SoC would suffer from huge levels of dc power dissipation caused by subthreshold currents, as was the case in the bipolar and BiCMOS LSI eras of the recent past. Thus, low-power techniques for bipolar and BiCMOS circuits might be revived to help reduce the power dissipation of the SoC. Reducing the number of random logic gates is one vital way of reducing the active-mode subthreshold current, since control of subthreshold currents from random logic gates at sufficiently high speed may remain impossible. Hence, new SoC architectures such as a memory-rich SoC, by which the subthreshold currents are effectively reduced, will be needed.

## 4. REFERENCES

- [1] K. Itoh, VLSI Memory Chip Design, Springer-Verlag, March 2001.
- [2] K. Itoh et al., "Reviews and Prospects of High-Density RAM Technology," CAS 2000, Sinaia (Romania), Oct. 2000.
- [3] O. Takahashi, et al., "1GHz Fully Pipelined 3.7ns Address Access Time 8kx1024 Embedded DRAM Macro," ISSCC 2000, pp. 396-397.
- [4] T. Inukai, et al., "Suppression of Stand-by Tunnel Current in Ultra-Thin Gate Oxide MOSFETs," SSDM 1999, pp. 264-265.
- [5] S. Miyano and M. Takahashi, "Embedded DRAM SOCs and Its Application for MPEG4 Codec LSIs," Proceedings of VLSI CIRCUITS SHORT COURSE, pp. 101-121, June 2001.
- [6] W. Leung, et al., "The Ideal SoC Memory: 1T-SRAM," 13th Annual IEEE Int. ASIC/SOC Conference, Sept. 2000.
- [7] M. Asakura, et al., "An Experimental 256-Mb DRAM with Boosted Sense-Ground Scheme," IEEE JSSC, vol. 29, no. 11, pp. 1303-1309, Nov. 1994.
- [8] T. Kawahara, et al., "A Circuit Technology for Sub-10ns ECL 4Mb BiCMOS DRAMs," 1991 Symp. on VLSI Circuits, pp. 131-132.
- [9] H. Mizuno, et al., "CMOS-Logic-Circuit-Compatible DRAM Circuit Designs for Wide-Voltage and Wide-Temperature-Range Applications," 2000 Symp. on VLSI Circuits, pp. 120-121.
- [10] B. Baas, et al., "An Energy-Efficient Single-Chip FFT Processor," 1995 Symp. on VLSI Circuits, pp. 164-165.
- [11] K. Zhang, et al., "The Scaling of Data Sensing Schemes for High Speed Cache Design in Sub-0.18 $\mu$ m," 2000 Symp. on VLSI Circuits, pp. 226-227.
- [12] K. Agawa, et al., "A Bit-Line Leakage Compensation Scheme for Low-Voltage SRAM's," 2000 Symp. on VLSI Circuits, pp. 70-71.
- [13] R. Krishnamurthy, et al., "A 0.13 $\mu$ m 6GHz 256x32b Leakage-tolerant Register File," 2001 Symp. on VLSI Circuits, pp. 25-26.
- [14] K. Takada, et al., "A 16Mb 400MHz Loadless CMOS Four-Transistor SRAM Macro," ISSCC 2000, pp. 264-265.
- [15] K. Itoh, et al., "A Deep Sub-V, Single Power-Supply SRAM Cell with Multi-Vt, Boosted Storage Node and Dynamic Load," 1996 Symp. on VLSI Circuits, pp. 132-133.
- [16] N. Rohrer, et al., "A 480 MHz RISC Microprocessor in a 0.12 $\mu$ m Leff CMOS Technology with Copper Interconnects," ISSCC 1998, pp. 240-241.

- [17] K. Yano, "Logic Circuit Families for Year-2005 Multi-Giga-Hertz Processors," IEEE Workshop on Design for Multi-Giga-Hertz Processors, San Francisco, Feb. 2000.
- [18] A. Keshavarzi, et al., "Effectiveness of Reverse Body Bias for Leakage Control in Scaled Dual Vt CMOS ICs," ISLPED 2001, pp. 207-212.
- [19] A. Alvandpour, et al., "A Conditional Keeper Technique for Sub-0.13 $\mu$  Wide Dynamic Gates," 2001 Symp. on VLSI Circuits, pp. 29-30.
- [20] M. Kubo et al., "A Threshold Voltage Controlling Circuit for Short Channel MOS Integrated Circuits," ISSCC 1976, pp. 54-55.
- [21] T. Kuroda, et al., "A 0.9V, 150MHz, 10mW, 4mm<sup>2</sup>, 2-D Discrete Cosine Transform Core Processor with Variable-Threshold-Voltage (VT) Scheme," ISSCC 1996, pp. 166-167.
- [22] Y. Oowaki, et al., "A Sub-0.1 $\mu$ m Circuit Design with Substrate-over-Biasing," ISSCC 1998, pp. 88-89.
- [23] M. Miyazaki, et al., "1000-MIPS/W Microprocessor using Speed-Adaptive Threshold-Voltage CMOS with Forward Bias," ISSCC 2000, pp. 420-421.
- [24] H. Mizuno, et al., "18- $\mu$ A-Standby-Current 1.8-V 200-MHz Microprocessor with Self Substrate-Biased Data Retention Mode," ISSCC 1999, pp. 280-281.
- [25] H. Tanaka, et al., "A Precise On-Chip Voltage Generator for a Giga-Scale DRAM with a Negative Word-Line Scheme," 1998 Symp. on VLSI Circuits, pp. 230-231.
- [26] M. Hiraki, et al., "A 63 $\mu$ W-Standby-Power Microcontroller with On-Chip Hybrid Regulator Scheme," 2001 Symp. on VLSI Circuits, pp. 225-228.
- [27] D. R. Ditzel, "Transmeta's Crusoe: A Low-Power x86-Compatible Microprocessor Built with Software," COOL Chips III, pp. 1-30.
- [28] T. Burd, et al., "A Dynamic Voltage Scaled Microprocessor System," ISSCC 2000, pp. 294-295.
- [29] M. Mizuno, et al., "Elastic-Vt CMOS Circuits for Multiple On-Chip Power Control," ISSCC 1996, pp. 300-301.
- [30] H. Mizuno, et al., "ChipOS: Open Power Management Platform to Overcome the Power Crisis in Future LSIs," ISSCC 2001, pp. 344-345
- [31] T. Miyake, et al., "Design Methodology of High Performance Microprocessor using Ultra-Low Threshold Voltage CMOS," CICC 2001, pp. 275-278.

# **Low-Voltage $0.25\mu\text{m}$ CMOS Improved Power Adaptive Issue Queue For Embedded Microprocessors**

Brian Curran, Mary Gifaldi, Jason Martin, Alper Buyuktosunoglu,  
Martin Margala, and David Albonesi

*Department of Electrical and Computer Engineering, University of Rochester,  
New York, 14627, USA.*

[margala@albonesi@ece.rochester.edu](mailto:margala@albonesi@ece.rochester.edu)

**Abstract:** In this paper, we present an improved adaptive issue queue for use in embedded microprocessors. The issue queue was designed, simulated, and implemented in  $0.25\mu\text{m}$  CMOS technology. Using an example of an 8-entry issue queue, we found that by shutting down the parts of the queue that are not in use, the power dissipation of the issue queue can be greatly decreased with only a small tradeoffs in speed and area. The additional logic, associated with the shutdown circuitry, introduces an insignificant overhead. Furthermore, low-power optimization has been performed on CAM and SRAM arrays which resulted in additional savings of power. It has been found that 4.8%-56% reduction in power can be achieved at 2.5V and at 1.5V the power consumption is reduced by 8.5%-60%. In addition, the overall power consumption drops 10 times compared to 2.5V operation.

**Key words:** low-voltage, power adaptive, issue queue, CMOS, embedded, architecture

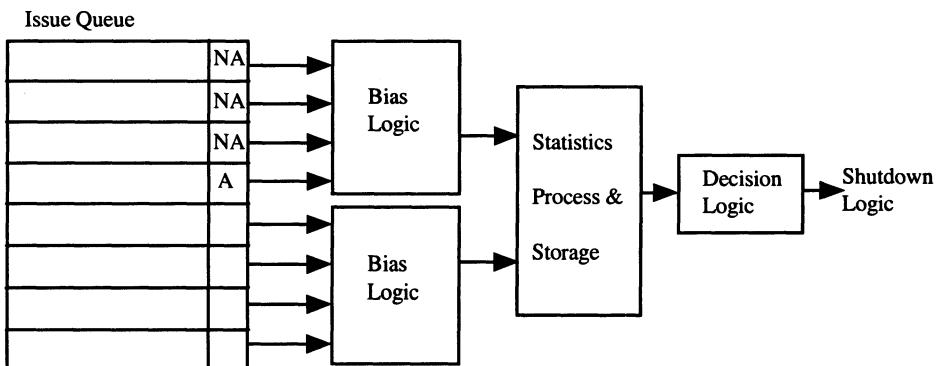
## **1. INTRODUCTION**

In modern electronics, increased power dissipation has become a major design constraint. An issue queue takes instructions and the destinations for the instructions and holds them until they are selected for processing. The issue queue contributes greatly to the overall power consumption of the

microprocessor, as well as complicating verification [1]. In a “floating point high power test,” a test designed by Intel that attempts to measure maximum power, the issue queue dissipates approximately 33% of the total processor power. Therefore, reducing power consumption in the queue should significantly reduce power consumption of the processor itself. A new type of issue queue has been proposed by Buyuktosunoglu et al. [1,2]. However, the authors have focused on large issue queues from 32 to 128 entries and most common queue sizes are between 8 entries and 27 entries. By using a simple model, an 8-entry issue queue, and using many techniques for low-power operation, we have demonstrated that additional significant savings in power can be achieved on a circuit-level combined with aggressive voltage scaling.

## 2. QUEUE STRUCTURE

The overall structure of the adaptive power issue queue is shown in Figure 1. Buyuktosunoglu et al. proposed separating the queue into “chunks” of equal size. Each chunk can be enabled or disabled based on its usage that is measured by the activity within the chunks. The bias logic generates an “active” or “not active” signal for the chunk it is assigned to. The statistics process and storage logic keeps track of how often chunks are active/not active, and the decision logic sends the enable/disable signal to the chunks.



*Figure 1.* Overall Issue Queue Structure.

As seen in Figure 2, the chunks are separated by a transmission gate on the bitlines, allowing the bitlines to be shortened if a chunk or chunks have been disabled, decreasing the capacitance on the bitline and increasing the speed of searching and retrieval of data from the queue.

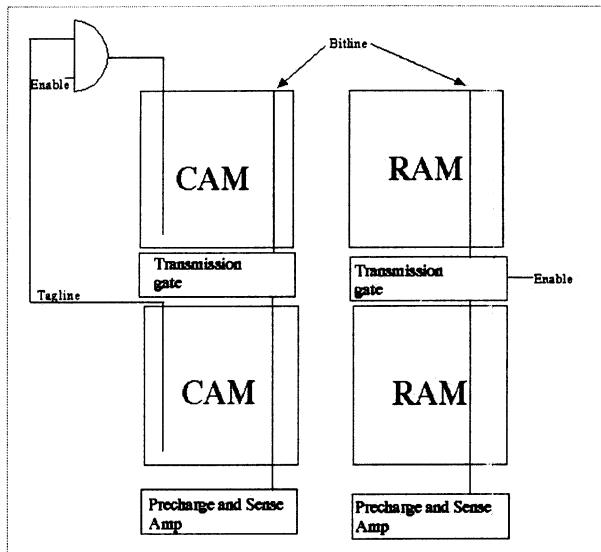


Figure 2. CAM/RAM section.

## 2.1.CAM Design

The CAM section of the queue is to hold the source registers of each instruction word. The CAM cell used is from Zhang and Asanovic [3], shown in Figure 3. The bitlines and search lines have been separated to reduce the capacitance during the searching operation. Each row is split into two, with the match detection logic on each half match line. This allows the worst case delay of the match line capacitance discharge to be divided in half. Energy is reduced on the match lines through the use of n-type transistors precharged to  $V_{dd} - V_{tn}$ . The match detection circuitry used is from Bellaouar and Elmasry [4], shown in Figure 4.

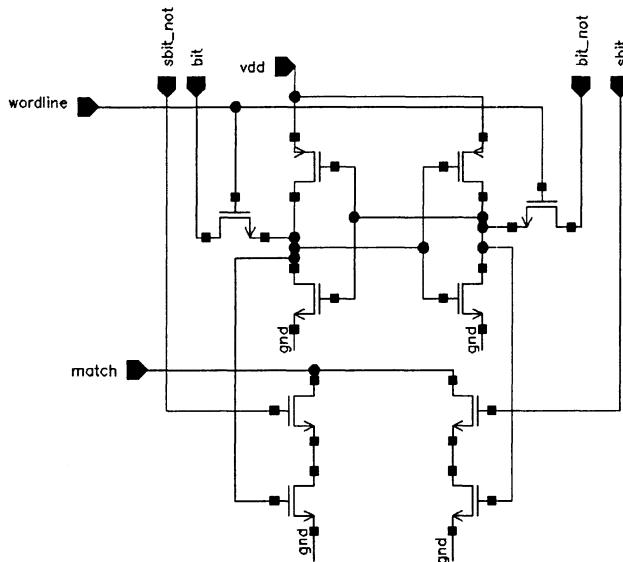


Figure 3. CAM Cell.

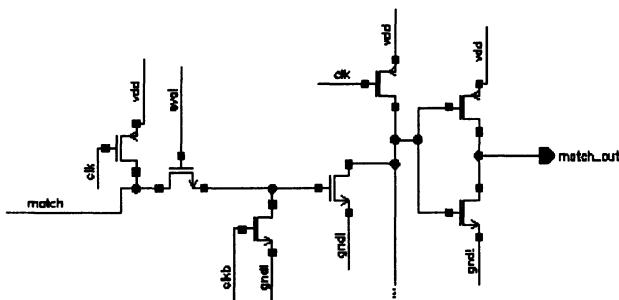


Figure 4. Match Detection Logic.

## 2.2. RAM Design

The RAM section of the queue is to hold the instruction, destination register, and any other information needed to complete the instruction's operation. We have utilized a low-power SRAM architecture originally proposed by Wang, et al. [5,6,7,8]. The SRAM cell is shown in Figure 5. It is a 7-T cell, however the sizing allows it to be 4.3% smaller than a standard 6-T cell. The additional transistor is for clearing the cell before a write operation. The sense amplifier used was a current-mode sense amplifier for a 1.5 power source, shown in Figure 6. The power consumption is reduced by

61 to 94% versus a conventional design. To perform a current-mode write operation, an n-type current conveyor (shown in Figure 7) is inserted between the data input cell and the memory cell.

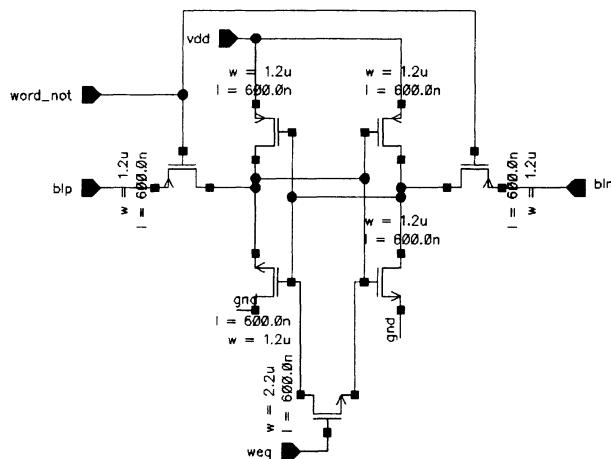


Figure 5. 7-T SRAM Cell.

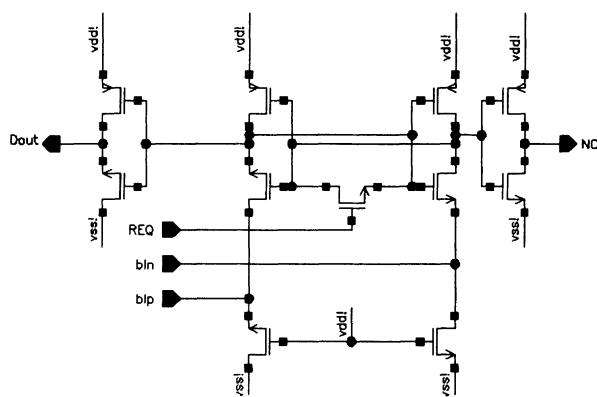


Figure 6. Current-mode Sense Amplifier.

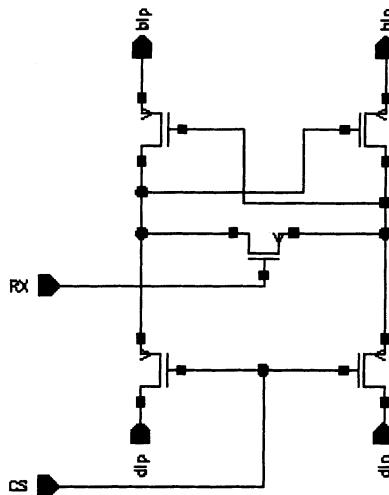


Figure 7. Current Conveyor.

### 2.3. Shutdown Circuitry

The barebones shutdown circuitry consists of a bias logic block for each chunk, and a detection logic block. The shutdown circuitry diagram for our sample 8-entry queue is shown below.

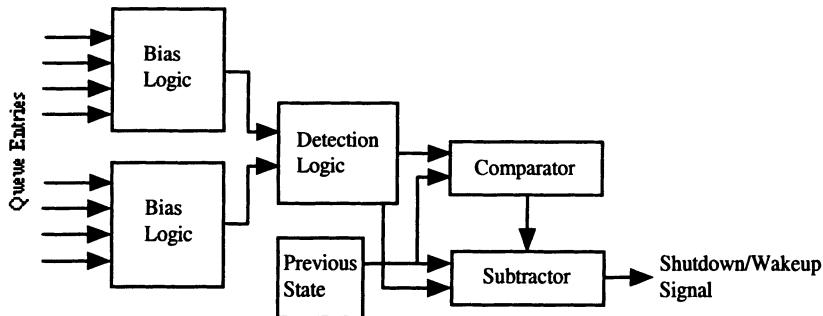


Figure 8. Shutdown Circuitry.

The bias logic block consists of a simple circuit that averages out the number of active entries per chunk, and generates an “active” or “not active” signal based on the logic table below:

*Table I.* Bias Logic Table

| Entry<br>1 | Entry<br>2 | Entry<br>3 | Entry<br>4 | Output |
|------------|------------|------------|------------|--------|
| NA         | NA         | NA         | NA         | NA     |
| NA         | NA         | NA         | A          | NA     |
| NA         | NA         | A          | A          | A      |
| NA         | A          | A          | A          | A      |
| A          | A          | A          | A          | A      |

A = Active, NA = Not Active

The detection logic can take a maximum of four bias logic blocks as inputs. Its function is to generate the number of active bias logic blocks. For our sample 8-bit shutdown circuitry with two bias logic blocks, only one detection logic block is needed, which can generate a binary zero, one, or two. In a case where there are multiple detection logic blocks, a binary counter is added to the circuit to add all outputs from the detection logic blocks. For example, if 32 bits are passed to the shutdown circuitry, and each bias logic block monitors four bits, eight bias logic blocks will be needed, which will then be connected to two detection logic blocks. If the first detection logic block generates a binary three, and the second generates a binary two, the counter will then add both numbers and generate a binary five.

The decision logic block shown in Figure 1 is composed of a comparator and subtractor, and is the most complicated part of the shutdown circuitry, and the final stage before the generation of the shutdown/wake up signal. The number of active queue entries (generated from the detection logic or the counter) from the previous state is stored in a register, and compared with the current number of active queue entries. If the previous state contained more active queue entries, then a signal is sent to shut down the difference between the number of active entries. If, however, the current state contains a greater number of active entries, a signal is sent to "wake up" the difference between the number of active entries. It is the shutting down of unnecessary queue entries that compensates for the extra power that the shutdown circuitry uses.

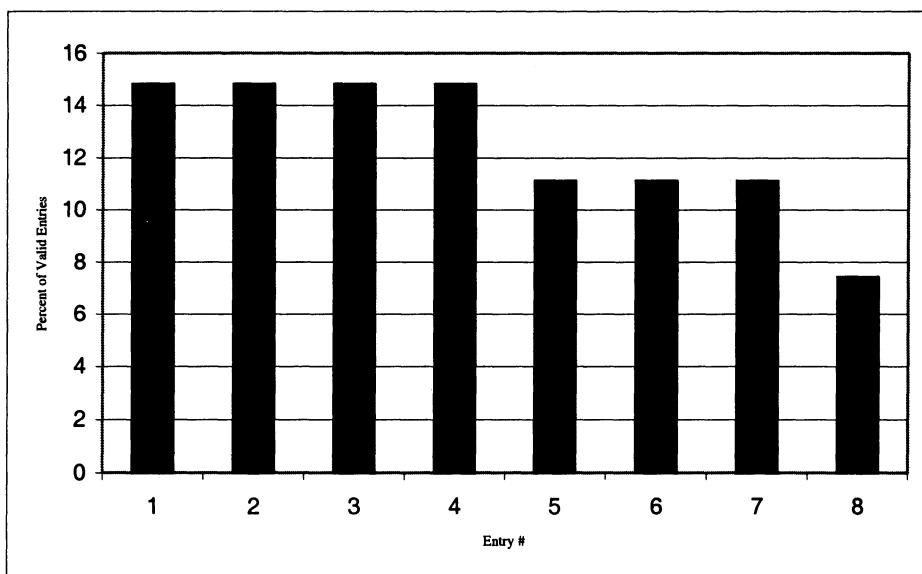
### 3. TEST SETUP

The characteristics of the simulated queue are as follows:

- 8 entry queue
- 2 chunks, 4 entries each
- 3 bit register addresses
- 16 bit instructions
- 1.5 V and 2.5V Voltage Source

Standard queue operation was simulated with and without the shutdown circuitry, and tested for power consumption and signal transmission with various combinations of active/inactive queue entries.

A set of instructions was created to simulate the same usage of the queue as the SPECint95 benchmark suite. The percentages are shown below in Figure 9. As the graph shows, more than 60% of the time only 4 entries of the queue are used, meaning that more than half of the time the queue is operating, a chunk can be turned off and power can be saved.



*Figure 9. Simulated Queue Entry Usage.*

## 4. RESULTS

Two source voltages were used for the simulation, 2.5V (the models used were optimized for this voltage, as well as our designs) and 1.5V. The early results show that up to 56% (third column) of power can be saved at 2.5V. Furthermore, the Figure 10 shows that the additional shutdown circuitry does not represent a major increase in power consumed over it not being present (first and second columns). The fourth column represents an average power consumed by the issue queue during the test.

The results for 1.5V are below in Figure 11. As seen from the Figure, the power is further reduced 10 times compared to 2.5V operation. This significant reduction is due to reduced energy waste and short-circuit power at low-voltages. Many signal spikes that are present at 2.5V become negligible at 1.5V. In addition,  $V_{dd}=1.5V$  reduces the area between  $V_{IL}$  and  $V_{IH}$  consequently reducing the time for a direct path between the power line and

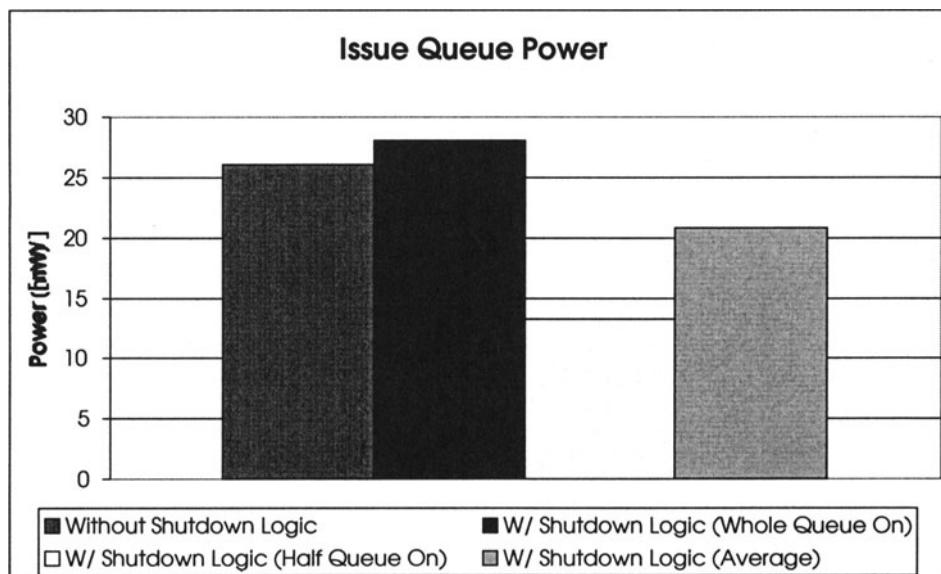
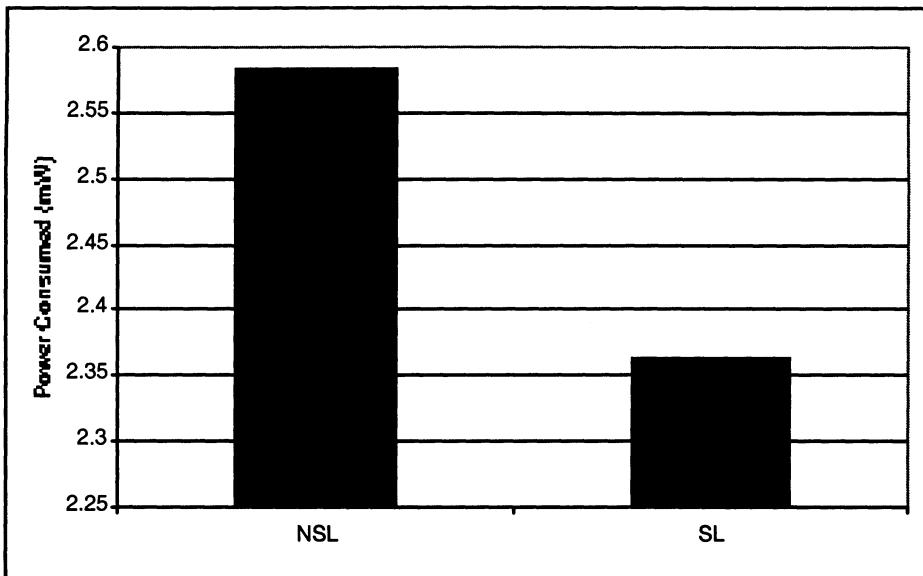


Figure 10. Issue Queue Power at 2.5V.



*Figure 11.* Power Consumed at 1.5V.

the ground. Furthermore, the improved issue queue consumes 8.5%-60% less power compared to the standard issue queue.

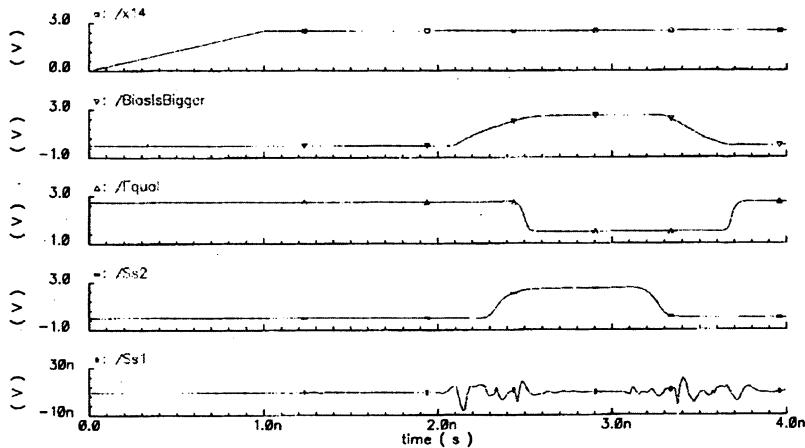
With the 1.5V power supply, there is a 8.5% power savings.

A timing diagram for one case of the shutdown circuitry is shown in Figure 12. The first signal (X14) shown is a logic "1," or 2.5 V, applied to both bias logic blocks to activate them. The second signal (BiasIsBigger) comes from the output of the comparator of the decision logic, indicating that there are more bias logic blocks active than in the previous stored state, which in this case is zero. The input signal reaches 2.5 V at 1 ns, and the comparator registers this at 2.1 ns. Thus, there is a 1.1 ns delay between the input of the bias logic and the input of the subtractor.

The third signal (Equal) also comes from the comparator, and is high when the number of active bias logic blocks is equal to the previous number of active blocks. When this "equal" signal is high, the number of queue entries is unmodified. This signal starts at logic "1," since both states are empty, and decreases to logic "0" at 2.5 ns, shortly after the bias logic blocks first become active. It then returns to logic "1" at 3.7 ns. At this point, there are two active bias logic blocks, and the previous state register also contains a binary two.

The fourth (Ss2) and fifth (Ss1) signals are the output shutdown/wakeup signals. The fourth signal represents a binary two, and the fifth represents binary one. The subtractor generates a binary two, the result of subtracting the number of previous active entries from the number of active bias blocks, at 2.4 ns. Since the comparator has flagged the bias logic as having the

greater number of active entries, the signal is sent to "wake up" two queue entries. Were the previous number of entries greater than the number of current bias entries, the signal would instead be sent to shut down queue entries.



*Figure 12.* Timing of Shutdown Circuitry.

It takes 1.4 ns from the time that the input signal reaches 2.5 V for the shutdown/wakeup signal to be generated. Most of this delay (1.1 – 1.2 ns) occurs between the input signal and the subtractor of the decision logic block.

## 5. CONCLUSION

In this paper, an improved adaptive issue queue for embedded microprocessors has been presented. The optimization has been focused on memory architectures for CAMs and RAMs that provide highest energy-efficiency (least energy for most performance). Early results show that 4.8–56% of power can be reduced at 2.5V and 8.5%–60% of power can be reduced by reducing the operating voltages to 1.5V. In addition, the issue queue consumes more than 10 times less power at 1.5V than at 2.5V. The improved issue queue has been designed in 0.25 $\mu$ m CMOS technology.

## 6. REFERENCES

- [1] A. Buyuktosunoglu, D. Albonesi, S. Schuster, D. Brooks, P. Bose, and P. Cook, "A Circuit Level Implementation of an Adaptive Issue Queue for Power-Aware Microprocessors", Proceedings of 11th IEEE Great Lakes Symposium on VLSI, pp. 73-78, March 2001.
- [2] A. Buyuktosunoglu, S. Schuster, D. Brooks, P. Bose, P. Cook, and D.H. Albonesi, "An Adaptive Issue Queue for Reduced Power at High Performance", Workshop on Power-Aware Computer Systems, held at the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, November 2000. Also to appear in Springer-Verlag Lecture Notes in Computer Science, Volume 2008.
- [3] M. Zhang and K. Asanovic, "Highly-Associative Caches for Low-Power Processors", Kool Chips Workshop, 33<sup>rd</sup> International Symposium for Microarchitecture, Monterey, CA, December 2000.
- [4] A. Bellaouar and M.I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems*, pg. 473, Kluwer Academic Publishers, Massachusetts, USA, 1995.
- [5] J.-S. Wang, et al., "Low-Power Embedded SRAM Macros with Current-Mode Read/Write Operations", Proceedings of IEEE/ACM International Symposium Low-Power Electronics and Design, pp. 282-287, Aug. 1998.
- [6] M. Khellah and M. I. Elmasry, "Circuit Techniques for High-Speed and Low-Power Multi-Port SRAMs", Proceedings of IEEE International ASIC Conference, pp. 157-161, Sep. 1998.
- [7] J.-S. Wang and H. Y. Lee, "A New Current-Mode Sense Amplifier for Low-Voltage Low-Power SRAM Design", in Proceedings of IEEE International ASIC Conference, pp. 163-167, Sep. 1998.
- [8] M. Margala. Low-Power Memory Circuits, in "*The VLSI Handbook*", Wai-Kai Chen ed., chapter 53, IEEE/CRC Press, 2000.

# Gate sizing for low power design

Philippe Maurine, Nadine Azemard, Daniel Auvergne  
*LIRMM, 161 Rue Ada, 34392 Montpellier, France*

**Abstract:** Low power design based on minimal size gate implementation induces great speed penalty. We present a new gate sizing method for improving the speed performance of static logic paths designed in submicron CMOS technologies without increasing the power dissipation obtained with a minimal surface implantation. This methodology is based on the definition of local gate sizing criterion. It has been deduced from analytical models of the output transition time and of the short circuit power dissipation which are briefly introduced. Validations are given, on a 0.18  $\mu\text{m}$  process using Hspice simulations(Bsim3v3 level69).

**Key words:** gate sizing, short circuit, power dissipation

## 1. INTRODUCTION:

Lowering the power consumption under speed constraint has emerged as a critical issue for VLSI designers. This requires an accurate estimation and a very good control of the different power components. Various analytical models [1-5] and methods [6-10] allowing to handle the speed-power-surface trade-off have been developed at all level of the design flow. On the whole, these heuristics aim at reducing the external power dissipation (eq.1) resulting from the voltage variations across the capacitance of the different nodes of the circuit as

$$P_{EXT} = \eta \cdot f \cdot C_{eff} \cdot V_{DD}^2 \quad (1)$$

where  $\eta$  is the activity rate,  $f$  the clock frequency,  $C_{\text{eff}}$  the effective capacitance and  $V_{\text{DD}}$  the supply voltage.

However the contribution of the power dissipation associated to the short circuit component, that may represent up to 20%-30% of the total power, is often neglected. In this work we show that considering the short circuit power component during the optimisation process may result in significant improvement of the speed and power performances of combinational paths.

To reach this goal, analytical models of the output transition time and short circuit power dissipation have been developed. They are briefly described in section 2 and 3, respectively (more detailed description can be found in [11-13]). In section 4, we deduced from these models a local sizing criterion that allows to minimize the power dissipated by a two inverter chain. Based on it, a gate sizing heuristic dedicated to the minimization of inverter tree power dissipation is developed in section 5. In section 6, some results obtained on really simple examples are presented and discussed, before to conclude in section 7.

## 2. OUTPUT RAMP MODEL:

The output transition time of CMOS gates and more precisely of CMOS inverter depends strongly on its current capability. For a falling output edge, modeling the N transistor as a current generator, allows to express the inverter output transition time as the ratio between the charge to be evacuated from the output node and the maximum discharge current that can provide the N transistor as

$$\tau_{\text{OUT}} = \frac{C_L \cdot V_{\text{DD}}}{I_{\text{MAX}}} \quad (2)$$

where  $C_L$  is the output load including parasitic capacitances. Obviously the determination of the maximum value of the discharge (charge) current in the structure is of prime importance in modeling the output transition time. This value depends of course on the N (P) transistor width, but also on the input ramp duration as illustrated in Fig.1 in which two switching characteristics corresponding to different input ramp duration domains, labelled ① and ②, are displayed.

In region 1, the set up of the current of the N transistor follows the input ramp variation to finally exhibits a constant maximum value (eq.3) during all the discharge process, this defines the fast input range.

$$I_{MAX}^{fast} = K_N \cdot W_N \cdot (V_{DD} - V_{TN}) \quad (3)$$

In region 2, the maximum current is obtained before the input ramp reaches its maximum value, resulting in a smaller value of the charge evacuated by time unit. This defines the slow input range where the maximum value of the discharging current decreases when the input transition time increases.

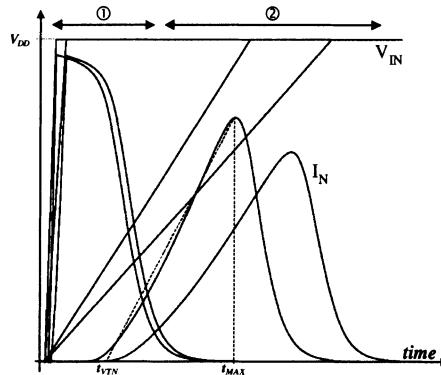


Figure 1. Sensitivity of the inverter discharging current to fast ① and slow ② input ramps.

In order to evaluate the value of  $I_{MAX}$ , we assumed that the discharge current  $I_N(t)$  varies linearly from the time at which it begins to rise ( $t=t_{VTN}$ ) and the time at which it reaches its maximum value  $I_{MAX}$  ( $t=t_{MAX}$ ). Under this assumption we obtain

$$I_{MAX}^{Slow} = \sqrt{\frac{K_N \cdot W_N \cdot V_{DD}^2 \cdot C_L}{\tau_{IN}}} \quad (4)$$

Combining eq.(2) with eq. (3) and (4), we finally get the output transition time expression for fast and slow input ranges as

$$\tau_{OUT}^{fast} = \tau_{ST} \cdot \frac{C_L}{C_N} = 2 \cdot T_{HLS} \quad (5)$$

and:

$$\tau_{OUT}^{Slow} = \sqrt{\frac{C_{OX} \cdot L_{GEO}}{K_N}} \cdot \sqrt{\frac{C_L \cdot \tau_{IN}}{C_N}} \quad (6)$$

where  $C_N$  is the thin oxide capacitance of the N transistor and  $T_{HLS}$  the time spent by the inverter to discharge the output voltage from  $V_{DD}$  to  $V_{DD}/2$ .  $\tau_{ST}$  is a parameter characteristic of the process speed and defined from

$$\tau_{ST} = \frac{V_{DD} \cdot L_{GEO} \cdot C_{ox}}{(V_{DD} - V_{TN}) \cdot K_N} \quad (7)$$

The extension to input falling edge can be easily obtained by exchanging p,n subscripts. Consideration of logic gates can be done replacing each gate by an equivalent inverter with identical current possibilities [11-13].

To validate the expressions (5) and (6), various comparisons between the model predictions and Hspice simulations (Most9, Bsim3v3 lvl 69) have been done on different processes ranging from  $0.35\mu m$  to  $0.18\mu m$ . The relative observed discrepancy is always lower than 10%. As an example Fig.2 illustrates the output transition time evolution with respect to the input ramp duration value for an inverter defined by  $W_N=1\mu m$   $W_P=2\mu m$  and  $L=0.18\mu m$ .

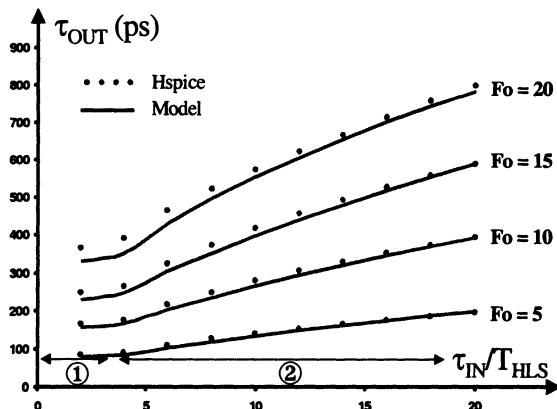


Figure 2. Output transition time values for an inverter loaded by 5 to 20  $C_{IN}$ ; ① and ② specify the fast and slow input ramp condition respectively.

As shown we obtain a very good agreement between simulated and calculated values of the output transition time.

### 3. SHORT CIRCUIT MODEL:

To be useful for designers, an analytical model must allow quick and direct comparison between the various components of the power dissipation. We adopted the equivalent capacitance concept developed in [3] that represents any power contribution in the form of an equivalent capacitance. Thus, considering that the external power dissipation (the most important) is directly proportional to the load  $C_L$  on the output node we get

$$P_{EXT} = \eta \cdot f \cdot C_L \cdot V_{DD}^2. \quad (8)$$

We then express the short circuit component by an equivalent capacitance ( $C_{SC}$ ) according to

$$P_{SC} = \eta \cdot f \cdot C_{SC} \cdot V_{DD}^2 \quad (9)$$

where  $C_{SC} \cdot V_{DD}$  is the amount of charge transferred from the supply rail to the ground during the short circuit process.

Then, assuming that the maximum short circuit current is reached while the P transistor operates in the linear mode, and that the short circuit current shape is symmetrical with respect to its maximum  $I_{MAX}$ , we can show [18-19] that the short circuit equivalent capacitance component value can be obtained from

$$C_{SC} = \frac{(1 - v_{THN} - v_{THP}) \cdot \tau_{IN}}{2 \cdot V_{DD}} \cdot \left( \psi_1 \cdot \frac{\tau_{IN}}{\tau_{OUT}} + \psi_2 \right) \cdot C_P \quad (10)$$

where  $v_{THN}$  and  $V_{THN}$  are the normalized threshold voltages values of P and N transistors and  $\psi_1 \psi_2$  are process parameters. These parameters must be calibrated on the process under consideration and are independent of the inverter configuration. Our approach has been validated by comparing  $C_{SC}$  calculated and simulated values on a wide design range, for various controlling and loading conditions. As illustrated in Fig.3 the relative observed discrepancy is always lower than 10%.

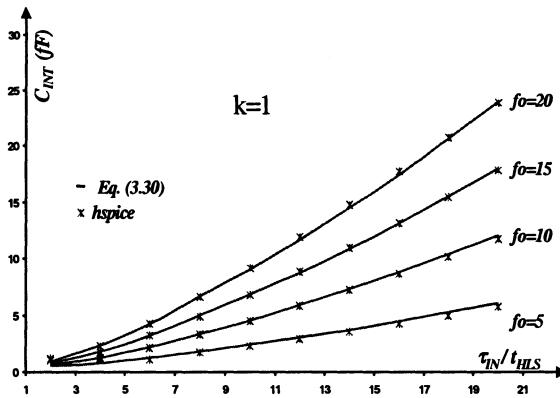


Figure 3. Comparison between simulated and calculated values of the short circuit equivalent capacitance for an inverter ( $W_N=1\mu m$   $W_P=2\mu m$   $L=0.18\mu m$ ) loaded by 5 to 20 times its input gate capacitance.

#### 4. CRITERION FOR LOW POWER ASSIGNMENT

We want to demonstrate in this part that the minimal surface implantation does not minimize the total power dissipation but only the external power dissipation (eq.8). Let us consider the structure represented in Fig.4. The input inverter is controlled by a step input,  $C_{i+1}$  can either be a single inverter or a stack of inverters loaded by the same capacitance.

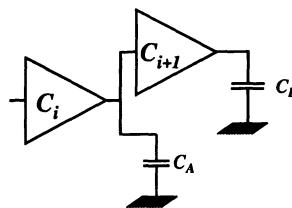


Figure 4. Inverter chain under consideration.

The existence of an optimum design that minimizes the total power dissipation can be justified by the following considerations:

- if the inverter (i) is sized too small then it provides a really slow ramp at its output that induces an important short circuit power consumption in the next stage,
- if (i) is chosen too large, of course the short circuit power dissipation in (i+1) is strongly reduced, but the external power dissipated in (i) is then much greater.

This clearly gives evidence of the existence of a sizing solution that results in a good trade off between short circuit and capacitive power components.

The total power dissipated by this inverter chain can be evaluated from

$$P = \eta f (C_i + C_{i+1} + C_{SC}^i + C_{SC}^{i+1} + C_A + C_L) V_{DD}^2 \quad (11)$$

Where  $C_k$ ,  $C_{SC}^k$  are respectively the gate and the short circuit capacitances of the stage  $k$ , and  $C_A$  models the parasitic capacitance, including drain and interconnect components.

Using the output transition and short circuit power models it is easy to express the total power in terms of  $C_i$  and  $C_{i+1}$ . Cancelling the derivative of eq.11 with respect to  $C_i$ , we get a six order polynomial which can be solved numerically. Although this solution accurately predicts (<10%) the optimal sizing of the stage (i), we decided to found an approximated but analytical solution based on

$$C_{SC} = \frac{(1 - v_{THN} - v_{THP}) \cdot \psi_1 \cdot C_p}{2V_{DD}} \cdot \frac{\tau_{IN}^2}{\tau_{OUT}} \quad (12)$$

Cancelling the derivative of eq.11 with respect to  $C_i$  we get finally the following approximated value of optimum sizing of the stage (i):

$$C_{i-OPT}^{5/2} \approx \frac{3}{2} \cdot A \cdot \frac{C_{i+1}^{3/2} \cdot (C_{i+1} + C_A)^{3/2}}{C_L^{1/2}}. \quad (13)$$

A is a process parameter defined by

$$A = \frac{(1 - v_{THN} - v_{TP}) (\psi_1^{HL} + R_\mu \cdot \psi_1^{LH}) \tau_{ST}}{2V_{DD} \cdot C_{OX} \cdot L_{GEO}}. \quad (14)$$

In order to validate this approach, we compared the total power dissipated by the structure represented in Fig.4, for different sizing conditions,  $C_i = C_{MIN}$  and  $C_i = C_{i-OPT}$ . Let us define by  $P_T^{MS}$  et  $P_T^{OS}$  the total power dissipated by a minimal surface implantation, and that following our proposal (eq.13). We define the gain of this sizing solution by

$$Gain = \frac{P_T^{MS} - P_T^{OS}}{P_T^{MS}}. \quad (15)$$

In Fig.5 we represent the evolution of this gain for different values of the active load  $W_{i+1}$ . As shown the improvement in power dissipation, with respect to a minimal size implementation may become significant and as large as 60% of the total dissipation for an important value of the terminal load.

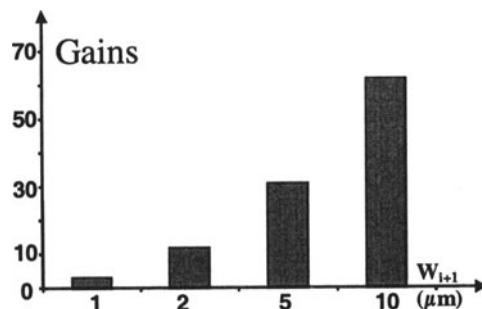


Figure 5. Gains obtained for the structure represented on the fig.4

## 5. SIZING METHODOLOGY

The application of the sizing criterion (13) to an inverter tree is almost straightforward, processing backward from the output to the input of the tree. Two problems have still to be solved:

- firstly, it is necessary to determine the size of the output drivers that allows to minimize the total power on the whole tree,
- secondly, we have to manage the divergences, and more precisely to find the optimal sizing of the (i-1) controlling stage as shown in Fig.7.

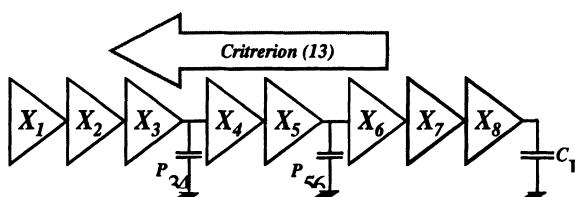


Figure 6. Illustration of how to apply the sizing criterion to an inverter chain

## 5.1 Output drivers:

In minimizing the power dissipated in an inverter tree, it appears that the optimal sizing of the output drivers depends strongly on the load content.

For example, in optimising the logic that drives a register or any sequential gates, we can consider that the output load is an active load or the sum of an active and passive load. Therefore, the sizing of the output driver has to be done using eq.13.

In the other hand, if the output driver controls a passive load, there is no short circuit power dissipation in the load and in this case the driver must be sized at the minimum value satisfying the delay constraint.

## 5.2 Divergence branches:

The case of divergence branches presents a difficulty because the sizing criterion developed in the preceding section does not allow to predict the optimal sizing of the stage (i-1).

The solution we adopted is based on the fact that the power is an additive characteristic of the structure. To justify our approach, let us consider the structure represented in Fig.7.

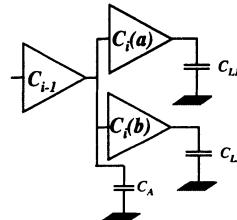


Figure 7. An example of divergence

The sizing criterion (eq.13) allows to predict the optimal value of  $C_{i-1}$ , only if  $C_{L1}=C_{L2}$  in which case the two inverters can be lumped in an unique inverter with an input gate capacitance equal to  $C_i(a)+C_i(b)$ . However in a general configuration  $C_{L1}$  and  $C_{L2}$  have different values.

Nevertheless, as the short circuit power dissipation is a decreasing function of  $C_L$ , we model the two inverter (a) and (b) by an unique inverter (see Fig.8) loaded by  $C_L=\text{MAX}(C_{L1},C_{L2})$  to avoid any overestimation of the short circuit power dissipated by (a) and (b).

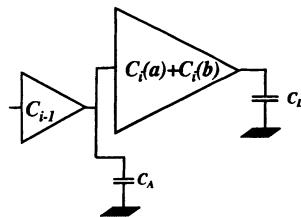


Figure 8. Equivalent structure to that of Fig.7.

## 6. EXPERIMENTAL VALIDATION

This sizing heuristic based on the sizing criterion defined by eq.13 has been applied to an inverter tree represented in the Fig.9. The total power dissipated in the different implementations has been obtained from Hspice simulations.

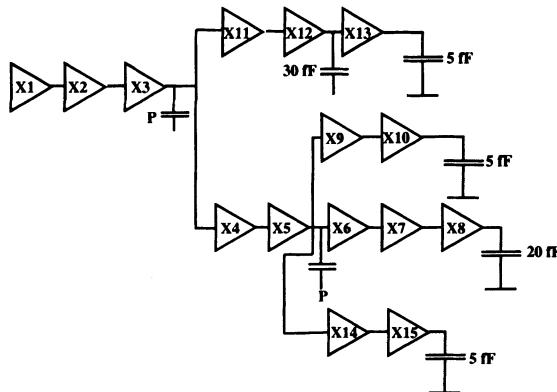


Figure 9. representation of the inverter configuration used to validate the sizing criterion (13)

In Fig.10 we compare the power gain and loss values (eq.15) obtained when comparing the sizing solution proposed to a minimal surface implementations. We considered different values the parasitic routing capacitance  $P$  to illustrate the sensitivity of the result to the parasitic content of the load.

As shown, depending on the value of  $P$ , the gains in power and speed are ranging from 3% to 15% and 13% to 45%, respectively.

The speed increase can be easily justified after a detailed analysis of the simulation results. For our example, the application the sizing criterion increases the size of the stages X12, X5 et X3. This induces a reduction of the ramp duration applied at the input of the stage X14, X11, X9, X6 et X4 reducing their switching delays.

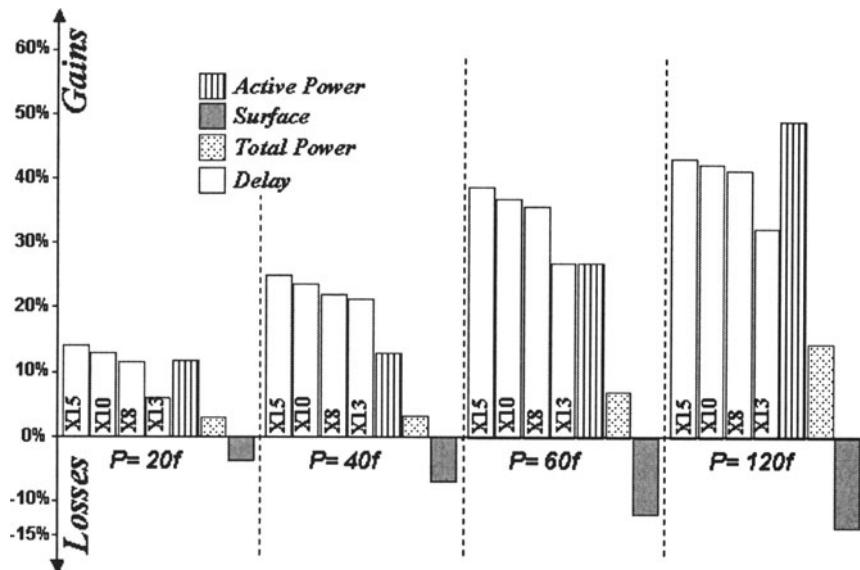


Figure 10. Gains and losses obtained for on the inverter tree plotted in Fig.9 with respect to the parasitic capacitance  $P=P_{3,4}=P_{5,6}$

## 7. CONCLUSION

Considering the power dissipation as a critical design parameter we have presented a sizing criterion for minimising the switching power dissipation component. This has been obtained by lowering the short circuit component through a control of the gate input transition time. Using an analytical model of the short circuit power dissipation and of the output transition time we showed that a sizing condition, that minimises the short circuit component, can be defined. Application has been given to general inverter configurations in various loading conditions. Comparison to minimal size implementations clearly shows that gain in power and speed as large as 15 and 45% can be obtained.

## 8. REFERENCES

- [1] H.J.M. Veendrick "Short circuit power dissipation of static CMOS circuitry and its impact on the design of buffer circuits" IEEE J. Solid State Circuits, vol. SC-19, pp.468-473, Aug. 1984.
- [2] A. Hirata, H. Onodera, K. Tamaru "Estimation of Short-Circuit Power Dissipation for Static CMOS Gates" IEICE Trans. Fundamentals, vol. E79-A, N°3 March 1996
- [3] S. Turgis, D. Auvergne "A novel macromodel for power estimation for CMOS structures" IEEE Trans. On CAD of integrated circuits and systems vol.17, n°11, nov.98.
- [4] T. Sakurai, R. Newton "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas" IEEE J. of Solid State Circuits, Vol. 25, N°2, April 1990
- [5] L. Bisdounis, S. Nikolaidis, O. Koufopavlou "Propagation Delay and Short Circuit Power Dissipation Modeling of the CMOS Inverter" IEEE Trans. On Circuits And Systems-I: Fund. Theory And Applications, Vol.45, N°3, March 1998
- [6] M. R.C. Berkelaar, P. H. W. Buurman, J. Jess "Computing The Entire Area /Power Consumption Versus Delay Tradeoff Curve For Gate Sizing With A Piecewise Linear Simulator" IEEE Trans. On CAD Of I.C. And Sys., Vol. 15, N° 11, Nov. 1996.
- [7] S. Saptnekar, V. B. Rao, P. Vaidya, S. M. Kang "An Exact Solution To The Transistor Sizing Problem For CMOS Circuits Using Convex Optimization" IEEE Trans. On CAD Of Integrated Circuits And Systems, Vol. 12, N° 11, Nov. 1993.
- [8] M. Borah , R. Owens, M. Irwin "Transistor Sizing Power Consumption Of CMOS Circuits Under Delay Constraint" Int. Symp. On Low Power Design 95, P.167
- [9] A. Chandrakasan, S. Sheng, R. Brodersen "Low-Power CMOS Digital Design" IEEE J. Of Solid State Circuits, Vol. 27, N° 4, April 1992
- [10] K. Usami M. Horowitz "Clustered voltage Scaling Technique for Low-Power Design", in proc. of Int. Symp. on Low Power Design 95, pp 3-8
- [11] P. Maurine , M. Rezzoug, D. Auvergne "Output transition time modeling of CMOS structures" To be published in Mai 2001 in the proc. of the IEEE Int. Symp. on Circuits And Systems, Sydney, Australia
- [12] P. Maurine , M. Rezzoug, D. Auvergne "Internal power dissipation modeling and minimization for submicronic CMOS design" PATMOS'2000: Gottingen, Germany. Sept 13-15, 2000, pp.531-536
- [14] J. Daga, D. Auvergne "A Comprehensive Delay Macro-Model of Submicron CMOS Logic" IEEE J. of Solid States Circuits, vol 34, n°1, pp.42-55, January 1999.

# Modeling and design of asynchronous priority arbiters for on-chip communication systems

J-B. Rigaud<sup>(†)</sup>, J. Quartana<sup>(\*)</sup>, L. Fesquet<sup>(†)</sup>, M. Renaudin<sup>(†)</sup>

<sup>(†)</sup>*TIMA Laboratory, 46 av. Felix Viallet 38031 Grenoble, France*

<sup>(\*)</sup>*Ph.D. work in co. with STMicroelectronics, AST Grenoble Lab, France*

*jean-baptiste.rigaud@imag.fr, jerome.quartana@st.com, laurent.fesquet@imag.fr,  
marc.renaudin@imag.fr*

**Abstract:** This paper addresses the design of complex arbitration modules, like those required in SoC communication systems. Clock-less, delay-insensitive arbiters are studied in the perspective of making easier and more practical the design of future GALS or GALA SoCs. The paper focuses on high-level modeling and delay-insensitive implementations of fixed and dynamic priority arbiter. Pre-layout simulations show that arbiters which are able to process several hundreds mega requests per second can be designed using the 0.18  $\mu\text{m}$  CMOS process of STMicroelectronics.

**Key words:** Asynchronous systems, priority arbiters, high-level modeling, CHP language, QDI circuits.

## 1. INTRODUCTION

One of the critical components of a SoC is the communication system, commonly named on-chip bus. Such an on-chip communication system has to be very flexible to interface in-house and external virtual components, providing high bandwidth, low latency, low power, arbitration mechanisms and routing capabilities.

In a SoC the on-chip bus connects the components to each other and dynamically allocates a path from one block to another. Several blocks running concurrently may require accessing the same resource leading to contentions. In this case, an arbiter is needed to solve the conflicts and to

ensure that only one block is accessing the resource. The choice is done with the help of priorities affected to each request.

Several arbitration algorithms were proposed in the past to solve the problem of accessing a unique resource from an arbitrary number of blocks. These algorithms can be classified according to the characteristics of their corresponding hardware implementation. To mention a few, arbitration structures can be distributed or centralized, can be linear like daisy-chain arbiters, or ring-based like token-ring and round-robin arbiters.

Most on-chip communication systems and the arbitration modules they include are today designed with synchronous circuits. In this paper, delay-insensitive asynchronous arbiters are considered, to be part of future on-chip busses of GALA (globally asynchronous locally asynchronous) or GALS (globally asynchronous locally synchronous) SoCs.

In the SoC design perspectives, delay-insensitive arbiters have this main advantage of being hundred-percent reliable (enough time is given to resolve metastability). Today, reliability of on-chip communication systems is a major issue since the increase transaction rates is drastically reducing the so-called Mean Time Before Failure characterizing clocked synchronizers.

As far as power consumption is concerned, such event driven communication/arbitration structures have a minimal electrical activity. Indeed, unlike clocked circuits, power consumption of delay-insensitive asynchronous arbiters is proportional to access rates [6]. Furthermore, delay insensitivity enables the design of fast “long distance” communication busses [5]. Finally yet importantly, such delay-insensitive communication systems are fully autonomous blocks, which can easily be reused in complex SoC architectures as soft, firm or hardware virtual components, hence decreasing design time and complexity.

Based on these motivations, the paper contributes to two fundamental issues: arbitration algorithms are modeled using a high-level description language called CHP (Communicating Hardware Processes), and the delay-insensitive arbiter architectures are derived from these CHP specifications.

Section 2 of the paper describes the global design flow from algorithm modeling to circuit synthesis and electrical simulation. Section 3 and section 4 present both the modeling and the circuit design of respectively fixed and dynamic-priority arbiters. Section 5 reports pre-layout simulation results. The last section concludes this work and mentions the main prospects.

## **2. DESIGN FLOW**

An overview of the global design methodology used to synthesize asynchronous circuits is presented in figure 1. The flow starts from a high-

level modeling using the CHP language (Communicating Hardware Processes). The CHP language, initially proposed by Martin [2], is naturally adopted in this work because i) it includes non deterministic choice structures required to model arbitration, and ii) it is very well suited to model and synthesize delay-insensitive circuits [2][3]. In [4], some new features were added to the CHP language to satisfy simulation requirements.

Delay-insensitive gate-level implementations are derived from the CHP specifications following a method that is not fully automated so far. However, the schematics are obtained following a formal procedure that is beyond the scope of this paper and will be detailed in future communications. It is based on the formal methodology presented in [2].

The design flow used is similar to the one presented in [4]. Gate-level implementations of CHP programs are all described using VHDL gate netlists that are verified by back-annotable logic simulations with timing. Two kinds of libraries are targeted: a standard-cell library and a specific asynchronous-cell library including Muller-gates. For each asynchronous cell, a VHDL functional view (including timing information), a schematic view and a layout view were developed. Then, circuit netlists are imported into the Cadence<sup>®</sup> framework for electrical simulations, placement and routing. The technology used for this study is the 0.18  $\mu\text{m}$  CMOS process from STMicroelectronics.

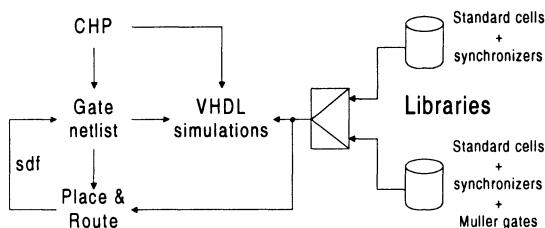


Figure 1. Design flow overview

All implementations use a four-phase handshaking protocol. Single-rail input channels are used for the fixed-priority arbiter, whereas dual-rail input channels are used for the dynamic-priority arbiter. The request signal associated to the single-rail channel R is denoted  $R_{\cdot i}$ . Signals  $R_{\cdot i\_0}$  and  $R_{\cdot i\_1}$  denote the two rails of the dual-rail input channel R. Signal  $R_o$  denotes the acknowledge signal of input channel R. Finally, the shared resource is assumed to be accessed through a single-rail channel S.

In the schematics, gates marked with a C are Muller C elements. An R (respectively S) is added when the gate output needs to be set to zero (respectively one) at reset. A "+" is added to specify asymmetric behavior. Input marked with a + only contributes to drive the output to one.

### **3. FIXED-PRIORITY ARBITER**

Fixed-priority arbiters have to choose among input requests with predefined hardware-coded priority values. This section considers several arbitration algorithms: a daisy chain, a particular binary-tree structure and a recent priority-arbitrer architecture proposed in [1].

#### **3.1 Fixed-priority arbiter modeling**

##### **3.1.1 Daisy-chain arbiter**

The CHP program of figure 2 describes a 4-input daisy-chain arbiter implementing a sequential priority scheme. The corresponding design is shown in figure 5. The circuit is by default in sleep mode, and wakes up each time an activity is detected on at least one of the input requests. The probe CHP operator (denoted #) is used to watch activity on inputs and trigger processing (Figure 2, line 1). Priority is simply modeled by sequentially testing the requests. Highest priority input R3 is analyzed first. It is tested active, #R3 statement in figure 2 (line 2). When active, the shared resource S is attributed to R3 by writing to S using the CHP write statement S!. Concurrently (";" CHP notation), request R3 is granted using the CHP read statement denoted R3?. The channel R3 is also tested not active, statement !#R3 in figure 2 (line 3). In this case, next channel R2 is analyzed the same way, and so on for R1 and R0.

Note that the stability of guards #R3 and !#R3 is not guaranteed [2]. In fact, the request signal level R3 may change while it is evaluated. In this case, a non-deterministic choice denoted "@@" is used in CHP. At the hardware level, it corresponds to using a synchronizer in charge of solving metastability that may occur while deciding whether R3 is zero or one.

```
* [ #( R0 ∨ R1 ∨ R2 ∨ R3 )
→ [ #R3 → S!, R3?
@@ !#R3 → [ #R2 → S!, R2?
@@ !#R2 → [ #R1 → S!, R1?
@@ !#R1 → S!, R0?
1 1 1 ]
```

*Figure 2. Daisy-chain arbiter CHP model*

##### **3.1.2 Binary-tree arbiter**

Another interesting arbitration structure, a binary-tree arbiter, is described in figure 3 and the corresponding gate level implementation presented in figure 6. The program clearly shows the tree structure of the selection process.

```

* | #( R0 ∨ R1 ∨ R2 ∨ R3 )
→ [ || #R3 → s1 := 2           (1)
  @@ #R3 → [ #R2 → s1 := 1
  @@ #R2 → s1 := 0
|| ],
[ || #R1 → s0 := 2           (2)
  @@ #R1 → [ #R0 → s0 := 1
  @@ #R0 → s0 := 0
|| ];
[ s1 = 2 → R3 ?, S!
@ s1 = 1 → R2 ?, S!
@ s1 = 0 → [ s0 = 2 → R1 ?, S!
@ s0 = 1 → R0 ?, S!
|| ]

```

Figure 3. Tree arbiter CHP model

Here again, line one of the program is used to sense input activities. Part one and part two of the program (respectively labeled 1 and 2 in figure 3), concurrently solve R3/R2 and R1/R0 contentions in the first stage. Priority of R3 over R2 (respectively R1 over R0) is modeled by a two-stage linear structure first checking R3 (respectively R1) and then R2 (respectively R0). Then, the CHP sequential operator “;” is used to specify that the concurrent parts of the first stage have to complete before stage 2 can process (label 3 in figure 3). In this last stage, deterministic choices are used since variables s0 and s1 are stable (“@” notation). According to s0 and s1 values, the winning input is granted and the shared resource is concurrently accessed.

### 3.1.3 Parallel-request-sampler priority-arbiter

Recently, Bystrov, Kinniment and Yakovlev introduced in [1] an enhanced version of priority arbiters that decouples request signals sampling (synchronizer module) and contention solving (priority module). This structure outperforms previously proposed structures in terms of complexity and latency. Moreover, unlike the daisy chain and the tree arbiters, this new arbiter structure is strongly modular, and priorities can be modified by only redesigning the priority module. However, in [1] the design of this priority arbiter family was very intuitive and performed by hand.

Figure 4 gives a formal CHP specification of a 4-way parallel-request-sampler fixed-priority arbiter (request-sampler FPA) like those proposed in [1]. Line 1 is used to sense request activities. Then, four identical subparts model the concurrent sampling of the request signals (labeled synchronizer 3 to 0). Variables s0 through s3 are used to store the samples, which are then exploited by the fixed-priority module to figure out which input has to be elected (labeled fixed-priority module).

```

* [ #( R0 ∨ R1 ∨ R2 ∨ R3 )           -- loop control
  → [ [   #R3 → s3 := 1 -- synchronizer 3
        @@ !#R3 → s3 := 0
      ],
      [   #R2 → s2 := 1 -- synchronizer 2
        @@ !#R2 → s2 := 0
      ],
      [   #R1 → s1 := 1 -- synchronizer 1
        @@ !#R1 → s1 := 0
      ],
      [   #R0 → s0 := 1 -- synchronizer 0
        @@ !#R0 → s0 := 0
      ],
      -- fixed-priority module
      [   s3 = 1 → R3 ?, S!
        @ s3 = 0 → [   s2 = 1 → R2 ?, S!
                      @ s2 = 0 → [   s1 = 1 → R1 ?, S!
                        @ s1 = 0 → R0 ?, S!
      ],
      ]
    ]
  ]
]

```

Figure 4. Parallel-request-sampler fixed-priority arbiter CHP model of [1]

## 3.2 Fixed-priority arbiter design

### 3.2.1 Daisy-chain arbiter

The structure of the 4-way daisy-chain arbiter is described in figure 5.

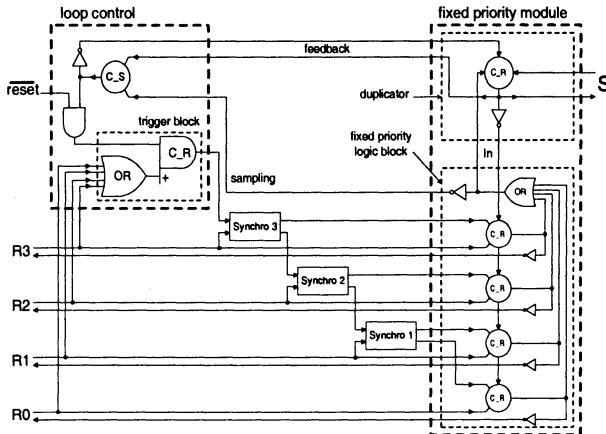


Figure 5. Daisy-chain arbiter architecture

It is composed of three blocks. The loop control block is in charge of reactivating the arbiter after an input request has been served and the shared resource accessed. This block includes the trigger block, which senses input activities to keep the arbiter quiet as long as no request occurs. The second block consists in a cascade of three synchronizers [2] that sequentially sample the input requests. The third block, the fixed-priority module, determines the input request to grant according to a priority order. Highest priority input R3 is sampled first and served if active. Otherwise, input R2 is analyzed the same way, and idem for R1. Note that no synchronizer is needed for R0: if no other input request is valid, input R0 is necessarily the one that activated the arbiter. The duplicator module controls the shared resource S.

### 3.2.2 Binary-tree arbiter

The binary-tree arbiter structure is presented in figure 6. Two pairs of cascaded synchronizers concurrently sample and analyze the input requests according to a priority order. Highest priority is R3 over R2 (respectively R1 over R0). Then a delay-insensitive logic implements a deterministic comparison in order to select the input to serve.

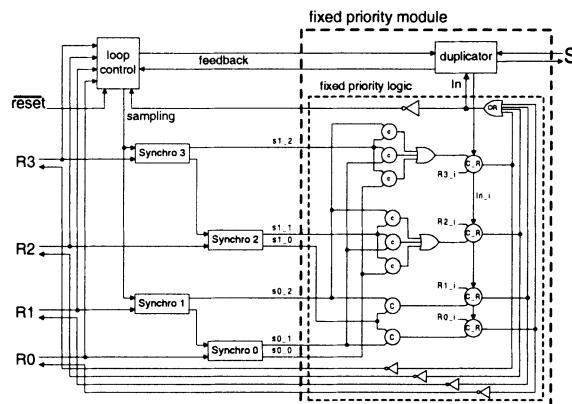


Figure 6. Tree arbiter architecture

### 3.2.3 Parallel-request-sampler FPA

The architecture of the 4-way parallel-request-sampler FPA of [1] is described in figure 7. The four synchronizers operate concurrently to sample the input request signals. The fixed-priority module determines the input request to grant according to hardware-coded priorities.

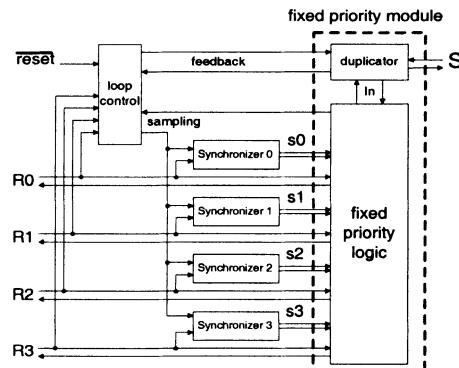


Figure 7. Parallel-request-sampler fixed-priority arbiter architecture of [1]

## **4. DYNAMIC-PRIORITY ARBITER**

In a dynamic-priority arbiter, each input channel is carrying requests as well as priority values. The dynamic-priority arbiter performs priority comparisons of active requests and grants the one with the highest value. The input having the highest index is by convention selected in case of identical priority levels.

According to the performances of the enhanced version of priority arbiters presented in [1], the structure of the parallel-request-sampler arbiter is implemented in this section with a dynamic-priority resolution algorithm. The time overload to resolve dynamic contention is compared in §5 with the fixed-priority version presented in figure 4.

### **4.1 Dynamic-priority arbiter modeling**

Figure 8 presents the program of a 4-way parallel-request-sampler dynamic-priority arbiter (request-sampler DPA). Each input  $R_i$  is a dual rail channel, encoding the request signal together with a priority value ranging from 0 (lower) to 1 (higher). Only two priority-levels are considered for the sake of simplicity. This can of course easily be extended and adapted to the requirements of real applications.

The first stage is very similar to that of the fixed-priority version of the arbiter (subparts labeled synchronizer 3 to 0). The priority module is much more complicated because it has to perform priority-level comparisons of competing inputs. The program proposed in figure 8 describes a two-stage comparator structure, but any other structure could be modeled leading to different tradeoffs in terms of complexity, speed and power.

The first stage concurrently analyses pairs of inputs and if necessary concurrently performs priority comparisons. Variable  $v1\_0$  indicates: no request occurred (value 0), request  $R0$  occurred (value 1) or request  $R1$  occurred (value 2). The request with the highest priority is passed to the following stage. Variable  $v3\_2$  plays the same role for inputs  $R3$  and  $R2$ . Variables  $v1\_0$  and  $v3\_2$  cannot be both zero since priority comparators are triggered on requests' activity. If only one request comes out stage one, the corresponding input is immediately granted (label “1<sup>st</sup> stage acknowledgment” in figure 8). If two requests come out stage one, then another comparison has to take place. A multiplexer identifies the two requests in competition and sends their priority values to the second stage comparator (subpart labeled “2<sup>nd</sup> stage: 2-way priority comparator” in figure 8). Finally, the issue of the comparison together with the couple of the inputs in contention is used to figure out which input is the winner (subpart labeled 2<sup>nd</sup> stage acknowledgment in figure 8).

```

* [ #( R0 ∨ R1 ∨ R2 ∨ R3 )           -- loop control
-- synchronizer 3
→ [ [ #R3 → s3 := 1 , [ #R3 = 0 → r3 := 0
                           @ #R3 = 1 → r3 := 1 ]
      @@ ]#R3 → s3 := 0
    ],
-- synchronizer 2
[ [ #R2 → s2 := 1 , [ #R2 = 0 → r2 := 0
                           @ #R2 = 1 → r2 := 1 ]
      @@ ]#R2 → s2 := 0
    ],
-- synchronizer 1
[ [ #R1 → s1 := 1 , [ #R1 = 0 → r1 := 0
                           @ #R1 = 1 → r1 := 1 ]
      @@ ]#R1 → s1 := 0
    ],
-- synchronizer 0
[ [ #R0 → s0 := 1 , [ #R0 = 0 → r0 := 0
                           @ #R0 = 1 → r0 := 1 ]
      @@ ]#R0 → s0 := 0
    ];
-- dynamic-priority module
-- 1st stage: 2-way request analysis and priority
comparators
[ [ (s0, s1) = (0, 0) → v1_0 := 0
   @ (s0, s1) = (1, 0) → v1_0 := 1
   @ (s0, s1) = (0, 1) → v1_0 := 2
   @ (s0, s1) = (1, 1) → [ r1 ≥ r0 → v1_0 := 2
                           @ r1 < r0 → v1_0 := 1 ]
    ],
[ (s2, s3) = (0, 0) → v3_2 := 0
  @ (s2, s3) = (1, 0) → v3_2 := 1
  @ (s2, s3) = (0, 1) → v3_2 := 2
  @ (s2, s3) = (1, 1) → [ r3 ≥ r2 → v3_2 := 2
                           @ r3 < r2 → v3_2 := 1 ]
  ],
];
-- 1st stage acknowledgment
[ (v1_0, v3_2) = (1, 0) → comp := 0 , R0? , S!
  @ (v1_0, v3_2) = (2, 0) → comp := 0 , R1? , S!
  @ (v1_0, v3_2) = (0, 1) → comp := 0 , R2? , S!
  @ (v1_0, v3_2) = (0, 2) → comp := 0 , R3? , S!
-- multiplexer
[ (v1_0, v3_2) = (1, 1) → comp := 1 , C0 := r0, C1 := r2
  @ (v1_0, v3_2) = (1, 2) → comp := 1 , C0 := r0, C1 := r3
  @ (v1_0, v3_2) = (2, 1) → comp := 1 , C0 := r1, C1 := r2
  @ (v1_0, v3_2) = (2, 2) → comp := 1 , C0 := r1, C1 := r3
];
-- 2nd stage: 2-way priority comparator
[ comp = 1 → [ C1 ≥ C0 → out := 1
                @ C1 < C0 → out := 0 ]
  @ comp = 0 → skip
];
-- 2nd stage acknowledgment
[ comp = 1 → [ out = 1 → [ (v1_0, v3_2) = (1, 1) → R2? , S!
                           @ (v1_0, v3_2) = (1, 2) → R3? , S!
                           @ (v1_0, v3_2) = (2, 1) → R2? , S!
                           @ (v1_0, v3_2) = (2, 2) → R3? , S!
  ]
  @ out = 0 → [ (v1_0, v3_2) = (1, 1) → R0? , S!
                 @ (v1_0, v3_2) = (1, 2) → R0? , S!
                 @ (v1_0, v3_2) = (2, 1) → R1? , S!
                 @ (v1_0, v3_2) = (2, 2) → R1? , S!
  ]
  @ comp = 0 → skip
];
];

```

Figure 8. Parallel-request-sampler dynamic-priority arbiter CHP model of [1]

## 4.2 Dynamic-priority arbiter design

Figure 9 presents the architecture of the 4-way request-sampler DPA specified by the CHP program of figure 8.

The two 2-way request analyzer & priority comparator blocks concurrently compare two pairs of requests. Each of them delivers the priority-holder request. Details of the gate-level structures are given in figure 10. It can be seen that the 2-way priority comparator is only activated when both input requests are active. The first stage of acknowledgment detects if a unique request is active at the output of the two 2-way request analyzers and priority comparators. In that favorable case, the corresponding input is immediately granted and the shared resource accessed, preventing from useless power consumption and latency.

The gate-level detailed blocks for fixed and dynamic parallel-request-sampler arbiters are available in [7].

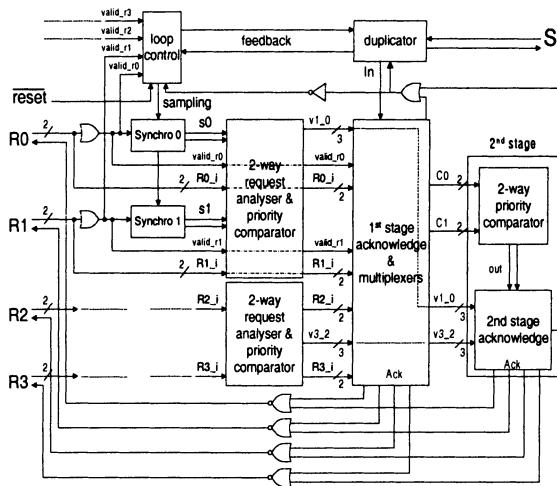


Figure 9. Parallel-request-sampler dynamic-priority arbiter architecture of [1]

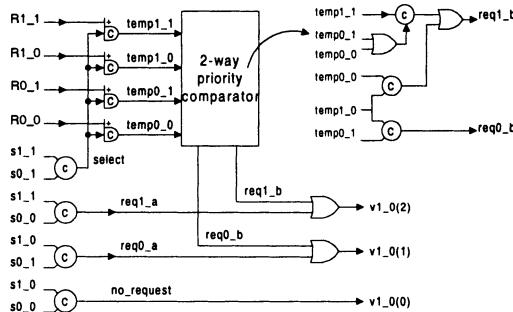


Figure 10. 2-way request analyzer and priority comparator

## 5. SIMULATION RESULTS

Table 1 summarizes latency and cycle time values obtained by electrical simulations of the gate-level netlists. The arbiters were simulated with two input-sets to measure best and worst case latencies.

For the daisy-chain arbiter, best and worst case latencies are respectively when R3 requests and when only R0 requests among the four inputs. For the tree arbiter, the best case corresponds to R1 active only and the worst case to R2 and R0 active. The parallel-request-sampler FPA best case occurs when R0 is the unique active request. The worst case occurs when only R1 is active. The dynamic-priority arbiter best and worst-case latencies were explained in Section 4.

|                     |            | Fwd latency | Cycle time | Throughput |
|---------------------|------------|-------------|------------|------------|
| Daisy chain arbiter | Best case  | 1.06 ns     | 1.66 ns    | 600 Mhz    |
|                     | Worst case | 1.42 ns     | 2.28 ns    | 440 Mhz    |
| Tree arbiter        | Best case  | 1.32 ns     | 2.15 ns    | 465 Mhz    |
|                     | Worst case | 1.52 ns     | 2.51 ns    | 400 Mhz    |
| Request-sampler FPA | Best case  | 1.26 ns     | 2.01 ns    | 500 Mhz    |
|                     | Worst case | 1.47 ns     | 2.46 ns    | 405 Mhz    |
| Request-sampler DPA | Best case  | 1.58 ns     | 2.46 ns    | 405 Mhz    |
|                     | Worst case | 2.53 ns     | 4.54 ns    | 220 Mhz    |

Table 1. Latency and cycle-time of simulated arbiters

**Fixed-priority arbiters** The daisy-chain arbiter responds here the fastest. It outperforms other structures in the best case of latency. However, the significant decrease of performance in the worst case confirms that the sequential sampling of the input requests strongly penalizes this structure. Furthermore, including wire delays in the simulations would heavily increase the latency. Such a linear architecture is not sufficient to cover the range of real applications, in which the number of peripherals is higher and in which the relative frequencies of the requesters have to be considered.

The binary-tree and the parallel-request-sampler arbiter present a more balanced structure, which exhibits similar delays for best and worst cases. When the number of inputs increases, the number of synchronizers increases linearly, whereas the fixed-priority logic complexity increases in  $O(\log_2)$ .

The advantages of the request-sampler FPA over the tree arbiter are: a simpler fixed-priority logic block and a one-level stage request sampler.

**Dynamic-priority arbiters** The dynamic-priority arbiter can sustain a request rate of about 405 MHz and 220 MHz. The modularity of the dynamic-priority arbiter enables an easy reconfiguration of the priority algorithm. Several priority levels offer a high degree of freedom for system-level designs.

**Optimizations** Using more aggressive optimizations based on fast handshaking components [8] would increase the speed. Another way to reduce the arbiter latency is to properly assign the arbiter inputs to requesters according to their respective priorities and request frequencies. For example, highest-priority requests should be connected to lowest-latency arbiter inputs in order to answer highest priority requests as quick as possible. Another strategy would be to connect lowest latency arbiter inputs to highest frequency requests in order to optimize average speed and power-consumption simultaneously.

## 6. CONCLUSION

In this paper, it is shown that arbiters can be modeled using the high-level language CHP and their corresponding delay-insensitive implementations derived. Pre-layout electrical simulations report that delay-insensitive priority arbiters processing several million requests per second can be designed using an up-to-date CMOS technology.

This work actually proves that it is today possible to cleanly and formally model and design delay-insensitive arbiter modules that are reliable, modular and fast. It also defines the fundamentals of an automated synthesis process devoted to arbiters. Finally, it constitutes an enabling factor for the asynchronous technology to be increasingly adopted in the design of SoCs.

Prospective works will be focused on the automation of the synthesis process and the improvement of arbiter architecture and circuit performances. “n initiators to p receivers” fixed or dynamic priority routers will also be investigated to address the design of complex on-chip routing systems.

## 7. REFERENCES

- [1] A. Bystrov, D. J. Kinniment, A. Yakovlev, "Priority Arbiters", in "International Symposium on Advanced Research in Asynchronous Circuits and Systems" (ASYNC), Eilat, Israel, April 2000, pp. 128-137.
- [2] A.J. Martin, "Synthesis of Asynchronous VLSI Circuits", Internal Report, Caltech-CS\_TR-93-28, California Institute of Technology, Pasadena, 1993.
- [3] J.-B. Rigaud, M. Renaudin, "Modeling and design/synthesis of arbitration problems", AINT'2000, Proceedings of the Asynchronous Interfaces: Tools, Techniques and Implementations Work-shop, TU Delft, The Netherlands, July 19-20<sup>th</sup> 2000, pp. 121-128.
- [4] M. Renaudin, P. Vivet, F. Robin, "A Design Framework for Asynchronous/Synchronous Circuits Based on CHP to HDL Translation", in "International Symposium on Advanced Research in Asynchronous Circuits and Systems", Barcelona, Spain, pp 135-144, April 19-21, 1999.
- [5] W.J. Bainbridge, S.B. Furber, "Delay Insensitive System-On-Chip Interconnect using 1-of-4 Data Encoding", in "International Symposium on Advanced Research in Asynchronous Circuits and Systems" (ASYNC), Salt Lake City, USA, March 11-14, pp 118-126, 2001.
- [6] C. Piguet, M. Renaudin, T. Omnes, "Special Session on Low-power SoCs", Design Automation and Test in Europe (DATE), Munich, Germany, March 2001, pp. 488-494.
- [7] J.-B. Rigaud, J. Quartana, L. Fesquet, M. Renaudin, "Modeling and design of asynchronous priority arbiters for on-chip communications systems", Proceedins of VLSI-SoC'01, 11<sup>th</sup> IFIP International Conference on VLSI, Montpellier, France, December 3-5 2001, pp. 424-429.
- [8] Andrew Lines, "Pipelined Asynchronous Circuits", Master Thesis, Caltech-CS-TR-95-21, 1995, revised June 1998.

# **Feasible delay bound definition**

N. Azemard, M. Aline, P. Maurine and D. Auvergne

*LIRMM : Laboratoire d'Informatique ,de Robotique et de Microélectronique de Montpellier,  
UMR 5506 Université Montpellier II //CNRS,  
161 rue ADA , 34392 Montpellier cedex 5, France  
Phone: (33) 4 67 41 85 21, Fax: (33) 4 67 41 85 00, E-mail: azemard@lirmm.fr*

**Abstract:** Minimizing the number of iterations when satisfying performance constraints in IC design is of fundamental importance to limit the design iterations. We present a method to determine the feasibility of delay constraint imposed on circuit path. From a layout oriented study of the path delay distribution, we show how to obtain the upper and lower bounds of the delay of combinatorial paths. Then we characterise these bounds and present a method to determine, , the average weighted loading factor allowing to satisfy the delay constraint. Example of application is given on different ISCAS circuits.

**Key words:** timing closure, delay bounds, fan out

## **1. INTRODUCTION**

Great interest [1-4] has been given to the research of optimal solutions to the problem of transistor sizing under delay constraint. But very few information is available on the direct determination of bounds on delay [5] allowing to evaluate the feasibility of constraints and/or the efficiency of an implementation. This evaluation is of great importance, for all stages of the design flow, in evaluating the quality of any synthesis or implementation style alternative.

In Fig.1 we illustrate the sensitivity of the delay of a path to the width of the transistors. For simplicity we have considered a uniform sizing ( $W$ ) of the transistors along the path but the trend observed is easily shown to be

conserved for more general irregular sizing conditions. As shown in this figure, for any circuit topology, a delay-area-power tradeoff can be defined [6-9].

This characteristic can be easily obtained for any path by varying the transistor sizes or, equivalently, the fan out factor [5,10,11], defined as the ratio of the output to input capacitance ( $F_{out} = C_{Load}/C_{IN}$ ) of the gates.

Considering a minimum transistor width implementation, the "maximum" delay value is obtained. We have to note here that this is a correctly designed maximum value, considering that any non justified extra loading on a path may lead to a greater value of the delay. This is also the minimum area solution but, due to controlling ramp effect, not necessarily the minimum power alternative [12,13]. On the other hand the value of the delay obtained for very large (and unpractical) transistor widths, gives the minimum delay that can be satisfied. If the plot in Fig.1 is determined from a post layout extraction of the path, this will give accurate bounds for the implementation from which optimization alternatives for delay-area-power may be selected. Obtained at the logical synthesis step, this plot will need to be updated after considering the interconnect parasitic capacitance introduced by the place and route step. This may induce numerous iterations implying prohibitive CPU time.

From first inspection, we can easily deduce that the difference  $\Theta_{Max} - \Theta_{Min}$  obeys a "b/W" law that will be justified later.

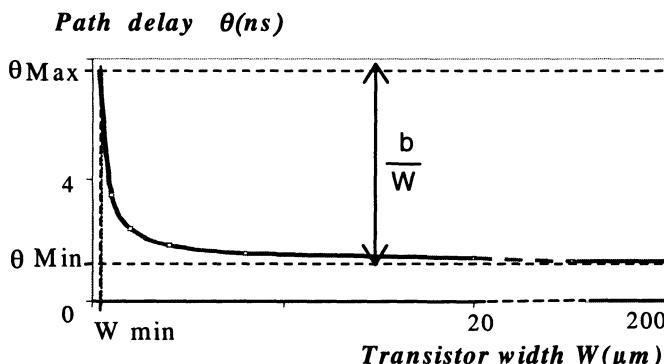


Figure 1. Path delay sensitivity to the transistor sizing.

We intend to show in this paper that the bounds of the delay can be easily determined and used to satisfy the delay constraint on critical paths, or, equivalently, to satisfy timing closure conditions at minimum iteration number.

This paper is organized as follows. In Section 2 we illustrate the path delay evolution of a standard circuit and demonstrate the existence of bounds

for delay. The upper and lower bounds of path delay are justified in Section 3. In Section 4, we present a new method for optimizing a circuit under delay constraint and we apply this new optimization method on ISCAS benchmarks. Conclusion is given in Section 5.

## 2. EVIDENCE OF DELAY BOUNDS

We have evaluated the delay profile of the 17000 paths of the C880 circuit (ISCAS benchmark) for different transistor sizing conditions, implemented in a  $0.25\mu\text{m}$  CMOS process. The resulting path distribution obtained with a path analyzer [14] is represented in Fig.2. Each curve corresponds to a specific uniform transistor size, varying from  $0.35\mu\text{m}$  to  $200\mu\text{m}$ . The layout of the circuit has been obtained from an automatic layout generator [15] and the timing analysis performed on a post layout extraction of the circuit. For a given transistor size ( $w$ ) the right limit of each curve determines the value of the critical path.

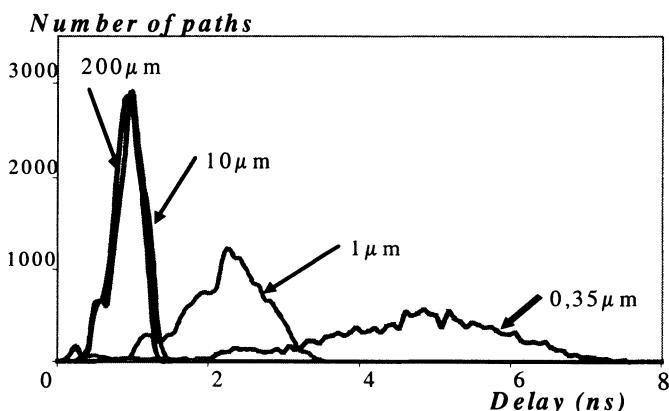


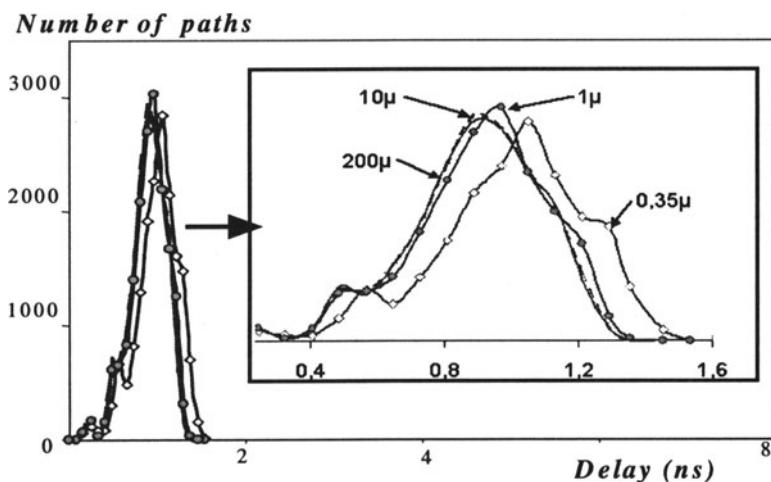
Figure 2. Distribution of delays on the paths of the C880 circuit implemented in a  $0.25\mu\text{m}$  CMOS process.

As expected from Fig.1 the distribution corresponding to the minimum transistor width results in the worse path delay with a widely spread distribution. The distribution obtained with very large transistor widths (nearly infinite for the process under consideration) identifies the minimum achievable delay on the circuit critical path. As predicted from Fig.1 the difference of distribution for large transistor sizing alternatives (10 and  $200\mu\text{m}$ ) is very small. The width of the distribution is also much narrow than

for the minimum size solution. This is a direct indication of the selected alternative for the path logical mapping.

Let us now only consider the parasitic diffusion capacitance. We obtain the delay distribution given in Fig.3 where all the distributions corresponding to different sizing conditions are superimposed. The weak difference of distribution (insert of Fig.3) will be identified later as due to the constant parasitic diffusion capacitance. We can observe, too, that the  $200\mu\text{m}$  sized distribution of Fig.2 is identical to that of Fig.3. The difference in distributions with smaller sizing is a direct illustration of the interconnect limited character of this minimum sized transistor implementation. This clearly shows that depending on the selected sizing alternative, the robustness of the circuit to design or process parameters may be exhausted at the expense (see Fig.1) of area and power consumption.

The distribution shown in Fig.4 has been obtained on the same circuit for two extreme sizing alternatives ( $200\mu\text{m}$  and  $0.35\mu\text{m}$ ) reducing the effect of the circuit parasitic capacitance to the parasitic diffusion part proportional to the transistor width. As shown, the two distributions are rigorously identical and directly give the limit of minimum delay feasible on this circuit.



*Figure 3.* Distribution of delays on the paths of the C880 circuit implemented in a  $0.25\mu\text{m}$  CMOS process, considering only the diffusion parasitic capacitance.

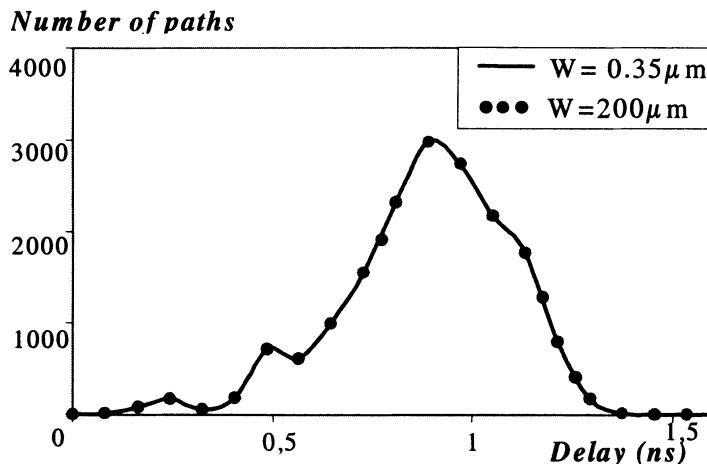


Figure 4. Distribution of delays on the paths of the C880 circuit implemented in a  $0.25\mu\text{m}$  CMOS process, considering only the transistor width dependent diffusion parasitic capacitance.

These figures illustrate an easy way to determine the reasonable bounds of delay in a circuit. It is just necessary to perform a delay profiling of this circuit, with minimum sized transistors, considering the complete parasitic capacitance or its value reduced to the transistor width-dependent diffusion part. If the full parasitic content is only known after the final circuit place and route, the reduced evaluation can be predicted much sooner. This supplies a very instructive indication on the efficiency of an implementation with respect to the imposed delay constraint. Or, alternatively, it ascertains the feasibility of the constraint that is of prime importance in evaluating architectural alternatives.

### 3. MODELING

As previously discussed, the path distributions in Fig.2-4 have been obtained from a timing evaluation based on a design oriented modeling of delays [16] allowing a good understanding of the performance sensitivity to the design and environmental parameters. In this representation the delay of the gate (i) is evaluated from

$$t_{\text{HL,LH}}(i) = A\tau_{\text{IN}} + t_{\text{HLS,LHS}}(i)(1 - \text{Cor}) \quad (1)$$

where  $\tau_{\text{IN}}$  is the input control ramp duration,  $t_{\text{HLS,LHS}}$  the fall or rise step response of the switching gate, Cor a correcting term associated to the carrier

speed saturation induced non linear variation of submicron process and  $A$ , a process dependent constant term [16].

$\tau_{IN}$  is directly associated to the step response of the controlling structure [16]. As a result, the switching delay of a gate can be expressed as a linear combination of step responses of the controlling and switching structures. Each step response can be evaluated from the ratio:

$$t_{HLS,LHS} = \frac{C_{Load} \cdot \Delta V}{I} \quad (2)$$

where  $\Delta V$  is the output voltage variation used to evaluate the step response and  $I$  the maximum current available in the structure to charge or discharge the output.

Applied to an inverter this results in:

$$t_{HLS,LHS}(INV) = \tau_{ST} \frac{C_{Load}}{C_{IN}} \quad (3)$$

where  $\tau_{ST}$  represents a metric for the process maximum speed, defined as the step response of an ideal inverter constituted of identically sized transistors, loaded by an identical one, and  $C_{Load}/C_{IN}$  is the fan out factor.

Extension to general inverter configurations and gates can be easily obtained by correcting eq.3 with a factor  $DW_{HL,LH}$  representing the ratio of current available, for each considered edge, between a gate and an inverter with identically sized transistors

This results in:

$$t_{HLS,LHS} = DW_{HL,LH} \cdot \tau_{ST} \frac{C_{Load}}{C_{IN}} \quad (4)$$

where  $DW_{HL,LH}$  introduced by [17] is a “digital weight” that allows to treat any structure of gate as an equivalent inverter. Its value has been explicitly defined in [18].

This demonstrates that with a very good approximation the delay on a path can be decomposed as a weighted sum of individual products: digital weight . fanout factor ( $DW \cdot Fout$ ).

Let us now consider the content of the terms in eq.4.  $C_{IN}$ , the gate input capacitance, is directly proportional to the transistor width.  $C_{Load}$  is generally constituted of active and parasitic capacitance components. The active capacitance defines the logic fan out ( $F_{Logic}$ ) of the switching gate (i). It is given by the sum of the input gate capacitance (i + 1) connected to the

output of gate (i), including the path and branch loading. The parasitic capacitance is constituted of the diffusion capacitance associated to the drains of the gate output (i) and of the interconnect metal capacitance. From direct inspection of a gate layout, the diffusion capacitance can be obtained from :

$$C_{\text{Diff}} = d_1 \cdot W + d_2 \quad (5)$$

where  $d_1$  and  $d_2$  are constants defined from the layout style and the design rules, and  $w$  is the transistor width of gate (i). The interconnect capacitance, known after the circuit place and route, is transistor width independent. Eq.5 can then be developed as:

$$DW(i) \cdot F_{\text{out}}(i) = DW(i) \cdot \frac{\sum_j C_{\text{IN}j} + C_{\text{Diff}}(i) + C_{\text{intercon}}(i, j)}{C_{\text{IN}}(i)} \quad (6)$$

$$= DW \cdot F_{\text{Logic}} + a + \frac{b}{W} \quad (7)$$

The total path delay is then obtained from the contribution of all the nodes (eq.4,7). This explains the evolution of the distribution given in Fig.2-4: for minimum sized transistor the first and third terms of eq.7 have a maximum value, this defines the value of the upper bound of delay. For large transistor sizes the third term is negligible. So  $F_{\text{Logic}}$  resumes to the average number of logical fan out and defines, with the transistor width dependent parasitic capacitance (“a” term), the minimum value of delay achievable on the path. Note as illustrated in Fig.3 and 4 that this minimum can also be reached considering only the “a” term for an implementation with equally sized transistors. This is illustrated in Fig.1 where the difference between upper and lower bounds for the delay is given by the parasitic contribution independent of the transistor width ( $b/W$  term).

#### 4. SIZING METHODOLOGY

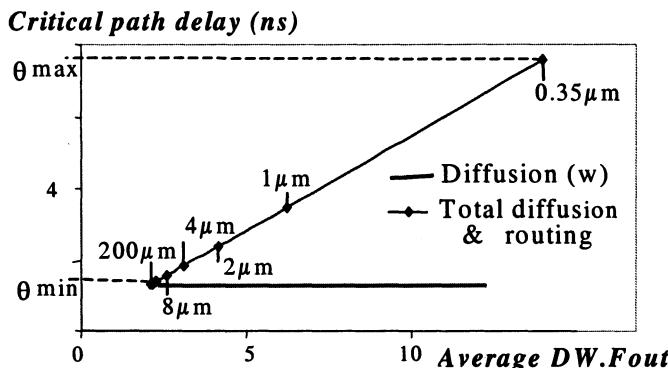
Any critical path can be characterised by delay bounds. Comparing the delay constraint imposed on the path to these limits gives indication of:

- the feasibility of the delay constraint,
- the cost of this constraint in terms of area (transistor width).

This can be a significant help in defining the average fan out factor to be imposed on the path nodes to satisfy timing constraints. This is illustrated in

Fig.5 where we plot for the C 880 circuit the variation of the delay with the average value of the product : "digital weight x fanout factor" of the path ( $DW.F_{out}$ ).

The delay value exhibits a linear variation between the min-max values previously defined. Indication is also given of the corresponding average transistor sizes for which, as illustrated in Fig.1, the delay is non linear. This gives the way to satisfy timing constraint on the path. To each constraint value corresponds an average value of the product  $DW.F_{out}$  which determines the sizing conditions of the different gates of the path.



*Figure 5. Illustration of the sensitivity of the critical path delay to the average value of the product "digital weight . fanout factor" of each gate on the path ( $DW.F_{out}$ ).*

This fan out factor can be determined as follows. Let us specify by  $\theta_{max,min}$ , the preceding bounds for delays and by  $F_{out,max,min}$  the corresponding fan out factors. From the preceding discussion the limits of delay variation shown in Fig.5 satisfy:

$$\begin{aligned}\theta_{max} &= DW \cdot (cF_{out\ max} + d) \\ \theta_{min} &= DW \cdot (cF_{out\ min} + d) \\ \theta_c &= DW \cdot (cF_{out} + d)\end{aligned}\quad (8)$$

where  $\theta_c$  is the delay constraint to be satisfied on each node of the circuit,  $DW$  and  $F_{out}$  is the fan out factor value to be imposed in order to satisfy the constraint. Solving for the product  $DW.F_{out}$  gives:

$$DW.F_{out} = \frac{1}{\Delta\theta} [\theta_c \Delta F + \theta_{max} F_{out\ min} - \theta_{min} F_{out\ max}] \quad (9)$$

where  $\Delta\theta$  and  $\Delta F$  represent the  $\theta_{\max}-\theta_{\min}$  and  $F_{\max}-F_{\min}$  differences as previously defined.  $c$  and  $d$  are the coefficients of the width-dependent and constant term of the parasitic contribution to the delay as given from eq.7 and illustrated in Fig.5.

In Table 1 we summarise the values of these parameters for different ISCAS benchmarks.  $\theta_{\min,\max}$  correspond to the previously defined bounds for delay evaluated on the post lay out description of the critical path of the circuits implemented in a  $0.25\mu\text{m}$  process.  $F_{\text{out},\min,\max}$  are the corresponding average fan out factors. The last column gives the number of gates on the considered critical path.

Considering a delay constraint  $\theta_c$  on these paths, it is then easy, using eq. 7 and 9, to determine the value of the average  $DW.F_{\text{out}}$  factor to be imposed on the path.

Table 2 summarizes the results obtained on the preceding circuits, considering for simplicity equally sized transistors.

|               | $\theta_{\min}$<br>(ns) | $\theta_{\max}$<br>(ns) | $F_{\text{out},\min}$ | $F_{\text{out},\max}$ | Gate<br>nb. |
|---------------|-------------------------|-------------------------|-----------------------|-----------------------|-------------|
| C1908         | 1.9                     | 12.3                    | 2.14                  | 16.5                  | 41          |
| adder<br>2x16 | 3.6                     | 17.5                    | 1.7                   | 8.8                   | 100         |
| C499          | 1.3                     | 7.9                     | 2.1                   | 14.7                  | 29          |
| C1355         | 1.4                     | 10                      | 2.3                   | 18.5                  | 28          |
| FPD           | 0.44                    | 1.82                    | 1.36                  | 6.32                  | 14          |
| C880          | 1.3                     | 7.6                     | 2.1                   | 14                    | 29          |

Table 1. Example of values of delay constraints and fan out parameters for different ISCAS benchmarks.

|               | $\theta_c$<br>(ns) | $DW.F_{\text{out}c}$ | $W_c$<br>( $\mu\text{m}$ ) | $\theta_{\text{sim}}$<br>(ns) |
|---------------|--------------------|----------------------|----------------------------|-------------------------------|
| C1908         | 3                  | 3.7                  | 3.1                        | 2.97                          |
| Adder<br>2x16 | 6                  | 2.9                  | 2                          | 5.73                          |
| C499          | 2                  | 3.44                 | 3.2                        | 1.96                          |
| C1355         | 2.5                | 4.37                 | 2.7                        | 2.4                           |
| FPD           | 0.8                | 2.64                 | 1.35                       | 0.75                          |
| C880          | 2                  | 3.42                 | 3.18                       | 1.96                          |

Table 2. Delay values simulated on paths sized under the  $\theta_c$  delay constraint.

Considering aggressive values for the delay constraint ( $\theta_c$ ) (of the order of two times the minimum available delay) we deduced the values of the

loading factor ( $F_{outc}$ ) and transistor width ( $W_c$ ) to be imposed on the different paths. The last column gives the value of the delay ( $\theta_{sim}$ ) simulated on the critical path after sizing all the transistors at the width defined from the delay constraint. As shown the resulting  $\theta_{sim}$  values are in good agreement with the  $\theta_c$  imposed constraint, demonstrating the validity of this approach.

We have to note that if the column  $W_c$  of Table 2 gives the average transistor sizing value. However eq.9 supplies an easy way to obtain a specific sizing of each gate that is digital weight dependent.

## **5. CONCLUSION**

Minimizing the number of iterations in satisfying timing constraints imposes as well a good prediction of the place and route interconnect capacitance than a good knowledge of the feasibility of the constraint imposed on the different circuit parts. We have first defined and determined the upper and lower bounds of delay on combinatorial paths. Then we characterized these bounds from which min max average values of the product (digital weight.loading factor) have been deduced. To each delay constraint we have associated an average value of the digital weight.loading factor product that we defined. Validations on ISCAS benchmarks ascertain the validity of the proposed methodology. Work in development considers gate level distribution of the delay constraint allowing specific gate sizing.

## **6. REFERENCES**

- [1] S.S. Sapatnekar, W.Chiang, "Power vs Delay in Gate Sizing: Conflicting Objectives", Proceeding of the IEEE/ACM ICCAD, pp. 463-466, 1995.
- [2] H.C. Chen, D.H.C. Du and L.R. Liu, "Critical Path Selection for Performance Optimization", IEEE trans. On CAD of Integrated Circuits and Systems, vol. 12, n°2, pp. 185-195, February 1995
- [3] N. Azemard, M. Aline, D. Auvergne, "Local Gate Resizing for Critical Path Optimization" DCIS'99, Palma de Majorque, Espagne, 16-19 November 1999.
- [4] E. Macii, M. Pedram, F. Somenzi, " High Level Power Modeling , Estimation and Optimization", IEEE trans. on CAD of Integrated Circuits and Systems, vol.17, n°11 pp. 1061-1079, Nov. 1998.
- [5] P. Rezvani, A.H. Ajami, M. Pedram, H. Savoj, "Fanout Optimization using a gain-based Delay model", In IWLS, Granlibakken, USA, 1999.
- [6] P.M.Vaidya, "A new algorithm for minimizing Convex Functions Over Convex sets", proceedings of IEEE Foundations of Computer Science, pp. 332-337, Oct 1989

- [7] M. Berkelaar, P. Buurman, J. Jess, "Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator", IEEE trans. on CAD of Integrated Circuits and Systems, vol.15, n°11 pp. 1424-1434, Nov. 1996.
- [8] I.E. Sutherland, R.F. Sproull, "Designing for Speed on the Back of an Envelops", Advanced Research in VLSI, Univ. Of Calif., Santa Cruz, 1991.
- [9] S.S. Sapatnekar, V.B. Rao, P.M. Vaidya, S.M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex functions", IEEE trans. CAD, pp. 1621-1634, Nov. 1993.
- [10] R. Murgai, "On the Global Fanout Optimization Problem", In IWLS, Granlibakken, USA, 1999.
- [11] C.L. Berman, J.L. Carter, K.F. Day, "The Fanout Problem: From Theory to Practice", In C.L. Seitz editor, Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conferences, pp. 69-99, MIT Press, March 1989.
- [12] D. Singh, J.M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, T.J. Mozdzen, "Power Conscious CAD tools and Methodologies: a Perspective", Proc. IEEE, vol.83, n°4, pp.570-593, April 1995.
- [13] S.Turgis, D. Auvergne, "A Novel Macromodel for power estimation in CMOS structures", IEEE trans. On CAD of Integrated Circuits and Systems, vol. 17, n°11, pp. 1090-1098, November 1998.
- [14] S.Cremoux, N.Azémard, D.Auvergne, "Path Selection based on Incremental Technique", Mixed Design of Integrated Circuits and Systems, Kluwer Academic Publishers, 1998.
- [15] JF. Moraes, M. Robert, D. Auvergne, "A virtual CMOS library approach for fast layout synthesis", VLSI'99, Lisboa, Portugal, pp. 415-426, December 1999.
- [16] J.M. Daga, D.Auvergne "A comprehensive delay macromodeling for submicron CMOS logics" IEEE J. of Solid State Circuits, vol. 34, n°1, pp. 42-55, January 1999.
- [17] I. Sutherland, B. Sproull, D. Harris, "Logical Effort: Designing Fast CMOS Circuits", Morgan Kaufmann Publishers, INC., San Francisco, California, 1999.
- [18] P. Maurine, M. Rezzoug, D. Auvergne " Output transition time modeling of CMOS structures" pp. V-363-V-366, ISCAS 01, Sydney, Australia

# **CMOS Mixed-signal Circuits Design on a Digital Array Using Minimum Transistors**

**Jung Hyun Choi and Sergio Bampi**

*Microelectronics Group (GME), Informatics Institute, Federal University of Rio Grande do Sul (UFRGS). Av. Bento Gonçalves, 9500 - Campus do Vale - Bloco IV. P.O. 15064 - 91501-970 - Porto Alegre - RS – Brazil. Phone/Fax: +55-51-33167036 / 33167308*

**Abstract:** This paper presents an analog semi-custom design using association of minimum channel length transistors for mixed-signal system applications on pre-diffused digital arrays. The Trapezoidal Association of Transistors (TAT) mimics the behavior of a full-custom single transistor, which should have an arbitrary geometry, using only a composition of regularly and linearly placed digital-based transistors in an array like SOT (Sea-Of-Transistors) style. TAT transistor achieves an analog performance that is similar to the single device. Basic analog building blocks were fabricated and tested using both semi-custom and full-custom design methodologies. Simulation and experimental results are for technologies at CMOS 1.0 $\mu$ m and 0.5 $\mu$ m.

**Key words:** TAT transistor, Pre-Diffused SOT array, Analog and Mixed-Signal Design, Building Blocks

## **1. INTRODUCTION**

To rapid ICs design, pre-diffused digital arrays are largely employed, which are good alternative to decrease the design turnaround and prototyping time. Silicon area and speed are often larger than the equivalent full-custom version. Advantages are that the design is at high level of abstraction and the remaining steps are automated (positioning&routing), use of pre-defined cells and IP (Intellectual Property) blocks from the libraries.

Available technologies are not well appropriate for analog applications and analog technologies are still costly alternative and mixing analog-digital circuits can be prohibitive. Therefore, several performance issues come up when implementing an analog circuit in an environment traditionally

dedicated for digital applications. Noise, DC and AC performances of MOSFETs are poor compared to bipolar devices. Mixed-signal and analog-digital circuits placed side-by-side on same silicon chip causes cross-talks and increased noise levels. Even in the traditional design methodologies, analog design is not straightforward. The design difficulty increases in the digital technology environment. It demands better device modeling and new design techniques. Also it needs improvement on CAD tools that reflect more accurately the linear and non-linear behavior of the devices, counting RF behavior and effects.

Consequently, analog design is a difficult task, as the digital array and technology are designed to stress the performance requirements of the large digital blocks, that most likely comprise the overwhelming majority of the transistors effectively used in the array. The alternative to overcome and/or improve on these characteristics is applying a specific discipline for designing the analog blocks on fixed-size transistors, that is, the principle of TAT. Unit transistors on SOT array were designed in such a way to facilitate migration to down-scaled technologies taking into account solely the digital circuits demand for speed (only  $L_{min}$  transistors) and adequate average digital load drive for local routing. Comparisons between TAT and single versions are presented for common-source amplifier, comparator and folded-cascode OTA designed and fabricated in both methodologies.

## **2. THE TAT TRANSISTOR TECHNIQUE**

The principle of TAT transistor was demonstrated earlier at the device level by Riccò [3], showing that enlarging the channel width at the drain end of the channel (trapezoidal shape – small source, large drain), shown in Fig. 1a, results in a substantial improvement on the output conductance in saturation region. Furthermore, Galup-Montoro [2] established that the series-parallel association of discrete transistors emulates an equivalent single transistor.

The TAT is a combination of the previous techniques, i.e., a trapezoidal series-parallel association of transistors, shown in Fig. 1b, using only minimum channel length transistors. The transistor variable sizes and geometry for electrically equivalent MOS channel lengths and widths required for analog design is achieved with appropriate association of the unitary transistors available on the SOT array. Details of this technique are better described in [1], [6].

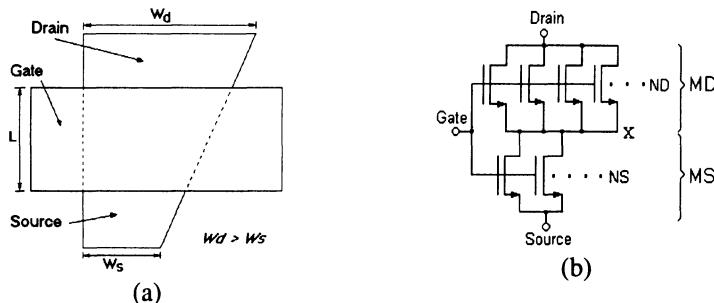


Figure 1. (a) Trapezoidal MOSFET [3]. (b) TAT: series-parallel association of unitary transistors.

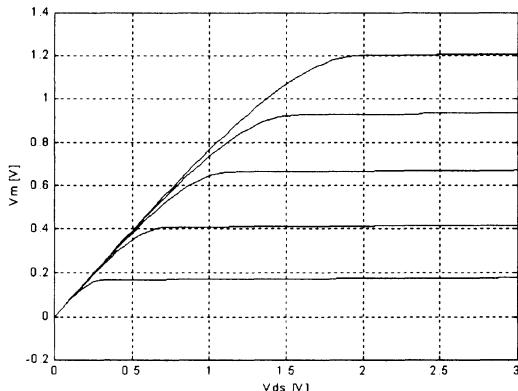


Figure 2. : TAT: simulated  $V_x \times V_{DS}$ . ND=17, NS=4,  $L_{min}=0.5\text{mm}$ .

Consider the series transistors, MD and MS at the drain and source ends, respectively shown in Fig. 1b. The equivalent conductance (linear region) is given by an equivalent aspect ratio of the association as [2]:

$$\left(\frac{W}{L}\right)_{Eq} = \frac{\left(\frac{W}{L}\right)_D \left(\frac{W}{L}\right)_S}{\left(\frac{W}{L}\right)_D + \left(\frac{W}{L}\right)_S} \quad 2.1$$

where  $(W/L)_D$  and  $(W/L)_S$  are, respectively, the aspect ratios of the top parallel association MD and the bottom parallel association MS transistors. Equation 2.1 can be rewritten as follows:

$$\left(\frac{W}{L}\right)_{Eq} = \frac{m}{m+1} \left(\frac{W}{L}\right)_s \quad 2.2$$

with:

$$m = \frac{\left(\frac{W}{L}\right)_D}{\left(\frac{W}{L}\right)_S} \quad (m > 1) \quad 2.3$$

Taking into account the limit of the function in 2.2, from  $m=1$  to  $m \rightarrow \infty$ , TAT aspect ratio will vary between a half of the MS aspect ratio and its physical individual value. Therefore, we must impose an aspect ratio to the MS greater than the required aspect ratio and smaller or equal than twice this value in order to get a trapezoidal association.

Regarding again Fig. 1b, to achieve improvement (decrease) on the output conductance [8], the channel width of MD must be larger than the width of the MS, as described in [2] and [3]. In fact, MD is obtained by having  $N_D$  unit transistors in parallel and similarly  $N_S$  unit transistors in parallel for MS, where  $N_S$  is smaller than  $N_D$ , in order to keep the principle of trapezoidal geometry. The association shown in Fig. 1b is the basic building block for analog design in a digital SOT array.

The intermediate node voltage  $V_x$ , Fig. 2, is clamped at a fraction of the pinch-off voltage of the MD source end terminal. This effectively limits the MS drain voltage to be the channel saturation voltage for a given  $V_G$ , which keeps MS in the linear region. MD and MS have the same gate voltage, then they have the same channel pinch-off voltage [9], [10], therefore, for both MD and MS in strong inversion, considering MD is saturated,  $V_x$  is given by:

$$V_x = \left[ 1 - \frac{1}{\sqrt{1 + N_D/N_S}} \right] V_P \quad 2.4$$

where  $V_P$  is TAT pinch-off voltage. Above expression shows clearly the advantage of the TAT to be trapezoidal, i. e.,  $V_x$  approaches to  $V_P$ . Since the transconductance of TAT is roughly the MS transconductance (equation 2.6) and it is in the linear region (onset to saturation), the TAT transconductance is directly proportional to MS drain-to-source voltage or  $V_x$ .

The MD and MS are the equivalent transistors obtained by several unit transistors physically laid out in parallel. Therefore, the equivalent threshold voltage is in fact an averaging over many unit transistor threshold voltages. This averaging in fact compensates the higher spread of  $V_T$  for minimum

lengths. Offset voltages should improve by using such composite TAT devices in deep sub-micron circuits.

TAT is trapezoidal and channel lengths are the same for both MD and MS transistors (only the electrically equivalent  $L_{eq}$  of TAT is variable). Then, MD is always made wider than MS and its width is not the minimum and the threshold voltage of MS will not be larger than that of MD, given the presence of body effect. Additionally, TAT can be used in several analog low voltage applications because it works as an intrinsic self-cascode (a second gate bias for MD is required for a traditional cascode).

Improvement, i.e. reduction, on the effective output conductance is effectively achieved for a TAT (in relation to minimum channel-length unit transistors), because it is similar to the traditional cascaded MOS transistors. Then, the output conductance is given by traditional expression:

From 2.5, the output conductance of MD  $g_{ds_{MD}}$  is represented by a large value, typical of the worst-case minimum channel length transistor, which is effectively reduced by the transconductance of MD. The effective transconductance of TAT transistor is given by:

$$g_{m_{TAT}} = \frac{g_{ds_{MS}}}{g_{mS_{MD}}} \left[ \frac{\frac{g_{mG_{MS}} \cdot (g_{mS_{MD}} + g_{S_{MD}})}{g_{ds_{MD}} + g_{mD_{MS}}}}{+ \frac{g_{mG_{MD}} \cdot g_{mD_{MS}}}{g_{mD_{MS}} - (g_{mS_{MD}} + g_{S_{MD}})}} \right] \equiv g_{mG_{MS}} \quad 2.6$$

According to 2.6, TAT transconductance is determined mainly by MS transconductance. Hence, to improve overall TAT characteristics one needs to increase the MS transconductance and improvement on the effective output conductance one has to increase mainly the output impedance or the MD transconductance. Thus, MD mainly determines the output conductance and MS determines the TAT transconductance.

In Fig. 3 is the experimental output conductance for TAT with channel length  $L=1.2\mu m$  and channel width  $W=51\mu m$ , compared to the equivalent single transistors. It shows clearly that the TAT output conductance is similar to the single transistor. At lower gate drive (1.5V) the short channel effects of  $V_T$  reduction and Drain Induced Barrier Lowering (DIBL) combine to give a larger  $g_{ds}$  both in conduction and saturation regions.

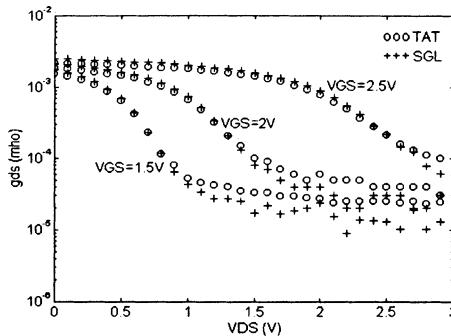


Figure 3. TAT: experimental output conductance at  $V_G=1.5V$ ,  $V_G=2V$  and  $2.5V$ . 0.5mm technology:  $L=1.2mm$ ,  $W=51mm$  ( $N_D=10$ ,  $N_S=10$ ,  $W_u/L_u=5.4/0.6$ ).

The total equivalent noise power at the input present in the TAT transistor is given by:

$$V_{n_{TAT}}^2 |_{in} = \left\{ \frac{N_D \cdot (g_{mG_{MD}}^2 |_u)}{[N_S \cdot (g_{mG_{MS}} |_u)]^2} + \frac{1}{N_S} \right\} \cdot V_{n_{f_k}}^2 |_u + \\ + \left\{ \frac{N_D + N_S}{[N_S \cdot (g_{mG_{MS}} |_u)]^2} \right\} \cdot I_{n_{th}}^2 |_u \quad 2.7$$

where  $V_{n_{f_k}}^2 |_u$  and  $I_{n_{th}}^2 |_u$  are the flicker noise and thermal noise of each unit transistor. Comparing above expression to the equivalent single transistor, TAT is noisier. Electrical simulation and experimental results (for OTAs) demonstrated that thermal noise is similar, however flicker noise is larger for TAT. The  $1/f$  noise can be minimized increasing the MS (or TAT) transconductance.

### 3. BASIC ANALOG BLOCKS USING TAT

Main parasitic capacitances in TAT are the gate-source capacitance from MD (in saturation) and from MS (in conduction) is the gate-source and gate-drain capacitance, shown in Fig. 4. The combination of parasitic capacitances is larger for the TAT than in its equivalent single transistor.

The presence of intermediate node in the TAT (pole) increases the total parasitic capacitance seen at the input. Nevertheless, due to the self-cascode effect the gain bandwidth is similar to the single transistor. Indeed, the product gain-bandwidth (GBW) is increased by this effect. In Fig. 5 it is shown clearly by electrical simulations for common-source amplifiers

(PMOS type current source). Channel lengths, widths and W/L ratios are exactly the same values for both TAT and single transistors for better comparison purposes, as indicated in the figure caption.

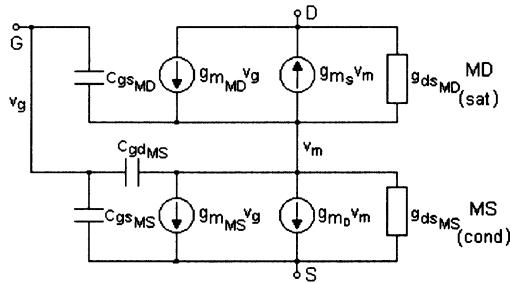


Figure 4. TAT: AC model with body effect.

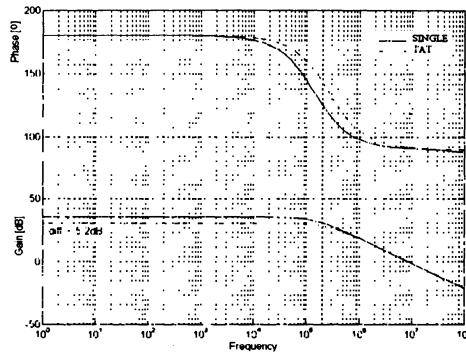
Table 1. Track-and-Latch comparator: measured performance in 0.5mm CMOS technology.

|                   | Full-custom | SOT array |
|-------------------|-------------|-----------|
| V <sub>DD</sub>   | 3V          | 3V        |
| C <sub>L</sub>    | 10pF        | 10pF      |
| f <sub>max</sub>  | 41.1MHz     | 30.5MHz   |
| V <sub>os</sub>   | 7.33mV      | 14.2mV    |
| I <sub>tail</sub> | ~ 80μA      | ~ 90μA    |

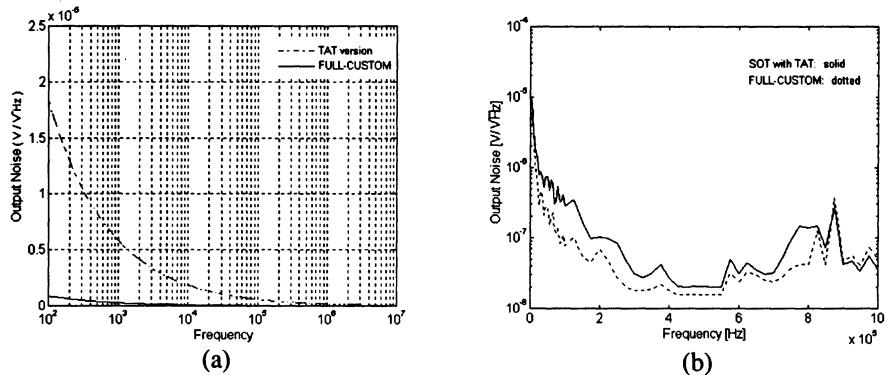
Table 2. OTA: measured performance in 1.0mm digital CMOS technology.

|                        | Full-custom | SOT array |
|------------------------|-------------|-----------|
| A <sub>v</sub> (db)    | 64.5        | 62.9      |
| GBW(MHz)               | 0.186       | 0.514     |
| @ C <sub>L</sub> (pF)  | 82.4        | 53.5      |
| GBW @ 50pF             | 0.307       | 0.550     |
| V <sub>os</sub> (mV)   | +30         | -15       |
| SR (V/μs)              | 0.42        | 1.10      |
| @ C <sub>L</sub> (pF)  | 82.4        | 53.5      |
| PM (°)                 | 79.4        | 51.7      |
| V <sub>o max</sub> (V) | +1.1/-0.7   | +0.9/-0.9 |

Fig. 6a shows the total equivalent noise power source at the input of common source amplifiers. It is clear that the flicker noise is almost 5 times larger in TAT transistor. In Table 1 is shown the simulated performance reached by the track-and-latch comparator. Maximum switching clock frequency for TAT version is approximately 25% lower and sensibility is twice larger. However, TAT has shown good performance.



**Figure 5.** Common-source amplifier: gain and phase margin. Electrical simulation using TAT (dotted) and single (solid) transistors. ( $W/L$ )<sub>eq</sub>=106/1.9 in  $0.5\mu\text{m}$ .



**Figure 6.** Total equivalent noise power in TAT (solid) and single (dotted) transistors. (a) Simulated noise for common-source amplifiers. ( $W/L$ )<sub>eq</sub>=106/1.9 in  $0.5\mu\text{m}$ . (b) Experimental noise for OTAs in  $1.0\mu\text{m}$ .

A single-ended folded-cascode OTA - Fig. 7b - in both SOT (using TAT transistors) and full-custom methodology [6], [7] were also implemented to allow better performance comparisons. In Table 2 the comparison of results for electrical simulations and experimental measurements are depicted for both SOT array and full-custom OTAs.

The load capacitances were estimated from electrical simulation of expected output current and measured slew-rate. It is worth noting that the SOT version of the OTA has similar (unit gain frequency 1.9x times higher) gain-bandwidth. It is expected because the TAT is an intrinsic cascode (self-cascode pointed out previously).

Offset voltage in SOT version of the OTA is smaller than the full-custom version due to its better intrinsic layout configuration, that is, naturally the TAT devices are very similar to the interdigitated layout technique that is widely used in full-custom layouts.

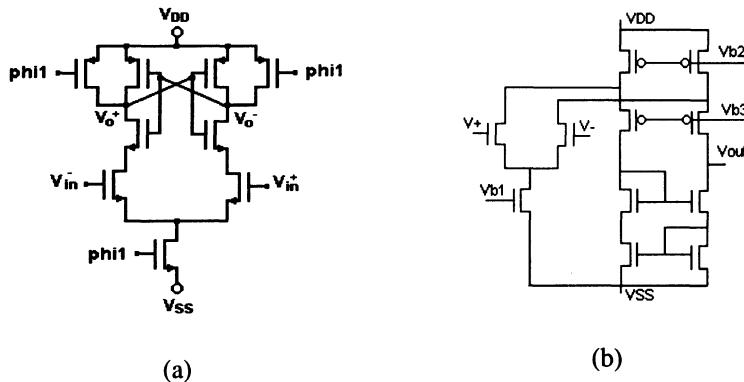


Figure 7. (a) Track-and-Latch comparator. (b) Folded-cascode OTA.

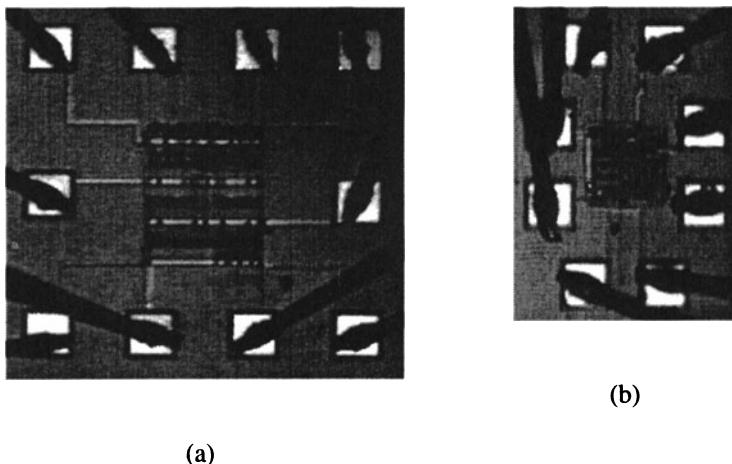


Figure 8. Die photos: OTA on SOT array (a) and the same OTA in full-custom (b).

Total noise power spectral densities measured in both OTAs are shown in Fig. 6b. The flicker noise in both are large due to its dependence on the OTA input differential pair geometry (area  $WL$ ) [4], [5]. Thermal noise in SOT version is almost similar (slightly higher) to the equivalent full-custom version, as predicted previously.

In Fig. 8 are shown the die photos of the folded-cascode and in Fig. 9 is shown the chip-test containing OTA, comparator and transistor structure in both SOT array (TAT devices) and full-custom.

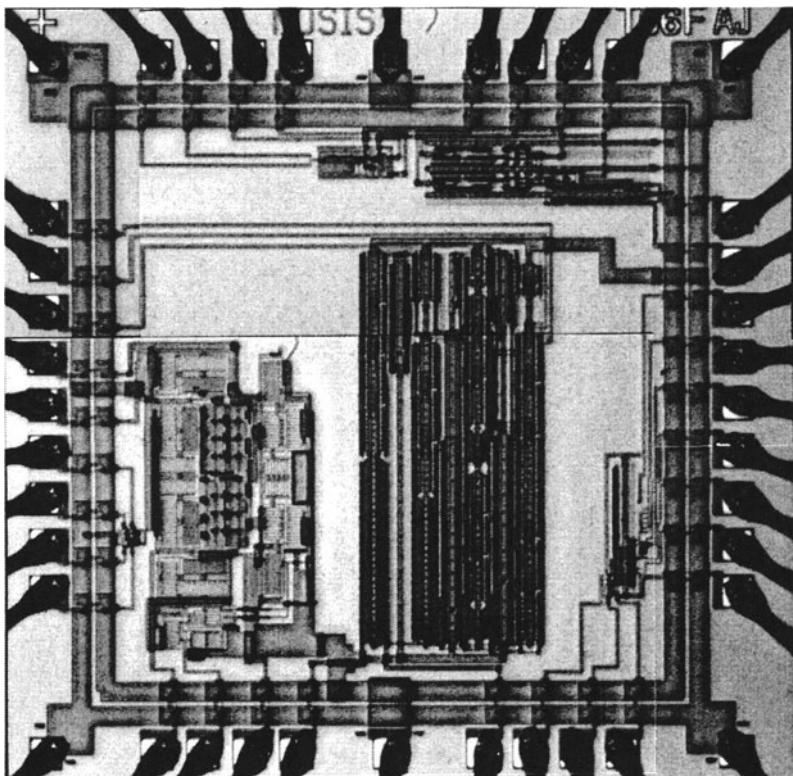


Figure 9. Test-chip die photo: Transistor structure, Track-and-latch comparator and folded-cascode OTA.  $0.5\mu\text{m}$  CMOS technology.

#### 4. CONCLUSION

This paper demonstrated the TAT technique, which can be used in many analog applications without major penalties. One has to deal with the demonstrated noise figures. It is noisier than the single transistor counterpart. However, using a large number of minimum-size transistors in the TAT, the worse characteristics with respect to noise can be compensated, while keeping the same equivalent  $W/L$ . TAT transistors, as shown herein, are not restricted only for semi-custom arrays. It can be used advantageously even in full-custom analog integrated circuits, as shown by the sub-circuits implemented in SOT array using TAT.

Even though the total noise is large and some natural limitations exist, TAT transistors in SOT methodology presented good performance. It is a good trade-off for a given application between performance, cost and design time turn-around.

## 5. ACKNOWLEDGEMENTS

The support of CAPES and CNPq Brazilian R&D Agencies and of the Analog VLSI Laboratory at OSU, USA, is gratefully acknowledged.

## 6. REFERENCES

- [1] Aita, A.L.; Bampi, S. "Design of Mixed Digital-Analog Circuits on a Digital Sea-of-Transistors", In: *International Symposium on Circuits and Systems*, Hong Kong. *Proceedings...*, ISCAS, 1997, p.2028-2031.
- [2] Galup Montoro, C.; Schneider, M.C.; Loss, I.J.B. "Series-Parallel Association of FET's for High Gain and High Frequency Applications", *IEEE Journal of Solid-State Circuits*, v.29, n.9, Sept. 1994, p.1094-1101.
- [3] Riccò, B. "Effects of Channel Geometries on FET Output Conductance in Saturation", *IEEE Electron Device Letters*, vol. EDL-5, Sept. 1984, p. 353-356.
- [4] Mikoshiba, H. 1/f Noise in n-Channel Silicon-Gate MOS transistors, *IEEE Transaction Electron Devices*, Vol. ED-29, June 1982, p.965-970.
- [5] Bertails, J. C. Low-Frequency Noise Considerations for MOS Amplifiers Design, *IEEE Journal of Solid-State Circuits*, Vol. SC-14, No. 4, Aug. 1979, p.773-776.
- [6] Choi, J. H.; Bampi S. Design of CMOS OTA Amplifiers and Oscillators in a Digital Sea-of-Transistors Array. In: 5<sup>th</sup> IEEE International Conference on Electronics, Circuits and Systems, ICECS'98. Lisboa, Portugal, 9-10/Sept/1998, p.321-324.
- [7] Choi, J. H.; Bampi, S. "Trapezoidal Association of Transistors for Mixed Analog-Digital Circuits Design on a SOT Array". In: IEEE International Workshop on Design of Mixed-Mode Circuits and Systems, III (Proceedings). Puerto Vallarta, July 1999, p.65-68.
- [8] Choi, J. H.; Bampi, S. "Conductances and Noise in Trapezoidal Association of Transistors for Analog Applications Using SOT Methodology". In: Symposium on Integrated Circuits and Systems Design, SBCCI. Proceedings... Natal, Brazil, Sept. 1999, p.22-25.
- [9] ENZ, C. High Precision Micropower Design, Ph.D. thesis, 1989, 198p.
- [10] ENZ, C. The EKV Model: a MOST Model Dedicated to Low-Current and Low-Voltage Analogue Circuit Design and Simulation, *Low-power HF microelectronics: a unified approach*, 1996, p.247-299.

# A VHDL-AMS Case Study:

*The Incremental Design of an Efficient 3<sup>rd</sup> generation  
MOS Model of a Deep Sub Micron Transistor*

C. Lallement, F. Pêcheux, Y. Hervé

*ERM-PHASE, ENSPS, Pôle API, Bld. S. Brant, 67400 Illkirch, FRANCE*

*E-mail: {lallem,pecheux,herve}@erm1.u-strasbg.fr*

**Abstract:** The paper presents an application of the VHDL-AMS formalism to state-of-the-art MOST simulation models. We present the principles, techniques and tools used to achieve the incremental implementation of an analytical third generation Spice transistor MOST model named EKV in VHDL-AMS, with relevant parameters set to match a deep submicron technology (gate length = 0.15 µm). The model includes the capacitances and resistors induced by the LDD structures as a function of gate voltage, and also considers thermo-electrical interactions between the transistor and its direct environment. Along with some considerations on the power of VHDL-AMS for modeling deep submicron devices, we give some examples of application of this innovative EKV MOS model.

## 1. INTRODUCTION

The design of innovative integrated devices, like MOEMS, involves a strong interaction of pluri-disciplinary objects on the same chip. These objects are tightly coupled by physical effects, and the corresponding exchanges can be modeled at various abstraction levels.

Our paper is an application of the innovative VHDL-AMS concepts to state-of-the-art MOS thermal-electronic simulation models. First, we present the principles, techniques and methodology used to achieve the design of an analytical third generation Spice transistor MOS model named EKV with

VHDL-AMS, with relevant parameters set [1] to match a deep submicron technology (gate length = 0.15  $\mu\text{m}$ ). Second, we add to this basic MOST model several characteristics specific to submicron technologies like the parasitic resistances and capacitances induced by the LDD structure. Third, we introduce the thermo-electronic interaction between the transistor and its environment and how it can simply be modeled in VHDL-AMS. Finally, we give some examples of application of the MOST model.

## **2. THE EKV MOSFET MODEL VERSION 2.6**

In the historical evolution of the MOSFET compact models, The EKV MOST Model belongs to the third and last generation [2]. The main characteristics of this generation are [3]:

- the “original intent” to simplicity (in contrast to the second generation: BSIM, BSIM2, HSPL28),
- a small number of physically-based parameters,
- an improved mathematical conditioning,
- a single model equation for all regions of device operation,
- the use of smoothing functions

The major models of this third generation (BSIM3v3, BSIM4, MM9, EKV2.6 and soon EKV3.0 and MM11) are dedicated to sub- and deep-submicron device.

The most used model in the design community (BSIM3v3, BSIM4) has forgotten the “original intent” of simplicity, and a small number of parameters in contrast with the other models [2]; this can be explained by the BSIM 3-4 quest for “extreme” precision and the modeling of all small effects, thus resulting in a complex core. On the contrary, the EKV MOST model keeps maintaining a good simplicity/efficiently ratio.

The EPFL-EKV MOSFET is a scalable and compact simulation model built on fundamental physical properties of the MOS structure. This model is dedicated to the design and simulation of low-voltage, low-current analog, and mixed analog-digital circuits using submicron CMOS technologies.

The EPFL EKV version 2.6 MOST model [4] is a charge-based compact model. It consistently describes effects on charges, transcapacitances, drain current and tranconductances in all regions of operation of the MOSFET transistor (weak, moderate, strong inversion) as well as conduction to saturation. The effects modeled in this model include all the essential effects present in submicron technologies. For quasi-static dynamic operation, both a charge-based model for the node charges and transcapacitances, and a simpler capacitances model are available.

## 2.1 The VHDL-AMS implementation

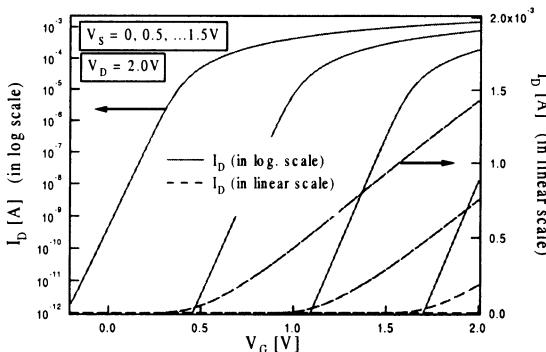
VHDL-AMS [5] is a language that allows the designer to describe mixed systems in the very same file. It has recently been chosen as the new IEEE standard for the modeling, the simulation, or the virtual prototyping of heterogeneous systems on a chip.

At the first order, writing the EKV MOST model in VHDL-AMS is a simple task, if we consider that any dynamic continuous model can easily be described as a set of simultaneous implicit or explicit differential algebraic equations (DAE). The VHDL-AMS analog solver is responsible for computing the values of the quantities such that the relationships hold (subject to tolerances), and the order of simultaneous statements does not matter. An example of VHDL-AMS implementation of a very simple EKV MOST model (case of a large n- MOST) can be found in [6].

## 2.2 VHDL-AMS Simulation results

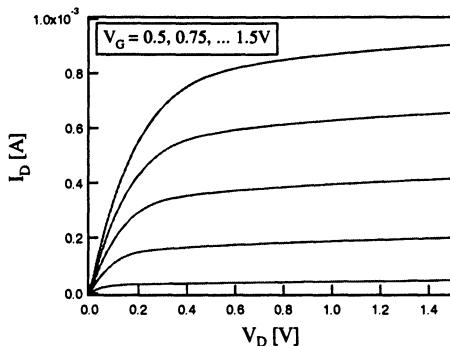
All simulation results in VHDL-AMS included in this paper are made with a n-channel transistor of  $W/L = 1.5\mu\text{m}/0.15\mu\text{m}$ , and with an p-channel transistor of  $W/L = 3.0\mu\text{m}/0.15\mu\text{m}$ . The simulation environment is Anacad Advance AMS v1.1.

Figs. 1 to 2 present the MOST behavior in static mode in all regions of operation of the MOSFET transistor (weak, moderate, strong inversion) as well as conduction to saturation. Fig. 1 shows the drain current  $I_D$  (in log. and linear scale) vs.  $V_G$  characteristic for different  $V_S$ , from weak to strong inversion.



*Figure 1.* Simulation of the  $I_D$  versus  $V_G$  characteristic, for different source voltages.

Then, the  $I_D$  versus  $V_D$  characteristic for different  $V_G$  is shown in strong inversion in Fig. 2.



*Figure 2.* Simulation of the  $I_D$  versus  $V_D$  characteristic, for different gate voltages.

### 3. FEATURES SPECIFIC TO SUBMICRON

The LDD regions in the sub- and deep- submicron CMOS technologies introduce additional parasitic resistances between the source/drain electrode and the channel, as well as parasitic capacitances.

A problem with all these parasitic elements is their non-linear and bias dependent behavior. An efficient MOST model dedicated to deep-submicron design must imperatively take into account these elements. Equation (2), and Figs. 3-4 show some simple and accurate solutions to the modeling of the LDD region, and this with a few parameters (Table 1).

#### 3.1 Series parasitic resistance

The source (or drain) resistance,  $RSeff$  (or  $RDeff$ ), taking the LDD region into account can be defined as:

$$RSeff \text{ (or } RDeff) = \frac{RS \text{ (or } RD)}{1 + 0.5 \cdot \left[ r + \sqrt{r^2 + 0.01} \right]} \quad (2.a)$$

$$\text{with } r = SVK \cdot [(VG - VT_{Oeff}) / VK - 1] \quad (2.b)$$

where  $VT_{Oeff}$  is the effective threshold voltage modified by bias and by the reverse short channel effect, and  $RS$  ( $RD$ ) is the maximum resistance of source (drain). For design, one can define a layout-dependent model; so eq. 2.a. can be rewritten as:

$$R_{eff} = \frac{1}{W_{eff}} \cdot \left[ HDIF \cdot RSH0 + \frac{(LDIF + LD) \cdot RS0}{1 + 0.5 \cdot \left[ r + \sqrt{r^2 + 0.01} \right]} \right] \quad (2.c)$$

where the definitions of  $RSH0$ ,  $LDIF$ ,  $LD$ ,  $HDIF$  and  $RS$ , can be found in Fig. 3.  $W_{eff}$  is the effective width of the channel.

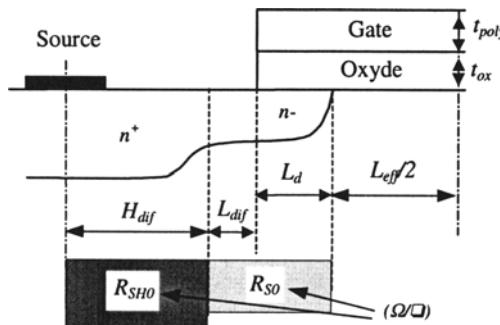


Figure 3. Definition of the resistive parameters in the LDD region.

### 3.2 The parasitic capacitances

The dynamic behavior of a MOST in deep submicron technology is strongly affected by its extrinsic capacitance formed by the overlap capacitance and the fringing capacitance (see Fig. 4). The fringing capacitance is constant, but the overlap capacitance is bias dependent. Its value is not equal to  $C_{ov} = W_{eff} \cdot L_d \cdot C_{ox}$ , but to  $C_{ov,eff} = W_{eff} \cdot L_{d,eff} \cdot C_{ox}$ .

$C_{ox}$  is the oxide capacitance per area unit, and  $L_{d,eff}$  ( $\leq L_d$ ) is a length essentially modulated by the bias.

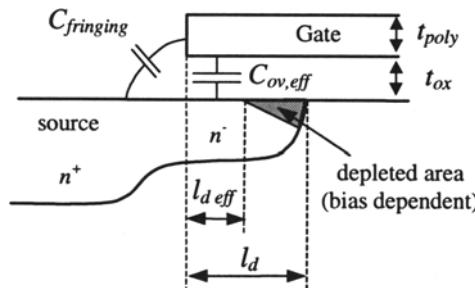


Figure 4. The parasitic capacitances in the LDD region (source or drain).

| Purpose             | Name                              | Description  |
|---------------------|-----------------------------------|--|
| Overlap Capacitance | LAMBDA                            | Overlap coefficient  |
|                     | TPOLY                             | Polysilicon thickness  |
| Series Parasitic    | RS/RD                             | Total resistance at zero bias  |
|                     | VK                                | Characteristic voltage   |
|                     | SVK                               | LDD coefficient  |
| Resistances         | Optional                          | <i>parameters</i>  |
|                     | HDIR                              | Path-length of the resistor in the heavily doped region (see fig. 3) |
|                     | LDIF                              | Partial path-length of the resistor in the LDD region (see fig. 3)   |
|                     | LD                                | Gate underdiffusion (see fig. 3)                                     |
|                     | RSH0<br>[in $\Omega/\square$ ]    | Sheet resistance (heavily doped region) at zero bias                 |
|                     | RS0/RD0<br>[in $\Omega/\square$ ] | Sheet resistance (LDD region) at zero bias                           |

Table 1. Parameters used to model the LDD regions.

### 3.3 Simulation results in VHDL-AMS

#### 3.3.1 Series parasitic resistance ( $R_{S\text{eff}}$ , $R_{D\text{eff}}$ )

A typical characteristic of series parasitic resistance can be observed in Fig. 5, for two different drain bias.

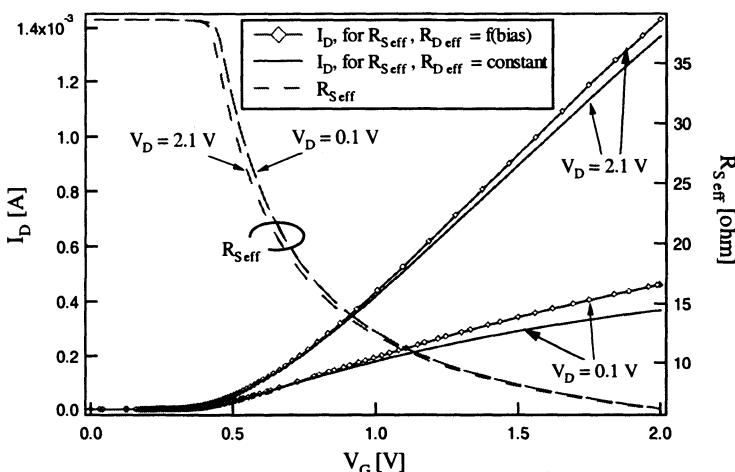


Figure 5. Simulation of  $I_D/R_{\text{Seff}}$  versus  $V_G$  characteristics for two drain bias.

Not taking into account the bias dependent of this resistance introduces some important errors on the drain current level, mainly for small  $V_D$  bias. These variations can considerably affect the parameters extraction procedures where, classically, channel length and series resistance are extracted altogether, at small  $V_D$  bias [7].

### 3.3.2 Parasitic capacitance

The parasitic capacitances represent a more and more important part of the global capacitance of the MOST (more 35% for a 0.15 $\mu\text{m}$  technology) as observed in Fig. 6, in the accumulation region. The influence of the fringing capacitance can be observed in the strong inversion region of Fig. 6; it represents the additional capacitance to COX (COX = W.L.Cox).

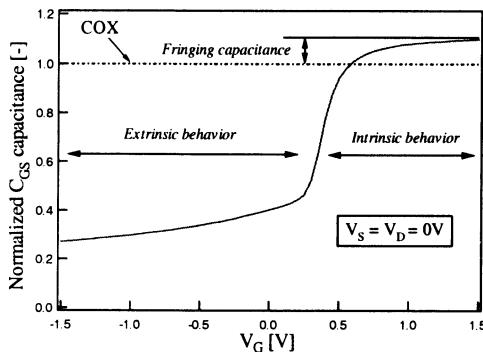


Figure 6. Simulation of the normalized global gate-source capacitance ( $C_{GS}/COX$ ) [8].

## 4. THERMAL-ELECTRONIC MODELING

As the transistor size decreases, thermal interactions between devices on the same chip increase. These thermal effects are constantly amplified by the growing power density, and a failure in their estimation at an early development stage of the design often means extra costs and delays.

For the system designer, one of the major interest of VHDL-AMS is the simplicity with which models involving various physical domains (electrical, optical, mechanical, etc) can be interconnected. Along with the ability to describe hierarchical/multi-abstraction models, the power of VHDL-AMS relies on the fact that it allows the model designer to only detail the constitutive equations of the inner submodels. VHDL-AMS takes advantage of a graph-based conceptual description to automatically compute

the equations describing the conservative laws of the global model. The vertices of the graph represent effort nodes (**across** quantities like voltage, temperature or pressure) in the circuit, and the edges represent branches of the circuit through which information flows (**through** quantities like current, heat/liquid flow rate).

Thus, for the designer, the assembling of complex systems is nothing more than the connection of elementary objects through compatible terminals (ie, capable of exchanging informations), in such a way that the conservative semantics of the generalized Kirchoff laws are preserved at all points.

## 4.1 VHDL-AMS Electrical and thermal model

In the EKV MOST electrical model, several parameters are strongly dependent on temperature. Their respective temperature variation are taken into account by appropriate coefficients in the equations.

We model the heat diffusion through solid materials by sourcing dissipated power into a thermal RC network [9], which represents the material properties of the different layers. The temperature profile is the result of a heat flow in the thermal network.

In such networks, energy conservation states that the sum of all power contributions at a thermal node equals zero, and that the temperature at all terminals connected to a thermal node be identical. Thermal evolution of a system is thus ruled by the very same Kirchoff laws dictating the behavior of conservative systems : voltage becomes the across quantity temperature, and current becomes the through quantity heat flow.

VHDL-AMS provides an elegant solution to thermal-electronic interactions. Among the various packages provided with the simulation environment, the **thermal\_system** package simply adds thermal capabilities to electrical models. The principle is to introduce a thermal terminal, with relative **through** and **across** quantities respectively bound to power and temperature.

The final EKV interface then becomes :

```
entity ekv1 is
    port (terminal d,g,s,b : electrical ;
          terminal j : thermal);
end;
```

with the following **quantity** relations :

```
quantity temp across power through j to thermal_ground ;
```

that define state variable **temp** as an extensive value, and **power** as the corresponding intensive value.

Using this electrical-thermal analogy, a thermal generator can simply be designed to model the ambient temperature with equation (3),

```

constant amb_temp: real := 300.0 ;
...
temp == amb_temp ;                                (3)
while thermal resistance and capacitance can respectively be described by equations
(4) and (5).
temp == rth * power ;                            (4)
power == cth * temp'dot ;                        (5)

```

Fig. 7 shows how thermal-electronic interactions between n-MOST and its direct environment can be modeled.

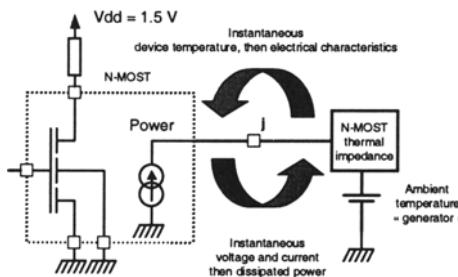


Figure 7. Modeling electro-thermal interactions.

## 4.2 VHDL-AMS simulation results

Electro-thermal simulations are given in Figs. 8-9, and Figs. 10-11.

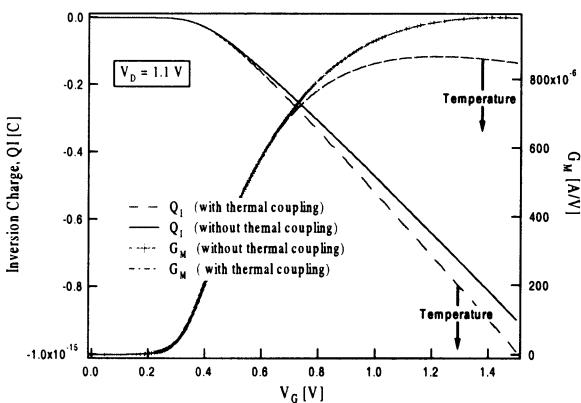


Figure 8. Characteristics of the inversion charge  $Q_I$  / transconductance  $G_M$  vs  $V_G$ .

The values of the capacitances and resistances of the thermal network have voluntarily been overstated. This exaggeration increases the chip temperature of the n-transistor by more than 100°K in Figs. 8-9.

In Fig. 8, one can observe that a positive variation of the chip temperature of the n-transistor increases the inversion charge ( $Q_I$ ), and downgrades the transconductance ( $G_M$ ). Fig. 9 shows the output characteristic of the MOST.

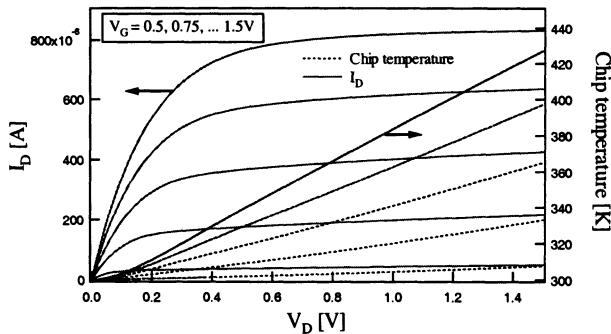


Figure 9. Simulation of the  $I_D$  / chip temperature versus  $V_D$  characteristics.

The  $I_D$  vs.  $V_D$  characteristic is downgraded with temperature variation in the chip compared to Fig. 2. obtained with the same electrical bias, but without thermo-electrical effect.

## 5. USING THE EKV MODEL IN VARIOUS DESIGNS

Fig. 10 presents an inverter design that validates the whole methodology.

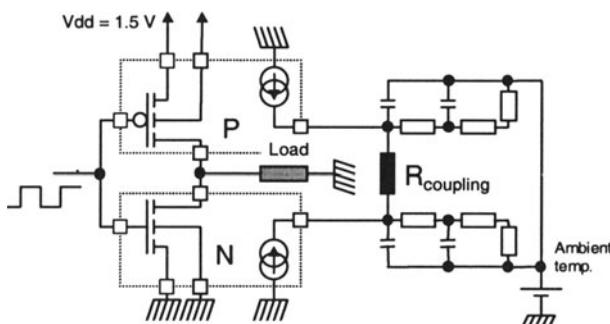
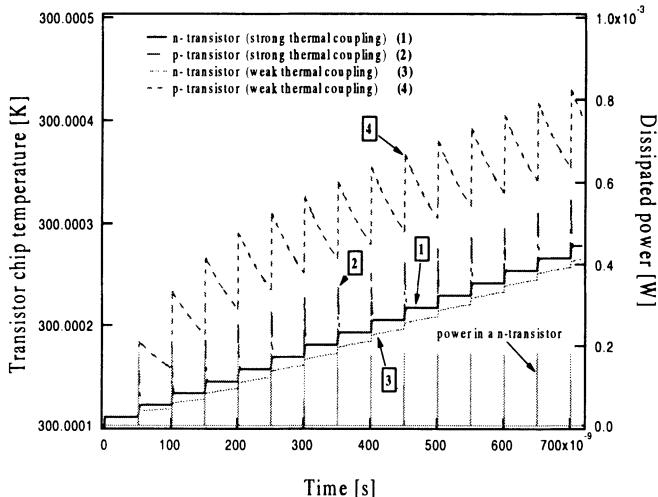


Figure 10. The inverter system, with RC network and thermal coupling.

The system details thermal coupling between the n-MOST and the p-MOST through **Rcoupling**. The inverter is excited by a squarewave stimulus and connected to a local thermal RC network. The two thermal networks represent the thermal constants of the various material layers.

For simulation purpose, as in §4.2. the values of the capacitances and resistances of the thermal network have voluntary been modified to point out the thermal effects.

Fig. 11 shows the temperature evolution in the inverter for two different values of **Rcoupling**. As expected, for a small value of **Rcoupling**, the temperature in the N and P transistors are tightly linked (dark curves in the figure). For a higher value, Fig. 11 shows the free temperature evolution of each transistor.



**Figure 11.** Simulation of the n/p - channel transistor chip temperature variation versus time during commutation (in an inverter), for two values of thermal coupling.

## 6. CONCLUSION

We are currently working on the VHDL-AMS release v.3 of the EKV model. It will notably include the quantum and polydepletion effects, and the NQS behavior [10]. This operational MOS transistor model can be seen as a possible starting point to simulate more complex structures. It is with such basic elements that VHDL-AMS provides optimal solutions to interconnect models of different domains, and the efficient modeling of MOEMS is almost within our grasp. For example, we have successfully instanciated the

EKV electrical model as a constitutive part of an analog opto-electrical system involving a light emitting diode and a photodiode [6] (both devices are very sensitive to temperature).

## 7. ACKNOWLEDGMENTS

The authors thank Matthias Bucher for discussion on the implementation of layout-dependent resistors model (without bias dependent) in standard simulators, and Fabien Pregaldiny for his help on overlap capacitances.

## 8. REFERENCES

- [1] R. J. Baker: CMOS: Mixed-signal Circuit Design, Chap. 33.  
<http://cmosedu.com/cmos2/cmos2.htm>.
- [2] C. Lallement, F. Pêcheux, Y. Hervé, "A VHDL-AMS Case Study : The Incremental Design of an efficient of a 3<sup>rd</sup> generation MOS Model of a Deep Submicron Transistor", in Proc VLSI-SOC, pp. 467-472, Montpellier, France, Dec. 2001.
- [3] D. Foty, '*MOSFET Modeling with SPICE - Principles and Practice*', ISBN 0-13-227935-5
- [4] M. Bucher, C. Lallement, C. Enz, F. Théodoloz, K. Krummenacher, "The EPFL-EKV MOSFET Model Equations for simulation, ver. 2.6", <http://legwww.epfl.ch/ekv/>.
- [5] E. Christen, K. Bakalar, "VHDL-AMS-a hardware description language for analog and mixed-signal applications," *IEEE Trans. on Circuits and Systems, part II*, Vol. 46 Issue: 10, pp. 1263 –1272, Oct. 1999.
- [6] C. Lallement, F. Pêcheux, Y. Hervé," *VHDL-AMS design of a MOST model including deep submicron and thermal-electronic effects,*" 2001 IEEE Workshop BMAS 2001, Oct. 10-12, 2001, Santa Rosa, USA
- [7] N. Arora, "Mosfet Models for VLSI Circuit Simulation: Theory and Practice, *Computational Microelectronics*, Springer Verlag, Wien New York, 1993.
- [8] F. Pregaldiny, C. Lallement, D. Mathiot, *Paper Submitted to SSE*.
- [9] C. Lallement, R. Bouchakour and T. Maurel, "One-dimensional Analytical Modeling of the VDMOS Transistor Taking Into Account the Thermoelectrical Interactions," *IEEE Transactions on Circuits and Systems, Part. I*, Vol. 44, N° 2, pp. 103-111, Feb. 1997.
- [10] J. M. Sallese, et al, "Advancements in DC and RF Mosfet modeling with the EPFL-EKV charge based model," *8th Int. Conf. MIXDES*, Poland, pp. 45-52, 21-23 June 2001.

# Speeding Up Verification of RTL Designs by Computing One-to-One Abstractions with Reduced Signal Widths

Peer Johannsen

*Corporate Technology CT-SE-4  
Siemens AG, 81730 Munich, Germany*  
[peer.johannsen@mchp.siemens.de](mailto:peer.johannsen@mchp.siemens.de)

Rolf Drechsler

*Institute of Computer Science  
University of Bremen, 28359 Bremen, Germany*  
[drechsle@informatik.uni-bremen.de](mailto:drechsle@informatik.uni-bremen.de)

**Abstract** Digital circuit designs are usually given as Register-Transfer-Level (RTL) specifications, but most of today's hardware verification tools are based on bit-level methods, using SAT or BDD-based techniques. RTL specifications contain more explicite structural information than bit-level descriptions. This paper presents a new approach to scale down design sizes before verification by exploiting word-level information. We introduce a one-to-one abstraction technique for RTL property checking, which computes a scaled-down abstract model of a design, in which signal widths are reduced with respect to a property. The property holds for the abstract RTL if and only if it holds for the original RTL. If the property fails, counterexamples for the original design are computed from counterexamples found on the reduced model. The verification task is completely carried out on the scaled-down version of the design; false-negatives cannot occur. Linear signal width reductions result in exponentially smaller state spaces and have a significant impact on the runtimes of verification tools. Experimental results on large industrial circuits have demonstrated applicability and efficiency of our method.

## Introduction

Verification has become one of the most important steps in digital circuit design. Today's circuit designs often contain up to several million transistors. The test for correct behavior before manufacturing becomes more and more important and a major economical issue. While design sizes are ever increasing, this test grows more complex, time-consuming, and expensive. Formal verification tasks often fail already due to design sizes. Automated abstraction techniques (e.g. CGL92) are a promising approach to enhance capabilities of formal verification tools.

Recently, *Bounded Model Checking* (cf. BCC<sup>+</sup>99) and *Bounded Property Checking* have gained increased significance in *Electronic Design Automation* (EDA), as recently surveyed in SS00. The majority of today's industrial hardware verification tools uses bit-level decision procedures, like SAT or BDD-based techniques (see e.g. Bry86; Sil95). However, circuit designs are usually given in terms of RTL specifications, for example coded in *Hardware Description Languages* (HDLs), like VHDL or Verilog. RTL specifications of digital circuits contain explicit structural information which is lost in bit-level descriptions. On bit-level, for example in Boolean formulae, all signals are of one-bit width, and all available functional units are Boolean gates. In contrast to that, on RTL, word-level data structures (e.g. bitvectors and busses) as well as high-level operators (e.g. adders, multipliers, and shifters) are still visible. Several approaches to formal circuit verification have been proposed which make use of such high-level information and which are based on word-level verification techniques, like for example *Word-Level Decision Diagrams* (e.g. Dre00), formal *Bitvector Theories* (e.g. BDL98; CMR97), *Integer Linear Programming* (e.g. ZKC01), *Symmetry Reductions* (e.g. CEJS98; ET99) and *Term Rewriting* (e.g. DJ90), to survey only a few.

## 1. Scaling Design Sizes before Verification

This paper presents a new word-level abstraction technique which is used as a preprocess in high-level property checking of digital circuits. The proposed method automatically scales down data-path widths while preserving design properties. We consider the property checking flow shown in Figure 1. Circuit designs are given as HDL specifications, and properties are described in a linear time logic used in *Symbolic Trajectory Evaluation* and specify the intended behavior of the design within a finite bounded interval of time. Typical properties are subdivided into an assumption part implying a commitment part and consist of temporal operators and state expressions, involving relationships among data words. As an example consider:

```

assume: (during [t, t+4]: reset = 0) and
        (at t: request = 1);
prove: (at t+3: acknowledge = 1) and
        (at t+4: data = 11111111);

```

The standard verification flow is indicated by white boxes. Design and property are transformed (*Synthesis, Unrolling*) into an instance of propositional SAT, i.e. a bit-level formula  $\varphi$ . Satisfiability of  $\varphi$  corresponds to invalidity of the property.  $\varphi$  is handed to a property checker, which uses bit-level verification techniques in order to either prove that the property holds, or to return a counterexample. The gray shaded areas in Figure 1 illustrate how the proposed abstraction technique is incorporated into such a flow. Instead of immediately going down to the bit-level, an RTL representation  $E$  of  $\varphi$  is generated, consisting of high-level primitives, like word-level signals (variables) and word-level operators (e.g. arithmetic units, comparators, multiplexors and memory

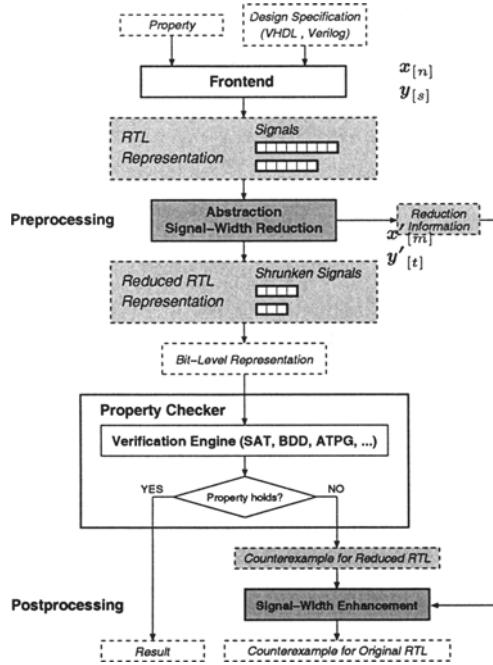


Figure 1. Property Checking Flow

elements). Each signal  $\mathbf{x}$  has a fixed width  $n \in \mathbb{N}_+$  and takes bitvectors of respective length as values. In a preprocessing step, our method takes the RTL representation and computes a second, scaled down RTL model  $E'$  by replacing each word-level signal  $\mathbf{x}$  of  $E$  by a corresponding shrunken signal of width  $m_{\mathbf{x}} \leq n$  (whereby  $n$  denotes the original width of  $\mathbf{x}$ ). Original and abstract model differ from each other only as far as signal widths are concerned. All other data-flow aspects are preserved. The width of each signal in the abstract RTL is the minimum width which is necessary and sufficient in order to establish a one-to-one abstraction with respect to design, property, and the reduction technique we propose (i.e. by solely changing signal widths):

$$\text{The property holds for the reduced RTL} \iff \text{The property holds for the original RTL}$$

The reduced RTL is transformed into a bit-level formula  $\varphi'$  which is given to the property checker instead of  $\varphi$ . Thus, the proposed abstraction technique can be used in combination with a variety of existing (powerful) property checking tools. The bit-level representations  $\varphi$  and  $\varphi'$  contain bit-level variables for each bit of each word-level signal of  $E$  and  $E'$ . Depending on the degree of reduction of signal widths,  $\varphi'$  can contain significantly less variables than  $\varphi$ . A linear reduction of a signal's width from  $n$  bits down to  $m$  bits,  $m < n$ , causes an exponential reduction of the size of the induced state space from  $2^n$  down to  $2^m$ , which can cause a significant speed-up of verification runtimes. If the property does not hold, the property checker returns a counterexample

for  $\varphi'$ , i.e. for the *reduced* RTL. Our method provides a postprocessing technique which takes such a counterexample and computes values for all inputs of the original design, for which the property fails.

## 2. Bitvector Equations

Circuit designs can be represented on RTL by systems of bitvector equations such that validity of design properties corresponds to satisfiability of the equations. We define an equational theory  $\mathcal{L}(\text{Bv})$  of fixed-size bitvectors, which is an extension of the core theory of bitvectors presented in CMR97. Let  $\mathbb{B} := \{0, 1\}$ . A **bitvector** of width  $n \in \mathbb{N}_+$  is a finite vector element  $v = \langle v_{n-1}, \dots, v_1, v_0 \rangle \in \mathbb{B}^n$ , consisting of  $n$  individual bits, which are indexed from right to left, starting with index 0. The set  $\mathbb{B}^n$  of bitvectors of width  $n$  is denoted by  $\mathbb{B}_{[n]}$ . A **bitvector variable** of width  $n \in \mathbb{N}_+$  is a typed variable  $x_{[n]}$ , representing fixed-size bitvectors  $v \in \mathbb{B}_{[n]}$  of width  $n$ . We use bold face characters for bitvector variables, and the width (i.e. the type) is always explicitly denoted. We write  $x_{[n]}[i]$  to refer to the  $i^{\text{th}}$  bit of the value of  $x_{[n]}$ . The set of well-formed  $\mathcal{L}(\text{Bv})$  terms is defined inductively over a finite set of bitvector variables and the set of bitvector operators shown in Table 1.

Table 1. Bitvector Operators and Equations

| Operator            | Syntax                     | Example  |
|---------------------|----------------------------|--|
| bitvector variables | $x_{[n]}$                  | $x_{[8]}, y_{[16]}, z_{[4]}, \dots$  |
| bitvector constants | $v_{n-1} \dots v_1 v_0$    | $0000, 1111, 00101011, \dots$  |
| concatenation       | $\otimes$                  | $x_{[16]} = y_{[12]} \otimes z_{[4]}$  |
| extraction          | $[j, i]$                   | $x_{[4]} = y_{[32]}[5, 2]$   |
| bitwise negation    | $\text{neg}$               | $x_{[8]} = \text{neg}(y_{[8]})$  |
| bitwise Boolean     | $\text{and, or, xor}$      | $x_{[16]} = y_{[16]} \text{ and } z_{[16]}$  |
| connectives         | $\text{nand, nor, xnor}$   | $x_{[16]} = y_{[16]} \text{ or } z_{[16]}$   |
| if-then-else        | $\text{ite}$               | $x_{[8]} = \text{ite}(a_{[4]} = b_{[4]}, y_{[8]}, z_{[8]})$<br>$x_{[8]} = \text{ite}(a_{[4]} < b_{[4]}, y_{[8]}, z_{[8]})$ |
| arithmetic          | $\oplus, \ominus, \otimes$ | $x_{[32]} = y_{[32]} \oplus z_{[32]}$  |
| memory read         | $\text{read}$              | $c_{[4]} = \text{read}(m_{[1024]}, i_{[8]})$   |
| memory write        | $\text{write}$             | $m_{[64]} = \text{write}(m_{[64]}, i_{[4]}, v_{[2]})$  |

Well-formedness of terms implies that variable widths have to comply with operator demands (e.g. index expressions must not exceed the widths of argument terms). Note that additional high-level operators, e.g. shifts and rotations, can already be expressed with the shown set of operators. Within our framework, RTL models consist of a system  $E$  of equations of bitvector terms over  $\mathcal{L}(\text{Bv})$  such that:

$$E \text{ is satisfiable} \iff \begin{array}{l} \text{The property does not} \\ \text{hold for the design} \end{array} \quad (1)$$

A system  $E$  is satisfiable if there exists a valuation of the variables of  $E$  such that all equations are satisfied. Multi-bit circuit signals directly

correspond to bitvector variables of  $E$ . A satisfying solution, if existent, yields a counterexample indicating values for all circuit signals such that the property does not hold for these assignments (*falsification, bug hunting*). The proposed abstraction technique generates a second system  $E'$  of bitvector equations, which differs from  $E$  solely in the manner that variable widths are reduced. The width of each bitvector variable is shrunken to the smallest possible number of bits (with respect to  $E'$  differing from  $E$  only by reduced variable widths), such that:

$$E' \text{ is satisfiable} \iff E \text{ is satisfiable} \quad (2)$$

$E'$  represents a scaled down version of the original design, and property checking can be done entirely on  $E'$ .

### 3. Signal Width Reduction

Bitvector equations describe data dependencies on word-level. The equations explicitly contain the high-level information, which individual bits belong to the same word-level signal, and how single bits are ordered within multi-bit signals. In the following we show how this information can be used to reduce computational complexity of satisfiability checks of bitvector equations.

If neighboring bits of bitvector variables are computed uniformly according to the same bit-level data flow, then such data dependencies are called a **uniform data flow**.

**Example 1 (Uniform Data Dependencies)** Let  $x_{[8]}, y_{[8]}$  and  $z_{[8]}$  be 8-bit signals, and consider the following bitvector equation:

$$x_{[8]} \text{ and } y_{[8]} = z_{[8]} \quad (3)$$

Equation (3) specifies functional data dependencies between  $x_{[8]}, y_{[8]}$  and  $z_{[8]}$ . Satisfiability of (3) corresponds to satisfiability of the following bit-level representation

$$(x_0 \text{ and } y_0 = z_0) \wedge \dots \wedge (x_7 \text{ and } y_7 = z_7), \quad (4)$$

involving 24 Boolean variables and 8 equations. Obviously it is not necessary to solve all 8 equations of (4) separately, because the data flow for bit-positions 0–7 is computed uniformly. Let  $x'_{[1]}, y'_{[1]}, z'_{[1]}$  denote new signals of width 1, derived from the variables of (3). It is sufficient to check satisfiability of

$$x'_{[1]} \text{ and } y'_{[1]} = z'_{[1]}, \quad (5)$$

because (3) is satisfiable if and only if (5) is satisfiable. A satisfying solution of (3) can be obtained from a solution of (5) by signed extension. For example,  $x'_{[1]} = 0$ ,  $y'_{[1]} = 1$ ,  $z'_{[1]} = 0$  yields  $x_{[8]} = 00000000$ ,  $y_{[8]} = 11111111$  and  $z_{[8]} = 00000000$ .  $\square$

Uniform data flow is formally characterized by **bitwise bitvector functions**.

**Definition 1 (Bitwise Bitvector Functions)** Let  $n, k \in \mathbb{N}_+$  and let  $\mathcal{F}_{[n]} : \mathbb{B}_{[n]} \times \dots \times \mathbb{B}_{[n]} \longrightarrow \mathbb{B}_{[n]}$  be a  $k$ -ary bitvector function of width  $n$  on bitvectors of width  $n$ .  $\mathcal{F}_{[n]}$  is a **bitwise bitvector function** if there exists a  $k$ -ary Boolean function  $\mathcal{B}_{[1]} : \mathbb{B}_{[1]} \times \dots \times \mathbb{B}_{[1]} \longrightarrow \mathbb{B}_{[1]}$  such that  $\mathcal{F}_{[n]}(\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k)[i] = \mathcal{B}_{[1]}(\mathbf{x}_{[n]}^1[i], \dots, \mathbf{x}_{[n]}^k[i])$  for all  $i \in \{0, \dots, n-1\}$  and all  $\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k \in \mathbb{B}_{[n]}$ . ■

$\mathcal{B}_{[1]}$  is called the **characteristic Boolean function** of  $\mathcal{F}_{[n]}$ . Satisfying solutions of bitvector equations with uniform data dependencies can be characterized by zeros of bitwise bitvector functions.

**Definition 2 (Bitwise Bitvector Equations)** Let  $e$  be a bitvector equation over  $\mathcal{L}(\text{Bv})$ , and let  $V = \{\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k\}$  be the set of bitvector variables occurring in  $e$ . Then  $e$  is called a **bitwise bitvector equation** if there exists a bitwise bitvector function  $\mathcal{F}_{[n]}$  such that the set of satisfying solutions of  $e$  is exactly the set of satisfying solutions of  $\mathcal{F}_{[n]}(\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k) = 0_{[n]}$ . ■

Satisfiability of bitwise bitvector equations for bitvectors of width  $n$  can be mapped onto satisfiability of equations for bitvectors of width 1 by simply reducing variable widths, as seen in Example 1 and formalized as follows.

**Corollary 1 (Bitwise Bitvector Functions)** Let  $n, k \in \mathbb{N}_+$  and let  $\mathcal{F}_{[n]} : \mathbb{B}_{[n]} \times \dots \times \mathbb{B}_{[n]} \longrightarrow \mathbb{B}_{[n]}$  be a  $k$ -ary bitwise bitvector function with characteristic Boolean function  $\mathcal{B}_{[1]}$ . Then  $\mathcal{F}_{[n]}(\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k) = 0_{[n]}$  is satisfiable if and only if  $\mathcal{B}_{[1]}(\mathbf{x}'_{[1]}^1, \dots, \mathbf{x}'_{[1]}^k) = 0_{[1]}$  is satisfiable. ■

In general, data flow must be analyzed for all equations of a given system. Reduction depends on structural and dynamical data dependencies imposed by the conjunction of all equations. Thus, even if uniform data flow exists for a specific equation, other equations can be the reason that reduction to only 1-bit width might not preserve satisfiability.

**Example 2 (Dynamical Data Dependencies)** Let  $\mathbf{x}_{[8]}, \mathbf{y}_{[8]}$  and  $\mathbf{z}_{[8]}$  be signals of width 8, for which uniform data dependencies exist. Consider a system of bitvector equations, which additionally contains the following expressions:

$$\begin{aligned} \dots &= \dots \text{ite}(\mathbf{x}_{[8]}=\mathbf{y}_{[8]}, \dots, \dots) \dots \\ \dots &= \dots \text{ite}(\mathbf{y}_{[8]}=\mathbf{z}_{[8]}, \dots, \dots) \dots \\ \dots &= \dots \text{ite}(\mathbf{z}_{[8]}=\mathbf{x}_{[8]}, \dots, \dots) \dots \end{aligned} \tag{6}$$

A satisfying solution of (6) might require that the values of  $x_{[8]}$ ,  $y_{[8]}$ ,  $z_{[8]}$  have to be mutually different, i.e.

$$x_{[8]} \neq y_{[8]} \wedge y_{[8]} \neq z_{[8]} \wedge z_{[8]} \neq x_{[8]}. \quad (7)$$

Then, reduction to only one bit width is not possible, because  $x'_{[1]} \neq y'_{[1]} \wedge y'_{[1]} \neq z'_{[1]} \wedge z'_{[1]} \neq x'_{[1]}$  is not satisfiable, while (7) is. Instead the following holds:

$$x_{[m]} \neq y_{[m]} \wedge y_{[m]} \neq z_{[m]} \wedge z_{[m]} \neq x_{[m]} \quad (8)$$

is satisfiable for all  $m \geq 2$ , and at the same time 2 is the minimum value for  $m$ , for which

$$(7) \text{ satisfiable} \iff (8) \text{ satisfiable}$$

holds. Therefore, satisfiability of (6) can be preserved by choosing reduced bitvectors of width 2. But, even if a solution of (6) maybe only requires  $x_{[8]} \neq y_{[8]} \wedge y_{[8]} \neq z_{[8]}$ , a reduction to one bit widths still might not be sufficient, although  $x'_{[1]} \neq y'_{[1]} \wedge y'_{[1]} \neq z'_{[1]}$  is satisfiable, because the uniform data dependencies between  $x_{[8]}$ ,  $y_{[8]}$ ,  $z_{[8]}$ , as imposed by further equations of (6), could be the following:

$$\begin{aligned} 11111111 = & (x_{[8]} \text{ and } y_{[8]} \text{ and } \neg(z_{[8]})) \text{ or} \\ & (\neg(x_{[8]}) \text{ and } y_{[8]} \text{ and } z_{[8]}) \end{aligned}$$

Such conditions are satisfiable for bitvectors of width 8, but the corresponding problem, where variables are reduced to 1-bit width, is unsatisfiable. Instead, satisfiability again is preserved when reduced bitvectors of width 2 are chosen.  $\square$

Data dependencies can exist between complete bitvectors or only between certain bits. Typically, different data dependencies exist for different chunks of a variable. Variables can be partitioned into contiguous parts in which all bits are treated uniformly with respect to data dependencies.

**Example 3 (Structural Data Dependencies)** Let  $x_{[8]}$ ,  $y_{[8]}$  and  $z_{[8]}$  be bitvector variables of width 8, and let  $a_{[2]}$ ,  $b_{[2]}$  be bitvector variables of width 2. Consider the following system  $E$  of bitvector equations:

$$E \quad \left\{ \begin{array}{l} x_{[8]} \text{ and } y_{[8]} = z_{[8]} \\ x_{[8]} = a_{[2]} \otimes b_{[6]} \end{array} \right. \quad (9)$$

The first equation specifies uniform data dependencies for  $x_{[8]}$ ,  $y_{[8]}$ ,  $z_{[8]}$ , but the second one imposes different structural dependencies for the upper two and the lower six bits of  $x_{[8]}$ .  $E$  can be decomposed into two

disjoint independent systems  $E_1$  and  $E_2$  of bitvector equations,

$$\begin{aligned} E_1 \quad & \left\{ \begin{array}{l} \mathbf{x}_{[8]}[7,6] \text{ and } \mathbf{y}_{[8]}[7,6] = \mathbf{z}_{[8]}[7,6] \\ \mathbf{x}_{[8]}[7,6] = \mathbf{a}_{[2]} \end{array} \right. \\ E_2 \quad & \left\{ \begin{array}{l} \mathbf{x}_{[8]}[5,0] \text{ and } \mathbf{y}_{[8]}[5,0] = \mathbf{z}_{[8]}[5,0] \\ \mathbf{x}_{[8]}[5,0] = \mathbf{b}_{[6]} \end{array} \right. \end{aligned} \quad (10)$$

such that the set of satisfying solutions of  $E$  is the same as the set of satisfying solutions of the conjunction of  $E_1$  and  $E_2$ , i.e.  $E$  is satisfiable if and only if  $E_1 \wedge E_2$  is satisfiable. Furthermore, all data dependencies in  $E_1$  and in  $E_2$  are uniform. Satisfiability of (10) then can be reduced to satisfiability of:

$$\begin{aligned} E'_1 \quad & \left\{ \begin{array}{l} \mathbf{x}'_{[1]} \text{ and } \mathbf{y}'_{[1]} = \mathbf{z}'_{[1]} \\ \mathbf{x}'_{[1]} = \mathbf{a}'_{[1]} \end{array} \right. \\ E'_2 \quad & \left\{ \begin{array}{l} \mathbf{x}''_{[1]} \text{ and } \mathbf{y}''_{[1]} = \mathbf{z}''_{[1]} \\ \mathbf{x}''_{[1]} = \mathbf{b}''_{[1]} \end{array} \right. \end{aligned} \quad (11)$$

and from (11) we can recompose

$$E' \quad \left\{ \begin{array}{l} \mathbf{x}'''_{[2]} \text{ and } \mathbf{y}'''_{[2]} = \mathbf{z}'''_{[2]} \\ \mathbf{x}'''_{[2]} = \mathbf{a}'''_{[1]} \otimes \mathbf{b}'''_{[1]} \end{array} \right. \quad (12)$$

with (12) is satisfiable if and only if (9) is satisfiable.  $\mathbf{a}'''_{[1]}$  relates to  $\mathbf{a}_{[2]}$ ,  $\mathbf{b}'''_{[1]}$  relates to  $\mathbf{b}_{[6]}$ ,  $\mathbf{x}'''_{[2][1,1]}$  relates to  $\mathbf{x}_{[8]}[7,6]$ , and  $\mathbf{x}'''_{[2][0,0]}$  to  $\mathbf{x}_{[8]}[5,0]$ , and so on. To obtain a solution of (9), signed extension is done separately for related chunks according to the prior decomposition, for example  $\mathbf{a}'''_{[1]} = 0$ ,  $\mathbf{b}'''_{[1]} = 1$ ,  $\mathbf{x}'''_{[2]} = 01$ ,  $\mathbf{y}'''_{[2]} = 11$ ,  $\mathbf{z}'''_{[2]} = 01$ , yields  $\mathbf{a}_{[2]} = 00$ ,  $\mathbf{b}_{[6]} = 111111$ ,  $\mathbf{x}_{[8]} = 00111111$ ,  $\mathbf{y}_{[8]} = 11111111$ ,  $\mathbf{z}_{[8]} = 00111111$ .  $\square$

## 4. Granularity Decomposition

According to Definition 2 and Corollary 1, satisfiability of bitwise bitvector equations can be mapped to satisfiability of corresponding equations over bitvectors of reduced width. This technique can be generalized to scale down signal widths for whole systems of bitvector equations as shown in Example 3.

A **chunk** of a bitvector variable  $\mathbf{x}_{[n]}$  is a triple  $(\mathbf{x}_{[n]}, j, i)$  with  $0 \leq i \leq j < n$ , representing the contiguous part of  $\mathbf{x}_{[n]}$  which is described by the bitvector term  $\mathbf{x}_{[n]}[j, i]$ . Two chunks  $\mathbf{x}_{[n]}[j_1, i_1]$  and  $\mathbf{x}_{[n]}[j_2, i_2]$  are **disjoint** if either  $i_1 > j_2$  or  $j_1 < i_2$ . A finite number of *mutually disjoint* sets  $C_1, \dots, C_q$  of chunks of bitvector variables is called a **granularity**, if  $\bigcup C_i$  is a set of *disjoint* chunks.

**Definition 3 (Granularity Decomposition)** Let  $V$  be a set of bitvector variables and let  $E$  be a system of bitvector equations over variables of  $V$ . A granularity decomposition of  $E$  is a partitioning of  $E$  into a finite number  $E_1, \dots, E_q$  of independent systems of  $\mathcal{L}(\text{Bv})$  equations and into sets of chunks  $C_1, \dots, C_q$  and sets  $I_1, \dots, I_q \subseteq V \times V$  s.t.

- $C_1, \dots, C_q$  is a granularity of  $V$ ,
- each  $C_i$  is exactly the set of chunks (variables) occurring in  $E_i$ ,
- all equations of each  $E_i$  are bitwise bitvector equations, and
- the set of satisfying solutions of  $E$  consists exactly of all compositions of satisfying solutions of  $E_1, \dots, E_q$ , which additionally satisfy the inequalities specified by  $I_1, \dots, I_q$ .

■

The process of scaling the widths of bitvector variables for a system  $E$  of bitvector equations is separated into two subsequent phases. Section 5 describes how first a granularity decomposition of  $E$  is computed, which, for each bitvector variable  $\mathbf{x}_{[n]}$  of  $E$ , describes a splitting of  $\mathbf{x}_{[n]}$  according to uniform data flow, as imposed by structural data dependencies. Then, for each chunk of  $\mathbf{x}_{[n]}$ , the necessary minimum width is computed, which preserves satisfiability as required by dynamical data dependencies (cf. Example 2). This is further explained in Section 6. According to these computed minimum chunk widths, the reduced width for the corresponding shrunken variable of  $E'$  is reassembled.

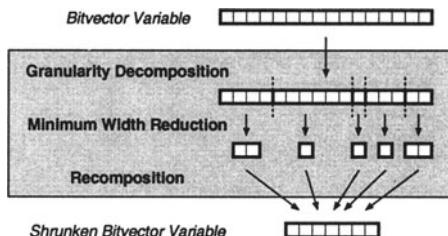


Figure 2. Basic Abstraction Technique

A granularity decomposition of a given system  $E$  always exists, because bitvector variables can always be decomposed into single bits and the data dependencies of  $E$  can always be described on bit-level. Usually, there exists a variety of possible decompositions. If, for example, uniform data dependencies exist for a chunk  $\mathbf{x}_{[8]}[5, 0]$ , then a finer splitting, e.g. into  $\mathbf{x}_{[8]}[5, 4]$  and  $\mathbf{x}_{[8]}[3, 0]$ , is also a valid decomposition. The highest amount of reduction can be achieved for bitwise decompositions with chunks of the largest possible width. Finding the coarsest possible decomposition for an arbitrary  $E$ , is a problem of detecting uniform data flow. Hence, it is a problem of deciding equality of Boolean functions, and thus is NP-complete.

## 5. Uniform Data Flow Analysis

Our methods computes a granularity decomposition of a given system  $E$  by syntactical analysis of the bitvector equations of  $E$ . According to the high-level information about operators and multi-bit signals, which is available within the  $\mathcal{L}(\text{Bv})$  formalism, chunks of neighboring bits of bitvector variables are determined, for which uniform data dependencies exist. The granularity decomposition is determined by means of an equivalence class structure, which groups chunks of bitvector variables between which functional dependencies can be described by a bitwise bitvector function.

The equivalence class computation can efficiently be done by employing a union-find algorithm, which, besides the known `union()` and `find()` operations, defines a new procedure `slice()`. Initially, one complete chunk for each bitvector variable  $\mathbf{x}_{[n]}$  resides in its own singleton equivalence class  $\{\mathbf{x}_{[n]}[n-1, 0]\}$ . A call of `find( $\mathbf{x}_{[n]}$ ,  $i$ )` yields the (non-ambiguous) equivalence class which includes a chunk of  $\mathbf{x}_{[n]}$  which contains bit position  $i$ ; `union()` performs the usual set union of two classes; and `slice( $\mathbf{x}_{[n]}$ ,  $j$ ,  $i$ )` calls `find( $\mathbf{x}_{[n]}$ ,  $i$ )` and `find( $\mathbf{x}_{[n]}$ ,  $j$ )` and splits all chunks of the respective classes at the bit positions corresponding to  $i$  and  $j$ . The originating parts are grouped in two new classes. Each bitvector equation  $e$  of  $E$  is sequentially (but in an arbitrary order) analyzed, and the next state of the equivalence class structure is computed by means of the procedure `gran( $e$ )`, which is outlined in Algorithm 1. The procedure recursively performs a case split according to the top-level operators occurring in the bitvector terms and computes a coarsest-possible<sup>1</sup> granularity decomposition of all variables occurring in  $e$ . When all bitvector equations have been processed, the granularity decomposition of  $E$  is given by the sets of chunks residing in each equivalence class.

## 6. Minimum Width Abstraction

The initial satisfiability problem for  $E$  is decomposed into a number of independent satisfiability problems  $E_i$  as described in Definition 3, which are associated with the computed equivalence classes. The specialty of our method is that these instances  $E_i$  do **not** have to be computed or represented explicitly. For the reduction technique, which we propose, it is sufficient to know, that the solutions of these problems can be characterized by satisfiability problems for bitwise bitvector functions and sets of inequalities.

**Definition 4 (BvSat)** Let  $k, n \in \mathbb{N}_+$  and  $V = \{1, \dots, k\}$  and let  $\mathcal{F}_{[n]} : \mathbb{B}_{[n]} \times \dots \times \mathbb{B}_{[n]} \longrightarrow \mathbb{B}_{[n]}$  be a  $k$ -ary bitvector function of width  $n$  on bitvectors of width  $n$ . Let  $I \subseteq V \times V$ . Then  $\text{BvSat}(\mathcal{F}_{[n]}, I)$  denotes the problem whether there exist  $\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k \in \mathbb{B}_{[n]}$ , such that  $\mathcal{F}_{[n]}(\mathbf{x}_{[n]}^1, \dots, \mathbf{x}_{[n]}^k) = 0_{[n]}$  and  $\mathbf{x}_{[n]}^i \neq \mathbf{x}_{[n]}^j$  for all  $\{i, j\} \in I$ . ■

BvSat is an NP-complete Problem, and is in detail investigated in Joh01b, where the following fundamental theorem on width reductions for bitwise bitvector functions and inequalities is presented.

**Algorithm 1** Granularity Analysis of Bitvector Equations

---

```

gran( $e$ ) {
    switch ( $e$ );
    case  $e \equiv 's_{[n]} = t_{[q]} \otimes u_{[r]}'$ :
        gran(' $s_{[n]}[n - 1, r] = t_{[q]}$ '); gran(' $s_{[n]}[r - 1, 0] = u_{[r]}$ ');
    case  $e \equiv 's_{[n]} = \text{neg}(t_{[n]})'$ :
        gran(' $s_{[n]} = t_{[n]}$ ');
    case  $e \equiv 's_{[n]} = \text{ite}(a_{[q]} = b_{[q]}, t_{[n]}, u_{[n]})'$ :
        gran(' $a_{[q]} = b_{[q]}$ '); gran(' $s_{[n]} = t_{[n]}$ '); gran(' $s_{[n]} = u_{[n]}$ ');
    case  $e \equiv 's_{[n]} = t_{[n]}$  and  $u_{[n]}$ ':
        gran(' $s_{[n]} = t_{[n]}$ '); gran(' $s_{[n]} = u_{[n]}$ ');
    ...
    case  $e \equiv 's_{[n]} = (t_{[q]} \otimes u_{[r]})[j, i]'$ :
        if ( $j < r$ ) {
            gran(' $s_{[n]} = u_{[r]}[j, i]$ ');
        } else if ( $i \geq r$ ) {
            gran(' $s_{[n]} = t_{[q]}[j - r, i - r]$ ');
        } else {
            gran(' $s_{[n]} = t_{[q]}[j - r, 0] \otimes u_{[r]}[r - 1, i]$ ');
        }
    case  $e \equiv 's_{[n]} = (t_{[q]}[l, k])[j, i]'$ :
        gran(' $s_{[n]} = t_{[q]}[k + j, k + i]$ ');
    case  $e \equiv 's_{[n]} = \text{ite}(a_{[q]} = b_{[q]}, t_{[r]}, u_{[r]})[j, i]'$ :
        gran(' $s_{[n]} = \text{ite}(a_{[q]} = b_{[q]}, t_{[r]}[j, i], u_{[r]}[j, i])$ ');
    case  $e \equiv 's_{[n]} = (t_{[q]} \text{ and } u_{[q]})[j, i]'$ :
        gran(' $s_{[n]} = t_{[q]}[j, i] \text{ and } u_{[q]}[j, i]$ ');
    ...
    case  $e \equiv 'x_{[n]}[j, i] = y_{[q]}[l, k]'$ :
        slice( $x_{[n]}$ ,  $j, i$ ); slice( $y_{[q]}$ ,  $l, k$ ); union( $x_{[n]} \langle j, i \rangle, y_{[q]} \langle l, k \rangle$ );
}

```

---

If two  $k$ -ary bitwise Bitvector functions  $\mathcal{F}_{[n]}^1$  and  $\mathcal{F}_{[m]}^2$  on bitvectors of width  $n$  and  $m$  operate according to the same characteristic Boolean function, then let this correspondence be denoted by  $\mathcal{F}_{[n]}^1 \simeq \mathcal{F}_{[m]}^2$ .

**Theorem 1 (Minimum Width Reduction)** Let  $k, n \in \mathbb{N}_+$  and let  $V = \{1, \dots, k\}$ . Let  $I \subseteq V \times V$  and let  $p \in \mathbb{N}_+$  be the number of connected components of the undirected graph  $G(V, I)$  with vertices  $V$  and edges  $I$ . Let  $m := \min(n, \max(1, k - p))$ . Then  $m$  is the minimum value for which the following holds:

$$\text{for all } k\text{-ary, } \begin{array}{c} \text{BvSat}(\mathcal{F}_{[n]}, I) \\ \text{bitwise } \mathcal{F}_{[n]} \end{array} \begin{array}{c} \text{satisfiable} \\ \iff \end{array} \begin{array}{c} \text{BvSat}(\mathcal{F}_{[m]}, I) \\ \text{satisfiable}. \end{array}$$

whereby for each  $\mathcal{F}_{[n]}$ ,  $\mathcal{F}_{[m]}$  denotes the corresponding bitwise bitvector function of width  $m$  width  $\mathcal{F}_{[n]} \simeq \mathcal{F}_{[m]}$ .  $\blacksquare$

Note that the reduced width  $m$ , which is computed in Theorem 1, depends only on the arity  $k$  of the bitwise functions and on the set  $I$  of inequalities. Thus, for each equivalence class  $C_i$  a reduced width  $\varphi(C_i)$  can be computed, which preserves satisfiability of the associated bitwise bitvector equations  $E_i$  in a one-to-one fashion.  $\varphi(C_i)$  only depends on the size of the equivalence class (i.e. on the number of chunks contained in  $C_i$ ) and on the number of connected graph components as induced by the inequalities, which are derived from guard expressions of *if-then-else* terms which involve variables of  $C_i$ .

The computation of the number of connected graph components for each class can efficiently be done by using a union-find algorithm, and moreover, can be embedded within the computation of the equivalence classes during the granularity decomposition.

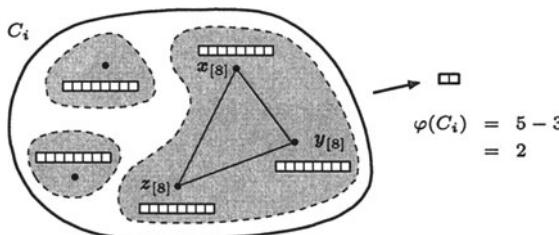


Figure 3. Minimum Width Abstraction

Figure 3 reconsiders Example 2, showing a sample equivalence class; inequalities are drawn as edges between chunks.

## 7. Experimental Results

The proposed abstraction technique has been implemented in C++ in a tool called BooSTER (Boolean String Length Reduction, Joh01a) and can easily be integrated into existing verification flows. BooSTER was tested in several case studies at the EDA departments of Siemens Corporation in Munich and Infineon Technologies in San Jose, CA. Experiments on large industrial circuits have demonstrated the applicability and efficiency of the presented technique. Good results have been achieved for specific types of digital designs, which provide a high degree of uniform data-flow, as for example memories, queues, stacks, bridges and interface protocols.

In an experiment we considered a design of roughly 3.000 lines of Verilog code with a synthesized netlist of 24.000 gates and 35.000 RAM cells. The RTL incorporates 16 FIFO queue buffers and complex control logic. Data packages are fed on 33 input channels to the management unit, stored in the FIFOs and upon request are output on one of 17 output channels, while the cell sequence has to be preserved and no package must be dropped from the management unit. Test cases were run on a PII 450 Mhz Linux PC with 128 MB. The tool operated as a preprocessor to the property checker used at Siemens and Infineon. All runtimes on reduced models were compared to those achieved on

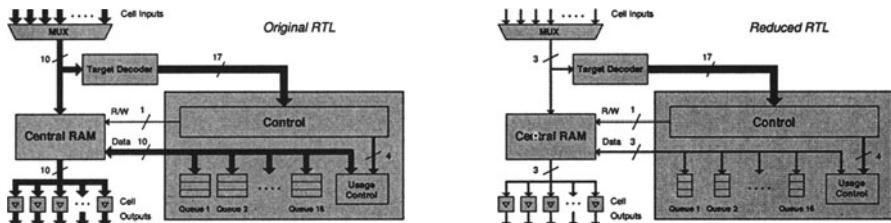


Figure 4. Sample Design before and after Scaling

the original designs without preprocessing. The width of the data-path could be reduced from 10 bits down to 3 bits. The size of the bit-level model was shrunken to  $\leq 30\%$  of the original size, and property checker runtimes dropped down to  $< 20\%$  of the former runtimes, see Table 2 for details.

Table 2. Experimental Results

|  | Property  | Original design  | Shrunken model   |
|--|---|--|--|
| Computation times for pre- and postprocessing          | <code>nop</code><br><code>read</code><br><code>write</code>                                 |  | 2.96 secs<br>6.53 secs<br>3.24 secs  |
| FIFO sizes on RTL                                      | <code>nop</code><br><code>read / write</code>   | 160 cells $\times$ 10 bit<br>160 cells $\times$ 10 bit | 160 cells $\times$ 2 bit<br>160 cells $\times$ 3 bit                             |
| # bits in all signals in cone of influence of property | <code>nop</code><br><code>read</code><br><code>write</code>                                 |  | 20925 (24.0 %)<br>31452 (33.6 %)<br>14622 (35.3 %)                               |
| # gates in synthesized netlist                         | <code>nop</code><br><code>read / write</code>   |  | 23801 (27.9 %)<br>23801 (33.3 %)   |
| # state bits   | <code>nop</code><br><code>read / write</code>   |  | 1658 (21.8 %)<br>1658 (31.6 %)   |
| Property checker runtimes                              | <code>nop</code><br><code>read</code><br><code>write_fail</code><br><code>write_hold</code> |  | 23:33 min (2.7 %)<br>42:23 min (8.1 %)<br>2:08 min (19.5 %)<br>27:08 min (4.2 %) |

## 8. Conclusion

Reduction of data path widths has always been a classical attempt of minimizing state place explosion for formal model checking methods. Many EDA companies today perform such reductions manually to reduce verification runtimes, often without having the guarantee that the chosen amount of scaling does not falsify verification results.

In this paper we presented a fully automated one-to-one RTL abstraction technique, which efficiently analyzes word-level data-flow in RTL descriptions with respect to a specified property. Designs are scaled down by reducing signal widths before property checking, while guaranteeing that the property holds for the scaled model if and only if it holds for the original design. The proposed technique can furthermore be applied in high-level equivalence checking and high-level simulation.

## Notes

1. Coarsest, in the sense of: as coarse as can be concluded by purely syntactical analysis. Yet in fact, for many practical applications in digital circuit design our technique yields the optimum decomposition.

## References

- [BCC<sup>+</sup>99] A Biere, A Cimatti, E M Clarke, M Fujita, and Y Zhu. "Symbolic Model Checking Using SAT Procedures instead of BDDs". In *Proc. DAC*, pages 317–320, 1999.
- [BDL98] C W Barrett, D L Dill, and J R Levitt. "A Decision Procedure for Bitvector Arithmetic". In *Proc. DAC*, pages 522–527, 1998.
- [Bry86] R E Bryant. "Graph-Based Algorithms for Boolean Function Manipulation". *IEEE Transactions on Computers*, 35(8):677–691, 1986.
- [CEJS98] E M Clarke, E A Emerson, S Jha, and A P Sistla. "Symmetry Reductions in Model Checking". In *Proc. CAV*, pages 147–158, 1998.
- [CGL92] E M Clarke, O Grumberg, and D E Long. "Model Checking and Abstraction". In *Proc. POPL*, pages 342–354, 1992.
- [CMR97] D Cyrluk, M O Möller, and H Rueß. "An Efficient Decision Procedure for the Theory of Fixed-Sized Bit-Vectors". In *Proc. CAV*, pages 60–71, 1997.
- [DJ90] N Dershowitz and J P Jouannaud. "*Handbook of Theoretical Computer Science, Formal Models and Semantics*", chapter "Rewrite Systems", pages 243–320. J.v. Leeuwen, Elsevier, 1990.
- [Dre00] R Drechsler. *Formal Verification of Circuits*. Kluwer Academic Publishers, 2000.
- [ET99] E A Emerson and R J Trefler. "From Asymmetry to Full Symmetry: New Techniques for Symmetry Reduction in Model Checking". In *Proc. CHARME*, pages 142–156, 1999.
- [Joh01a] P Johannsen. "BooSTER : Speeding Up RTL Property Checking of Digital Designs by Word-Level Abstraction". In *Proc. CAV'01*, pages 373–377, 2001.
- [Joh01b] P Johannsen. "Reducing Bitvector Satisfiability Problems to Scale Down Design Sizes for RTL Property Checking". In *IEEE Proc. HLDVT'01*, 2001.
- [Sil95] J P M Silva. "*Search Algorithms for Satisfiability Problems in Combinational Switching Circuits*". PhD thesis, University of Michigan, 1995.
- [SS00] J P M Silva and K A Sakallah. "Boolean satisfiability in electronic design automation". In *Proc. DAC*, pages 675–680, 2000.
- [ZKC01] Z Zeng, P Kalla, and M Ciesielski. "LPSAT: A Unified Approach to RTL Satisfiability". In *Proc. DATE*, pages 398–402, 2001.

# Functional Test Generation Using Constraint Logic Programming

Zhihong Zeng, Maciej Ciesielski

*Dept. of Electrical & Computer Engineering,  
University of Massachusetts,  
Amherst, MA 01003, USA,  
{zzeng, ciesiejl}@ecs.umass.edu*

Bruno Rouzeyre

*LIRMM, Universite de Montpellier,  
34090 Montpellier, France  
rouzeyre@lirmm.fr*

**Abstract:** Semi-formal verification based on symbolic simulation offers a good compromise between formal model checking and numerical simulation. The generation of functional test vectors, guided by miscellaneous coverage metrics to satisfy the simulation target, can be posed as a satisfiability problem (SAT). This paper presents a novel approach to solving SAT based on Constraint Logic Programming technique. The proposed SAT solver allows efficiently handling the designs with mixed word-level arithmetic operators and Boolean logic. It is applicable for designs specified at different levels, including HDL, RTL, and Boolean. The experimental results are quite encouraging compared with classical CNF-based, BDD-based, and LP-based SAT solvers.

**Key words:** Functional test generation, Satisfiability, Constraint Logic Programming, Validation, Verification

## 1. INTRODUCTION

Numerical simulation remains a dominant design validation method in industry since it scales well with the design complexity. A typical design verification scenario includes random and pseudo-random directed tests, bringing the functional coverage of the design specification to a desired level. Functional coverage metrics typically include line coverage, state coverage, transition coverage, branch coverage, etc. In the early design phase, both random and directed test vectors can help to find design bugs easily and

improve functional coverage quickly. When the functional coverage reaches certain level (say 90%) of coverage, improving the coverage and discovering corner cases by adding more random or manual test vectors becomes very inefficient. At this point, the remaining gap in functionality coverage can be solved by applying deterministic tests. The generation of such tests must satisfy a predefined simulation target, such as reaching a particular state of the design, exercising a branch, or covering a piece of HDL code, and is guided by miscellaneous coverage metrics and monitors. The functional test generation problem guided by such constraints can be posed as a satisfiability problem (SAT).

Several tools have been developed in industry and academia to facilitate the generation of deterministic test vectors. SIVA [9] is an example of such a tool, used to generate input vectors to exploit a larger state space and check the desired properties. The core algorithms in SIVA use a combination of BDD-based and ATPG tools to solve satisfiability. In our context of semiformal verification, a *symbolic simulation* engine is used to generate a set of symbolic expressions according to the simulation targets. The set of symbolic expressions is then transformed into a SAT instance. A solution to this SAT problem gives a sequence of input vectors to exercise the simulation target, or proves that it is not possible to find such vectors.

SAT belongs to the class of NP-complete problems, with algorithmic solutions having exponential worst-case complexity. Hence the efficiency of the semi-formal verification approach is largely determined by the performance of the SAT solvers. In this paper, we investigate a method for solving SAT problems that originate from RTL designs with mixed arithmetic and control logic that are common in the datapath of modern microprocessor and DSP designs.

## 2. PREVIOUS APPROACHES IN SOLVING SAT

### 2.1 Boolean Satisfiability

Classical approaches to SAT are based on variations of the well-known Davis and Putnam procedure [4] that works on CNF formulae. Typical versions of this procedure incorporate a chronological backtrack-based search [8]; at each node in the search tree, it selects an assignment and prunes the subsequent search by iterative application of the unit clause and pure literal rules. Recent approaches incorporate learning techniques and other conflict analysis procedures with *non-chronological* backtracks to prune the search space [15].

Another popular approach to solving the Boolean satisfiability problems is based on Binary Decision Diagrams (BDDs) [2]. Given a circuit for which a SAT instance needs to be solved, a set of BDDs can be constructed representing the output value constraints. The conjunction of all the constraints expressed as a Boolean product of the corresponding BDDs, referred to as a product BDD, represents the set of all satisfying solutions [13]. However, a major limitation of this approach is the memory explosion problem associated with the construction of the *product* BDD. Kalla *et al.* [14] proposed a BDD-based SAT technique that overcomes the problems related to BDD size by exploiting elements of the unate recursive paradigm. This technique searches for SAT solutions in the cofactors of the individual constraint BDDs, thus restricting the growth of the entire BDD search space.

CNF-based SAT solvers can be directly integrated into the semi-formal verification framework. However, practical RTL or behavioral descriptions often have word-level operators. Collapsing those word-level operators into single CNF formulae destroys the regularity of the problem and often makes the problem much harder to solve. On the other hand, BDD-based techniques suffer from the size explosion problems. For example, the size of a BDD for a multiplier is exponential, regardless of the variable ordering.

## 2.2 Hybrid Approaches

To overcome these drawbacks, Fallah *et al.* [7] proposed a *hybrid* satisfiability approach, HSAT, to generate functional test vectors for structured HDL designs. Working on the RTL descriptions, the hybrid method generates a set of *CNF clauses* for random Boolean logic, and *linear arithmetic constraints* for arithmetic blocks in the design. Then, a 3-SAT solver is applied to solve SAT for Boolean logic, while a Linear Programming (LP) technique is used to check the feasibility of linear constraints for arithmetic portion of the design. It should be noted that the two problems, SAT for Boolean logic and LP for arithmetic blocks, are solved separately, each in its own domain.

Constraint propagation techniques between different domains have also been explored to generate test vectors and check assertions for HDL descriptions [10], where word-level ATPG and modular arithmetic constraint-solving technique are combined to solve SAT problems. During the justifications in the Boolean domain, Boolean constraints are propagated to arithmetic domain using word-level implications as early as possible. However, the *word-level implications* on arithmetic operators are much weaker than the constraint propagation of a generic constraint solving technique, for example techniques described in [12], can provide. In other

words, constraint propagation in such framework is not efficient and relies on heuristics.

The following aspects are the main disadvantages of a hybrid approach with separate domains, compared to a unified approach.

(1) *Constraint propagation across two domains is inefficient.* This can be demonstrated with an example in Figure 1. Assume that  $a=1$  is the objective we want to satisfy. Figure 2(a) shows a possible decision tree for a hybrid approach, where either arithmetic constraints are not propagated to the Boolean domain as early as possible [7] or implication engine is not robust enough [10]. A better solution is shown in Figure 2(b), where putting Boolean constraints and word-level arithmetic constraints together results in a decision tree with early reduction of the search space.

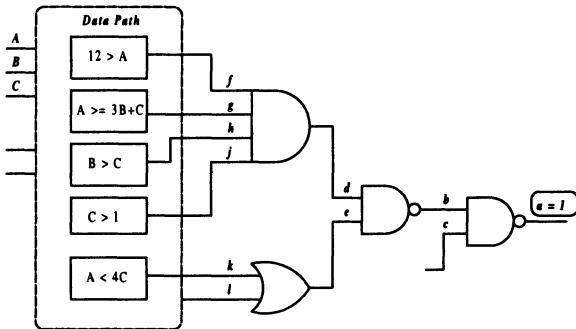


Figure 1. A circuit example for constraint propagation

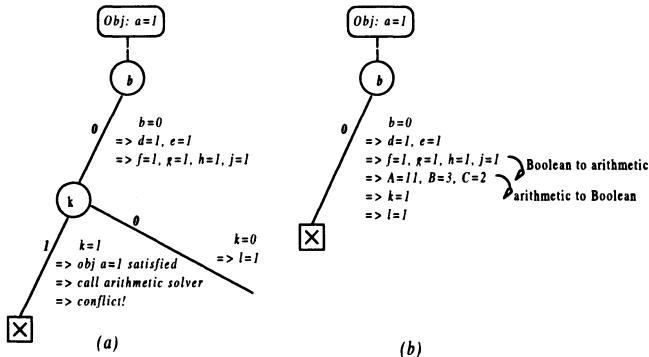


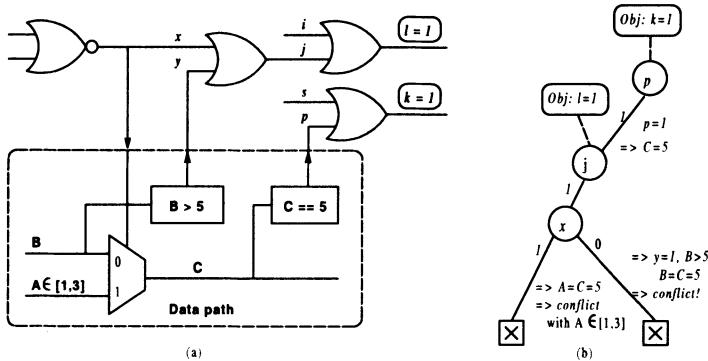
Figure 2. Decision tree with constraint propagation between two domains, (a) after objective justified in Boolean domain (b) as early as possible

(2) *Conflict-based learning across the boundary is difficult, if not impossible.* Assume that we are using a unified approach to solve a SAT problem in Figure 3(a), where in the middle of a Branch-and-Bound process the objectives  $k=1$  and  $l=1$  have to be justified. Figure 3(b) is a possible decision tree corresponding to this search. Upon a conflict occurring at node

$x$ , we can learn that  $[p=1 \Rightarrow j=0]$  and  $[C=5 \Rightarrow j=0]$  and by contra positive law,  $[j=1 \Rightarrow p=0]$  and  $[j=1 \Rightarrow C \neq 5]$ . Such a learning, especially when involving both Boolean and integer variables, is difficult to be derived and maintained in a hybrid approach.

### 2.3 Unified Word-Level Satisfiability

It would be desirable to use an infrastructure that can represent both the Boolean as well as arithmetic constraints in a single *unified* domain. By doing so, constraint propagation between the arithmetic and Boolean parts can be handled implicitly and efficiently. Zeng *et al.* [16] presented an enhanced word-level satisfiability solver, LPSAT, based on linear programming. By generating linear constraints for both the Boolean logic and the arithmetic operators, this approach allows to solve the SAT problem in a unified integer linear programming (ILP) domain. By doing so, LPSAT utilizes the implicit constraint propagation of the ILP solver.



*Figure 3.* A circuit for conflict based learning (b) a possible decision tree in a unified solver.

However, such generic ILP solvers tend to be inefficient in solving satisfiability problems encountered in RTL verification. First, generic LP solvers are based on numerical procedures that are designed predominantly to solve optimization problems, rather than satisfiability. As a result, they suffer from numerical convergence problems, and are sensitive to a number of internal parameters. Also, they tend to be inefficient in the Branch-and-Bound part for solving the decision problems, which are at the heart of SAT problems. Secondly, any nonlinear arithmetic operator in LP-based SAT has to be explicitly linearized into linear constraints. This includes the modeling of mixed arithmetic/Boolean blocks, such as comparators, shifters, multiplexors, etc., with integer decision variables that may lead to the numerical convergence problems. Finally, the ILP can only compute a single

solution; it is computationally expensive to force it to produce several different solutions during subsequent runs.

In this paper we investigate a new satisfiability solver based on *Constraint Logic Programming* (CLP). By transforming the SAT instances into predicates in *Logic Programming*, we preserve the regularity of the word-level operators. Compared to LP, the modeling of implications, often encountered in the verification problems, is simpler and more natural for CLP. Also the modeling of mixed blocks is easier: it does not require the introduction of (integer) variables and is not plagued with the numerical convergence problems. Another important aspect of this approach is that it allows generating multiple vectors, needed for simulation-based functional validation. Finally, efficient modeling of both arithmetic and Boolean domains inherent to CLP makes it applicable not only to satisfiability (or justification), but also to simulation (numerical and symbolic), and equivalence checking.

The rest of the paper is organized as follows: Section 3 explains how to generate a SAT problem from symbolic simulation using Prolog predicates. Section 4 discusses a practical aspect of modeling wide arithmetic operators. Finally, Section 5 gives the experimental results, and Section 6 contains concluding remarks.

### 3. FORMULATING SAT PROBLEM FOR RTL VALIDATION

#### 3.1 Functional Test Generation

*Deterministic functional test generation* is used to improve the functional coverage, especially targeting corner cases and hard-to-detect functional faults. Figure 4 shows a design validation flow on top of the functional test generation [11]. First, random simulation can be used to bring a design into a certain state, called *seed environment*. Starting at the seed environment, symbolic simulation is performed for several consecutive clock cycles (i.e. bounded time frames). A set of symbolic expressions is then generated and converted to a SAT problem.

The generation of symbolic expressions is guided by a *simulation target*, which is defined as a set of properties that needs to be verified through a simulation run. A simulation target can be specified by the user or derived from a coverage metric. It could be stated as simply as: “*Output signal A must take value h’109 after 5 clock cycles from the current simulation time*”. Or it could be as complex as exercising different branches at different time

frames. Static property checking can also become a simulation target since monitor statements can be added for each static property. An example of such a target is: “*Driving a common data bus from multiple sources is not allowed at the same time*”. Putting the symbolic expressions together with the constraints encoding the simulation targets or properties, a SAT instance is obtained.

In the test generation flow shown in Figure 4, any generic SAT solver can be invoked to solve a SAT problem. In this paper, we focus on solving SAT problems using Constraint Logic Programming that belongs to word-level SAT solvers.

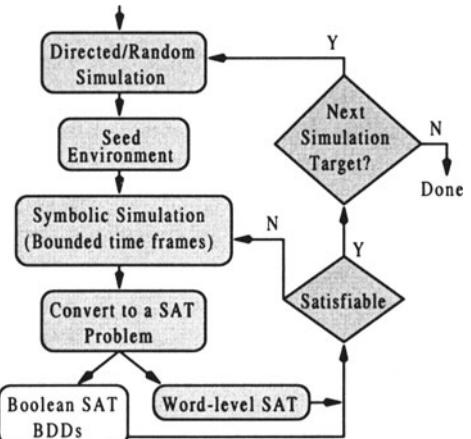


Figure 4. Functional test generation for design validation

### 3.2 Symbolic Simulation

Given a simulation target, symbolic expressions are generated using symbol propagation techniques. The resulting design description remains in the text format, thus minimizing a risk for memory explosion, commonly encountered in BDD-based representation [3]. Then, the symbolic expressions of a SAT instance are translated into a common intermediate representation (such as BLIF format) so that different kinds of SAT solvers can be applied to solve the SAT instance. The generated expressions capture only the portion of the design, which lies in the *cone of influence* of the simulation target, or a static assertion property. Thus a SAT problem generated by this approach remains small even when originating from a large design.

Figure 5 shows a 4-state finite state machine (FSM) of a simple datapath circuit with initial state S1. Assume, that through numerical simulation state S2 is reached from initial state S1 by applying a sequence of input vectors. Starting from state S2 (seed environment), we want to verify the following simulation target: “*Is the machine able to return to the initial state S1 in two clock cycles?*” Through a combination of symbolic simulation and SAT checking, we are able to formally answer the above question. First, a symbolic trace is generated as follows:

$$\begin{aligned}
 S^I &= f_1(A^0, B^0, Ctl^0, (S^0 = S2)) \\
 S^2 &= f_2(A^1, B^1, Ctl^1, S^I) = f_2(A^1, B^1, Ctl^1, F_1(A^0, B^0, Ctl^0, (S^0 = S2)))
 \end{aligned} \tag{1}$$

where  $f(A, B, Ctl, S)$  is the next state function for state variable  $S$ ,  $A^i$  denotes the symbolic value of input variable  $A$  at the  $i$ 'th clock cycle,  $S^i$  is the state value at the  $i$ 'th clock cycle. Together with the simulation target,  $S2 == S1$ , a SAT problem is formed.

In case when the SAT problem cannot be solved within reasonable or allowed time, we can decrease the problem size by fixing some symbolic variables, such as  $Ctl$  in Figure 5, to a constant value. In this case, we trade the SAT performance for the completeness. In the above example, if the symbolic variable  $Ctl$  is fixed to constant '0' then we may fail to explore some parts of state space by the two-cycle symbolic simulation.

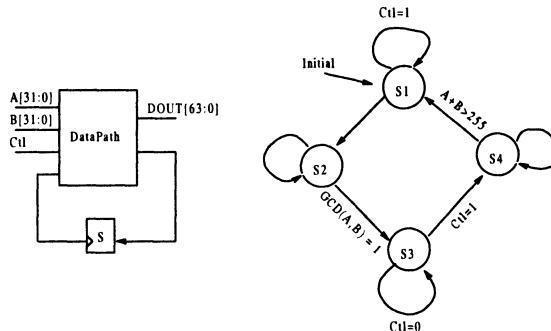


Figure 5. A FSM for a simple datapath circuit

### 3.3 Symbolic Expressions in Prolog

Constraint Logic Programming (CLP) is a constraint solving method based on logic programming. In recent decades, CLP drew extensive research interest and made a lot of progress in solving practical problems [12]. There are many publicly available CLP solvers based on different constraint solving techniques. Among them *GNU Prolog* (GProlog) [6, 5] has reported a good performance, even comparable to some of the commercial tools. It is a native Prolog compiler with constraint solving over the finite domain, which makes it especially suitable for solving our SAT problems.

| Type of Operators/Gates   | GNU Prolog Predicates          |
|---------------------------|--------------------------------|
| $Z = \text{and}(A, B)$    | $A \# \wedge B \# \leq \geq Z$ |
| $Z = \text{or}(A, B)$     | $A \# \vee B \# \leq \geq Z$   |
| $Z = \text{not}(A)$       | $A \# \backslash Z$            |
| $Z = A < +   - * > B$     | $Z \# = A < +   - * > B$       |
| $z = A < B$               | $z \# \leq \geq A \# < B$      |
| $A \Rightarrow B$         | $A \# \Rightarrow B$           |
| $z = A == B$              | $z \# \leq \geq A \# = B$      |
| $Z = \text{mux}(A, B, s)$ | $Z \# = s * A + (1-s)*B$       |

Table 2. Modeling of Boolean logic and arithmetic operators by Logic Predicates

The symbolic expressions are first translated into the widely accepted BLIF format, with the annotation made for any sub-module, if it is a word-level operator. The BLIF file is then transformed into a Prolog program. The GNU Prolog solver we used supports many built-in predicates in finite domain. These built-in predicates made our translation task from BLIF to Prolog quite straightforward. Table 1 has some examples illustrating how to model Boolean gates and arithmetic operators in terms of GNU Prolog predicates. Here ‘ $\# \wedge$ ’ stands for AND, ‘ $\# \vee$ ’ for OR, ‘ $\# \backslash$ ’ for NOT and ‘ $\# \leq \geq$ ’ means equivalence. For more details of the usage of GNU Prolog predicates, the reader is referred to [5].

## 4. HANDLING WIDE WORD-LEVEL OPERATORS

In realistic designs and RTL specifications, wide word-level signals (with bit width larger than 32 bits) are common. Unfortunately, the largest integer domain that can be allowed in GNU Prolog solver is currently limited to  $2^{28}$ . Any wide operator greater than 28 bits has to be decomposed into smaller blocks. For example, a 32 bits comparator  $c=(A[31:0] < B[31:0])$  can be decomposed into three smaller arithmetic operators and two Boolean gates, as shown in Figure 6.

There are two ways to decompose wide operators. In our experiments, the decomposition is done during the translation of symbolic expressions into BLIF representation. The other possibility is to perform the decomposition during the translation from BLIF (or any other intermediate format) to Prolog. This requires creating user-defined predicates (macros), with wide word-level operands decomposed into a set of sub word-level vectors.

## 5. EXPERIMENTS

We did a preliminary implementation of our SAT solver by integrating GProlog into our satisfiability-solving framework. The experimental results are quite encouraging compared with those of other satisfiability solvers. The whole process of reading the RTL Verilog design, decomposing wide operators, generating symbolic expressions, and solving SAT using GProlog is done automatically, without any human intervention. It is implemented in the framework of VIS system [1].

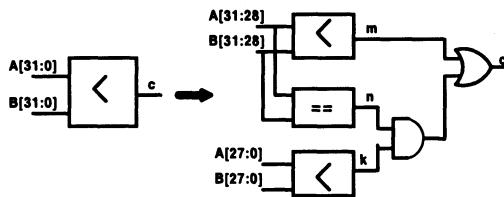


Figure 6. Decomposing a wide word-level comparator

We compared our CLP-based SAT solver, called *CLP-SAT*, to another word-level solver, LPSAT [16]; two CNF-based solvers, SATO [17] and GRASP [15]; and a BDD-based tool, B-SAT [14], over a range of available benchmarks. The overhead associated with transforming the SAT instance into CNF formulas were ignored. In order to get a fair comparison, we also ignored the overhead associated with translating SAT instance into linear constraints for LPSAT or predicates for GProlog. We observed that such a translation was within seconds or less for the experiments conducted here. All experiments were performed on a Pentium III/500MHz PC running Linux.

### 5.1 Description of Benchmarks

In order to have a better comparison with LPSAT, we used the SAT instances generated for the functional vector generation purpose reported in [16]. The experimental results are shown in Table 2.

The circuit square corresponds to a design whose output asserts high if ( $Z^2 = X^2 + Y^2$ ), where  $X$ ,  $Y$  and  $Z$  are 16-bit wide operators. The SAT instances *square(1)* and *square(0)* correspond to two different output requirements. The benchmark *quadratic* is an implementation of a solution to the quadratic equation  $X^2 + a*X + b = 0$ , where  $a$  and  $b$  are constants and  $X$  is a 16-bit variable. Given the constants  $a$  and  $b$ , the SAT instance corresponds to computing the value of  $X$ . Examples *linear(1)* and *linear(2)* are circuits with a relatively simple structure (a chain of comparators) but with a large number of primary inputs (over 1200). The two instances differ in their size.

*gcd20* and *gcd40* are extensions of the greatest common divisor (GCD), a 24-bit input sequential circuit. They are generated by symbolic simulation of GCD circuit over 20 and 40 time frames, respectively. *m13* and *m16* are 13-bit and 16-bit multipliers. Two different SAT instances for each were created: (*sat*) with a feasible solution, and (*non*) with a non-satisfiable requirement. Finally, *mdpe(1)/(2)*, is a circuit composed of a multiplier feeding a dynamic priority encoder, taken from a realistic design. The two cases differ in the size of the Boolean part of the circuit.

It should be emphasized, that all the test cases were comprised of both, the arithmetic and the Boolean parts, including the 16-bit multiplier circuits (certain amount of connecting Boolean logic is required due to wide operator decomposition).

In Table 2, column 2 (# *lines*) gives the code size of the corresponding GProlog program. Column 4 (# *constr*) gives the number of linear constraints generated by LPSAT, and column 6 (# *clauses*) gives the number of clauses in the CNF formulae.

## 5.2 Experimental Results

Table 2 shows the experimental results, where ‘—’ means not finishing within 3600 seconds. The CPU time is given in seconds. Column 3 shows a CPU time for CLP-SAT using GProlog. It is composed of two parts: one for the compilation (from Prolog input file to executable program), and the other for the actual execution. The compilation time ranges from about 1 to 12 seconds, which is a significant portion of the total solving time. The remaining columns report the size and performance of LPSAT, SATO, GRASP, and BSAT [14].

From the results, we can conclude that the satisfiability solver (CLP-SAT) based on GProlog competes very well with the established CNF-based solvers SATO and GRASP, and with the BDD-based SAT solvers, and is comparable to the performance of LPSAT. An interesting note is that LPSAT and CLP-SAT each failed on only one test case, *square(1)* and *mdpe(2)*. As a general observation, the word-level SAT approaches exemplified by CLP-SAT and LPSAT work well on large yet simple sequential designs like GCD. For CNF-based solvers these designs are too hard due to a large number of CNF clauses. Similarly, the BDD-based satisfiability tool, BSAT [14], could solve but small examples because of the excessive time/memory needed to create BDDs for the test circuits.

| tests     | CLP-SAT    |                    | LPSAT       |       | CNF-SAT      |           |            | BSAT  |
|-----------|------------|--------------------|-------------|-------|--------------|-----------|------------|-------|
|           | # of lines | time<br>comp / exe | # of constr | time  | # of clauses | SATO time | GRASP time | time  |
| m13(sat)  | 78         | 0.23 / 0.00        | 68          | 0.04  | 16704        | 2.51      | 187.24     | 137   |
| m13(non)  | 78         | 0.24 / 0.00        | 68          | 0.60  | 16704        | 12.12     | 1355.8     | 520   |
| m16(sat)  | 116        | 0.29 / 0.37        | 149         | 44.09 | 24720        | 722.35    | 2819.3     | —     |
| m16(non)  | 116        | 0.24 / 53.1        | 149         | 2.34  | 24720        | 132.12    | —          | —     |
| square(1) | 529        | 0.58 / 0.00        | 701         | —     | 77361        | —         | 1344       | —     |
| square(0) | 529        | 0.82 / 0.00        | 701         | 0.96  | 77361        | —         | —          | —     |
| quadratic | 413        | 0.78 / 4.29        | 469         | 0.05  | 72015        | 10.68     | 14.38      | 923.8 |
| linear(1) | 1109       | 1.53 / 0.00        | 950         | 0.37  | 36914        | 5.01      | 2.98       | —     |
| linear(2) | 3527       | 11.7 / 0.01        | 2749        | 1.34  | 77887        | 1.27      | 6.73       | —     |
| gcd20     | 876        | 1.10 / 0.01        | 542         | 0.03  | 117785       | —         | —          | —     |
| gcd40     | 1515       | 1.90 / 0.01        | 1062        | 0.08  | 248449       | —         | —          | —     |
| mdpe(1)   | 147        | 0.46 / 0.67        | 2933        | 1.12  | 29560        | 75.2      | 572.27     | —     |
| mdpe(2)   | 685        | 5.32 / —           | 3673        | 8.98  | 30851        | 4.4       | 59.1       | —     |

Table 2. Comparison of different SAT results

## 6. SUMMARY AND FUTURE WORK

We investigated a new word-level satisfiability checker based on Constrained Logic Programming. The new SAT checker has been successfully applied to solve problems posed from semiformal verification area. The preliminary results demonstrate that the proposed CLP-based SAT solver is a good alternative to other word-level SAT solvers, such as LPSAT [16], in verifying RTL designs with mixed arithmetic and Boolean logic.

In our future work we will continue to examine other CLP solvers besides GProlog. We shall also try to gain a better understanding of the inner workings of GProlog to explain the inconsistencies in some of the obtained results, such as *mdpe(2)* in Table 2. It is also worthy to investigate the applications of our CLP-based SAT solver for other verification purposes, such as equivalence checking and counterexample finding in modeling checking.

## 7. REFERENCES

- [1] R. K. Brayton and et al. Vis: *A system for verification and synthesis*. Proceedings of the Computer Aided Verification Conference, pages 428-432, 1996.
- [2] R. E. Bryant. *Graph based algorithms for Boolean function manipulation*. IEEE Transactions on Computers, C-35:677-691, August 1986.
- [3] R. E. Bryant. *Symbolic simulation-techniques and applications*. In Proc. of 27th Design Automation Conf., pages 517-521, June 1990.
- [4] M. Davis and H. Putnam. *A computing procedure for quantification theory*. Journal of the ACM, 7:201-215, 1960.
- [5] Daniel Diaz and Philippe Codognet. gnu.org/software/prolog, 1999.
- [6] Daniel Diaz and Philippe Codognet. *The GNU prolog system and its implementation*. In SAC (2), pages 728-732, 2000.
- [7] F. Fallah, S. Devadas, and K. Keutzer. *Functional vector generation for HDL models using linear programming and 3-satisfiability*. In Proc. Design Automation Conf., pages 528-533, 1998.
- [8] J. W. Freeman. *Improvements to propositional satisfiability search algorithms*. Ph.D. Dissertation, Dept. of Comp. and Inf. Sc., Univ. of Penn., May 1995.
- [9] M. K. Ganai, A. Aziz, and A. Kuehlmann. *Enhancing simulation with BDDs and ATPG*. In Proc. of Design Automation Conf., pages 385-390, June 1999.
- [10] C. Huang and K.-T. Cheng. *Assertion checking by combined word-level ATPG and modular arithmetic constraint-solving techniques*. In Proc. of Design Automation Conf., pages 118-123, 2000.
- [11] C. L. Huang. *Private communication*. Avery Design Systems, Inc.
- [12] Joxan Jaffar and Michael J. Maher. *Constraint logic programming: A survey*. The Journal of Logic Programming, 19 & 20:503-582, 1994.
- [13] S. Jeong and F. Somenzi. *A new algorithm for the binate covering problem and its application to the minimization of Boolean relations*. In ICCAD, pages 417-420, 92.
- [14] P. Kalla, Z. Zeng, M. J. Ciesielski, and C. Huang. *A BDD-Based satisfiability infrastructure using the unate recursive paradigm*. In Proc. DATE, pages 232-236, 2000.
- [15] J. Marques-Silva and K. A. Sakallah. *GRASP - a new search algorithm for satisfiability*. In ICCAD-6, pages 220-227, 1996.
- [16] Z. Zeng, P. Kalla, and M. Ciesielski. *LPSAT: A unified approach to RTL satisfiability*. in Proc. DATE, pages 398-402, March 2001.
- [17] H. Zhang. *Sato: An efficient propositional prover*. In Proc. of 14th Conference on Automated Deduction, pages 272-275, 1997.

# An Industrial Approach to Core-Based System Chip Testing

Erik Jan Marinissen

*Philips Research*

**Abstract:** System chips are increasingly being designed by embedding reusable pre-designed and pre-verified cores. Modular, core-based test development is an attractive proposition for such large and complex ICs. This paper outlines the approach developed and used in Philips for core-based testing. It consists of four components: (1) a standardized set of test deliverables for a core, (2) standardized on-chip test access hardware, (3) a tool for translation of core tests into SOC tests, and (4) a tool for test scheduling.

**Key words:** system chip, manufacturing test, test wrapper, test access mechanism, test expansion, test scheduling, Macro Test

## 1. INTRODUCTION

Modern semiconductor process technologies enable the manufacturing of a complete system on one single die, the so-called *system chip* or SOC. Such system chips typically are very large ICs, consisting of millions of transistors, and containing a variety of hardware modules. In order to design these large and complex system chips in a timely manner and leverage from external design expertise, increasingly reusable cores are utilized. *Cores* are pre-designed and pre-verified design modules, meant to be reused in multiple SOC designs. Examples of cores are CPUs, DSPs, media co-processors, communication modules, memories, and mixed-signal modules. Core-based design divides the IC design community into *core providers* and *core users*. Core-based design takes place within companies, as well as between companies, i.e., one company acts as core provider, while another

company is the core user. Especially in the latter scenario, legal issues with respect to intellectual property rights and liability come to the table.

Due to imperfections in their manufacturing process, all electronic systems need to be tested for manufacturing defects. System chips are no exception to that rule. In traditional system *boards*, the components (ICs) used are tested by their provider (the IC manufacturer), and hence the system integrator only needs to test the rest of the board — often mainly interconnect wiring (cf. Figure 1(a)). For system *chips*, this situation is quite different. The components (cores) cannot be tested for manufacturing defects by their providers, as what is transferred from provider to user is merely a description, and hence not yet manufactured [1]. Manufacturing only takes place once the entire system chip is assembled (cf. Figure 1(b)). Therefore, a system chip integrator is responsible for testing both the user-defined circuitry surrounding the embedded cores, but also the embedded cores themselves. Even though the core provider might have tested his product for design errors and other users of the same core might have tested it in their ICs for manufacturing defects, that does not take away the obligation of the core user to test his system chip for manufacturing defects.

Testing of core-based system chips is best done in a core-based fashion: the core providers prepare their products with the right design-for-testability hardware and create test patterns for the cores, while the system chip integrator adds SOC-level design-for-testability and creates a system chip test using the core-level tests as building blocks [1]. In this core-based way of SOC test development, the system chip integrator does not need to be bothered with the content of the cores he is using in order to develop a test of sufficient quality. In many cases, core providers also do not want to share the implementation details of their cores in order to protect their intellectual property in the design of the core. They might deliver their cores as layouts or encrypted netlists, without providing RTL or netlist-level sources. In such a situation, the core user needs to rely on the tests as provided by the core provider, as he does not have access to the core's implementation details and hence cannot generate the tests himself. Furthermore, core-based testing allows for test reuse. A test developed once for a given core, can be reused for many instantiations of that core in different system chips, hence optimizing the usage of scarce engineering resources. And, if proper procedures and tools are in place, this way of working should also save precious development time on the side of the system chip integrator, thereby contributing to a shorter time-to-market.

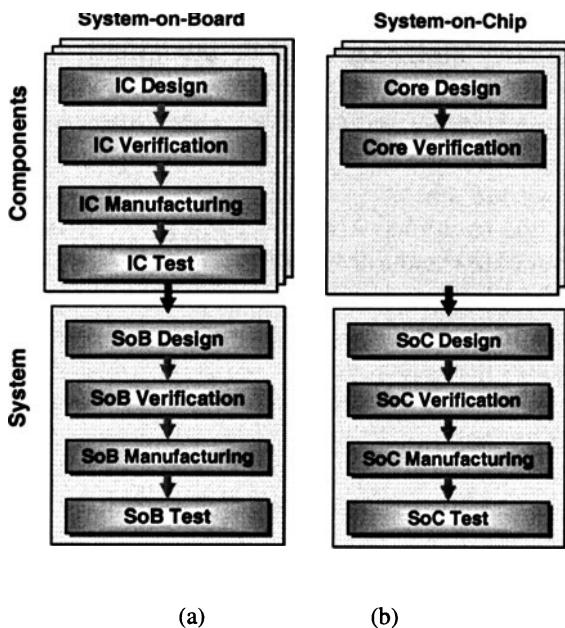


Figure 1. System-on Board (a) vs. System-on-Chip (b) trajectory.

Efficient design of system chips is very important to Philips Semiconductors. Hence, Philips has strongly embraced the paradigms of core-based system chip design, and, consequently, core-based testing. This paper details what solutions have been put into place within Philips to enable effective and efficient core-based testing.

The sequel of this paper is organized as follows. Section 2 lists the three main challenges involved in core-based testing and their proposed solutions. Subsequently, Section 3 provides the overview of and motivation for the Philips-internal solutions to these challenges. Sections 4, 5, and 6 provide more detail on each of the solutions. Section 7 briefly describes some experiences with using this core-based test approach on industrial SOCs. Finally, Section 8 concludes this paper.

## 2. CHALLENGES IN CORE-BASED TESTING

The traditional challenge for IC test engineers is to find a right balance between (1) test quality, (2) test costs, and (3) test development time.

Next to this traditional challenge, which of course remains in full force, also for core-based SOC designs, there are new challenges, specifically related to core-based testing. Marinissen and Zorian [2] listed the following core-based test challenges.

**1. Distributed test development.**

Core-based design and test development is an effort distributed over multiple parties, spread over location, time, and company. The joint responsibility of all parties involved for the SOC test requires the transfer of ‘core test knowledge’ from core provider to core user.

Required solution: A standardized set of deliverables, preferably delivered in standardized directory structures and file formats.

**2. Test access to embedded cores.**

Cores are typically deeply embedded inside the SOC design. Testing takes place at the SOC pins, and hence electrical test access has to be created from the SOC pins to the core-under-test and vice versa.

Zorian et al. [1] introduced a generic conceptual architecture for test access to embedded cores. It consists of (1) a *source* and *sink*, that generate the test stimuli and evaluate the test responses, (2) a *test access mechanism* (TAM) that bridges the physical distance between source/sink and core, and (3) a *core test wrapper*, that isolates the core during testing and provides switching between functional and test access.

Required solution:

- A standardized on-chip test access infrastructure.
- Tools for test translation from core to SOC.

**3. SOC-Level test optimisations.**

A modular SOC test approach allows optimization trade-offs. Which cores will be tested as stand-alone units, and which cores will be merged and tested as part of the overall SOC-level test? How to design and optimise the test access infrastructure? How to schedule the various tests, such that the test application time is minimized, but for example a power dissipation budget is not exceeded?

Required solution: Tools to evaluate and implement the various trade-offs.

### **3. OVERVIEW OF PHILIPS’ CORE-BASED TEST APPROACH**

Modular testing, in which modules are tested as separate entities, is nothing new in Philips. Already in the mid 1980s, *Macro Test* was defined by Beenker et al. [3]. Already back then, ICs were composed of multiple modules, termed *macros*. These macros had different circuit structures, such

as logic, memories, PLAs, register files, etc. Different circuit structures exhibit different defect behavior, and hence require dedicated test pattern generation and independent testing. Tools were developed for automatic test expansion from macro level to IC level, and those have been successfully used for numerous industrial ICs [4].

In finding test access paths from the embedded macro terminals to the IC pins, one inevitably has to make use of other on-chip circuitry. In 1980-90, adding dedicated additional test access hardware was considered too expensive. Hence, the tool for test protocol expansion was made such that it was capable of utilizing existing functional access paths through neighboring macros. It would make use of the scan chains in neighboring macros, as well as exploit functional transparent paths, both at the gate level as well as at higher levels of abstraction.

Times have changed since the mid 1980s. What used to be entire ICs are now only cores on a system chip. The sheer size of some system chips is impractical or intractable for some DfT insertion and/or test generation tools, and hence this asks for a ‘divide-and-conquer’ solution. Some cores come from external sources and have their implementation details hidden, i.e., we do not have available transparent paths through them for test access to neighboring cores. Silicon area is still a cost factor, but has relatively decreased in importance, whereas time-to-market has gained importance.

In 1997, Philips formed an internal team to study core-based testing. This team was named CTAG: Core Test Action Group. Its mission was to define additional methods and tools, on top of Macro Test, that would improve and ease core-based testing. CTAG defined two items.

- *Standardized deliverables.* These standardized deliverables have been integrated into the CoReUse Standards & Constraints, that define similar items in other areas than testing.
- *Standardized wrapper and TAM,* named TestShell and TestRail. These have been documented in a Philips-internal standard. Some aspects of TestShell and TestRail have been patented. Furthermore, tools have been developed for the automatic generation and verification of TestShells and TestRails.

TestShell and TestRail are dedicated on-chip test access hardware, and therefore mark a clear difference with the early years of Macro Test usage, when such dedicated access hardware was considered to be too expensive. TestShell and TestRail ensure guaranteed test access; the accessibility of a core does not depend on neighboring circuitry, but is handled via dedicated ‘highways’. This also makes the task of test protocol expansion a lot easier and hence contributes to reduced test development times.

Table 1 summarizes the core-based test challenges and required solutions, as discussed in Section 2 and adds to that an overview of the Philips-internal variants of those solutions.

*Table 1. Summary of challenges and solutions in core-based testing.*

| Challenge                        | Required Solution   | Philips Solution  |
|----------------------------------|---|---|
| 1. Distributed test development  | Standardized set of deliverables                                  | CoReUse Standards & Constraints                             |
| 2. Test access to embedded cores | Standardized on-chip access hardware + Tools for test translation | CTAG TestShell and TestRail                                 |
| 3. SOC-level optimization        | Tools for evaluating trade-offs                                   | CAT Test Protocol Expansion<br>CAT Test Protocol Scheduling |

## 4. COREUSE STANDARDS AND CONSTRAINTS

CoReUse *Standards* define required and optional delivery views for soft, firm, and hard cores, as well as standardized directory structures and file naming conventions. The standard delivery views ensure that core providers know what should be delivered, and core users know what they can expect. Delivery views include the following.

- Inserted DfT, such as scan chains or BIST.
- Inserted TestShell and its verification report.
- Test pattern lists and their respective fault coverages.
- Test protocols for the various test pattern lists.
- Diagnostic information.

CoReUse *Constraints* define requirements regarding the contents of the delivered files. There are Constraints with respect to RTL design, Verilog and VHDL, Verilog coding, VHDL coding, coding of scripts, test bench design and test suite development, packaging, test and testability, and testability port naming conventions.

## 5. CTAG TESTSHELL AND TESTRAIL

The CTAG TestShell, as depicted in Figure 2, has been published first by Marinissen et al. [5]. The TestShell supports four basic modes: (1) normal functional mode, (2) an inward-facing IP test mode, (3) an outward-facing interconnect test mode, and (4) the bypass mode.

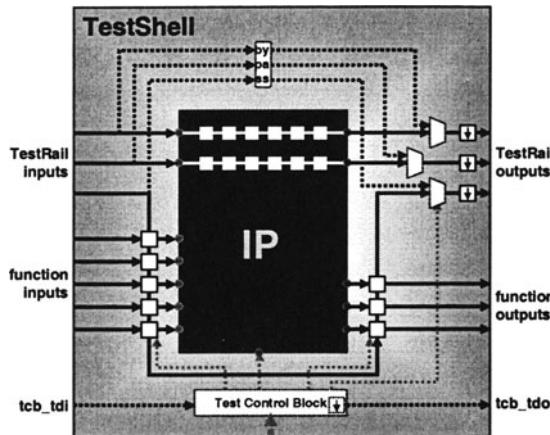


Figure 2. Conceptual view of the TestShell.

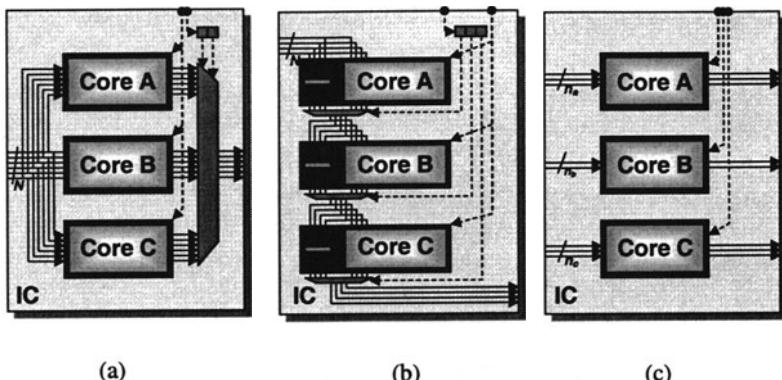
The TestShell has the following interface.

- Functional inputs and outputs, directly corresponding to the functional inputs and outputs of the unwrapped core.
- A one-bit test control input and output.
- A scalable multi-bit test data input and output, also referred to as the TestRail plug.

The TestShell consists of the following components.

- A *Test Cell* for every core terminal. The test cell provides controllability as well as observability.
- An (optional) *Bypass Register*, which allows a TAM to bypass core and wrapper, in order to test another core that is connected to the same TAM.
- A *Test Control Block* (TCB). The TCB has a bit-slice nature and consists of a shift and an update register. The TCB is primarily meant to control the operation of the TestShell, through several mandatory bit slices. Additional user-defined bit slices can be added for control of core-internal test modes.

A TestRail is a dedicated TAM in between TestShells. In principle, a TestShell is connected to the same TestRail at both input and output. Therefore, the TAM input plug and the TAM output plug of a TestShell normally have the same width. At SOC-level, the number, widths, and configuration of the TestRail is completely user-defined. In [6], three alternative TAM configurations were published (see Figure 3); TestRail supports all of these and others.



**Figure 3.** Examples of three TAM configurations [6]: (a) Multiplexing, (b) Daisychain, and (c) Distribution architectures.

## 6. MACRO TEST TOOLS

Macro Test has originally been defined by Beenker et al. [3]. The application of its concepts in the domain of core-based testing have been reported in [7,8].

One of the fundamental concepts behind Macro Test is that a test of a core is split up into a *test protocol* and a list of *test patterns*. The test protocol describes how to apply one test pattern in a pattern-value independent way. It does not specify how many patterns need to be applied, nor what the data content of the test patterns is.

Initially, both test protocol and test patterns are defined at the core terminals, as if the core terminals are directly accessible. While the bulk of the data, contained in the test patterns, remains untouched, subsequent tasks that translate the core-level test into a SOC-level test are performed on test protocols only.

*Test protocol expansion* translates the initial test protocol, defined at the core terminals, into a SOC-level test protocol, defined at the IC pins. This process involves exhaustive path searching. It is capable of exploiting *transfer* descriptions of neighboring cores at any level of hierarchical abstraction. The general test protocol expansion problem has been proven to be *NP-hard*. However, the test protocol expansion tool has a built-in preference for test access via TestShells and TestRails, as they provide a dedicated test access infrastructure and hence guarantee an efficient and successful expansion.

*Test protocol scheduling* tries to overlap the expanded test protocols of various cores in order to minimize the resulting test vector set and test

application time [8]. Our definition of the test protocol scheduling problem has been shown to be *NP-hard* and similar to the No-Wait Job Shop Scheduling problem [9].

*Test assembly* creates test vectors by filling in the test patterns into the corresponding test protocols. Our test assembly tool can write out test vectors for netlist simulation (together with a simulation test bench), or for the various ATE models in use. Test assembly can be performed on the unexpanded test protocols, in order to create verification suites for the core's test patterns. Also, test assembly can be performed for the expanded and scheduled test protocols, to generate the final test vector set for a core at SOC level [7,8].

## 7. APPLICATION EXAMPLES

The core-based test approach as described in this paper has been applied to many industrial Philips SOCs. In this section, we briefly report on two case studies.

The first case study reported by Arendsen and Lousberg [10] applied a core-based test strategy to a Digital Still Camera (DSC) IC. The IC was manufactured in a  $0.35\text{ }\mu\text{m}$  five metal layer CMOS technology with a total die area of  $50\text{ mm}^2$  (see Figure 4(a)). It contained a mix of nine hard, firm, and soft cores. One of the cores, a bus control unit, was considered too small to be treated as a stand-alone unit, and hence tested as part of the overall interconnect test. The other eight cores were tested as individual cores. TestShells and distributed TestRails were added to them, and their SOC-level tests were generated by means of the Macro Test tools.

The design benefits included a clear separation of tasks between core provider and core user, easy integration with respect to testing, and strongly reduced ATPG run times due to the divide-and-conquer strategy. The relative area costs are listed in Table 2. The table shows that the additional area costs for TestShell and TestRail depend strongly on the size of the core. In this case, the additional area costs attributed to TestShell and TestRail varied between 0.5% for the largest core and 22.1% for the smallest core. In total 7.7% of the total chip area was devoted to test infrastructure; 4.5% for the core-internal scan chains and an additional 3.2% for TestShell and TestRail, which enabled the core-based test approach. These additional area costs were considered acceptable in the view of the associated benefits.

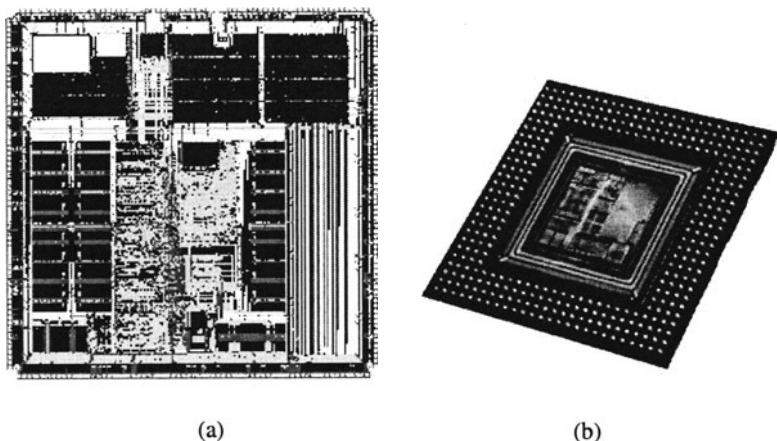


Figure 4. Layout of DSC-IC (a) and chip photo of CPA-IC (b).

Table 2. Relative DfT area costs for DSC-IC

| Core               | Core Area<br>as Part of<br>Total IC | Functional<br>Area | DfT Area         |                         |       |
|--------------------|-------------------------------------|--------------------|------------------|-------------------------|-------|
|                    |                                     |                    | Core<br>Internal | TestShell +<br>TestRail | Total |
| RISC Core 1        | 69.7%                               | 96.8%              | 2.7%             | 0.5%                    | 3.2%  |
| RISC Core 2        | 11.2%                               | 92.3%              | 5.7%             | 2.0%                    | 7.7%  |
| Peripheral 2       | 7.5%                                | 76.7%              | 11.6%            | 11.7%                   | 23.3% |
| UART (2×)          | 4.9%                                | 77.6%              | 9.6%             | 12.7%                   | 22.3% |
| SDRAM Interface    | 3.0%                                | 83.0%              | 12.0%            | 5.0%                    | 17.0% |
| Debug Support Unit | 1.8%                                | 72.8%              | 10.8%            | 16.4%                   | 27.2% |
| Peripheral 1       | 1.5%                                | 69.7%              | 8.1%             | 22.1%                   | 30.2% |
| Bus Control Unit   | 0.4%                                | 90.3%              | 9.7%             | 0.0%                    | 9.7%  |
| Total              | 100.0%                              | 92.3%              | 4.5%             | 3.2%                    | 7.7%  |

A second case study was reported by Van Beers and Van Herten [11]. It involves a Co-Processor Array (CPA) for digital video applications, consisting of twelve large video co-processors and a switch matrix, and manufactured as 7.6 M transistors (incl. 0.5 M logic gates and 90 K flip flops) in a 0.35  $\mu\text{m}$  five metal layer CMOS technology with a total die area of 170  $\text{mm}^2$  (see Figure 4(b)). Also this IC was tested in a core-based fashion. In total 13 cores were equipped with TestShells and connected into one 48-bit wide daisychained TestRail.

## 8. CONCLUSION

In order to improve on design productivity and import external expertise, large system chips are increasingly being designed by embedding reusable

pre-designed and pre-verified cores. For such core-based system chips, modular, core-based, test development is an attractive proposition.

Philips' core-based test strategy includes these four solutions.

1. The CoReUse Standards and Constraints define standardized deliverables, both with respect to form and content.
2. The CTAG TestShell and TestRail are the Philips-internally standardized, but scalable wrapper and TAM.
3. Test protocol expansion is the Philips tool for test translation. By working only with test protocols and abstracting from the actual test pattern values, the computational complexity involved is significantly reduced.
4. Test protocol scheduling is the Philips tool that implements one of the possible tools for SOC-level optimization.

Philips' core-based test strategy has been and continues to be successfully used on numerous industrial SOC designs, and has become the de-facto DfT and test generation strategy within the company. For the future we expect that some of our internal standards will be replaced by industry-wide standards, such as, IEEE P1500 SECT [12].

## **9. ACKNOWLEDGEMENTS**

I gratefully acknowledge the contributions of many Philips colleagues to the achievements described in this paper. Special thanks go to my original fellow members of CTAG: Robert Arendsen, Gerard Bos, Hans Dingemanse, Maurice Lousberg, and Clemens Wouters. The data on the various applications of core-based testing have been obtained by Robert Arendsen, Jos van Beers, and Harry van Herten.

## **10. REFERENCES**

- [1] Y. Zorian, E.J. Marinissen, and Sujit Dey. Testing Embedded-Core Based System Chips. In Proceedings IEEE International Test Conference, pages 130–143, Washington, DC, October 1998.
- [2] E.J. Marinissen and Y. Zorian. Challenges in Testing Core-Based System ICs. IEEE Communications Magazine, 37(6):104–109, June 1999.
- [3] F. Beenker, K. van Eerdewijk, R. Gerritsen, F. Peacock, and M. van der Star. Macro Testing: Unifying IC and Board Test. IEEE Design & Test of Computers, Vol. 3(No. 4):26–32, December 1986.
- [4] F. Bouwman, S. Oostdijk, R. Stans, B. Bennetts, and F. Beenker. Macro Testability; The Results of Production Device Applications. In Proceedings IEEE International Test Conference, pages 232–241, September 1992.

- [5] E.J. Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In Proceedings IEEE International Test Conference, pages 284–293, Washington, DC, October 1998.
- [6] J. Aerts and E.J. Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In Proceedings IEEE International Test Conference, pages 448–457, Washington, DC, October 1998.
- [7] E.J. Marinissen and M. Lousberg. The Role of Test Protocols in Testing Embedded-Core-Based System ICs. In Proceedings IEEE European Test Workshop, pages 70–75, Konstanz, Germany, May 1999.
- [8] E.J. Marinissen. The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs. Journal of Electronic Testing: Theory and Applications, Vol. 18(No. 4), August 2002.
- [9] M. Pinedo. Scheduling - Theory, Algorithms, and Systems. Prentice Hall, Englewood Cliffs, New Jersey, 1995.
- [10] R. Arendsen and M. Lousberg. Core Based Test for a System on Chip Architecture Framework. In Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems, pages 5.1–1–8, Washington, DC, October 1998.
- [11] J. van Beers and H. van Herten. Test Features of a Core-Based Co-Processor Array for Video Applications. In Proceedings IEEE International Test Conference, pages 638–647, Atlantic City, NJ, September 1999.
- [12] E.J. Marinissen, R. Kapur, and Y. Zorian. On Using IEEE P1500 SECT for Test Plug-n-Play. In Proceedings IEEE International Test Conference, pages 770–777, Atlantic City, NJ, October 2000.

# **Power-constrained Test Scheduling for SoCs under a "no session" scheme**

**Marie-Lise Flottes, Julien Pouget, Bruno Rouzeyre**

*Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier,  
161 rue Ada, 34392 Montpellier cedex 5, France*

**Abstract:** This paper considers the scheduling problem of core tests in a system. Our objective is to minimize the total system test time while respecting system constraints in terms of power consumption and test resource sharing. A simple and effective scheduling heuristic is proposed based on a no sessions based scheme for better overall test time optimisation.

**Key words:** System-on-Chip, test scheduling

## **1. INTRODUCTION**

Present need of high-performance complex systems leads to modify the classical design style, based on standard IC assembling, to System-on-a-chip design (SoC), based on core and user-define logic (UDL) integration. While individual functions were tested before assembling in the classical approach, the SoC design technique counter part is that it postpones the test of every core and UDL at the end of the system implementation leading thus to prohibitive test time. In this context, an optimised test scheduling of individual SoC functions allows minimizing the system test time and thus the system cost. Obviously, the minimal system test time would be achieved by simultaneously testing all the individual functions (cores for simplicity). But, most of the time, design constraints prevent this full parallelism. Individual tests are conflicting because 1/ they share common test resources (e.g. test bus, test response compactors...), or 2/ the power consumption during simultaneous testing exceeds the device power allowance.

Since many years, numerous techniques have been reported in the literature for testing individual functions and thus cores (memories, processors, application specific functions...). An overview of BIST methods for several core types can be found in [1] for instance. More recently, mechanisms for accessing SoC cores have also been explored [2], [3], [4], [5]. Test access mechanism provides access to the cores from the test "sources" and "sinks" (system IOs in external testing, test pattern generators and response compactors in BIST) but limits the parallelism of test application.

The power consumption is another factor that may impact the test parallelism. Test power dissipation is a function of time, it depends of the switching activity resulting from the application of a new test vector to the system. For simplicity, the core power dissipation can be assigned to a fixed value [6]. A pessimistic point of view consists in defining this value as the maximum power dissipation over all test vectors send to the core (peak power). With this assumption, two cores cannot be tested in parallel if the sum of their peak power goes beyond the maximum allowable system power dissipation, even if their peak power do not occur at the same time! However, such assumption prevents any test parallelism that could damage the system.

This paper presents a heuristic for solving core test scheduling in a system, taking into account the two test parallelism constraints mentioned above. We target the minimal system test time. With this heuristic, test plan is not scheduled into test sessions; every core test starts as soon as resource and power constraints allow it.

The organization of the paper is as follow. In section 2, we present the advantages and drawbacks of several test control schemes: organizing tests in sessions of equal or unequal length, or with no session. Corresponding algorithmic complexities are discussed in section 3. In section 4, we present an algorithm for solving the "no session" scheduling problem. Experimental results are detailed in section 5 and section 6 concludes the paper.

## **2. TEST CONTROL SCHEMES**

The basic approach in test scheduling is to organize tests for the target modules into so-called test sessions. A test session brings together the tests of compatible modules. This compatibility is checked with respect to the test resource sharing needs and the power threshold to not exceed. Related test scheduling techniques assume either equal length test sessions or unequal length test sessions.

With the first assumption, the tests to perform are first assigned to test sessions. After completion, the length of each session can be set to the length of the longest test in that session for further test time optimisation.

This technique does not necessarily lead to the optimal scheduling solution in terms of system test time. Actually, when all test sessions are assumed to have the same length, the scheduling problem consists in minimizing the number of test sessions and not really the overall test time: let's consider an hypothetic SoC composed of three cores, 1, 2 and 3 with respective test lengths  $\text{Length}(1)=1000$ ,  $\text{Length}(2)=800$  and  $\text{Length}(3)=500$  - test lengths are expressed in terms of number of clock cycles-. Their respective power consumption is 500 mw, the maximal power consumption for the system is 1w and there is no test resource conflict between the cores. In this example,  $\text{Max}_i \text{Length}(i)=1000$  for  $i \in \{1,2,3\}$ . Consequently, the "equal session length" assumption leads to consider test sessions of 1000 clock cycles. A first scheduling solution Sol1 consists in testing cores C1 and C3 in parallel then to test core C2 in a second test session. A second solution Sol2 consists in testing cores C1 and C2 in parallel then to test C3 in a second test session. There is no solution with only one test session due to the power consumption constraint. The solutions Sol1 and Sol2 are equivalent in terms of number of test sessions for a total test time of  $2 \times 1000 = 2000$  clock cycles. Any algorithm based on the assumption of equal length test sessions can equally generate them. However, the test can be stopped after  $1000+800=1800$  clock cycles in the first solution, and after  $1000+500=1500$  clock cycles in the second solution.

This example shows that assuming equal lengths for the test sessions and thus targeting the minimal number of test sessions is not sufficient for guarantying the minimal test time at system level.

The second classical scheduling approach assumes unequal length test sessions. In this context, a session length is the time taken to test the core requiring the longest time in the session ( $\text{Length}(S_j) = \text{Max}_{i \in S_j} \text{Length}(i)$ ). With this approach, solution 2 is obtained. Compared to the previous scheme, an additional problem here is to map core tests to test sessions.

Finally, test for cores can be organized without session using a test scheme where the test of every core starts as soon as possible considering resource and power constraints.

Let's compare these two last approaches with help of the following example: the SoC under test includes fourteen cores whose characteristics are given in table 1. Resource constraints are very relaxed in order to let the scheduling solution space sufficiently large (the degree of incompatibility is about 10%). Following pairs (i, j) means that the tests of cores i and j are incompatible due to resource conflicts: (1,3), (1,6), (1,14), (2,9), (2,12), (2,13), (3,9), (3,12), (4,7), (7,12). The power limit is assumed to be  $P_{max}=30$  units.

| Cores | Test length Di<br>(# clock cycles) | Power consumption<br>during test mode Pi<br>(unit) |
|-------|------------------------------------|--|
| 1     | 20000                              | 11   |
| 2     | 11000                              | 9  |
| 3     | 10000                              | 11   |
| 4     | 19000                              | 6  |
| 5     | 10000                              | 16   |
| 6     | 4000                               | 15   |
| 7     | 16000                              | 10   |
| 8     | 7000                               | 2  |
| 9     | 16000                              | 9  |
| 10    | 6000                               | 6  |
| 11    | 9000                               | 7  |
| 12    | 3000                               | 3  |
| 13    | 7000                               | 15   |
| 14    | 5000                               | 8  |

*Table 1.* A SoC example: core's test length and power characteristics

## 2.1 Unequal length test sessions scheduling

According to this session-based scheme, the SoC test example can be scheduled as described in Figure 1. The core 1 fixes the length of the first test session (S1) to 20000 clock cycles. In the same way, the core 9 fixes the length of the second test session, and cores 4, 14 and 5 respectively fix the length of test sessions S3, S4 and S5. The total test length is equal to 72000 clock cycles. For information, this solution represents a gain of 30000 clock cycles compared to a solution generated under the assumption of equal length test sessions.

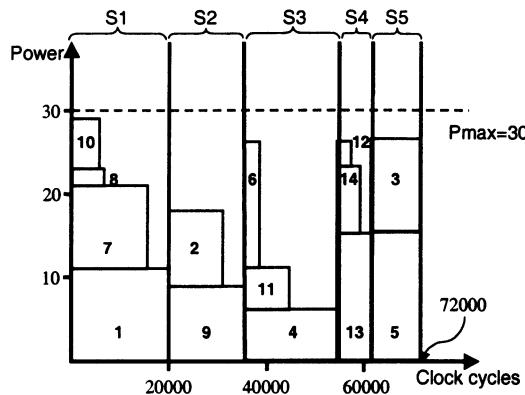


Figure 1. Power and timing report for unequal length session scheduling

## 2.2 "Sessionless" scheduling

In this scheme, no test sessions are considered. One scheduling solution for the given SoC example is reported in Figure 2. For instance, cores 1, 7 and 9 are simultaneously tested at the beginning of the test mode. The test of core 2 (resp. 8) starts right after the end of the test of core 7 (resp. 9) while test of core 1 is still running. The core-test starts are scheduled as soon as possible considering resource and power conflicts. The system test time is partitioned in several time slots delimited by dashed lines in Figure 2. The maximum allowable system power dissipation is respected for every time slot during the test mode.

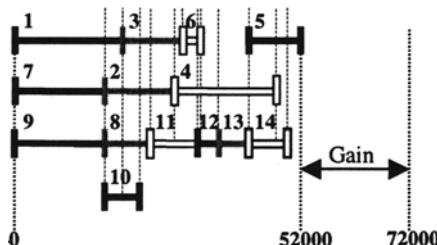


Figure 2. Timing report for the "sessionless" scheduling

Test length is now of 52000 clock cycles, representing an improvement of 20000 clock cycles over the solution based on unequal test session lengths. However, the sessionless scheme is more complex from the synchronization point of view since each core test depends of other(s) core(s) test end signal.

### 3. SCHEDULING

For all three BIST control schemes, the scheduling problem consists to minimize the total test time while obeying the power constraint limit and the resources sharing possibilities that is, organizing the tests in such a way that:

- 1- At any time, the sum of the individual power consumption does not exceed the limit  $P_{max}$  (constraint  $C1$ )

- 2- the tests of two cores sharing test resources can not overlap in time (constraint  $C2$ ). Constraints  $C2$  can be modelled as an incompatibility graph in which nodes represent tests, and an edge links two nodes if they share a test resource.

Let's denote  $i = \{1, 2, \dots, n\}$  the cores to test,  $D_i$  their test durations,  $P_i$  their power consumption,  $T_i$  their test starting date and  $P_{max}$  the maximum system power limit.

#### 3.1 Equal length sessions

Under a equal length session scheme, the scheduling problem can be stated as follows:

Problem P1: Minimize the total test time  $T_{total}$  with  $T_{total} = \# \text{ sessions} * (D_{max})$  under constraints  $C1$  and  $C2$ .  $D_{max}$  is the duration of the longest core test over all core tests.

As previously mentioned, this problem consists simply to minimize the number of sessions and assign the core tests to the sessions [6] while respecting power and test resource sharing constraints.

Theorem 1: P1 is NP-complete.

Proof: Let's restrict P1 to the case in which  $C2$  is null (no resource conflict). The problem sums up to the well-known minimum bin-packing problem (with "bins" being the test sessions). The bin packing problem is NP-complete ([7]). q.e.d.

Remark: by restricting P1 to the case in which  $P_{max} = \sum P_i$  with  $P_i$  the individual test power consumptions, i.e.  $C1$  is null, P1 sums up to the minimal graph coloring problem which is well-known to be NP-complete. Thus, P1 is made of two interleaved NP-complete problems (bin-packing and minimal graph-coloring).

Heuristics such as the one proposed in [6] can be used for solving P1.

#### 3.2 Unequal length sessions

Whereas in the previous scheme, the assignment of core tests to sessions does not impact on the total test time, this point is crucial here since the

sessions lengths directly depends on it. This point adds an extra difficulty to the problem stated as follows:

Problem P2: Minimize the total test time  $T_{\text{total}}$  with  $T_{\text{total}} = \sum D(s)$ ,  $s \in \text{Sessions}$ , where  $D(s)$  is given by the longest core test length of the current session  $s$  i.e.  $D(s) = \text{MaxDi}$ ,  $i \in s$ , under constraints  $C1$  and  $C2$ .

Theorem 2: Problem P2 is NP-complete

Proof: Let's consider the case in which all  $Di$  are constant. P2 sums up to P1. Thus P2 is NP-complete.

Several scheduling heuristics proposed in the literature (e.g. [8], [9], [10]) can be used for unequal length test session scheduling.

### 3.3 No sessions

Now the test scheduling problem is stated as follows:

Problem P3: Minimize the total test time  $T_{\text{total}}$  based on the actual individual test lengths under constraints  $C1$  and  $C2$ .

Theorem 3: P3 is NP-complete.

Proof: Let's restrict P3 to the case in which all  $Di$  are constant. P3 sums up to P1. Thus P3 is NP-complete.

In this test scheme, the test scheduling is more complex than in the previous schemes, in which it sums up to a mapping problem of core tests to sessions. Recently, an ILP formulation has been proposed [11] for problem P3 but to the best of our knowledge, no scheduling heuristic have been proposed in the literature for solving it.

It is clear from the example in section 2, that in general a "good" solution for P3 cannot be directly derived from a good solution for P2. In the next section, we propose simple and efficient heuristics for such a control scheme.

## 4. SCHEDULING HEURISTIC WITH NO SESSIONS

Let's recall that P3 contains P1 that in turn contains the minimal graph-coloring problem.

It is known from the literature that the graph-coloring problem cannot to be approximated in bounded limits when the graph has no special structure [7]. Thus, we developed the following heuristic.

## 4.1 Algorithm

```

L1 = list of cores sorted by decreasing Di values
L2 = Ø
Tmax=0
While L1 ≠ Ø
    Tmax =Di+Place (first core in L1)
    For all others cores i in L1
        Ti=Place(i)
        if Ti+Di > Tmax
            remove i from the placed cores
        L2 = L2 ∪ {i}
    L1 = L2

```

*Figure 3.* Test scheduling heuristic

in which function Place (i) positions i as soon as possible i.e. find the earliest date  $T_i$  to start i test taking into account that for any other already placed core j such that:  $(T_j \leq T_i \text{ and } T_j + D_j > T_i)$  or  $(T_i \leq T_j \text{ and } T_i + D_i > T_j)$  – i and j tests overlap:-

- $\sum P_j + P_i < P_{\max}$  ( $C_1$ )
- i does not conflict with j ( $C_2$ )

Property 2: the complexity is  $O(n^3)$ .

Proof: In the worst case, only one core is actually placed in each iteration of the while loop. At the  $i^{\text{th}}$  iteration, the for-loop iterates  $i$  times and executes the Place function which is  $O(i)$ . Thus, it comes that the complexity is  $O(\sum(i(n-i))) = O(n^3)$ .

## 5. RESULTS

In order to show the effectiveness of the no session test scheme, we compare our approach with the one presented in [10]. The characteristics of the SoC example used for this comparison are summarised in Table 2. In [10], the authors propose an improvement of the unequal length session approach by allowing several cores to be sequentially tested within a session.

On this example, the system power limit is set to 12 units. It can be seen on Figure 4 that the system test time reduces from 31k clock cycles with the approach presented in [10] to 23k clock cycle with the approach proposed here. In fact, the session-based test scheme results in partitioning the system test time into 4 sessions (S1 to S4), leaving unused time slots that could be used to start test of cores 4 and 3.

| Core | Pi | Di    | Share test resource with |
|------|----|-------|--------------------------|
| 1    | 6  | 16000 | 6 7 8                    |
| 2    | 5  | 10000 | 4 5 6 7                  |
| 3    | 4  | 9000  | 6 7 9                    |
| 4    | 2  | 7000  | 5                        |
| 5    | 8  | 4000  | 2 4 7                    |
| 6    | 2  | 3000  | 1 2 3                    |
| 7    | 2  | 2000  | 1 2 3 5                  |
| 8    | 2  | 1000  | 1                        |
| 9    | 1  | 3000  | 3                        |

Table 2. A SoC example for comparison with a session-based test scheduling approach

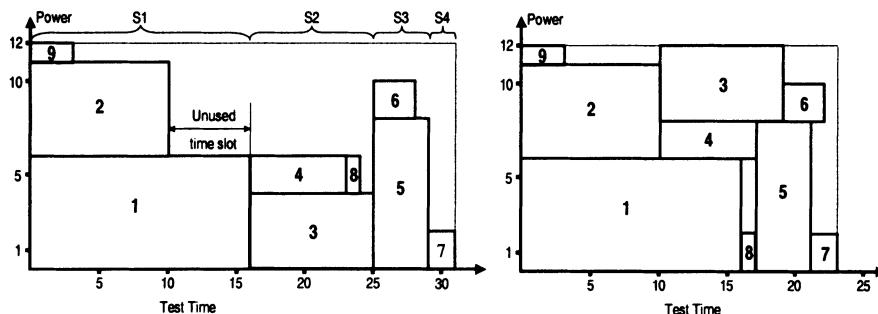


Figure 4. Scheduling results with 1/ unequal length test sessions [10] and 2/ no session

We also compare our technique with another method proposed in the literature targeting the no session based approach [11]. The authors solve the scheduling problem using a MILP formulation. They assume two test runs for every core, a BIST procedure in order to test most of the faults with a relatively low number of patterns and, when necessary, an external test procedure for detecting remaining hard-to-detect faults. Inequalities are used for expressing conflicts and constraints between core tests. Note that the main drawback of the ILP formulation is the exponential growth in term of inequalities. The authors report a total test time of 7985 cycles for the SoC example whose characteristics are given in Table 3.

We adapted our algorithm to support hybrid tests (BIST + external) so that the comparison with [11] can be possible. Since external test and BIST cannot be applied simultaneously on a core, the adaptation simply consists

to add incompatibility constraints between these two tests on every core. We applied this new version of our heuristic to the same SoC example with the same constraints on resource sharing limitations and power dissipation limit ( $P_{max}=950$  mW). Our solution is presented in Figure 5. Each bloc ii (resp. i) represents an external test (resp. BIST), for the core number i.

| Core/number | BIST test time<br>(cycles) | External test<br>time (cycles) | Power dissipated<br>in BIST mode (mW) |
|-------------|----------------------------|--------------------------------|---------------------------------------|
| c880/1      | <b>256</b>                 | 134                            | 54                                    |
| c2670/2     | <b>2048</b>                | 2543                           | 159                                   |
| c7552/3     | <b>2048</b>                | 1357                           | 453                                   |
| s953/4      | <b>256</b>                 | 454                            | 57                                    |
| s5378/5     | <b>256</b>                 | 1903                           | 324                                   |
| s1196/6     | <b>256</b>                 | 242                            | 72                                    |
| s13207/7    | <b>2048</b>                | -                              | 792                                   |
| s1238/8     | <b>1024</b>                | 176                            | 75                                    |

Table 3: SoC example for a comparison with another NS-based approach

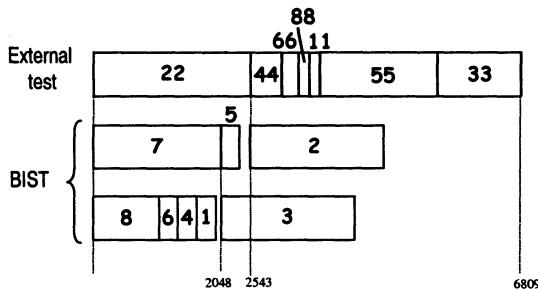


Figure 5. : Test scheduling for the SoC example described in Table 3

We obtained a total test time of 6809 cycles versus 7985. This example shows that in spite of its simplicity, our algorithm leads to good results.

Moreover, the CPU time is in the order of the second for about 100 cores in a SoC. Thus, we can foresee using our tool to explore different solutions. Figure 6 presents several scheduling solutions for the SoC example described in Table 1. As expected, the system test time decreases when the power dissipation constraint increases. The curve shows that different power dissipation limits lead to the same system test time, it also indicates the resulting increase in test time when the power constraint becomes stronger.

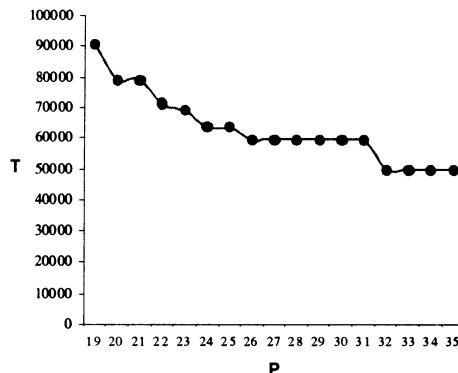


Figure 6. Exploration of the test scheduling solution space

## 6. EXTENSION

As mentioned above, the first extension for our tool was to adapt our algorithm in order to support hybrid tests during which several test runs are applied to every core (see section 3.2). For instance, a core is first tested with pseudo-random patterns delivered by internal test resources, then tested with deterministic patterns delivered by the ATE.

As mentioned in [11] for an "abort at first fail" industrial test strategy, it is more convenient to run test processes that are more likely to detect defect first. This point introduces the notion of precedence in a test suite. In practice precedence constraints appear, if cores have to be tested first for many reasons like: they are bigger and have more chances to be faulty, they are purchased from external vendors and so tested before cores designed in-house. The last reason comes from the fact that core test sets can contain a BIST part and a deterministic part. The cores first tested should be the ones with the best rate (number of detected faults)/(test time), just to test the highest number of faults in the shortest possible time decreasing in this way the test time of a faulty core. Since the external test only targets few hard-to-detect faults in comparison with BIST that targets all other ones, BIST should be applied first.

Consequently the proposed algorithm has been modified in order to deals with precedence constraints. These user-given constraints concern any test and any core in a test suite. The function Place() in our algorithm (see section 3.1) takes into account these new constraints. The algorithm postpones a test if the ones that must be applied first have not been yet scheduled.

## 7. CONCLUSION

We have presented an efficient scheme for organizing the test at system-level and a corresponding power constrained test-scheduling algorithm. This approach outperforms classical ones, which are based on test sessions.

In spite of its simplicity, the proposed algorithm also outperforms other "no session" solution. Reasonable CPU times allow exploring a wide range of solutions. Proposed extensions allow to run several tests on the same core and to partially order tests on a test suite.

Finally, present work assumes that the test architecture is fixed before to solve the test scheduling problem. We expect a better test area and test time tradeoff by assigning dynamically the test resources when needed.

## 8. REFERENCES

- [1] H.J. Wunderlich, "BIST for systems-on-a-chip", Integration, the VLSI journal, 26, pp 55-78, 1998.
- [2] I. Ghosh, N. K. Jha, S. Dey, "A low overhead design for testability and test generation technique for core-based systems", Proc. Int. Test Conf., pp 50-59, 1997.
- [3] E.J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, C. Wouters, "A structured and scalable mechanism for test access to embedded reusable cores", Proc. Int. Test Conf., pp 130-143, 1998.
- [4] P Varma, S Bhatia, "A structured test re-use methodology core-based system chips", Proc. Int. test Conf., pp 294-302, 1998.
- [5] K. Chakrabarty, "Design of system-on-a-chip test access architectures using integer linear programming", Proc. VLSI Test Symp., pp 127-134, 2000.
- [6] R. Chou, K. Saluja, V. Agrawal, "Scheduling Tests for VLSI Systems under Power Constraints", IEEE Trans. on VLSI Systems, Vol. 5, No. 2, pp 175-185, June 1997.
- [7] M.R. Garey, D. Jonhson: "Computers and Intractability: guide to the theory of NP-completeness", W.H. Freeman and Company, San Francisco.
- [8] C.P. Ravikumar, A. Verma, G. Chandra, "A Polynomial-Time Algorithm for Power Constrained Testing of Core Based Systems", ATS 99, pp 107-112.
- [9] V. Muresan, X. Wang, M. Vladutiu, "List scheduling and Tree Growing Technique in Power-Constrained Block-Test Scheduling", ETW 99, pp 27-32.
- [10] V. Muresan, X. Wang, M. Vladutiu, "A comparison of classical Scheduling Approaches in Power-Constrained Block-Test Scheduling", ITC 2000, pp 882-891.
- [11] V. Yengar, K. Chakrabarty, "Precedence-Based, Preemptive and Power-Constrained Test Scheduling for System-on-a-Chip", Proc. VTS '01, pp:368-374.

# **Random Adjacent Sequences**

## *An Efficient Solution for Logic BIST*

**René David**

*Laboratoire d'Automatique de Grenoble  
BP 46, 38402 St-Martin-d'Hères, France  
Rene.David@inpg.fr*

**Patrick Girard, Christian Landrault, Serge Pravossoudovitch,  
Arnaud Virazel\***

*Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier  
161, rue Ada, 34392 Montpellier Cedex 5, France  
Patrick.Girard@lirmm.fr  
Christian.Landrault@lirmm.fr  
Serge.Pravossoudovitch@lirmm.fr  
virazel@lirmm.fr*

\* *The author works now in the Computer Architecture Lab – University of Stuttgart  
Breitwiesenstrasse 20-22, 70565 Stuttgart, Germany  
Arnaud.Virazel@informatik.uni-stuttgart.de*

**Abstract:** High defect coverage requires good coverage of different fault types. In this paper, we present a comprehensive test vector generation technique for BIST, called Random Single Input Change (RSIC) generation, that can be used to generate tests for many arbitrary misbehaviors that can occur in digital systems, thus providing a single on-chip test generation solution. By proving the effectiveness of universal test sequences produced by such a generation technique in detecting stuck-at, path delay and bridging faults, we demonstrate that using RSIC generation is one of the best and most practical way to reach a high level of defect coverage during BIST of digital circuits.

**Key words:** BIST, Stuck-at Fault, Bridging Fault, Delay Fault, Universal Sequence.

## 1. INTRODUCTION

In the near future, the recent advances in deep-submicron IC process technology and core-based IC design technology will lead to a widespread use of logic BIST in industry. This is confirmed by the ITRS (International Technology Roadmap for Semiconductors) statement that by 2014 it may cost more to test a transistor than to manufacture it unless techniques like logic BIST are employed [1]. Logic BIST, which test logic circuits through the use of built-in pattern sources and response evaluators, offers a number of advantages compared with external testing, which is becoming more and more difficult and costly [2].

A first problem that occurs with external testers is that they are several times slower than the circuits they have to test. Purchasing high speed testers that meet the performances of new designs requires a huge investment. Moreover, even with those high speed testers, it is not always possible to have a timing accuracy comparable to the IC internal speed [2]. In this context, BIST represents an attractive test solution since it allows at-speed testing, thus solving timing accuracy and test time related problems encountered with traditional external testers. In addition, BIST drastically reduces the amount of test data exchanged with the tester, thus reducing the need for complex external testing equipment. BIST can hence be run on a very low cost tester. Finally, BIST may relieve the problem of tester capacity [3] and the problem of low accessibility of internal nodes of the design, which increases the test complexity [4].

The growing need for performance-related testing (called delay testing) also demands BIST solutions. Deep-submicron technologies introduce new performance-related defect types, and the increasing clock frequencies in high-speed designs impose aggressive timing margins. While the internal clock frequencies have risen by 30% per year, the accuracy of external test equipment has improved at a rate of only 12% per year [1]. Hence, it is becoming increasingly difficult to do performance-related testing using external test equipment. BIST may solve this problem since it allows to do accurate delay testing and precision measurement on-chip.

Test pattern generation for BIST can be broadly classified into deterministic, mixed-mode, pseudo-random (flat or weighted) and exhaustive generation. Deterministic BIST consists in storing pre-computed test patterns on-chip by using a special-purpose hardware [5]. Deterministic BIST has been shown to be efficient in terms of fault detection but the price for obtaining complete fault coverage usually is a relatively large silicon area overhead. Moreover, deterministic BIST is always related to a specific fault model. Mixed-mode BIST consists in using a limited number of pseudo-random patterns to eliminate easy to detect faults and deterministic patterns

to cover the remaining random patterns resistant faults [6]. The deterministic patterns are either stored on-chip in a compressed format and expanded during BIST [7] or directly embedded into an LFSR sequence by “bit-fixing” or “bit-flipping” techniques [8]. Mixed-mode BIST is less hardware consuming compared with deterministic BIST, but is also less efficient in terms of test quality (fault coverage vs test length) and is often extremely tailored to a specific circuit. Moreover, mixed-mode BIST is related to a specific fault model. Pseudo-random BIST consists in applying test patterns that exhibit randomness but which are generated using special-purpose hardware (LFSR or Cellular Automata) and are thus repeatable [9,10,11]. Pseudo-random BIST is the least hardware consuming solution but requires test lengths which may be long in some cases. Pseudo-random BIST is not related to a specific fault model. Exhaustive (or pseudo-exhaustive) BIST consists in applying all (or nearly all) the logic combinations at the circuit inputs [12]. Pseudo-exhaustive BIST is not related to a specific fault model, but cannot be used in practice due to high testing time and circuit segmentation requirement.

The goal in this paper is to propose a new test generation technique for BIST that can be used to generate tests for many arbitrary misbehaviors that can occur in digital systems, thus providing a single on-chip test generation solution. The reason for using such kind of “universal” test sequences can be explained as follows. The classical single-stuck fault (SSF) model continues to be the most commonly used fault model in digital systems testing. However, defects in new nanometer CMOS technologies do not always behave like stuck-at faults [13]. Therefore, test generation based on the SSF model alone is no longer sufficient for obtaining high defect coverage [14,15]. On the other side, the use of multiple test generation techniques that each targets a specific fault type (such as transition, bridging or stuck-open fault type) would be costly and unpractical from a BIST point of view. For this reason, we propose an alternative solution consisting in the use of a single on-chip test pattern generator providing universal test sequences that target both conventional (stuck-at) and non-conventional (delay, bridging, stuck-open) fault types.

The question now is: what kind of BIST test pattern generation do we need to use for generating such “universal” test sequences? According to the above presentation of BIST test generation approaches, the only practical solution is to use pseudo-random BIST as this approach is not related to a specific fault model, i.e. test patterns are determined irrespective of the targeted fault type. However, the only problem that remains with pseudo-random BIST is the test length, which is acceptably long to test for SSFs, but which is prohibitively long to test for delay faults [16]. A solution to solve this problem is to use Random Single Input Change (RSIC) test sequences,

that have been shown to be efficient to test path delay faults with a reasonable test length [17]. RSIC test sequences are single input change (also called adjacency) test sequences that have to be generated in such a way that they have practically the properties of pure random sequences. This can be performed through the use of an LFSR with improved random properties for example. Moreover, the selection of the changing bit (between two consecutive patterns) has also to be performed randomly.

The remainder of this paper is organized as follows. Section 2 gives preliminary definitions on the considered fault models and the test sequences used. Section 3 highlights the effectiveness of RSIC generation, first for delay fault detection and next for SSF and bridging fault detection. Concluding remarks are given in Section 4.

## **2. BASICS AND BACKGROUND**

### **2.1 Basics on fault models**

Fault models describe the logical behavior of a faulty system [18]. In this work, several fault models have been considered to exhibit the effectiveness of test sequences produced from a RSIC generation. The classical and most widely-used SSF model is obviously considered. Two other fault models are also used: the bridging fault (BF) model and the path delay fault (PDF) model. Basic definitions and notations on these fault models are now given.

A bridging fault can be viewed as an unintentional short between two lines. This short can be a non-resistive short such that the two lines are always brought into equilibrium at the same potential or a resistive short such that the potential between the shorted lines is different. Therefore, after the popular wired-AND and wired-OR models used to model the effects of bridging faults in bipolar logic [19], several models for the resistive and non-resistive fault types of bridging faults were proposed. However, these models suffer from a number of limitations. The primary drawback of the non-resistive fault model is that it is not guaranteed to detect resistive bridging faults. While resistive BF models are more realistic, they fall short of modeling all bridging type anomalies since the number of possible BF anomalies is intractable for most circuits [20].

Although the coverage of a bridging fault by both wire-AND and wire-OR behavior does not always guarantee detection, these models have the advantage to be easy to consider during BF simulation. Moreover, it has been proven recently that a high gate level SSF coverage implies high BF coverage [21]. Consequently, the rest of this paper is based on the use of such models to represent bridging defects that occur in digital circuits. The

BF site extraction problem has not been dealt with since the lists of realistic bridging faults considered during the evaluation of RSIC test sequences were provided by the Lisboa Technical University (INESC/IST).

Some defects in a manufactured circuit do not affect the logic function of the circuit, rather they change the delays of some gates, thus altering the speed at which the circuit can operate. Such defects are referred to as delay defects and delay fault testing is the term used for the methodology of detecting such defects by checking whether a manufactured circuit meets its timing characteristics. The fault model most widely studied in this context is the path delay fault model [22]. The main advantage of using the path delay fault model is that it models the real distributed delay defects very well compared with other existing delay fault models, namely the gate delay fault model and the transition delay fault model. However, an important feature of the path delay fault model is that the single fault assumption is not realistic since a single defect usually affects a large number of paths. For this reason, a robust test is preferred to detect a path delay fault. A test for a path is robust if it can detect a delay fault on the path irrespective of other delays and delay faults in the circuit, otherwise it is non-robust [22].

Detection of path delay faults requires two-pattern tests. An initialization vector is applied and the circuit is allowed to stabilize. Then, the test vector is applied and the circuit outputs are sampled at clock speed. The response is then compared to that of the fault-free circuit to determine the presence or the absence of a delay fault. Hence, correct operation of a circuit at the intended speed can only be guaranteed if there is no path delay that exceeds the value determined by the clock period [23].

## 2.2 Theoretical basis of RSIC generation

In general, two-pattern tests may differ in multiple bit positions. In this case, they are called multiple input change (MIC) pattern pairs. Test pairs that differ only in one bit position are called single input change (SIC) pattern pairs. Let us now define what a RSIC sequence should be from a theoretical point of view (we assume equal likelihood of all vectors). Let

$$S = V(1)V(2)\dots V(l)\dots V(L), \quad (1)$$

be a test sequence composed of  $L$  successive  $n$ -bit vectors  $V(l)$ . Each vector takes a value from the set

$$V = \{V_0, V_1, \dots, V_{2^n - 1}\}, \quad (2)$$

where  $V_j$  corresponds to the  $n$ -bit vector  $(x_1, x_2, \dots, x_n)$  associated with the decimal value  $j$ . For example, for  $n = 5$ ,  $V_9 = 01001$ , i.e.,  $x_1 = x_3 = x_4 = 0$  and  $x_2 = x_5 = 1$ . In a Random MIC (called RMIC) sequence, the probability  $\Pr[V(l) = V_j] = 1 / 2^n$  for any  $l$  and any  $j$ , and the probability  $\Pr[V(l) = V_j]$  is

independent of the values  $V(i)$ , where  $i = 1, 2, \dots, l-1$ . In a RSIC sequence, this probability is:

$$\Pr[V(l) = V_j \mid V(l-1) = V_k] = \frac{1}{n}, \text{ iff } |j - k| = 2^a, \quad (3)$$

where  $a$  is a non-negative integer. In other words, for any  $l > 0$ ,  $V(l)$  differs from  $V(l-1)$  by exactly one bit randomly drawn, and this bit must be independent of the bits previously drawn.

Figure 1 represents the basic principle of an RSIC generation. This principle is taken from [24]. Basically, a  $k$ -bit random (in fact pseudo-random) source is used; this source may be a random number obtained from a maximal length LFSR. The value of the vector  $Q_1Q_2\dots Q_k$  changes at each clock cycle of the test session. At each time  $l$ , a subset of  $m$  bits is used ( $m \leq k$ ). The vector  $R(l) = R_1(l)R_2(l)\dots R_m(l)$  is transformed into a  $n$ -bit vector  $V(l) = x_1(l)x_2(l)\dots x_n(l)$  which is applied to the circuit under test. In order to agree with the definition of a RSIC sequence given above, the following points have to be addressed:

1.  $R(l)$  is independent of  $R(l-1)$ .
2. The transcoding between  $R(l)$  and  $V(l)$  satisfies Equ.(3), at least approximately.
3. The period of the sequence  $S = V(1)V(2) \dots V(l) \dots V(L)$  is at least equal to the test length  $L$ .

Solutions to points 1 and 2 are given in [24]. The third point is addressed in [25] where it is demonstrated that the period of the RSIC sequence generated from an appropriate structure based on the above principle is  $2 \times (2^k - 1)$ . For this point to be verified, the length  $k$  of the pseudo-random source (i.e., an LFSR) must be chosen such that  $2 \times (2^k - 1) > L$ , i.e.,

$$k > \log_2 \left( \frac{L}{2} + 1 \right) \approx \log_2 \frac{L}{2} \quad (4)$$

The advantage of using RSIC generation is that the probability of delay test invalidation due to hazards or multiple delay faults is greatly reduced as a single transition is applied at the primary inputs of the CUT at each clock cycle of the test session. Moreover, RSIC test pairs are sufficient to detect all robustly detectable path delay faults [22].

The structures producing MIC or SIC test sequences with random properties similar to those discussed above were taken from [24]. More specifically, RMIC test sequences have been obtained from a modified LFSR in which more than one shifting are used between two consecutive vectors. Similarly, RSIC test sequences have been obtained from the hardware structure proposed in [24]. More details and discussion on the RSIC generator and the corresponding properties can be found in [24] and in [25]. In particular, it is demonstrated in [25] that random testing from the

above hardware structure provides the same results than those obtained from a software generation, which is easy to perform thanks to the random function Rand() of the C language. This confirms the fact that pure random test sequences can be materially generated.

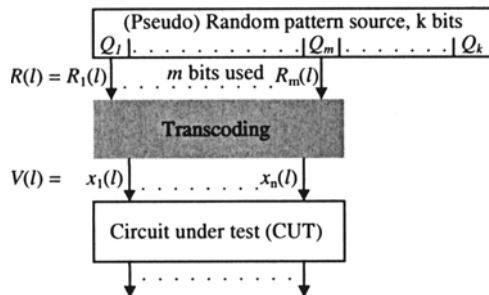


Figure 1. Principle of generation of a RSIC test sequence

### 3. EFFECTIVENESS OF RSIC GENERATION

The aim of this section is to demonstrate the effectiveness of RSIC generation, first for delay fault detection and next for SSF and bridging fault detection. For this purpose, we proceed in two steps. First, we remind the results presented in [26] on the comparison between random and pseudo-random generation of SIC or MIC sequences, and in [17] on the comparison between RSIC and RMIC generation. These results were obtained based on the path delay fault model. Next, we present new results on the detection capability of RSIC generation for SSF and bridging fault coverage. A synthesis is drawn at the end of this section.

#### 3.1 Comparison between random and pseudo-random generation

All the generated sequences are pseudo-random by construction (*i.e.* repeatable). In the sequel, comparison is made between two kinds of generation. The first one is the usual generation : at time  $l$ , the vector made of  $n$  bits in the LFSR is applied. After a shifting in the LFSR, a new  $n$ -bit vector is applied at time  $l+1$ . For short, this generation will be called usual pseudo-random (in case of SIC sequence, there is an additional transformation as described in Section 3.3). The second kind of generation (software or hardware, as it will be specified) is such that the sequence has properties very close to a pure random sequence (RSIC or RMIC sequence). For short, such a sequence will be called random in the sequel.

In order to validate our statement that SIC test sequences must be random (rather than usual pseudo-random) to be efficient in detecting conventional and non-conventional faults, we first compare random and usual pseudo-random generation in [26]. The results obtained demonstrate that using random test pairs (produced from a software generation) to test path delay faults in a given circuit produces higher delay fault coverage than that obtained with usual pseudo-random test pairs. The comparison has been further extended, showing that the same conclusion can be derived when SSF or bridging fault coverage is targeted rather than delay fault coverage [26].

A sample of the results given in [26] is reported in Table 1 and Table 2. Table 1 lists the robust and non-robust delay fault coverage achieved by RSIC test sequences and usual pseudo-random SIC test sequences respectively. The results are expressed in terms of fault efficiency. Similarly, Table 2 lists the robust and non-robust delay fault efficiency achieved by RMIC test sequences and usual pseudo-random MIC test sequences respectively.

| Circuit | L      | Robust Eff |         | Non-Robust Eff |         |
|---------|--------|------------|---------|----------------|---------|
|         |        | RSIC       | SIC     | RSIC           | SIC     |
| s1238   | 499200 | 89.11 %    | 74.94 % | 97.16 %        | 93.55 % |
| s1494   | 351800 | 94.51 %    | 81.47 % | 100 %          | 99.54 % |
| s3330   | 158620 | 62.28 %    | 14.14 % | 77.63 %        | 17.01 % |
| s5378   | 297300 | 64.77 %    | 46.18 % | 67.23 %        | 56.95 % |

Table 1. Comparison between RSIC and Pseudo-random SIC - delay fault efficiency

| Circuit | L      | Robust Eff |         | Non-Robust Eff |         |
|---------|--------|------------|---------|----------------|---------|
|         |        | RMIC       | MIC     | RMIC           | MIC     |
| s1238   | 499200 | 29.35 %    | 18.42 % | 99.84 %        | 70.32 % |
| s1494   | 351800 | 30.83 %    | 23.24 % | 99.78 %        | 72.62 % |
| s3330   | 158620 | 23.47 %    | 23.18 % | 90.71 %        | 90.62 % |
| s5378   | 297300 | 24.51 %    | 17.87 % | 97.09 %        | 65.09 % |

Table 2. Comparison between RMIC and Pseudo-random MIC - delay fault efficiency

As can be seen on these results, the delay fault efficiency obtained with random (RSIC or RMIC) test sequences is always better than that obtained with usual pseudo-random (SIC or MIC) test sequences. Hence, these results clearly illustrate the fact that random testing is more efficient than usual pseudo-random testing for delay fault detection.

### 3.2 Comparison between RSIC and RMIC generation

According to the above results, it can be stated that random testing is more efficient than usual pseudo-random testing. So, the next step has been to concentrate on this kind of testing and compare RSIC and RMIC generation for delay fault detection.

A first comparison between SIC and MIC generation for delay fault detection has already been proposed in [9], where it is shown that SIC test sequences are more efficient than MIC test sequences when a high robust delay fault coverage is targeted. However, the study considers usual pseudo-random vectors produced from an LFSR and concentrates the analysis on the coverage of faults having at least one robust test. For these reasons, the efficiency of RSIC test sequences for delay fault testing was further analyzed in [17]. The performance measurement took into account both robust tests and non-robust tests, and a random generation was assumed. The main conclusion drawn from this study is that, even with a lower non-robust delay fault coverage, a RSIC test sequence may often give rise to a better test quality than that obtained with RMIC delay test sequences.

To illustrate our statement, a sample of the results given in [17] is reported in Table 3. As can be seen, these results only concern the robust fault efficiency. For results concerning non-robust fault efficiency, the reader can refer to the discussion developed in [17].

| Circuit | L      | RSIC    | RMIC    |
|---------|--------|---------|---------|
| s1238   | 499200 | 89.11 % | 29.35 % |
| s1494   | 351800 | 94.51 % | 30.83 % |
| s3330   | 158620 | 62.28 % | 23.47 % |
| s5378   | 297300 | 64.77 % | 24.51 % |

Table 3. Comparison between robust fault efficiency of RSIC and RMIC testing

### 3.3 New results on the effectiveness of RSIC generation

In this subsection, we present new results on the effectiveness of RSIC generation for SSF and bridging fault coverage. We proceed in the same way than that used in our previous evaluations, *i.e.* we compare the efficiency of RSIC test sequences to that of RMIC test sequences (for SSF and bridging fault detection). Results in terms of SSF coverage are given in Table 4. Results in terms of bridging fault coverage are given in Table 5. Bridging faults simulated to evaluate the efficiency of RSIC test sequences are realistic bridging faults provided by the Lisboa Technical University (INESC/IST - Portugal). For each fault, we considered the following behaviors: WAND (Wire-AND), WOR (Wire-OR), WAND&WOR (the

fault is tested if the two behaviors are tested), WAND||WOR (the fault is tested if at least one of the two behaviors is tested). As the WAND&WOR and the WAND||WOR behaviors are more representative, only results on these models are reported in Table 5. Note that results in Table 4 are expressed in terms of fault efficiency (*i.e.*, coverage of detectable faults) while results in Table 5 are expressed in terms of fault coverage (we have no information on whether some bridging faults are redundant or not).

| Circuit | #faults | L     | RSIC    | RMIC    | L      | RSIC    | RMIC    |
|---------|---------|-------|---------|---------|--------|---------|---------|
| s420    | 424     | 10000 | 77.12 % | 83.25 % | 132200 | 96.10 % | 97.88 % |
| s510    | 538     | 10000 | 100 %   | 100 %   | 136600 | 100 %   | 100 %   |
| s1238   | 1 332   | 10000 | 90.99 % | 97.94 % | 499200 | 99.84 % | 100 %   |
| s1494   | 1 489   | 10000 | 99.66 % | 99.94 % | 351800 | 100 %   | 100 %   |

Table 4. Comparison between RSIC and RMIC generation for SSF efficiency

| Circuit | #faults | L      | RSIC     |           | RMIC     |           |
|---------|---------|--------|----------|-----------|----------|-----------|
|         |         |        | WAND&WOR | WAND  WOR | WAND&WOR | WAND  WOR |
| s420    | 521     | 132200 | 81.23 %  | 97.59 %   | 81.96 %  | 98.89 %   |
| s510    | 1 073   | 136600 | 100 %    | 100 %     | 100 %    | 100 %     |
| s1238   | 2 924   | 499200 | 99.9 %   | 100 %     | 99.9 %   | 100 %     |
| s1494   | 3 851   | 351800 | 99.95 %  | 100 %     | 99.95 %  | 100 %     |

Table 5. Comparison between RSIC and RMIC generation for bridging fault coverage

By looking at the results in Table 4, it can be seen that the effectiveness of RSIC generation compared to RMIC generation mainly depends on the test length L of the SSF sequence. For a test length L = 10 000, the fault efficiency of RMIC test sequences is slightly higher than that of RSIC test sequences. This is also true for test lengths L < 10 000 (on the considered circuits). But when the test length increases, the efficiency of RSIC test sequences becomes comparable to that of RMIC test sequences. For a test length equal to the test length used for delay fault testing (*cf.* sections 3.1 and 3.2), the fault efficiency is nearly the same for both type of test sequences. This is the most important result we can derive from these experiments: for a sufficiently long RSIC test sequence, the SSF efficiency is comparable to that of a RMIC test sequence. Actually, the same conclusion can be derived when bridging fault detection is considered. This is illustrated in the Table 5, which compares the bridging fault coverage of RSIC and RMIC test sequences for various ISCAS'89.

The fact that RSIC generation requires sufficiently long test sequences to reach the same level of test quality as that of RMIC generation (for SSF and Bridging fault coverage) confirms the basic assumption according to which

producing efficient universal test sequences from RSIC generation (for SSF and Bridging fault coverage) can only be done at the expense of a longer test time. But this is not a problem! According to the results reported previously on delay fault testing by RSIC test sequences, we know that quite long test lengths are required. So, the only way to cover both SSF, delay and bridging faults with universal RSIC test sequences is to use test lengths which are at least as long as test lengths for delay fault detection. In this context, RSIC generation is as efficient as RMIC generation for SSF and bridging fault coverage.

## 4. CONCLUSION AND FUTURE WORK

In this paper, we have presented a comprehensive test generation technique for BIST, called RSIC generation, that can be used to generate tests for many arbitrary misbehaviors that can occur in digital systems. We have demonstrated the effectiveness of universal test sequences produced by such a generation technique in detecting stuck-at, path delay and bridging faults. Although the effectiveness of the proposed generation technique has been clearly shown, it is possible to continue this study by looking at a tradeoff between the proposed RSIC generation and the conventional MIC generation. In other words, it would be interesting to analyze the efficiency of test sequences in which two or three bits rather than one bit change between two consecutive vectors. This solution would probably reduce the test length while achieving the same fault and defect coverage. The problem in this case would be to find an appropriate hardware generating structure. This problem will be consider in our future work.

## 5. ACKNOWLEDGEMENT

The authors would like to thank Pr. P. Teixeira and his team from the Technical University of Lisboa (INESC/IST) who provided us with the lists of realistic bridging faults experienced.

## 6. REFERENCES

- [1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors (ITRS)", 1999 Edition.
- [2] Y. Zorian, "Testing the Monster Chip", IEEE Spectrum, Vol. 36, N° 7, pp. 54-60, 1999.
- [3] G. Hetherington et al., "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", IEEE Int. Test Conf., pp. 358-367, 1999.

- [4] J. Rajski and J. Tyszer, "Arithmetic Built-In Self-Test for Embedded Systems", Prentice Hall PTR, 1998.
- [5] R. Dandapani, J. Patel and J. Abraham, "Design of Test Pattern Generators for Built-In Test", IEEE Int. Test Conf., pp. 315-319, 1984.
- [6] C. Fagot, P. Girard and C. Landrault, "On Using Machine Learning for Logic BIST", IEEE Int. Test Conf., pp. 338-346, 1997.
- [7] K. Chakrabarty, B.T. Murray and V. Iyengar, "Built-In Test Pattern Generation for High Performance Circuits Using Twisted-Ring Counters", VLSI Test Symp., pp. 22-27, 1999.
- [8] G. Kiefer, H. Vranken, E.J. Marinissen and H.J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits", IEEE Int. Test Conf., pp. 105-114, 2000.
- [9] W. Wang and S.K. Gupta, "Weighted Random Robust Path Delay Testing of Synthesized Multilevel Circuits", IEEE VLSI Test Symp., pp. 291-297, 1994.
- [10] P. Girard, C. Landrault, V. Moreda and S. Pravossoudovitch, "An Optimized BIST Test Pattern Generator for Delay Testing", IEEE VLSI Test Symposium, pp. 94-99, 1997.
- [11] C. Fagot, O. Gascuel, P. Girard and C. Landrault, "On Calculating Efficient LFSR Seeds for Built-In Self Test", IEEE European Test Workshop, pp. 7-14, 1999.
- [12] A. Vuksic and K. Fuchs, "A New BIST Approach for Delay Fault Testing", IEEE VLSI Test Symp., pp. 284-288, 1994.
- [13] R.C. Aitken, "Nanometer Technology Effects on Fault Models for IC Testing", IEEE Computer, Vol. 32, N° 11, pp. 46-51, 1999.
- [14] P. Nigh et al., "An Experimental Study Comparing the Relative Effectiveness of Functional, Scan, Iddq and Delay Fault Testing", VLSI Test Symp., pp. 459-464, 1997.
- [15] S.C. Ma, P. Franco and E.J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results", IEEE Int. Test Conf., pp. 663-672, 1995.
- [16] C.Chen and S.K. Gupta, "BIST Test Pattern Generators for Stuck-Open and Delay Testing", IEEE Euro. Design & Test Conf., pp. 289-296, 1994.
- [17] A. Virazel, R. David, P. Girard, C. Landrault, and S. Pravossoudovitch, "Delay Fault Testing: Effectiveness of Random SIC and Random MIC Test Sequences", IEEE European Test Workshop, pp. 9-14, 2000.
- [18] M. Abramovici, M. Breuer and A. Friedman, "Digital System Testing and Testable Design", IEEE Press, Piscataway, NJ, 1990.
- [19] K. Mei, "Bridging and Stuck-at Faults", IEEE Trans. on Computers, Vol. C-23, N° 7, pp. 720-727, 1974.
- [20] D. Lavo, B. Chess, T. Larabee and F. Ferguson, "Diagnosing Realistic Bridging Faults with Single Stuck-at Information", Trans. on CAD, Vol. C-17, N° 3, pp. 255-267, 1998.
- [21] S. Ma, I. Shaik and R. Fetherston, "A Comparison of Bridging Fault Simulation Methods", IEEE Int. Test Conf., pp. 587-595, 1999.
- [22] G.L. Smith, "Model for Delay Faults Based upon Paths", IEEE Int. Test Conf., pp. 342-349, 1985.
- [23] A. Krstic and K.T. Cheng, "Delay Fault Testing for VLSI Circuits", Kluwer Academic Publishers, Boston, 1998.
- [24] R. David, "Random Testing of Digital Circuits: Theory and Applications", Marcel Dekker, Inc., New York, 1998.
- [25] R. David, P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "On Hardware Generation of Random Single Input Change Test Sequences", IEEE European Test Workshop, PP.117-123, 2001.
- [26] P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "Comparison Between Random and Pseudo-Random Generation for BIST of Delay, Stuck-at and Bridging Faults", IEEE On-Line Testing Workshop, pp. 121-126, 2000.

# On-chip generator of a saw-tooth test stimulus for ADC BIST

F. Azaïs, S. Bernard , Y. Bertrand and M. Renovell

*LIRMM - University of Montpellier*

*161, rue Ada - 34392 Montpellier Cedex 5 France*

*azais, bernard, bertrand, renovell@lirmm.fr*

**Abstract:** In the context of analog BIST for A-to-D converters, this paper presents an implementation of an on-chip ramp generator. It is demonstrated that the proposed original adaptive scheme allows the internal generation of a highly saw-tooth signal with a very precise control of the signal amplitude. In addition, the implementation of the adaptive ramp generator exhibits a very low silicon area.

**Key words:** ADC testing, BIST, analog stimulus generation

## 1. INTRODUCTION

The use of Built-In Self-Test (BIST) for high volume production of mixed-signal ICs is desirable to reduce the cost per chip during production testing by the manufacturers. Within the past few years, analog and mixed-signal BIST have received the growing attention of industry and research community in order to alleviate increasing test difficulties. In particular, a number of papers deal with the problem of ADC and DAC testing.

For mixed-signal ICs with an ADC and a DAC, many of the proposed BIST techniques uses an all-digital approach [1-3] based on a reconfiguration in test mode such that the circuit appears all digital by connecting the analog output of the DAC to the input of the ADC, possibly via some analog block under test. Another all-digital BIST approach [4,5] exploits the digital signal processing (DSP) capabilities to determine characteristic parameters of the converters. Finally, an interesting approach has been proposed more recently, which is based on a polynomial-fitting algorithm to implement DAC and ADC BIST [6].

In case of mixed-signal ICs including solely an ADC, an original approach is detailed in [7], which relies on a reconfiguration in test mode that creates oscillation in the circuit. Measurements on these oscillations guarantee some tests. A more classical ADC BIST scheme implies the generation of an analog test stimulus and the digital processing of the ADC outputs. In this context, a specification-oriented approach is the only way to determine ADC parameters. A BIST module is proposed in [8] that partially overcomes this drawback since it permits to evaluate the converter linearity. Only the LSB is used for the determination of the linearity, the global functionality of the converter being tested with the comparison between the remaining bits and a counter clocked by the LSB. A more complete evaluation of the converter characteristics can be obtained by means of the histogram test method and BIST analyzers implementing this technique are presented in [9,10].

Concerning on-chip test stimulus generation, only a limited number of BIST solutions have been proposed. Original generators providing single or multi-tone analog signals are described in [11-13] to make frequency-domain test of converters. Time-domain testing is addressed in [14,15] with solutions for generating a precise analog ramp signal.

The objective of this paper is to develop an on-chip generator dedicated to linear histogram testing of ADCs. The histogram method is one of the most popular techniques for ADC testing in the industrial context. It actually relies on probabilistic methods to deduce the behavior of the circuit under test. The basic idea behind this approach is that the probability distribution behavior of the output signal is directly related to the input probability distribution through the circuit's transfer characteristic. The histogram method therefore involves the application of a given analog signal to the ADC input and the record of the number of times each code appears on the ADC outputs. Processing the measured data against a reference histogram then permits extracting the circuit's characteristics. To achieve statistically satisfactory results, this technique requires a lot of samples. However in practice, it is difficult to collect such a high number of samples applying a single ramp test stimulus. A multi-cycle approach is usually adopted which consists in sampling several cycles of the input signal within the test interval. In particular for linear histogram testing, the input test stimulus should be either a saw-tooth or a triangle signal. It is therefore the purpose of this paper to develop an on-chip saw-tooth generator. A drastic reduction of the testing costs would then be obtained since the ADC test can be performed on standard digital test equipment instead of a sophisticated and costly mixed-signal tester.

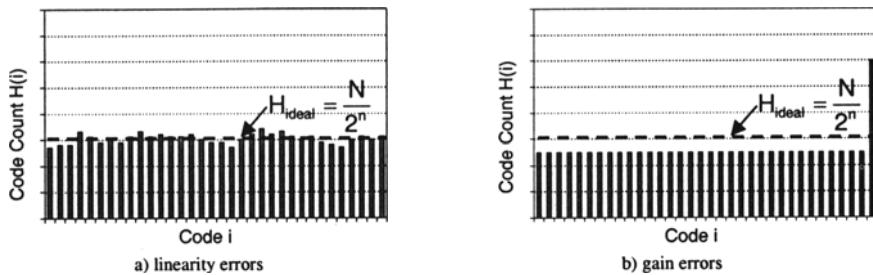
The paper is organized as follows. Section 2 gives the generator requirements needed to ensure accurate ADC characterization in terms of ramp linearity and amplitude control. The basic principles of the ramp generation are recalled in section 3 and the need of self-calibration to take into account parameter fluctuations is shown. Then our proposal for accurate self-calibration saw-tooth generator is described in section 4 and the calibration procedure is detailed. Also simulations are implemented that demonstrate the validity of our approach. Finally, the performance of the generator are analyzed and discussed in section 5.

## 2. GENERATOR REQUIREMENTS

One first obvious constraint concerns the silicon area since the generator has to be fabricated on the same IC as the circuit under test. The generator must occupy minimal silicon area so that the resulting overhead to the system is economically viable.

The second constraint concerns the quality of the test signal. Indeed, the histogram technique relies on the fact that the probability distribution behavior of the output signal is directly related to the input probability distribution through the circuit's transfer characteristic. The ADC characteristics are then extracted by comparison between the measured histogram and a reference histogram corresponding to the ideal input signal distribution. Obviously, the correct determination of these characteristics depends on the quality of the input signal whose probability distribution should be as close as possible to the ideal one. In particular, because Differential Non-Linearity (DNL) and Integral Non-Linearity (INL) are two key parameters of the converter measured with the histogram technique, the input signal should exhibit a quality exceeding that of the circuit under test in terms of linearity. Another important parameter of the input signal involved in the determination of the ADC characteristics is the amplitude of the test stimulus.

For illustration, figure 1 compares the histogram measured through a perfect n-bit converter in case of an ideal saw-tooth signal and a degraded saw-tooth signal containing linearity and gain errors. In the ideal case, we have a uniform distribution with an equal count for all codes, the value of this count being directly proportional to the number of samples ( $N$ ) collected to build the histogram.



*Figure 1.* Influence of the errors on the input signal

In case of linearity errors included in the input saw-tooth signal, we can observe some variations in the code counts (figure 1.a). Even if these variations are due to non-idealities of the input signal, they will be considered as linearity errors of the converter when comparing the measured histogram to the ideal reference one. Consequently, it is clear that the input signal must exhibit linearity errors much lower than the DNL and INL values to be measured for the converter under test.

Ideally, the input signal should exactly cover the full scale range of the converter in order to obtain the ideal reference histogram. Now if we assume a variation in the amplitude of the saw-tooth signal, for instance an increase, we obtain the histogram of figure 1.b. In this case, we observe a uniform distribution for all codes but with a lower code count, and a much higher count for the last code. However, this histogram is very different from the ideal reference histogram demonstrating that an increase of the input signal amplitude will strongly affect the measurement of the ADC characteristics.

These results clearly reveal that it is of crucial importance to be able to control the amplitude of the generated saw-tooth signal. This points out a specific requirement for the test generator in the context of a BIST solution. Because the saw-tooth signal is generated on-chip, unavoidable process variations have to be taken into account and the generator should be insensitive to these variations with regard to its amplitude.

As a summary, a practical saw-tooth generator for linear histogram testing of ADCs should respect some specific test constraints. It must be capable of generating a signal with quality exceeding that of the circuit-under-test in terms of linearity and should be insensitive to process variations with regard to the signal amplitude. These two features guaranty the test quality. In addition, it must occupy minimal silicon area so that the resulting overhead to the system is economically viable.

### 3. BASIC RAMP GENERATOR

#### 3.1 Ramp generation principle

A very simple approach to generate a ramp voltage consists in charging a capacitor by a constant current. The simplified schematic of such a ramp generator is given in figure 2.a. It comprises in addition to the constant current source and the charging capacitor, a switch to control the duration of the ramp and a switch to initialize the structure.

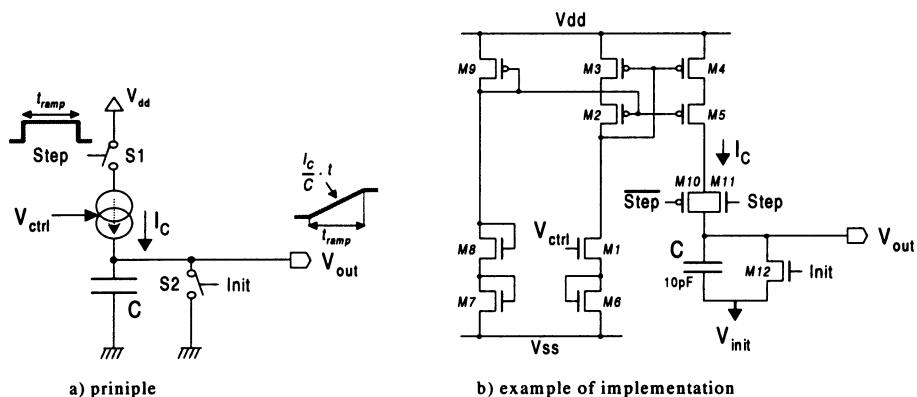


Figure 2. Voltage ramp generation

As an example of integrated circuit implementation, we consider the ramp generator described in figure 2.b. Transistor  $M_1$  is the current source, transistors  $M_2$  through  $M_5$  form a cascode current mirror, transistors  $M_6$  through  $M_9$  correspond to biasing transistors and  $C$  is the charging capacitor. Transistors  $M_{10}$ - $M_{11}$  operate as a switch to stop the charging of the capacitor when "Step" is at logic 0. Transistor  $M_{12}$  is used to initialize the structure at a given predetermined voltage  $V_{init}$ , typically ground for a ramp with positive values only and a negative voltage for a symmetrical ramp around 0.

As a case study, this structure has been implemented with a charging capacitor of 10pF and a charging current of  $0.3\mu\text{A}$ . It allows to generate a symmetrical ramp of 3V amplitude with a duration of 0.1ms using a negative initialization voltage of -1.5V. Iterating the ramp generation process therefore should permit to generate a saw-tooth signal with a frequency around 10kHz. This corresponds to the desired input signal frequency to be applied to the ADC for performing the histogram test.

### 3.2 Performances of the basic ramp generator

As stated in the previous section, the quality of the generated ramp is a crucial point since the ramp has to be used as test stimulus for the ADC. In particular, the generated ramp should exhibit high-linearity and precise amplitude to be used for histogram testing.

So first, we determine the linearity of the generated ramp voltage. This linearity actually depends on the ability of the current source to deliver a constant current and we can expect good performance due to the use of the cascode current mirror. The basic ramp generator has been simulated using Hspice and results are detailed in figure 3.a with the ramp signal along with the corresponding INL.

We obtain a ramp signal of 3V amplitude with a duration of 0.1ms and we measure a maximum INL of  $80\mu\text{V}$ . This corresponds to a slope distortion lower than 0.003%. The basic ramp generator therefore exhibits good linearity performance.

This linearity has to be compared to the resolution of the converter to be tested. We obtain a linearity higher than 15 bits for the ramp generator (considering a 3V input range). According to the requirement of at least 2bits better resolution for the test stimulus than for the converter-under-test, the basic ramp generator is therefore suitable to test a 12-bit ADC.

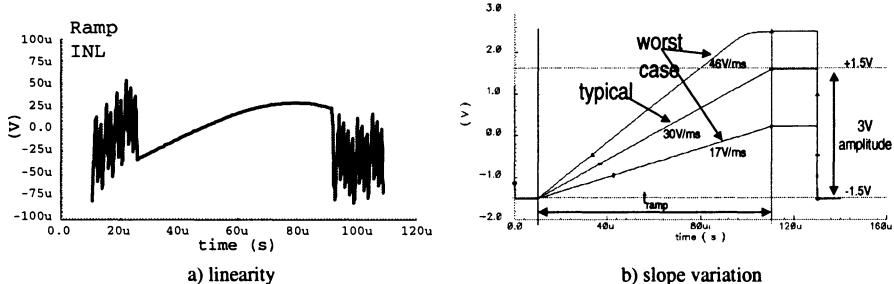


Figure 3. Linearity and slope variation of the basic ramp generator

Then we study the sensitivity of the basic ramp generator to process fluctuations. Since the duration of the ramp is controlled by the *Step* signal, the ramp amplitude actually depends on the slope of the generated ramp voltage. This slope is determined by the current source and capacitor values. Hence, the slope precision just depends on the ability of precisely controlling the value of the charging current and the value of the capacitor. However, it is extremely difficult to obtain a precise control of these values due to fluctuations in the manufacturing environment. To illustrate this point, various electrical simulations have been performed taking into account the

typical and worst case models provided in a CMOS 0.6 $\mu$ m technology. Simulation results are summarized in figure 3.b.

The extreme sensitivity of the structure to process variation clearly appears on these results. Indeed in the typical case, we obtain the desired ramp voltage with an amplitude of 3V for a ramp duration of 0.1ms, which corresponds to a ramp slope of 30V/ms. Looking at worst case simulation results, we observe that the final output voltage varies from 0.2V up to 2.5V, which corresponds to variations in the ramp slope as high as  $\pm 50\%$ . This imprecision mainly comes from the uncorrelated variations of the current source and capacitor values. These results clearly demonstrate that the basic ramp generator suffers from very poor performance in terms of slope precision, which prevents its direct use as test stimulus generator for histogram testing of ADCs.

#### 4. CALIBRATED SAW-TOOTH GENERATOR

In order to correct the inaccuracy of the basic ramp generator, the authors have developed an original adaptive scheme [15]. The basic principle consists in comparing the generated ramp voltage to a predetermined reference voltage and feeding back an adjustment signal so that the ramp voltage converges with the reference voltage within a given period. The general block-diagram of the corrective scheme comprises 3 blocks :

- the **ramp generator circuit** delivers a ramp voltage during a given period according to the *Step* signal,
- the **comparator** indicates whether the ramp voltage has reached the predetermined reference voltage during the period,
- the **ramp rate control circuit** provides the adjustment signal to the generator according to the comparison result.

Such a corrective scheme uses two reference values, i.e. a clock signal to define the ramp period (*Step*) and the reference voltage ( $V_{ref}$ ). The goal is to adjust the current source value of the ramp generator to a proper value so that the ramp voltage converges with the reference voltage within the given period. The proposed implementation of the ramp rate control circuit is based on the use of a very simple capacitive structure. The idea is to progressively accumulate on a capacitor the amount of charge required to drive the proper analog control voltage. Due to this progressive accumulation, the process of ramping and adjusting the control voltage has to be iterated a number of times to complete the system calibration. It is therefore an iterative adaptive scheme.

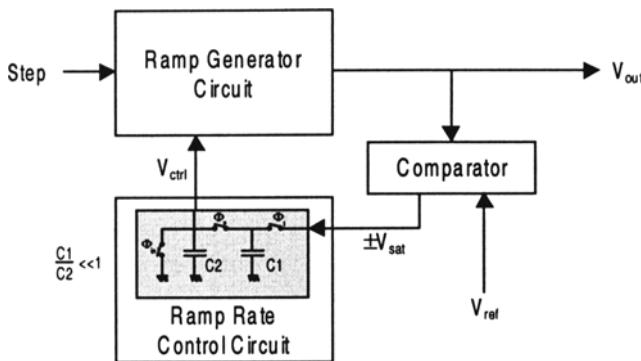


Figure 4. Implementation of the corrective scheme

Figure 4 shows the implementation of this adaptive scheme. The feedback circuitry simply consists of a capacitor bridge controlled by the comparator output. At each iteration, we have an adjustment of the control voltage according to the comparison result. This adjustment is actually performed by the capacitor bridge in a two-phase operation, i.e. first pre-charge of capacitor C1 to either the positive or negative saturation voltage according to the comparison result, and then charge distribution between the 2 capacitors. The resulting output voltage is expressed as:

$$V_{ctrl}(i) \equiv V_{ctrl}(i-1) \pm \Delta \quad (\text{if } \frac{C1}{C2} \ll 1 \text{ and } \Delta = \frac{C1}{C2} \cdot V_{sat})$$

As an illustration of the calibration procedure, we consider the case of a reduction of the ramp slope due to process variations. The calibration procedure starts with a number of cycles in which the control voltage is progressively augmented of  $\Delta$  at each iteration, until the ramp voltage reaches the reference voltage within the given period. Then, the control voltage oscillates around the proper value in the following iterations, indicating that the calibration is completed. Once the system is calibrated, we can then use the generator to deliver a saw-tooth signal by continuously iterating the process of ramping without any intermediate calibration step. In this case, the *Step* signal is maintained at logic 1 and only the *Init* signal is used to initialize ramp generation at each cycle.

The calibrated saw-tooth generator has been implemented in a CMOS 0.6 $\mu$ m technology. In order to achieve high precision on the ramp calibration, we have to implement a very low capacitor ratio. For instance, the adjustment of the control voltage within 1mV requires a capacitor ratio of  $4.10^{-4}$  (assuming a  $\pm 2.5$ V saturation voltage for the comparator). To ensure such a low capacitor ratio while maintaining a low area overhead, we suggest to use only the parasitic capacitors of the transmission gates operating as switches for C1 and choose 10pF for C2.

The complete structure has been simulated to validate the calibration process. We can see on figure 5.a that the ramp voltage calibrates itself to the reference voltage in less than 10 cycles. Then, we have oscillations of the control voltage around -13mV with an amplitude less than 1mV. Measurements of the ramp slope once calibrated reveal an average slope error of 0.4%, which clearly demonstrates the effectiveness of the calibration scheme to correct the sensitivity of the ramp generator to process fluctuations.

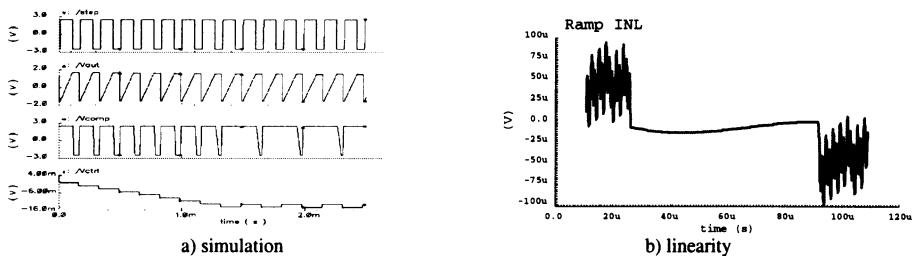


Figure 5. Simulation and linearity of the calibrated ramp generator

We also verify that the linearity of the ramp generator is not degraded by the additional circuitry. A detailed view of the calibrated ramp along with the corresponding INL is given in figure 5.b. We measure a maximum INL of  $91\mu\text{V}$ , which translates in a linearity of 15 bits for the ramp voltage. This result is comparable to the linearity of the basic ramp generator, demonstrating that we have a very low impact of the additional circuitry on the linearity performance.

## 5. PERFORMANCES AND DISCUSSION

The previous section has detailed the calibration procedure of the saw-tooth generator. In particular, it has been shown that each single ramp of the saw-tooth generator ramp exhibits good linearity and slope precision once calibrated. In this section, we now use the calibrated saw-tooth generator as test stimulus generator for an ADC and we want to evaluate the quality of this test stimulus in the context linear histogram testing. More precisely, we want to demonstrate that the histogram built up collecting the samples on the ADC output corresponds to the histogram of a perfect converter.

The experimental setup is the following. We consider a perfect converter of 10-bit resolution, 3V full scale range and 50MHz clock rate. We want to perform the histogram test on this converter using 64261 samples collected at-speed.

The first step consists in defining the appropriate input stimulus amplitude and period to test this converter. Ideally, the input stimulus amplitude should corresponds to the full scale range of the converter. However in practice, an input signal slightly larger than full scale is usually applied to the circuit under test. This permits to ensure that all converter codes are fully exercised. In addition, this also permits to limit the effect of unavoidable non-linearity errors at the initialization of each ramp of the saw-tooth signal. So we choose  $V_{init} = -1.6V$  and  $V_{ref} = +1.6V$  so that the saw-tooth generator delivers a 3.2V amplitude signal once calibrated.

Then, we have to define the period of the input signal. We actually want to collect 64261 with a sampling frequency at 50MHz. In order to minimize test time, the coherent testing approach is usually adopt [16]. Indeed, coherent testing provides a means to gather information using the least number of samples. Basically the idea is that, given a number of samples, these samples can be distributed over a controlled interval in a way that is informationally equivalent to a uniform distribution over one signal period. The fundamental requirement for coherent testing is described by the equation:

$$\frac{F_{in}}{F_s} = \frac{M}{N}$$

with  $M$  and  $N$  relatively prime, where  $F_{in}$  is the input test frequency,  $F_s$  the sampling frequency,  $M$  the number of input test cycles and  $N$  the number of samples.

So according to this equation, we determine that the samples have to be collected on  $M=13$  periods on the input test signal. Indeed, remember that the basic ramp generator is designed to deliver a ramp signal in the 3V amplitude range with a 0.1ms duration under typical mean conditions. The input test frequency consequently lies in the 10kHz range, implying 13 periods of the input signal to collect the 64261 samples. The test frequency is then selected according to the coherent testing requirement as  $F_{in} = 10.111\text{kHz}$ , corresponding to a signal period of  $98.9\mu\text{s}$ . So we define the *Init* signal as a pulse at logic 1 each  $98.9\mu\text{s}$  with a duration of  $0.3\mu\text{s}$ , while the *Step* signal is maintained at logic 1 for the duration of the input test stimulus.

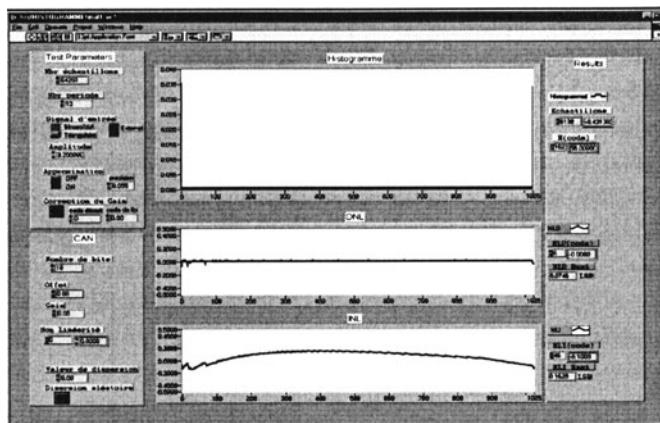


Figure 6. Experimental histogram test results

Finally, we apply 13 periods of the calibrated saw-tooth signal to the converter and we build up the corresponding histogram. From this histogram, we can now evaluate the DNL and INL of the converter. Experimental results are given in figure 6. The first graph corresponds to the measured histogram on the ADC output. As expected, this histogram appears perfectly flat with an equal code count for all codes (excepted the extreme ones because of the overloaded input signal). The second graph plots the DNL of the converter. The maximum measured DNL value is equal to 0.07 LSB. Such a low value is in agreement with the assumption of an ideal converter, taking into account that the finite number of samples used to built up the histogram introduces an inaccuracy in the parameter measurement. Finally, the last graph shows the INL of the converter. The maximum measured INL value is 0.16 LSB. At first glance, this value seems to be less performing than the DNL one. Nevertheless, it remains close to that obtained from an ideal converter. In any case, it remains under that usually derived from real converters.

Concerning the critical criterion of extra area the solution we propose is found to be highly performing. Indeed, the area of our generator is less than 0.05 mm<sup>2</sup>. Which corresponds to an area overhead of 2 % when compared to a classical flash 6-bit ADC.

## 6. CONCLUSIONS

In the context of BIST for analog and mixed signal circuits, this paper presents an on-chip generator dedicated to the test of A-to-D Converters using a linear histogram approach. In order to fulfill reliable characterization of ADCs a very original adaptive scheme is proposed that ensures both high

linearity of the ramp and precise amplitude control while maintaining low silicon overhead. The basic principle of the generator consists in comparing the generated ramp voltage to a predetermined reference voltage and feeding back an adjustment signal so that the ramp voltage converges with the reference one. The effectiveness of the calibration is clearly demonstrated on an example where the ramp voltage calibrates itself to the reference voltage in less than 10 cycles with an average slope error of 0.4 %. We measured a maximum INL of 91  $\mu$ V which translates in a linearity of 15 bits for the ramp signal. In addition, the implementation of the circuit requiring only about ten transistors, two capacitors and one comparator, exhibits an extremely small area.

## 7. REFERENCES

- [1] M.J. Ohletz, "Hybrid Built-In Self-Test (HBIST) for Mixed Analogue/Digital Integrated Circuits", *Proc. European Test Conference*, pp. 307-316, 1991.
- [2] N. Nagi, A. Chatterjee, J. Abraham, "A Signature Analyzer for Analog and Mixed-Signal Circuits", *Proc. ICCD*, pp.284-87, 1994.
- [3] K. Damm, W. Anheier, "HBIST Of Nonlinear Analog Building Blocks In Mixed-Signal Circuits", *International Mixed Signal Testing Workshop*, pp.257-62, June 1995.
- [4] E. Teraoca, T. Kengaku, I. Yasui, K. Ishikawa, T. Matsuo, "A Built-In Self-Test for ADC and DAC in a Single-Chip Speech CODEC", *Proc. International Test Conference*, pp. 791-796, 1993.
- [5] M.F. Toner and G.W. Roberts, "A BIST Scheme for a SNR, Gain Tracking and Frequency Response Test of a Sigma-Delta ADC", *IEEE Trans. Circuits & Systems II*, Vol. 42, pp. 1-15, 1995.
- [6] S Sunter, N. Nagi, "A Simplified Polynomial-Fitting Algorithm for DAC and ADC BIST", *Proc. International Test Conference*, pp. 389-395, 1997.
- [7] K. Arabi, B. Kaminska, "Efficient and Accurate Testing of Analog-to-Digital Converters Using Oscillation-Test Method", *Proc. European Design & Test Conference*, pp. 348-352, 1997.
- [8] R. de Vries, T. Zwemstra, E. Bruls, P. Regtien, "Built-In Self-Test Methodology for A/D Converters", *Proc. European Design & Test Conference*, pp. 353-358, 1997.
- [9] M. Renovell, F. Azaïs, S. Bernard, Y. Bertrand, "Hardware Resource Minimization for a Histogram-based ADC BIST", *Proc. VLSI Test Symposium*, pp. 247-252, 2000.
- [10] F. Azaïs, S. Bernard, Y. Bertrand, M. Renovell, "Towards an ADC BIST Scheme using the Histogram Test Technique", *Proc. European Test Workshop*, pp. 53-58, 2000.
- [11] X. Haurie, G.W. Roberts, "Arbitrary-Precision Signal Generation for Bandlimited Mixed-Signal Testing", *Proc. International Test Conference*, pp.78-86, 1995.
- [12] A.K. Lu, G.W. Roberts, "An Analog Multi-Tone Signal Generator for Built-In Self-Test Applications", *Proc. International Test Conference*, pp. 650-659, 1994.
- [13] G.W. Roberts, A.K. Lu, "Analog Signal Generation for Built-In Self-Test of Mixed-Signal Integrated Circuits", *Kluwer Academic Publishers*, ISBN 0-7923-9564-6, 1995.
- [14] B. Provost, E. Sanchez-Sinencio, "Auto-Calibrating Analog Timer for On-Chip Testing", *Proc. International Test Conference*, 1999.
- [15] F. Azaïs, S. Bernard, Y. Bertrand, X. Michel, M. Renovell "A Low-Cost Adaptive Ramp Generator for Analog BIST Applications", *Proc. VLSI Test Symposium*, 2001.
- [16] M. Mahoney, "DSP-based Testing of Analog and Mixed-Signal Integrated Circuits", *IEEE Computer Society Press*, ISBN 0-8186-0785-8, 1987.

# Built-in test of analog non-linear circuits in a SOC environment

Luigi Carro<sup>1,2</sup>, André C. Nácul<sup>1</sup>, Daniel Janner<sup>2</sup> and Marcelo Lubaszewski<sup>1,2</sup>

<sup>1</sup>*Universidade Federal do Rio Grande do Sul -- Instituto de Informática*

<sup>2</sup>*Universidade Federal do Rio Grande do Sul -- Departamento de Engenharia Elétrica*

**Abstract** This work describes a methodology for the built-in test of non-linear analog circuits, developed at the system level, by the use of digital non-linear adaptive filters. Non linear analog circuits are widely used on data transmission applications, such as wireless communication, radio and portable multimedia systems. Several non-linear circuits like mixer, automatic gain control amplifiers, peak-detectors and log-amplifiers were simulated, and some prototypes assembled, in order to allow fault injection in the circuit under test. The proposed built-in test methodology is very precise, has low cost when seen at the system level, and allows complete fault coverage with a very small testing time.

**Key words:** analog built-in testing, non linear circuits (mixer, agc, peak-detector, log-amplifier), non linear adaptive filters

## 1. INTRODUCTION

Although there has been a lot of work in the area of built-in test of linear analog circuits [1-6], the built-in test of non-linear analog circuits has been limited to AD and DA converters, and some experiments on PLLs. Other important circuits, used mainly in communication devices (like mixers, log-amplifiers and others presented here) are non-linear in nature. To the authors' knowledge, built-in test for mixers and other non-linear circuits has not been extensively addressed before. One of the reasons is that all the

theory that can easily be applied to linear circuits (*s* and *z* transforms) is not adequate to deal with non-linear components. In [7-8] a discussion on where to insert test points in communication circuits using mixers and data converters has been presented. A built-in strategy to test the mixer itself was not considered. In [9], some design for testability issues were addressed to both reduce production test cost and improve test quality for automatic gain controllers.

In the era of Systems-on-Chip (SOC), a built-in test strategy that can benefit from other hardware already available in the system is likely to have lower area and performance penalties. Nowadays, almost all applications rely on some sort of digital processing of signals, from low level signal acquisition and filtering to complex multimedia applications like voice and video processing. A real-life example of such complex mixed signal devices is the Software Radio, a SOC by itself [10-12].

This way, aiming at extensive reuse of components and accurate and low cost built-in analog test, this work presents a built-in test methodology that uses common digital hardware found in most present-days SOC to test non-linear analog circuits, also present in the SOC. All testing is performed using non-linear adaptive filters developed in the digital domain. Adaptive Filtering has already been proved to be a powerful technique for linear systems testing and identification [13–14]. The main goal of this work is to use adaptive filters for identification and testing of non-linear systems.

The proposed test technique extracts the analog signature of the circuit under test and stores it as filter coefficients in the digital domain. Then, no precise knowledge of the circuit must be acquired. To the author's knowledge, this is the first proposed methodology targeting built-in test of different integrated non-linear analog circuits at the SOC level.

This paper is organized as follows: section 2 presents the proposed testing procedure, while section 3 presents the results achieved for this methodology, both in simulation and real life prototypes. Section 4 presents an analysis on the required resolution and frequency of the analog to digital converters used during testing. Finally, section 5 presents our conclusions and future work.

## **2. THE TESTING PROCEDURE**

The theory regarding linear and non-linear system recognition using Digital Signal Processing techniques (DSP) is well developed [14] and is the basis of the work presented in [13]. The methodology of analog circuit recognition using adaptive filters is shown in figure 1. In the first phase (figure 1a), known as training phase, a fault-free system and the adaptive

filter are submitted to the same excitation and their outputs are compared. The error signal helps to calculate the filter coefficients to the next iteration in the training phase.

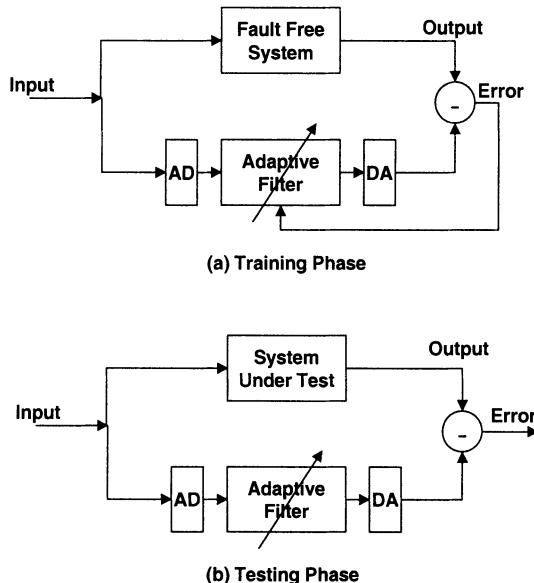


Figure 1. Adaptive Tester Topology

After the training phase, the filter coefficients have stabilized, and the filter can now reproduce the circuit behaviour under all situations for which it has been trained. Then, the fault-free system is replaced by the system to be tested and the error signal is used to detect system failures (figure 1b). This is called the testing phase. Both the circuit and the filter are submitted to the same input and their outputs are compared, generating an error signal. Hopefully, any fault in the system, either a hard, soft or catastrophic fault, will change the output of the tested system, which will be detected by the filter and compared to a tolerated error to indicate if the system is error free.

The error signal (figure 1b) indicates whether the system is fault free or not. It is important to notice that, although it is very close to zero, the error signal for the fault free system is not zero. However, the error signal for the faulty system is much larger than the fault free signal, thus indicating a failure in the tested system. A threshold value that indicates whether a system is faulty or not must be determined. Usually, this is done experimentally. A higher threshold implies that more systems will be classified as fault-free system, and then one can say that the tester is more tolerant or flexible.

As it is shown in figure 1, the proposed testing methodology requires the use of Analog to Digital converters. This is because the adaptive non-linear filter is developed in the digital domain, while the system under test works in the analog domain. However, in a mixed signal SOC like the Software Radio, for example, fast DA and AD converters are readily available [10-11, 22]. The use of a digital filter is favoured because of available design automation and construction robustness, not to mention its spread use on most present days SOCs. The influence of the resolution of the required analog to digital converters and some comments about their speed is studied in section 4.

## 2.1 Filter Algorithm and Topology

The most common topology for linear adaptive filters uses a Finite Impulse Response Digital Filter (FIR), mainly because of stability reasons. Although FIR filters are stable, an adaptive FIR filter can be unstable under some conditions [15]. On the other hand, for non-linear systems, many topologies have been studied, and probably the Volterra equations is the most common equation set for modelling these systems [15].

The topology proposed here is implemented with a non-linear FIR filter. The implemented filter can be represented by equation (1).

$$\begin{aligned} Y(n) = & \sum_{m=0}^{N_1-1} (c_1(m)x_1(i-m) + c_2(m)x_2(i-m)) \\ & + \sum_{m=0}^{N_2-1} c_x(m)x_1(i-m)x_2(i-m) \end{aligned} \quad (1)$$

$Y(n)$  is the output of the filter at discrete time  $n$ .  $N_1$  and  $N_2$  are the number of taps on the linear and non-linear components of the filter, respectively. The filter coefficients are  $c_1(0\dots N_1-1)$  and  $c_2(0\dots N_1-1)$ , in the linear, and  $c_x(0\dots N_2-1)$  in the non-linear components.

The filter adapts to any circuit, provided it has enough taps (pair coefficient-sample) to represent all poles and zeros of the original system. To decide which signal to use to train the filter and test the system, we should consider that the signal must last enough for the filter to converge and must be rich in frequency components, in order to provide excitation of the circuit under test at all frequencies of interest.

The main difference between our work and other non-linear filter topologies is that we develop a non-linear FIR filter with two inputs, instead of only one. This comes from the concept that, while in a linear system, the output can only present a sum of all inputs in a scaled and phase modified

version. In a non-linear system, frequencies that are not present at the input can be generated internally. Since  $\cos(a) * \cos(b) = 1/2[\cos(a+b)+\cos(a-b)]$ , it is clear that the multiplication of two cosines produces as a result two new frequencies, that were not present at the original inputs, but are a combination of both. If the difference between these two frequencies is large (say, 10 times, as it happens in heterodyne receivers [16-17]), it is very hard to obtain, with only one non-linear input, all the necessary frequencies to test the circuit in its entire frequency spectrum.

The algorithm for updating the filter coefficients during the training phase is the LMS, one of the simplest adaptive algorithms. Some experimentation is needed to determine the algorithm coefficients. Further details can be found in [18].

## 2.2 The input signal generation

In the testing case presented above, one relies on the accurate generation of an excitation signal. Again, considering the built-in test of the proposed analog non-linear circuits in a SOC environment, it is very likely that a DA converter might be available for a loopback test mode. This way, one can generate the excitation signal in the digital domain, and after use a DA converter to reach the desired analog circuits. For example, in the context of the Software radio or other application that requires bi-directional communication or interfacing with the real world, a pair AD/DA converter is most likely to be present.

The set of frequencies to be used as excitation for the circuit under test is another issue. Generally, white noise is the preferred choice, since it has enough frequency components spread through all the spectrum of interest, and it is easily generated within the digital domain (with a LFSR, for example), considering the pass-band limitations of the circuits to be tested.

## 3. EXPERIMENTAL RESULTS

In order to check the coverage of the test procedure presented in this work, this section provides some test cases that were both simulated and prototyped in our laboratories. The simulation results were obtained using Matlab<sup>TM</sup> as the modelling and simulation tool. Also, an analog mixer was built by the authors with real life discrete components, so that it was possible to inject faults in the mixer and observe its output signal. This data was acquired with a digital oscilloscope and later used in Matlab<sup>TM</sup>, where the non-linear adaptive tester was implemented. All testing procedure was done off-line.

### 3.1 Analog Mixer

A prototype mixer [23] was assembled and is shown in figure 3. The input signals  $x_1$  and  $x_2$  are represented in the diagram by the AC sources  $vin_1$  and  $vin_2$ . The output signal is differential, measured between points  $V_1$  and  $V_2$ . With this experiment we are interest in making actual fault insertion to check the behaviour of the pair mixer-adaptive tester.

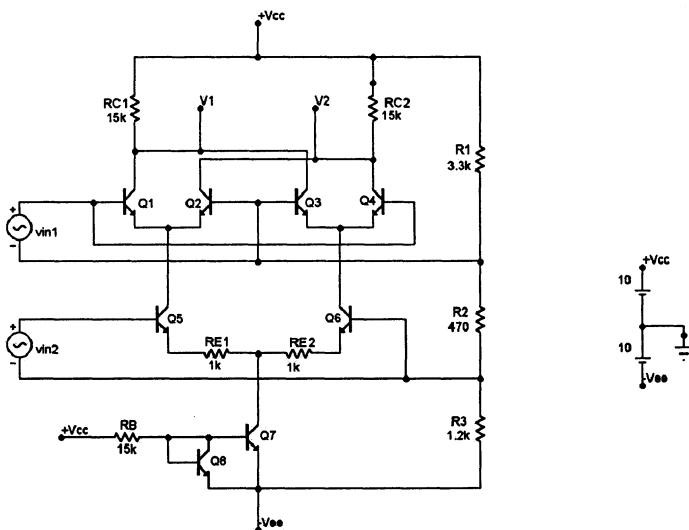


Figure 2. The prototyped mixer

#### 3.1.1 Component Deviations

The first type of failure that was simulated in the mixer was a deviation in the nominal value of the components of the circuit. We simulated faulty resistors for components  $RC_1$ ,  $RC_2$ ,  $RE_1$  and  $RE_2$ , and after open and short faults for all the transistors in the system. Table 1 and table 2 show the error signal ratio for the faulty system and the fault free system in some test cases. For a complete table, with all the component deviation that was simulated, please refer to [18]. A unitary ratio means that the fault was not detected by the proposed testing procedure. Observing the results, it is possible to verify that the procedure is able to detect component deviations as small as 1% very easily. Also, shorts and open faults in the transistors produce an error signal that is much higher than the error signal of the fault free mixer.

**Table 1.** Output error ratio – resistor faults

|     | 0%   | +1%    | -1%    | +10%     | -10%     |
|-----|------|--------|--------|----------|----------|
| RC1 | 1.00 | 163.41 | 163.68 | 1.63e+03 | 1.64e+03 |
| RC2 | 1.00 | 163.46 | 163.59 | 1.62e+03 | 1.64e+03 |
| RE1 | 1.00 | 18.39  | 18.90  | 177.56   | 195.97   |
| RE2 | 1.00 | 18.68  | 18.54  | 177.51   | 195.30   |

**Table 2.** Output error ratio – transistor faults

|    | open     | Short    |
|----|----------|----------|
| Q1 | 1.50e+04 | 1.52e+04 |
| Q5 | 2.12e+03 | 3.93e+03 |

### 3.1.2 Third Harmonic Interference

This section addresses the third harmonic interference problem, an important design parameter for communication circuits. The results presented here were simulated in Matlab™. A third harmonic component was inserted in the local oscillator (input signal  $x_2$ ) and the error signal was analysed. Table 3 summarizes the results.

As one can see in table 3, a very small (1/1000) third harmonic component in the output signal produces an error signal that is much larger than an error free output, being easily detected.

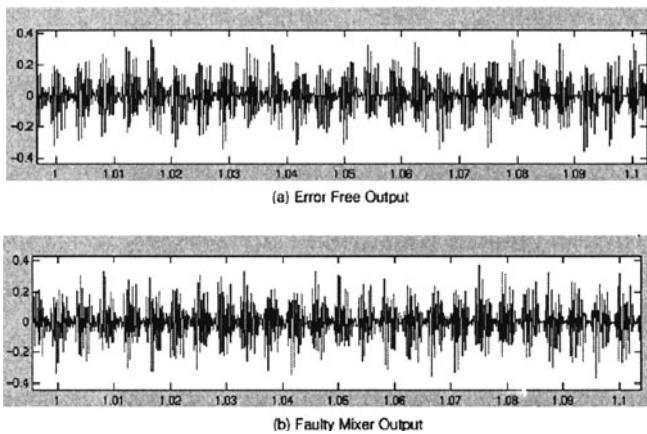
**Table 3.** Output error ratio

| Amplitude | Error Power Ratio |
|-----------|-------------------|
| 0         | 1.00              |
| 1/10      | 6.62e+03          |
| 1/100     | 6.62e+02          |
| 1/1000    | 66.11             |

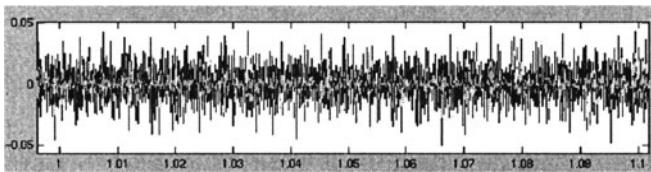
### 3.1.3 Voice Signal Testing

In order to verify the fault detection of the proposed testing methodology at the presence of complex signals, the RF mixer was excited with a voice signal (like in an actual modulation procedure, for example). First, a fault free system was tested. Later on, a faulty mixer, with a 5% frequency deviation in the local oscillator, was tested. Figure 4 shows the output signal and figure 5 shows the error signal for both testing cases.

As it can be seen, both outputs are very similar, but the error signal power for the faulty system is larger. While the error signal in the fault free mixer has a mean power of 0.0088, the faulty mixer has a mean power of 0.0171, that is 93% larger.



*Figure 3.* The input voice signal



*Figure 4.* Error signal for the voice signal

### 3.2 Automatic Gain Controller

An automatic gain controller (AGC) was simulated using Matlab<sup>TM</sup>. The input is a sinusoidal signal attenuated by an exponential signal,  $v_i = \text{Acos}(\omega t) \cdot 0.9k$ . The desired output of the AGC is the sinusoidal input with no attenuation. In the adaptive filter, the input signal  $x_1$  is the input signal of the AGC, while the gain signal of the AGC is the input signal  $x_2$ .

It is possible to observe in table 4 that a catastrophic fault like a short capacitor is easily detected by the testing methodology. Some component deviations are also detected by the proposed testing methodology. If the AGC produces a gain signal that is 1% larger than expected, the error signal is almost 10% larger when compared to the fault free system.

*Table 4.* Output error ratio

| Fault simulated         | Error Power Ratio |
|-------------------------|-------------------|
| Short capacitor         | 3.28              |
| Gain stuck at 1         | 2.29              |
| 1% error in gain signal | 1.09              |

### 3.3 Peak Detector

A peak detector was also simulated in the Matlab<sup>TM</sup> environment. As a peak detector has only one input, the inputs of the adaptive filter were short-circuited, so that  $x_1 = x_2$ . During simulations, we inserted faults so that the peak detector under test could not sustain the peak voltage for a long time. Table 5 shows the error signal ratio for the faulty and the fault free peak detector.

*Table 5. Output error ratio*

| Voltage sustained (%) | Error Power Ratio |
|-----------------------|-------------------|
| 95                    | 3.00e+04          |
| 99                    | 1.19e+04          |
| 99.9                  | 2.35e+03          |
| 99.99                 | 351.42            |

### 3.4 Log-amplifier

A logarithmic amplifier was modelled and simulated in Matlab<sup>TM</sup>. A log-amp is a one input circuit, like the peak detector, and then again the inputs of the adaptive filter were shortened. Equation 2 was used to model the log-amp in Matlab<sup>TM</sup>.

$$V_{out}(i) = (R_2 / R_1 + 1)V_t \ln(i / i_{ref}) \quad (2)$$

Faulty resistors and sources were simulated and table 6 shows the error power ratio for the fault free system and the faulty system.

*Table 6. Outpur error ratio*

|      | R1   | R2   | I <sub>ref</sub> |
|------|------|------|------------------|
| +10% | 1.44 | 1.55 | 1.71             |
| -10% | 1.69 | 1.55 | 1.71             |
| +20% | 2.33 | 2.84 | 3.05             |
| -20% | 3.48 | 2.72 | 3.66             |

## 4. THE REQUIRED RESOLUTION AND SPEED OF A/D CONVERTERS

All the results presented up to now considered a 16 bit A/D converter, that is almost an ideal assumption. In real life systems, there are different precision A/D converters, from 6 to 16 bits.

To determine the sensitivity of the proposed testing methodology to different precision A/D converters, we simulated several converters, from 6 to 16 bits. Table 7 shows the mean error value for the fault free system. Note that the more resolution available in the converter, the better is the error value for a fault free system. This means that one can assume a smaller threshold value in determining whether the system under test is good or not, which leads to more precise testing results.

*Table 7.* Mean error value

|    | Mean error value (fault free system) |
|----|--------------------------------------|
| 6  | 8.2e-03                              |
| 8  | 2.8e-03                              |
| 10 | 8.2e-04                              |
| 12 | 1.9e-04                              |
| 14 | 4.6e-05                              |
| 16 | 1.2e-05                              |

The third harmonic interference problem was also simulated with different A/D converters and table 8 reports the different error ratios. Observing table 8, it is possible to see that as the precision of the A/D converter increases, the testing procedure performs better. However, even with lower precision converters, the testing was also successful. Among all the third harmonic interference that were simulated, only one failure was not detected, when the third harmonic component had an amplitude 1/1000 and the system used an 8 bit A/D converter. With the same amplitude (1/1000) for the third harmonic, a system with a 10 bit A/D converter reported an error signal that is 37% higher than the error signal in the fault free mixer.

*Table 8.* Error power ratio

| Amplitude | 8-bit A/D | 10-bit A/D | 12-bit A/D |
|-----------|-----------|------------|------------|
| 0         | 1.00      | 1.00       | 1.00       |
| 1/10      | 33.33     | 104.22     | 447.05     |
| 1/100     | 3.32      | 10.28      | 44.76      |
| 1/1000    | 1.00      | 1.37       | 4.28       |

One important point concerning the use of the proposed method is the speed of used AD and DA converters. It is true that although mathematically robust, for high-speed circuits the method presented in this paper depends heavily on the availability of high-speed analog to digital converters. Present day AD and DA converters have a limited sample ratio, when one thinks on actual communication frequencies requirements. For example, one can find commercial AD and DA converters reaching 1G samples/second, with 8 bits resolution [10,22]. However, the power consumption of such devices makes them inadequate for portable equipment.

For some RF frequencies, fast and low-power converters are available [19], but for Gigahertz applications they are still missing. Nevertheless, there

have been appearing ever more publications regarding the integration of  $\Sigma\Delta$  converters directly on the RF signal path, as in [20]. Moreover, some of the present work into RF AD converters focus on the integration of the mixer and the converter in a single unit, like [21]. This way, it is possible that in a near future this same testing technique can be applied to online testing of non-linear analog circuits working at Gigahertz speeds.

## 5. CONCLUSIONS AND FUTURE WORK

This paper addressed the difficult topic of built-in testing of non-linear analog circuits like RF mixers, automatic gain controllers, peak-detectors, and log-amplifiers. The test methodology here presented allows the construction of a built-in self test mechanism that is precise, and can take benefit from other modules available at the system level. The proposed technique can be applied to RF communication systems and multimedia portable SOCs, like the Software Radio, for example.

Faulty mixers were both prototyped and simulated, and the testing procedure was able to detect component deviations as small as 1%, and third harmonic component interference as low as 1/1000. All other circuits were simulated in Spice and Matlab<sup>TM</sup>, and small component deviation faults were successfully detected. Also, the required AD and DA converters resolution and speed was addressed.

The area penalty of the proposed approach is minimal, since most of the hardware is already available in the SOC. Area penalty will come mainly from switches to maintain the loopback configuration during test.

As a future research topic, the authors intend to verify the possibility of using low-speed and hence low-power converters in the test path, so that together with extra analog hardware, one can reach GHz frequencies without power penalties. Also, it would be interesting to evaluate the power consumption of the proposed technique with a purely non-linear analog built-in technique, which has yet to be developed.

## 6. REFERENCES

- [1] K. Arabi and B. Kaminska. Oscillation built-in self test (OBIST) scheme for functional and structural testing of analog and mixed-signal circuits. Procs. of Intl. Test Conf, pp. 786–795, 1997.
- [2] G. Huertas, D. Vázquez, A. Rueda, and J. L. Huertas. Effective oscillation-based test for application to a DTMF filter bank. Procs. of Intl. Test Conf., pp. 549–555, 1999.
- [3] N. Nagi, A. Chatterjee, A. Balivada, and J. A. Abraham. Efficient multisine testing of analog circuits. Procs of Intl. Conf. on VLSI Design, pp 234–238, New Delhi, India, 1995.

- [4] M. Lubaszewski, S. Mir, and L. Pulz. ABILBO: Analog Built-In Block Observer. Procs of IEEE/ACM Intl. Conf. on Computer Aided Design, pp 600–603, San Jose, CA, 1996.
- [5] M. Renovell, M. Lubaszewski, S. Mir, F. Azais, and Y. Bertrand. A multi mode signature analyzer for analog and mixed circuits. Procs of IFIP Intl. Conf. on Very Large Scale Integration, pp 65–76, Gramado, Brazil, 1997.
- [6] E. Cota, M. Renovell, L. Carro, M. Lubaszewski, F. Azais, and Y. Bertrand. Reuse of existing resources for analog BIST of a switch capacitor filter. Procs of ACM/IEEE Design Automation and Test in Europe Conf., pp 220–226, Paris, France, 2000.
- [7] S. Ozev and A. Orailoglu. Block-based test integration for analog integrated circuits. Procs of Latin American Test Workshop, pp 128–132, Rio de Janeiro, Brazil, 2000.
- [8] S. Ozev, I. Bayraktaroglu, and A. Orailoglu. Test synthesis for mixed signal SOC paths. Procs of ACM/IEEE Design Automation and Test in Europe Conf., Paris, France, 2000.
- [9] A. Lechner, A. Richardson, B. Hermes, and M. Ohletz. A design for testability study on a high performance automatic gain control circuit. Procs IEEE VLSI Test Symposium, pp 376–385, 1998.
- [10] P. Leppänen et al. Software radio – an alternative for the future in wireless Personal and Multimedia Communications. Procs of IEEE ICPWC99, pp.364–368.
- [11] E. Buracchini The Software Radio Concept. *IEEE Communications Magazine*, 38(9):138–143, Sep. 2000.
- [12] J. Mitola The Software Radio Architecture. *IEEE Communications Magazine*, 33(5):26–38, May 1995.
- [13] M. Negreiros and L. Carro. Efficient analog test methodology based on adaptive algorithms. In Procs of 35th Design and Automation Conf., pp 32–37, San Francisco, 1998.
- [14] W. Jenkins and et. al. Advanced Concepts in Adaptive Signal Processing. Kluwer Academic Publishers, USA, 1996.
- [15] V. J. Mathews. Adaptive polynomial filters. *IEEE Signal Processing Magazine*, pages 10–26, July 1991.
- [16] J. Crols and S. J. Steyaert. Low-IF topologies for high performance analog front ends of fully integrated receivers. *IEEE Transactions on Circuits and Systems II*, 45(3):269–282, Mar. 1998.
- [17] S. Haykin. Communication Systems. John Wiley & Sons, New York, 1978.
- [18] A. Nácul, L. Carro, D. Janner and M. Lubaszewski. Testing of RF mixers with adaptive filters. Procs of IEEE Intl. Mixed Signal Test Workshop, Atlanta, USA, 2001.
- [19] J. A. P. Van Engelen, R. Van de Plassche, E. Stikvoort, and A. G. Venes. A sixth-order continuos-time bandpass sigma-delta modulator for digital radio IF. *Journal of Solid-State Circuits*, 34(12):1753–1764, Dec. 1999.
- [20] A. I. Hussein and W. B. Kuhn. Bandpass  $\Sigma\Delta$  modulator employing undersampling of RF signals for wireless communication. *IEEE Transactions on Circuits and Systems II*, 47(7):614–620, July 2000.
- [21] A. Namdar and B. Leung. A 400-MHz, 12 bit, 18-mW, IF digitizer with mixer inside a sigma-delta modulator loop. *Journal of Solid-State Circuits*, 34(12):1765–1776, Dec. 1999.
- [22] R.H. Walden. Analog-to-Digital Converter Survey and Analysis. *IEEE Journal on Selected Areas in Communications*, v17, n4, Apr. 1999, p.539–550.
- [23] D. O. Pederson, K. Mayaram. Analog Integrated Circuits for Communication. Kluwer Academic Publishers, 1991, p.568.

# Design of a Fast CMOS APS Imager for High Speed Laser Detections

B. Casadei<sup>(1)</sup>, J. P. Le Normand<sup>(1)</sup>, Y. Hu<sup>(1)</sup> and B. Cunin<sup>(2)</sup>.

(1) LEPSI , 23 rue du LOESS, 67037 Strasbourg; (2) GOA, 23 rue du LOESS, 67037 Strasbourg

**Abstract:** In this paper, the results of the first temporal resolution imager CMOS for high-speed laser pulse characterisation are presented. This new imager can replace the conventional streak camera using a charge-coupled device (CCD) sensor for some applications. It produces the intensity information in function of time and one spatial dimension ( $I_{ph} = f(x, t)$ ). The time information is obtained for the prototype to delay successively the integration phase in each pixel of the same row. The different noise sources for an APS sensor such as the shot noise due to the photo sensor, the flicker noise and the thermal noise due to the transistors are studied to determine the fundamental limits on image sensor. The prototype imager named FAMOSI (FAst MOS Imager) consists of an array of  $64 \times 64$  active pixel sensors that are integrated in AMS  $0.6 \mu\text{m}$  CMOS technology. Each pixel size is  $13 \mu\text{m} \times 13 \mu\text{m}$  and has a fill factor of 28%. At 50 frames/s, theoretical and experimental results show a total noise of  $95 \pm 4$  electrons, a fixed pattern noise of 2% (saturation voltage is 760 mV), a dynamic range of 64 dB and a power consumption of 25 mW. The conversion gain is  $6.73 \pm 0.25 \mu\text{V}/\text{electrons}$  and the time resolution is 0.8 ns.

**Key words:** CMOS sensors, noise analysis and optimisation, high time resolution imager.

## 1. INTRODUCTION

The streak camera using CCD sensor [1] performs the measurement in temporal domain to study of the fast speed light pulses. It produces the light intensity variation in function of time and one spatial dimension. The streak camera parts use a vacuum tube where the photons are collected and converted in electrons. These electrons are accelerated, deflected by a transverse electric field, which varies linearly with the time. Then they are multiplied with a multi-channels plate (MCP) and projected on a phosphor

screen. The CCD camera reads this information. In repetitive synchroscan mode that means the successive accumulation of short light pulses at regular time interval, the temporal resolution can reach two picoseconds. Unfortunately these cameras are voluminous, quite expensive (\$ 70000), fragile because the streak parts use a vacuum tube, and request a large power supply (a few kV) [2].

Our research work is to design a fast imager for observation of the short light pulses by using the microelectronic techniques such as the CMOS technology. The major advantage of CMOS technology in which sensor, analogue and digital circuits are integrated in the same silicon substrate (system on chip), is to be able to realise the smart sensor. These VLSI systems are low cost and low power consumption.

In the visible imaging, a lot of research works has been done in which the CMOS sensors [3], [4] are a viable alternative to replace CCD camera. The advantages of a CMOS sensor compared to CCD are the random readout, windowing and the integrating time different per pixel of the light flux.

Actually, the most CMOS or CCD cameras give the light intensity in function of the two spatial dimensions ( $I = f(x, y)$ ). The innovation of our CMOS imager is to give the light intensity in function of time and one spatial dimension ( $I = f(t, x)$ ). To obtain this temporal dimension from the spatial array ( $x, y$ ), we make a splitting (streak role) of the light intensity by checking the time integration of each pixel of the same row of the matrix.

The following sections describe the architecture and operation of our imager. We present a full analysis of the different noise sources in the imager and the experimental results.

## **2. ARCHITECTURE AND OPERATION**

### **2.1 Architecture**

FAMOSI is not a classical imager. The originality of this imager is that the intensity information in function of the one-dimensional optical information of one image and the time can be restored. In general, there are two main families of CMOS sensors. The passive pixel sensor (PPS) is simpler than the active pixel sensor (APS) but the noise is higher and the readout is achieved one time with the PPS, even then the readout can be done many times with an APS [5], [6]. For the camera FAMOSI, an APS structure has been chosen. In APS, a photo detector is integrated in a pixel together with selection switches and the source follower amplifier, which connect the photo sensor directly to the output line for the readout. The topology of this imager is shown in figure 1. Each pixel consists of a photodiode  $N^+ P_{well}$ , a

reset transistor Mrst, a source follower transistor Msf, and an access transistor Msr (row switch). Column circuits include a bias transistor Mc that acts as current source and the column switch Msc.  $C_L$  is the total capacitance of the column bus.  $C_{FN}$  is the total capacitance at the floating node at the photodiode equal to 22 fF. It is the sum of the parasitic capacitances of the source follower and the photodiode capacitance.

## 2.2 Operation principle

The operations of this imager are described in figure 2. When the sensor is illuminated by an optical pulse source (i.e. a laser), the voltage signal obtained at the output of the chip is proportional to the function of the temporal repartition of the light intensity. The output from our sensor is first amplified by a low noise amplifier, and then converted to 12 bits digital signal. The image reconstruction for the initial optical signal can then be performed by the data processing.

A frame is achieved in three operations. For each pixel, first, the reset transistor Mrst is turned on to initialise the photodiode capacitance and to eliminate the charges stored in the capacitance of the source follower. The light pulse appears in the acquisition phase. Then the charge signal that is proportional to the optical intensity and the time, is integrated in the photodiode capacitance when the transistor Mrst is switched off. Finally, the selection of the row control transistor Msr and the column control transistor Msc provide the series readout of the voltage signal.

In order to obtain the information intensity in function of time, the reset signal is delayed for each column to retard the start of the integration in the photodiode capacitance. This delay is realised by an array of the logic gates and it controls the reset transistor Mrst for each column.

The advantage to use a 2D array is to have 64 temporal resolution imagers CMOS in one device. Each column gives the time information and each line provides the spatial dimension. Then this chip allows to observe multiple phenomena in the same time for example in fluorescence application when the light comes from several light sources.

## 3. NOISE ANALYSIS AND OPTIMISATION

For a low noise CMOS imager with high resolution, noise sources must be carefully analysed because it can limit the performance of the active pixel sensors, especially with regard to the small signal produced by a weak optical source.

Noise during reset: During the reset, the transistor M<sub>sr</sub> is turned off and a positive voltage pulse is applied to the gate of M<sub>rst</sub>. This transistor is saturated during a short time and then it goes below the threshold for the reset phase. If the reset time  $t_r$  is much greater than the settling time  $t_{\text{settle}}$ , it can be said that steady state has been achieved. The time  $t_{\text{settle}}$  represents the moment when the reset transistor subthreshold current is equal to the value of the diode leakage current. The mean square thermal noise voltage can be calculated. It is thermal noise given by the switch-on resistance  $R_{\text{dson},\text{Mrst}}$  of the transistor RST and passed through a low pass filter. Then the mean square thermal noise voltage can be calculated by

$$\overline{V_{n,\text{Mrst}}^2} = 4 kT R_{\text{dson},\text{Mrst}} \int_0^\infty f_c \frac{1}{1+x^2} dx \quad (1)$$

where  $f_c = 1/(2\pi R_{\text{dson},\text{rst}} C_{\text{FN}})$  is the cut-off frequency of this low pass filter and  $x = f/f_c$ . Then this voltage due to the KTC noise source is obtained by (2) and the Equivalent Noise Charge can be also calculated by (3)

$$\overline{V_{n,\text{Mrst}}^2} = \frac{kT}{C_{\text{FN}}} \quad (2)$$

$$\text{ENC}_{\text{Mrst}}(e^-) = \frac{1}{q} \sqrt{kT C_{\text{FN}}} \quad (3)$$

Noise during integration: The dominant noise source during the integration is the shot noise due to the diode leakage current  $I_{\text{leak}}$ . However, this kind of noise should be taken into account when the integration time increases. For a more precise analysis, the change of the diode capacitance should also be taken into account during the integration time. However, it can be considered negligible as 2nd order effect. The mean square value of the noise sampled at the end of integration  $t_{\text{int}}$  is given by:

$$\overline{V_{n,\text{int}}^2(t_{\text{int}})} = \frac{q I_{\text{leak}}}{C_{\text{FN}}^2} t_{\text{int}} \quad (4)$$

Noise during readout: During readout, the transistors M<sub>2</sub> and M<sub>3</sub> as well as the column switch M<sub>s</sub> and current source for the transistor M<sub>c</sub> with line capacitance  $C_L$  are the main noise sources. Normally the noise performance of a MOS transistor is fully characterised by an output noise source  $i_{\text{out}}$  with noise spectral density.

$$S_{i_{\text{out}}}(f) = \frac{8 kT}{3} g_m + \frac{K_F I_D}{C_{\text{ox}} L^2 f} \quad (5)$$

where  $g_m$  is the transconductance of the transistor,  $C_{ox}$  the oxide capacitance per unit area and  $K_F$  the flicker noise coefficient. In (5) the first term presents the channel thermal noise [7], [8] and the second term the flicker noise [9]. As the low cut off frequency for flicker noise is inversely proportional to the circuit on time, the choice of a correct reset voltage to the gate of the input transistor of the source follower for a short period on time in order to reduce flicker noise is very important.

Normally the mean square thermal noise voltage for this transistor can be calculated by

$$\overline{V_{n,Mc}^2} = \int_0^\infty \left( \frac{8}{3} kT g_{m,Mc} \right) \cdot R_t^2 \frac{1}{1 + (2\pi f C_L R_t)^2} df \quad (6)$$

where  $R_t = \frac{1}{g_{ds,Msr}} + \frac{1}{g_{ds,Msc}} + \frac{1}{g_{m,Msf}}$ . By straightforward integration, this voltage is given by (7) and by the same way, the thermal noise contribution of the transistor M2 is (8).

$$\overline{V_{n,Mc}^2} = \frac{2 kT}{3 C_L} g_{m,Mc} R_t \quad (7)$$

$$\overline{V_{n,Msf}^2} = \frac{2 kT}{3 C_L} \frac{1}{1 + g_{m,Msf} \left( \frac{g_{ds,Msr} + g_{ds,Msc}}{g_{ds,Msr} g_{ds,Msc}} \right)} \quad (8)$$

where  $g_{m,Mx}$  and  $g_{ds,Mx}$  are respectively the transconductance and the output conductance of the transistor Mx. Since the transistors Msr and Msc are used as switch transistor, their contributions on the thermal noise can be calculated as a contribution of a resistance given by (9). The thermal noise is obtained by (10). For the transistor Msc the thermal noise contribution is given by the expression (11) and the Equivalent Noise Charge at the input by (12)

$$dv_r^2 = 4 kT R_{on} df \quad (9)$$

$$\overline{V_{n,Msr}^2} = \frac{kT}{C_L} \frac{1}{g_{ds,Msr} R_t} \quad (10)$$

$$\overline{V_{n,Msc}^2} = \frac{kT}{C_L} \frac{1}{g_{ds,Msc} R_t} \quad (11)$$

$$ENC_{Mx}(e^-) = \frac{C_{FN}}{q} \sqrt{\frac{\overline{V_{n,Mx}^2}}{A_g^2}} \quad (12)$$

where  $A_g$  is the total gain of the source follower in pixel and Mx the different transistors. Because these noise sources are not correlated, the total equivalent noise charge of the imager can be calculated by:

$$\text{ENC}_T = \sqrt{\text{ENC}_{\text{Mrst}}^2 + \text{ENC}_{\text{Msf}}^2 + \text{ENC}_{\text{Msr}}^2 + \text{ENC}_{\text{Mc}}^2 + \text{ENC}_{\text{Msc}}^2} \quad (13)$$

According to the formula, the calculation results of equivalent input noise charge are 97 electrons. The thermal noise source is greater than the flicker noise source and we can neglect its effect in the study of the sensor. (see Appendix 1)

As mentioned above, to decrease the reset noise, the photodiode size must be small as possible to decrease the photodiode capacitance as shown the equation (3). An optimisation design has been carried out to trade off low noise behaviour with high quantum efficiency. The optimal value is 28%. The thermal noise is inversely proportional to the  $g_{ds,sf}$ . To obtain a small thermal noise due to this transistor, the conductance must be large but in this case, the fill factor decreases. An optimisation must be realised the two parameters. To get the sensor time response as small as possible, it is important to design a large reset transistor conductance. For this, we make the reset transistor with the minimal size and we supply the gate voltage bigger than the drain. This condition allows to eliminate the light pulse effect in the acquisition phase when the some pixels are still in the reset phase.

#### **4. EXPERIMENTAL RESULTS**

The imager FAMOSI is composed of an array of 64 x 64 pixel sensors fabricated in 0.6  $\mu\text{m}$  AMS three metal CMOS process. In this section, noise measurements are presented and compared with the theoretical results. Figure 3 shows the measured output response for an input laser wavelength of 532 nm with a light pulse width of 6 ns at the middle of the input pulse. The laser pulse is uniformly illuminated on the entire array. On this figure, each line is a streak camera. The reconstituted information is presented in figure 4 and compared with the laser pulse achieved with a photodiode and a digital oscilloscope (input 50  $\Omega$ ). It is observed that the two pulses are approximately identical.

In a CMOS camera, the reset noise and the readout noise cannot be separated at the output. Figure 5 presents the readout noise in one pixel and the photon shot noise. The noise ( $kTC + \text{readout}$ ) in the dark is equal to 100 electrons. The photon shot noise, due to the statistical variation of the number of photons, varies in square root of the total photon flux. The results are  $144 \pm 5$  electrons for light power of  $2\mu\text{W}$  and  $251 \pm 10$  electrons for  $12\mu\text{W}$ . We show the noise source in pixel for an entire matrix (Figure 6), is in practice to  $95 \pm 4$  electrons with a leakage current of 5.4 fA. The calculated reset noise is equal to 84 electrons, the integration time noise, depending of the leakage current to 4 fA, is 36 electrons and a readout noise is 24

electrons. The total noise sources in pixel are 97 electrons. The experimental, simulation and calculated results are approximately the same. A random phenomenon, known as the fixed pattern noise (FPN), affects the output voltage on each pixel due to the variation of the size of each pixel components and process parameters when the imager is illuminated with a constant light. The effect of this noise source is shown in figure 7. Thanks to the experimental results obtained, we calculate the effect of the FPN of 2 % of saturation. The dynamic range of this circuit is to 64dB and the power consumption is to 25 mW for an acquisition rate of 50 frames per second. The conversion gain of  $6.73 \pm 0.25 \mu\text{V}/\text{electrons}$  for this version has been obtained. The measured results are presented in table 1.

## 5. CONCLUSIONS

The first temporal resolution imager using CMOS APS for laser detection has been described in this paper. A full analysis based on calculations and measurements for an active pixel camera has been performed. The advantages of this imager are its small size, low cost, high speed and low power consumption. Total noise sources are  $95 \pm 4$  electrons and a power consumption of 25 mW. For this chip, the fixed pattern noise of 2% and the conversion gain is  $6.73 \pm 0.25 \mu\text{V}/\text{electrons}$  for a fill factor of 28%. Moreover, the chip can perform the reconstitution of the optical information as a function of time. Because of these features, this imager can be widely used in biomedical, environmental and astronomy applications.

## APPENDIX 1: FLICKER NOISE SOURCES

To calculate the flicker noise at the input, the output power spectral density in current and in voltage is used. As the different gain of the system is determined, all of the noise sources can be obtained at the level of the sensor.

Noise during reset: During reset the flicker noise contribution is due to the reset transistor.

$$\overline{V_{n,\text{Mrst}}^2} = \frac{K_F I_{\text{PH}}^{A_F}}{L_{\text{rst}} W_{\text{rst}} C_{\text{ox}}} (\log[f] - \log[g_{m,\text{Mrst}} + 2\pi C_{\text{FN}} f]) \quad (14)$$

Noise during integration: During integration, the flicker noise source is due to the photodiode.

$$\overline{V_{n,FN}^2} = I_{PH}^{A_F} K_F \left[ \log [f] - \frac{1}{2} \log \left[ (g_{ds,Mrst} + g_{m,Mrst})^2 + (2\pi f C_{FN})^2 \right] \right] \quad (15)$$

Noise during readout: During readout, each transistor in pixel introduces flicker noise and the noise contribution of the current mirror source Mc is calculated (16) and the contribution of the others transistors in the imager is given by (17).

$$\overline{V_{n,Mc}^2} = \frac{R_t^2 K_F I_{ds}^{A_F}}{C_{ox} W_{Mc} L_{Mc}} \left[ \log [f] - \frac{1}{2} \log \left[ 1 + (2\pi R_t f C_L)^2 \right] \right] \quad (16)$$

$$\overline{V_{n,Mx}^2} = \frac{K_F}{C_{ox} L_x W_x} \left[ \log [f] - \frac{1}{2} \log \left[ 1 + (2\pi R_t f C_L)^2 \right] \right] \quad (17)$$

where  $A_F$  is the flicker noise exponent parameter and  $K_F$  is the flicker noise coefficient parameter,  $C_{ox}$ , the oxide capacitance,  $L_x$  and  $W_x$  the transistor size and  $f$  the frequency.  $R_t$  and  $C_L$  are described before. Then we can calculate the total equivalent flicker noise charge in electrons by the same way of (12) and we find for the frequency variation to 0.1 Hz up to 200 kHz (the readout frequency) the equivalent noise charge equal to 5 electrons.

## 6. REFERENCES

- [1] Hamamatsu - "Guide to Streak Cameras", *Technical information booklet* Hamamatsu Photonetics K. K.
- [2] Y. Reibel "Développement et Caractérisation d'une Caméra Vidéo Numérique Rapide (500i/s) a Haute résolution (10 bits). Application à la Reconstruction 3D de Surface Microscopique " Thèse année 2001.
- [3] Michael Schanz, Werner Brockherde, Ralf Hauschild, Bedrich J. Hosticka and Markus Schwarz, " Smart CMOS Image Sensor Arrays", *IEEE Transaction on Electron Devices*, vol. 44, no. 10, October 1997, pp. 1699-1705.
- [4] Sunetra K. Mendis, Sabrina E. Kemeny, Russell C. Gee, Bedabrata Pain, Craig O. Staller, Quiesup Kim and Eric R. Fossum, " CMOS Active Pixel Image Sensors for Highly Integrated Imaging Systems" *IEEE Journal of Solide State Circuits*, vol 32, no 2, February 1997 pp 187-197
- [5] Eric R. Fossum, "CMOS Image Sensor Electronic Camera On A Chip", *IEEE Transaction on Electron Devices*, vol. 44, no. 10, October 1997, pp. 1689-1698.
- [6] Steven Decker, R. Daniel McGrath, Kevin Brehmer, and Charles G. Sobini, "A 256\*256 CMOS Imaging Array with Wide Dynamic Pixels and Column Parallel Digital Output", *IEEE Journal of Solid State Circuits*, vol. 33, no 12, December 1998, pp. 2081-2091.

- [7] Joseph Calderer, Mauricio Moreno and Marco Braam "Integration of Phototransistor in CMOS Circuits", *Sensors and Materials*, vol. 8, no.4, 1996, pp 199-208.
- [8] Hui Tian, Boyl Folwer and Abbas El Gammal, "Analysis of Temporal Noise in CMOS Photodiode Active Pixel Sensor", *IEEE Journal of Solid State Circuits*, vol 36, no 1, January 2001 pp 92-101
- [9] Hui Tian and Abbas El Gammal, "Analysis of 1/f Noise in Switch MOSFET Circuits", *IEEE Transaction on Circuits and System I: Analogue and Digital Signal Processing*, vol 48, no 2, February 2001

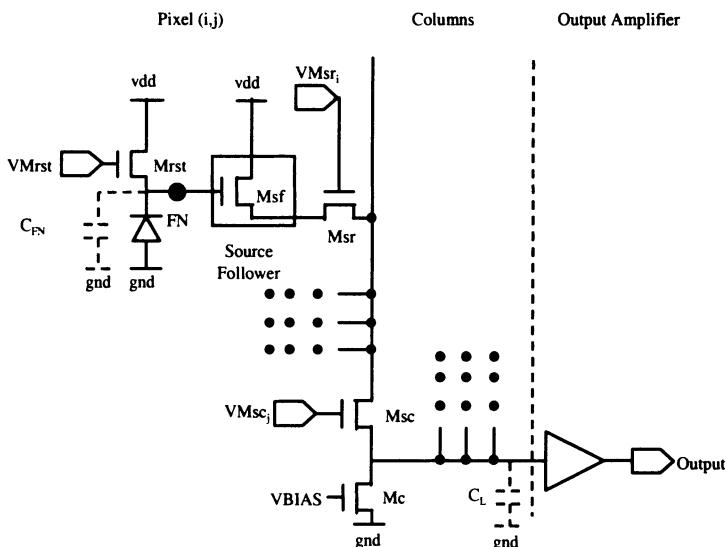


Figure 1. Imager architecture

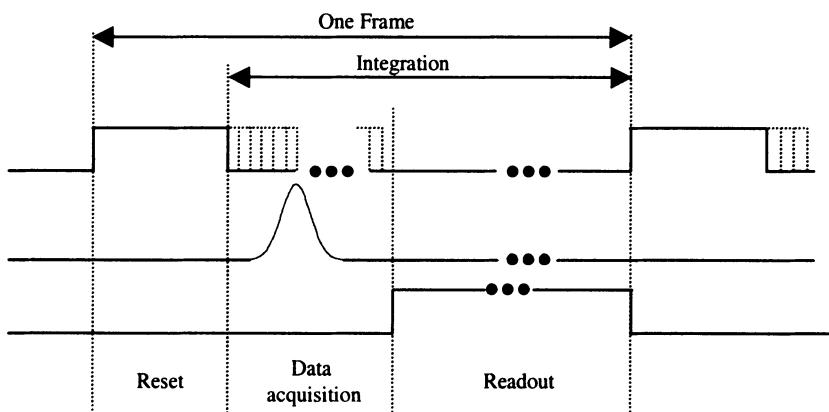
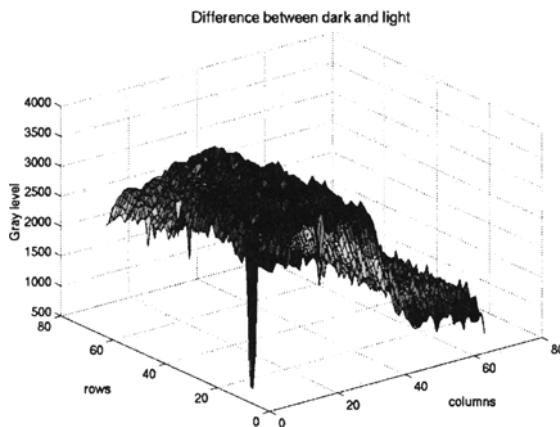
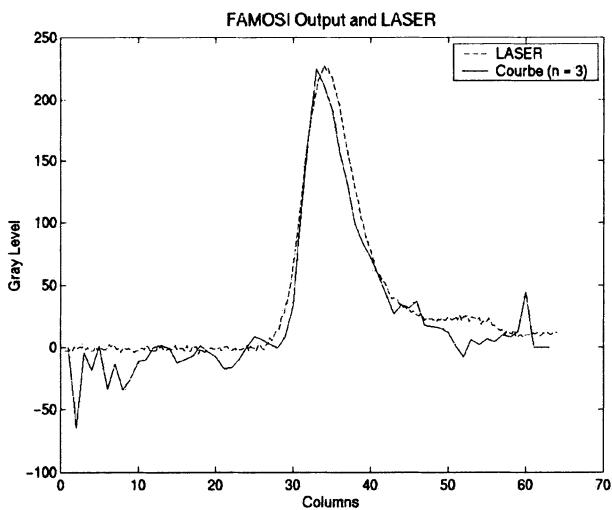


Figure 2. Timing diagram



*Figure 3.* Output response for the entire array



*Figure 4.* Reconstituted information

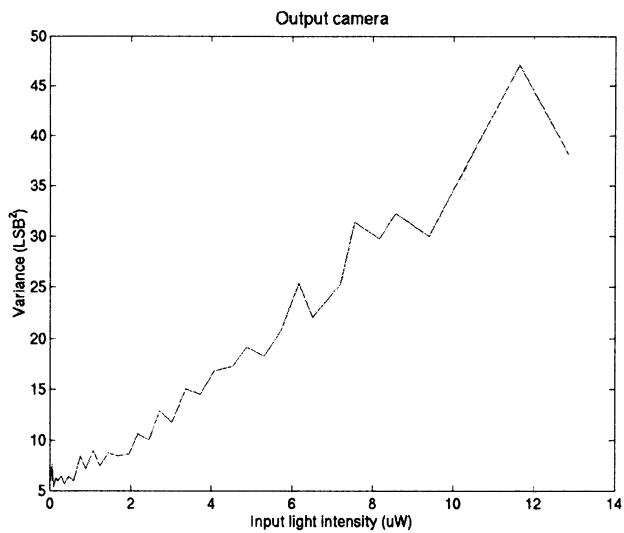


Figure 5. Photon shot noise for one pixel

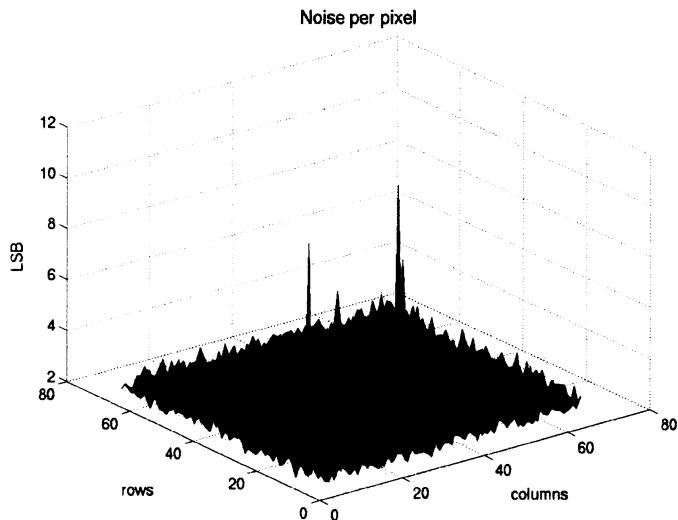
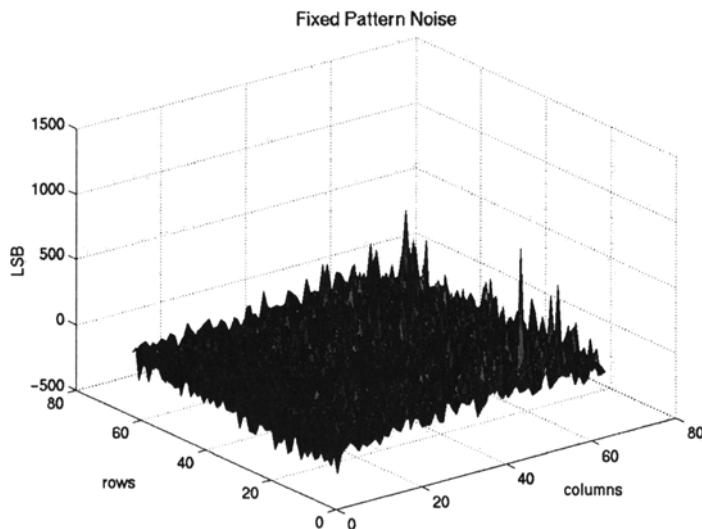


Figure 6. Readout noise



*Figure 7. Fixed pattern noise*

|                            |                                |
|----------------------------|--------------------------------|
| Conversion gain            | $6.73 \pm 0.25 \mu\text{V/e-}$ |
| Junction capacitance       | 24 fF                          |
| Input noise                | $95 \pm 4 \text{ e-}$          |
| FPN                        | 2%                             |
| Dynamic                    | 64 dB                          |
| Leakage current            | 5.4 fA                         |
| Power consumption          | 25 mW                          |
| Frequency data acquisition | 200 KHz                        |
| Rate                       | 50 frames/s                    |
| Time resolution            | 800ps                          |

*Table 1. Measured results*

# Noise optimisation of a piezoresistive CMOS MEMS for magnetic field sensing

Vincent Berouille, Yves Bertrand, Laurent Latorre, Pascal Nouet

*Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (France)*

Contact author : [latorre@lirmm.fr](mailto:latorre@lirmm.fr)

**Abstract:** Using 100% industrial fabrication processes, the design and the fabrication of a monolithic CMOS MEMS magnetic field sensor, targeting noise reduction, are developed in this paper. The sensor is based on a resonant cantilever structure with optimized electronics for signal treatment and noise filtering in order to achieve competitive performances. With a final sensitivity of  $350\text{V}_{\text{rms}}/\text{T}$  and a  $2\mu\text{T}$  resolution, the system is able to measure earth natural magnetic field.

**Key words:** CMOS MEMS, magnetic sensor, noise

## 1. INTRODUCTION

Magnetic field sensing devices regain interest since new applications have been demonstrated for automotive systems including navigation and mechanical parts (e.g. wheels) motion detection (ABS). Because magnetic sensing does not involve mechanical contact and wear, it offers the reliability level required by security concerns.

In such applications, major concerns are reliability, performances and cost. Monolithic integration of intelligent sensing devices using a standard microelectronic process (CMOS) provides significant cost reduction leading to high volume production. Also reliability is improved by reducing the number of wired connections. Regarding performances, it is obvious that the best magnetic field sensor will be achieved using dedicated processes (e.g. GMR [1]), however, the purpose of this paper is to show that using a CMOS

MEMS sensing device, on-chip signal treatment allows for overall performance improvement.

Electromechanical CMOS magnetic field sensors have been under investigation in our laboratory for several years [2]. The first part of the paper is dedicated to the electromechanical cell: the "U-Shape" cantilever. Sensing principle, and modeling are addressed with detail on the methodology we used to integrate the structure into a standard ASIC design flow.

The second part concerns an evaluation of the sensor performance, using a stand-alone cantilever device first, and with dedicated low-noise electronics for signal treatment. Sensitivity and resolution of overall magnetic field sensing system is finally given.

## 2. THE U-SHAPE CANTILEVER DEVICE

The "front-side bulk micromachining" (FSBM) technology is today fully industrialized and made available to fabless designers thought multi-project services for cheap prototyping [3]. This technology is based on a simple and low-cost post-process that etches selected parts of the CMOS die silicon substrate, releasing structures such as beams or membranes. Force sensing or thermal effects related applications might then be addressed using those structures.

From our point of view, monolithic MEMS is a standard mixed-signal ASIC that includes mechanical analog cells (beams, membranes...), the same way as it includes resistances or capacitors. Integrated system design is to be performed with standard EDA CAD tools including schematic capture, system level simulation and fabrication masks layout edition. Considering electromechanical blocs, this approach supposes that basic components have been previously characterized and translated into standard cells data with associated symbols, simulation models and layout synthesis helps.

Thus, in the following, the structure we used for magnetic field sensing is studied in order to provide the design environment with these required information.

### 2.1 Sensing principle

The proposed magnetic field sensor is based on the interaction between the unknown external magnetic field  $B$  to be measured and a known current  $I$  generated into the device itself (current loop).

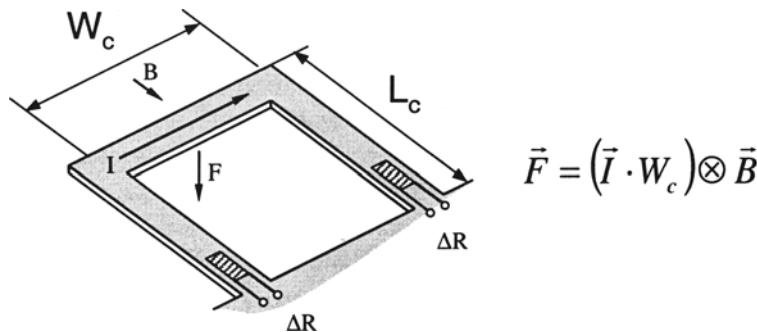


Figure 1: Magnetic field sensing principle.

The basic mechanical structure (Figure 1) consists in a U-shaped cantilever embedding an aluminum planar coil. The coil is supplied with an electrical current  $I$  that can be continuous (static mode) or alternative (dynamic mode). When an external magnetic field  $B$  is present, cantilever beam is therefore deflected due to the action of the Lorentz force  $F$  applied to its free side. Two polysilicon strain gauges are located into the mechanical structure close to anchor points in order to detect such deflections with a maximum sensitivity.

## 2.2 Modeling

As illustrated on figure 2, the sensor is a transducer that converts a force into a resistance variation (the same variation for the two gauges).

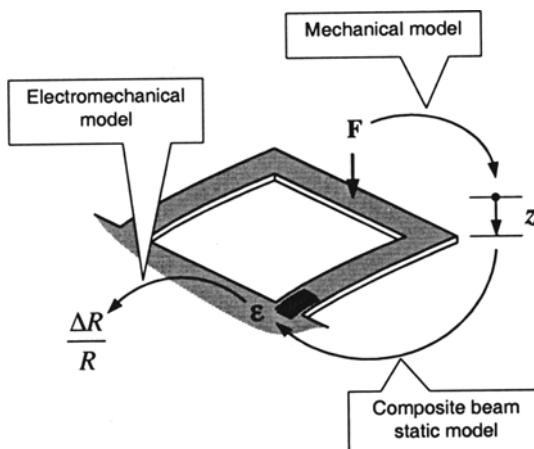


Figure 2: Modeling of the complete transducer.

Three basic models are used in series to mathematically represent this conversion:

- A mechanical model, that describes the relation between the applied force  $F$  and the vertical displacement of the cantilever  $z$ ,
- A structural model for the CMOS beam that takes into account the heterogeneous composition of the structure in order to provide a relation between the vertical displacement  $z$  and the strain applied to the gauge,
- And finally, an electromechanical model, that translates the gauge strain into a resistance variation.

Thanks to our previous work on characterization and analytical modeling [4], we have established that the associated mechanical model is a classical second order system. Table 1 reports differential equation and Laplace transfer function where  $k$ ,  $M$  and  $D$  represent respectively the stiffness, the mass and the damping factor of the structure:

|  |  |                                      |
|--|--|--------------------------------------|
| $M \frac{d^2 z}{dt^2} = -k.z - D \frac{dz}{dt} + F$          | $z = S_{stat} \times \frac{B}{1 + 2 \cdot \xi \frac{p}{\omega_0} + \frac{p^2}{\omega_0^2}}$  |                                      |
|  | (i)<br>$\omega_0 = \sqrt{\frac{k}{M}}$   | (ii)<br>$\xi = \frac{D}{2\sqrt{kM}}$ |
| Differential Equation of the mechanical second order system. | Laplace transfer function of the mechanical second order system.(i) cantilever natural pulsation $\omega_0$ .(ii) system damping coefficient $\xi$ |                                      |

Here, the vertical displacement is expressed as a function of the magnetic field  $B$  and the static sensitivity  $S_{stat}$  ( $\mu\text{m}/\text{T}$ ) given by:

$$S_{stat} = \frac{I \cdot W_c}{k}$$

The structure consists in an heterogeneous stack of various process layers, namely silicon oxides, metals, polysilicon and passivation nitride. In the composite beam model, the width of each layer is normalized regarding its young modulus. This operation transforms the heterogeneous beam into a equivalent homogeneous beam of equivalent Young's modulus  $E_n$ , from which we can calculate mechanical parameters such as the moment of inertia  $I_n$  and the vertical distance  $v$  between neutral axis and the gauges.

The stiffness  $k$  can be calculated from this model:

$$k = \frac{3 \cdot E_n \cdot I_n}{L_c^3}$$

Where  $L_c$  is the length of the anchored beams. Using strength of material theory, the strain in the gauge is given by:

$$\varepsilon = \frac{F \cdot L_c}{E_n \cdot I_n} = \frac{k \cdot z \cdot L_c}{E_n \cdot I_n} = \frac{3 \cdot z \cdot v}{L_c^2}$$

At last, piezoresistivity is a phenomenon that linearly links the strain and the material resistivity. As a result, the electromechanical model is simply:

$$\frac{\Delta R}{R} = Gf \times \varepsilon \quad \text{Where } Gf \text{ is called the polysilicon gauge factor.}$$

### 3. SYSTEM LEVEL INTEGRATION

#### 3.1 Stand-alone cantilever performance

While sensitivity can be useful metric, the real parameter of interest in most application is resolution. We discuss, in this part, about the minimum magnetic induction  $B$  that can be measured.

Piezoresistive cantilevers suffer from an unexpected 1/f noise [5]. Unfortunately, 1/f noise in resistors from AMS process are not characterized, and corresponding spice equation is empty. So, in the following, bandwidths avoid low frequencies where 1/f noise can not be neglected.

White noise spectral density is constant over frequency, given for a resistor  $R$  about  $1.7\text{k}\Omega$  by :

$$V_{noise}^2(f) = 4k_b T R \quad V_{noise,rms}(f) = 5.3 \text{nV}/\sqrt{\text{Hz}}$$

Where  $k_b$  is the Boltzmann's constant,  $T$  is the temperature in Kelvin and  $R$  is the resistance value in Ohms.

Two constant resistors and two gauges, equal to the same value  $R$ , are actually connected together in a Wheatstone bridge circuit that compensates for temperature variation. It gives a total output noise equivalent to the previous single resistor noise level.

The noise spectral density at the output of a magnetic sensor may be interpreted as a result of an noise equivalent magnetic induction (NEMI), acting on a noiseless device [6]. The following curve (figure 3) reports this NEMI spectral density due to White noise in resistors.

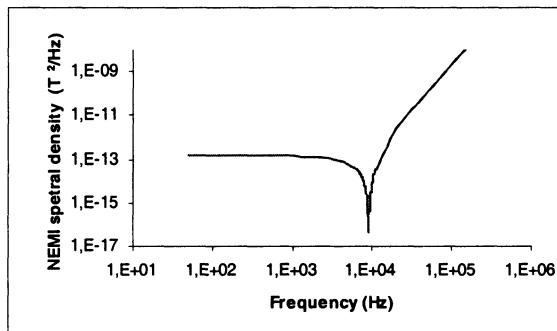


Figure 3: NEMI spectral density due to White noise in Wheatstone bridge.

Our sensor is a electro-mechanical component, thus mechanical noise must be considered. This noise is a thermal noise which can be described with a single relation giving force noise  $F_m$  acting on the cantilever:

$$F_m = \sqrt{4k_b T D} \quad \text{With damping, } D=7.19 \cdot 10^{-7} \text{ kg.s}^{-1}$$

Then, this force conduces to the NEMI spectral density [6]:

$$S_{B_n/U_{mec}}(f) = \frac{4k_b T D}{(W.I)^2}$$

In our application,  $S_{Bn/U_{mec}}(f)$  is about  $10^{-18}$ , thus, mechanical noise can be neglected.

### 3.2 Signal processing

Looking at results of figure 3, it is obvious that the best sensor performances in terms of sensitivity and resolution will be achieved using the resonant mode. The signal is then mechanically filtered and the signal

bandwidth is restricted to the  $-3\text{dB}$  mechanical bandwidth around resonant frequency. This bandwidth is in fact only few hundreds hertz large.

It is a general rule to optimize noise performance that one should not design circuits for larger bandwidth than the signal requires. We thus designed amplification circuit and a filter in order to minimize total noise RMS value accordingly with the already restricted signal bandwidth. A schematic of the integrated circuit is presented on figure 4.

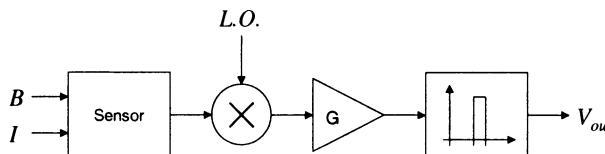


Figure 4: Schematic of the measuring system.

In this circuit, amplification stage and bandpass filter are preceded by a multiplier, which is used to shift the amplifier input signal in the frequency domain. If the cantilever is used in resonant mode, this feature is required for adjustment purpose as long as it is difficult to precisely predict both sensor natural frequency and filter cutoff frequencies. In the case of a static actuation of the cantilever, the multiplier shall be used to shift the signal from DC to higher frequencies where the  $1/f$  noise of amplifier becomes negligible (from the lock-in principle).

Any circuit located before amplification is under major concern concerning noise injection. For this reason, noisy operational amplifiers have been avoided in the design of the mixer. Figure 5 shows a schematic view of the Gilbert's multiplier that we used instead:

Equivalent input noise in this structure strongly depends on the differential pair transistors length. In order to find the optimal length for those transistors, a parametric noise analysis has been performed (figure 6). Because the transistors length also affects the multiplier gain ( $A$  decreases when  $L$  increases) there is no benefit regarding equivalent input noise in increasing the length too much. Finally,  $L$  has been fixed to  $5\mu\text{m}$ , resulting in a mixer gain  $A$  about 10. The layout of the integrated Gilbert's cell is presented on figure 7.

Amplification and filtering use classic, operational amplifier based, circuits. The filter is a second order band-pass filter with a cutoff frequency of  $25\text{kHz}$  and a  $3\text{dB}$  bandwidth of  $3\text{kHz}$ .

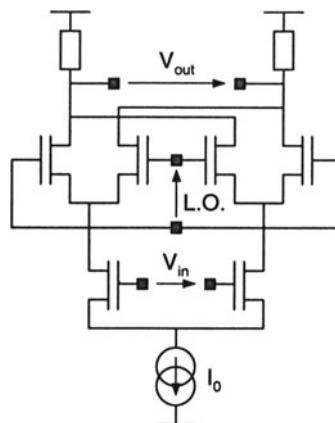


Figure 5: Schematic of the Gilbert's multiplier.

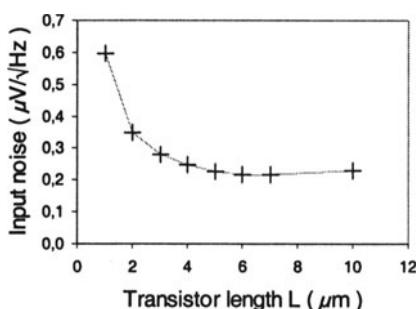


Figure 6: Equivalent input noise as a function of the differential stage transistors length.

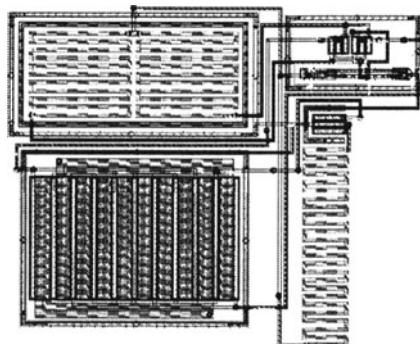


Figure 7: Layout of the multiplier

### 3.3 Simulations

In order to include the mechanical device in the design environment, the complete analytical model of the U-Shaped cantilever has been translated in Analog HDL language that authorizes simulations in a microelectronic designer framework.

This model, which is attached to the mechanical transducer, is used by analog electronic circuits simulation such as Spice. Real numbers are manipulated during the simulation regardless their physical significance. Doing this, a voltage generator can be used to generate any physical stimuli (e.g. a magnetic field) and electrical measurement (voltage or current) might represent any physical information such as displacement or speed.

In figure 8, a magnetic field step from 0 to 1mT is applied to the sensor at  $t=0$ . From top to bottom the figure displays the following waveforms: (i) The “force input” which is the voltage source applied across the planar coil, (ii) the local oscillator control signal, (iii) the vertical displacement of the U-Shape in microns, (iv) the amplifier output signal and, (v) the whole system output. Spectrums have been calculated using a time window far enough from the simulation beginning, so that permanent regime has been reached.

In order to shift the 9kHz cantilever resonant frequency to the 25kHz filter central frequency, the local oscillator signal frequency has been fixed at 16kHz. Looking at FFT's the multiplication produces the awaited frequency shift and the filter removes the undesired 6kHz peak.

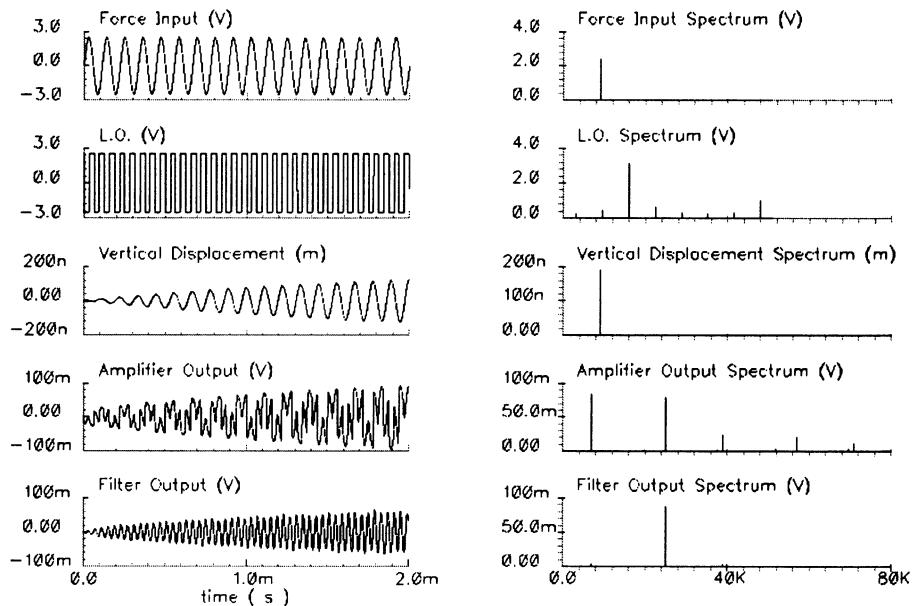


Figure 8: Example of system level simulations.

### 3.4 Noise performance analysis

Figure 9 shows noise simulation results in terms of noise power spectral density for the complete signal processing circuit including the sensor gauges, the multiplier, the amplifier and the filter. However, as long as non-linear circuit are not completely supported for noise analysis, the L.O. signal has been fixed to a constant level, transforming the multiplier into a simple amplification stage with a gain of 10. Doing this, the multiplier input noise is not shifted in the frequency domain. This operation does not affect the result

regarding thermal noise into the sensor gauges since its noise power spectral density is constant over the frequency range.

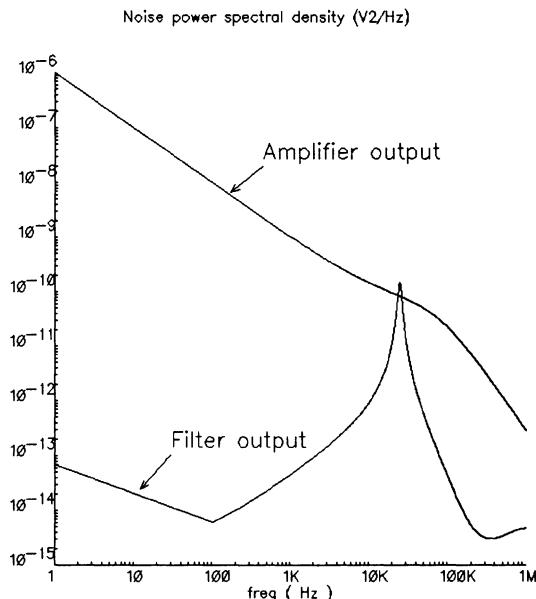


Figure 9: Noise power spectral density  
before and after filtering

According to simulation results, the total noise level is about 8mVrms before filtering and  $820\mu\text{V}$  rms when filtered with a 3kHz bandwidth.

Since total amplification factor with mixer and square local oscillator is 660 , the sensitivity of the resonant sensor becomes 350Vrms/T with a  $2\mu\text{T}$  resolution.

### 3.5 System fabrication

The system has been fabricated using  $0.6\mu\text{m}$  CMOS technology from AMS [7]. The figure 10 presents the circuit after CMOS process. The gray square around the U-Shape device is the bare silicon defined during the mask design step by stacking all the oxides openings (diffusion, contact, via and pad). After CMOS fabrication, dies are anisotropically wet etched in a TMAH solution, releasing the cantilever as shown on figure 11.

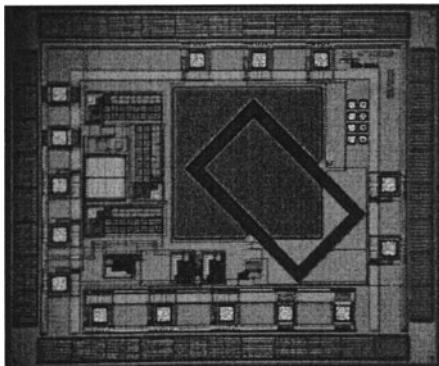


Figure 10: The sensing system after CMOS process.

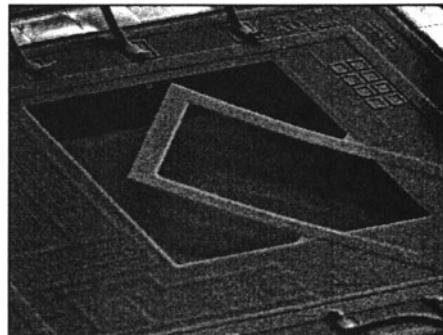


Figure 11: The sensing system after FSBM post-process.

#### 4. CONCLUSION

An advantage of monolithic integration is the possibility to add electronic functions without significant increase of the total system cost. The U-Shape cantilever performance in the magnetometer application does not reach the performance level of sensors built on dedicated technologies in term of sensitivity. Although, the use of polysilicon resistances gauges in a Wheatstone bridge structure offers many advantages such:

- Effect of temperature on the gauge is cancelled,
- Noise power spectral density in resistances is low and good resolutions are achievable with appropriate signal processing,
- It is easy to interface with electronics.

After amplification and noise filtering, the sensor resolution has been taken to  $2\mu\text{T}$ , giving the system the ability to measure earth magnetic field.

Further improvements of the signal processing circuit will implement oversampling techniques in order to reduce noise level.

## 5. REFERENCES

- [1] G.Rieger, K. Ludwig, J. Hauch, W. Clemens, "GMR sensors for contactless position detection", Sensors and Actuators A 91 (2001) 7-11.
- [2a] L. Latorre, Y. Bertrand, P. Nouet, "On the use of Test Structures for the Electromechanical characterization of a CMOS compatible MEMS technology", Proc. IEEE 1998 Int. Conf. On Microelectronic Test Structures, March, 1998.
- [2b] L. Latorre, P. Nouet, Y. Bertrand, P. Hazard and F.Pressecq, "Characterization and modeling of a CMOS-compatible MEMS technology", Sensors and Actuators, Vol. 74, 1999, pp. 143-147.
- [2c] L.Latorre, V. Berouille, Y. Bertrand, I. Salesse, P. Nouet, « Electro-Mechanical Magnetic Field sensors in a CMOS Technology », *Proc. of DCIS 2000, pp694-699, 15<sup>th</sup> Design of Circuits and Integrated Systems Conference, Montpellier, Le Corum, France, November 21-24, 2000*.
- [2d] V. Berouille, Y. Bertrand, L. Latorre, P. Nouet, « Micromachined CMOS Magnetic Field Sensors with Low-Noise Signal Conditioning », *15<sup>th</sup> IEEE INTERNATIONAL MICRO ELECTRO-MECHANICAL SYSTEMS CONFERENCE, January 20-24, 2002, Las Vegas, Nevada, USA*.
- [3] Circuits Multi-projets (CMP) is a broker in Ics, MCMs and MEMS. See <http://cmp.imag.fr/>
- [4] L. Latorre and al., "Characterization and modeling of a CMOS-compatible MEMS technology", Sensors and Actuators, Vol. 74, 1999, pp. 143-147.
- [5] Jonah A. Harley, "Advances in piezoresistive probes for atomic force microscopy", Dissertation of Ph'D degree, march 2000.
- [6] R. S. Popovic, « Hall effect Devices, Magnetic Sensors and Characterization of semiconductors », The Adam Hilger Series On Sensors, Adam Hilger, ISBN 0-7503-0096-5, 1991
- [7] See <http://www.ams.co.at>

## Authors Index

|                    |          |                     |           |
|--------------------|----------|---------------------|-----------|
| Albonesi, D.       | 289      | Chillet, D.         | 51        |
| Aline, M.          | 325      | Choi, J. H.         | 337       |
| Araujo, C.         | 145      | Ciesielski, M.      | 375       |
| Ascia, G.          | 157      | Cunin, B.           | 449       |
| Auguin, M.         | 217      | Curran, B.          | 289       |
| Auvergne, D.       | 301, 325 | Dardalhon, M.       | 241       |
| Azaïs, F.          | 425      | David, Ra.          | 51        |
| Azemard, N.        | 301, 325 | David, Re.          | 413       |
| Bampi, S.          | 337      | De Mello, B. A.     | 181       |
| Barros, E.         | 145      | Demigny, D.         | 3, 39, 75 |
| Becker, J.         | 97       | Drechsler, R.       | 361       |
| Belleudy, C.       | 217      | Dutertre, J.M.      | 229       |
| Benoit, P.         | 63       | Fesquet, L.         | 313       |
| Bernard, D.        | 133      | Flandre, D.         | 169       |
| Bernard, S.        | 425      | Flottes, M-L.       | 401       |
| Berouille, V.      | 241, 461 | Galy, J.            | 63        |
| Bertrand, Y.       | 425, 461 | Gebotys, C.H.       | 205       |
| Boudouani, N.      | 75       | Gharsalli, F.       | 193       |
| Bourguiba, R.      | 75       | Gifaldi, M.         | 289       |
| Buyuktosunoglu, A. | 289      | Girard, P.          | 413       |
| Cambon, G.         | 63       | Glesner, M.         | 97        |
| Camposano, R.      | 87       | Guitton-Ouhamou, P. | 217       |
| Carro, L.          | 437      | Hervé, Y.           | 349       |
| Casadei, B.        | 449      | Hu, Y.              | 449       |
| Catania, V.        | 157      | Indrusiak, L. S.    | 97        |
| Cathebras, G.      | 229      | Ishikawa, M.        | 15        |

|                   |               |                      |          |
|-------------------|---------------|----------------------|----------|
| Itoh, K.          | 277           | Palesi, M.           | 157      |
| Janner, D.        | 437           | Pêcheux, F.          | 349      |
| Jerraya, A. A.    | 193           | Perez, G.            | 241      |
| Johannsen, P.     | 361           | Pillement, S.        | 51       |
| Karabernou, M.    | 3, 75         | Pisuk, S. M.         | 265      |
| Kessal, L.        | 3, 39, 75     | Pons, J.             | 39       |
| Kimura, M.        | 109           | Pouget, J.           | 401      |
| Komuro, T.        | 15            | Pravossoudovitch, S. | 413      |
| Lallement, C.     | 349           | Pressecq, F.         | 241      |
| Lamaty, P.        | 3             | Quartana, J.         | 313      |
| Landrault, C.     | 133, 413      | Reis, R.             | 97       |
| Latorre, L.       | 241, 461      | Renaudin, M.         | 313      |
| Le Normand, J. P. | 449           | Renovell, M.         | 425      |
| Lubaszewski, M.   | 437           | Rigaud, J-B.         | 313      |
| MacMillen, D.     | 87            | Robert, M.           | 63       |
| Margala, M.       | 289           | Roche, F.M.          | 229      |
| Marinissen, E. J. | 389           | Roma, N.             | 253      |
| Martin, E.        | 27            | Rousseau, F.         | 193      |
| Martin, J.        | 289           | Rouzeyre, B.         | 375, 401 |
| Maurine, P.       | 301, 325      | Sassatelli, G.       | 63       |
| Mazar, B.         | 3             | Senn, E.             | 27       |
| Meftali, S.       | 193           | Sentieys, O.         | 51       |
| Miki, M. H.       | 109           | Shirakawa, I.        | 109      |
| Mizuno, H.        | 277           | Sousa, L.            | 253      |
| Muresan, R.       | 205           | Torres, L.           | 63       |
| Nácul, A. C.      | 437           | Virazel, A.          | 413      |
| Nève, A.          | 169           | Wagner, F. R.        | 121, 181 |
| Nouet, P.         | 133, 241, 461 | Wu, P. H.            | 265      |
| Onoye, T.         | 109           | Zeng, Z.             | 375      |
| Otero, J. C.      | 121           |                      |          |
| Oudea, C.         | 241           |                      |          |

# Keywords Index

## A

|                                |     |
|--------------------------------|-----|
| Active Pixel Sensor            | 27  |
| ADC testing                    | 425 |
| Analog and Mixed-Signal Design | 337 |
| Analog built-in testing        | 437 |
| Analog stimulus generation     | 425 |
| Architecture                   | 289 |
| Architecture refinement        | 193 |
| Array Architectures            | 253 |
| ASIC                           | 3   |
| Asynchronous systems           | 313 |

## B

|                 |          |
|-----------------|----------|
| BIST            | 413, 425 |
| Block Matching  | 253      |
| Bridging Fault  | 413      |
| Building Blocks | 337      |

## C

|                  |     |
|------------------|-----|
| Capacitance      | 133 |
| Characterization | 241 |
| CHP language     | 313 |

Closed-form models 133

CMOS 289

CMOS MEMS 461

CMOS sensors 449

Code transformation 193

Constraint logic

programming 375

Cooperation 181

Current 205

## D

Data flow 63

Data Latch 229

Delay bounds 325

Delay Fault 413

Design environments 121

Design space exploration 157

Digital design 169

Digital Signal Processing 63

Distributed co-simulation 181

DRAM 277

DSP 205

DSP processor 217

Dynamic reconfiguration 75

Dynamic VT 277

**E**

|                 |     |
|-----------------|-----|
| Edge chaining   | 3   |
| Edge detection  | 3   |
| Embedded        | 289 |
| Embedded system | 109 |
| Energy          | 205 |

**F**

|                            |     |
|----------------------------|-----|
| Fan out                    | 325 |
| FPGA                       | 75  |
| Functional test generation | 375 |

**G**

|                        |     |
|------------------------|-----|
| Gain cells             | 277 |
| Gate sizing            | 301 |
| Gate tunneling current | 277 |
| Gate-source backbias   | 277 |
| Gaussian filter        | 39  |
| Genetic algorithms     | 157 |

**H**

|                              |     |
|------------------------------|-----|
| Hardware engine              | 109 |
| High time resolution imager. | 449 |
| High-level modeling          | 313 |
| High-Performance             | 51  |
| HLA                          | 181 |

**I**

|                    |       |
|--------------------|-------|
| IDQD test          | 277   |
| Image processing   | 3, 75 |
| Image sensor       | 15    |
| Industrial Control | 27    |
| Interconnects      | 133   |
| Issue queue        | 289   |

**J**

|      |     |
|------|-----|
| Java | 109 |
|------|-----|

**L**

|                              |     |
|------------------------------|-----|
| Layout Extraction            | 133 |
| Low Voltage                  | 289 |
| Low Voltage Low Power design | 169 |
| Low-Power                    | 51  |

**M**

|                              |     |
|------------------------------|-----|
| Macro Test                   | 389 |
| Magnetic sensor              | 461 |
| Manufacturing test           | 389 |
| Memory-rich architectures    | 277 |
| MEMS                         | 241 |
| Model                        | 205 |
| Motion Detection             | 27  |
| Motion Estimation            | 253 |
| Multi-objective optimization | 157 |
| Multi-static VT              | 277 |

**N**

|                                  |     |
|----------------------------------|-----|
| Noise                            | 461 |
| Noise analysis and optimisation, | 449 |
| Non linear adaptive filters      | 437 |
| Non linear circuits              | 437 |

**O**

|                                   |     |
|-----------------------------------|-----|
| Object-oriented hardware modeling | 121 |
|-----------------------------------|-----|

**P**

|                               |     |
|-------------------------------|-----|
| Parallel processing           | 15  |
| Parameterized systems,        | 157 |
| Pareto-optimal configurations | 157 |
| Pattern Recognition           | 27  |
| Peripheral circuits           | 277 |

|                             |     |                                   |          |
|-----------------------------|-----|-----------------------------------|----------|
| Polygonal approximation     | 3   | Single Event Upset                | 229      |
| Polynomial filter           | 39  | Software kernel                   | 109      |
| Power                       | 205 | SOI Technology                    | 169      |
| Power adaptative            | 289 | Specialized Processors            | 253      |
| Power consumption model     | 217 | SRAM                              | 277      |
| Power dissipation           | 301 | Stuck-at Fault                    | 413      |
| Power management            | 277 | Subthreshold current              | 277      |
| Power/performance-tradeoffs | 157 | System chip                       | 389      |
| Pre-Diffused SOT array      | 337 | System-on-a-chip<br>architectures | 157      |
| Priority arbiters           | 313 | System-on-Chip                    | 401      |
| Process Scatterings         | 241 |                                   |          |
| Processor architectures     | 121 |                                   |          |
| <b>T</b>                    |     |                                   |          |
| QDI circuits                | 313 | TAT transistor                    | 337      |
| <b>R</b>                    |     |                                   |          |
| Radiation Hardened Design   | 229 | Teaching environments             | 121      |
| Reconfigurable Architecture | 51  | Test access mechanism             | 389      |
| Reconfigurable computing    | 63  | Test expansion                    | 389      |
| Recursive filter            | 39  | Test scheduling                   | 389, 401 |
| Region labelling            | 3   | Test wrapper                      | 389      |
| Reliability                 | 229 | Timing closure                    | 325      |
| Robustness,                 | 229 | <b>U</b>                          |          |
| Run time reconfiguration    | 75  | UMTS                              | 51       |
| <b>S</b>                    |     |                                   |          |
| Satisfiability              | 375 | Universal Sequence                | 413      |
| Shared memory               | 193 | <b>V</b>                          |          |
| Short circuit               | 301 | Validation                        | 375      |
| Simulation backbone         | 181 | Verification                      | 375      |
|                             |     | Vision chip                       | 15       |
|                             |     | Vision System On Chip             | 27       |
|                             |     | Visual feedback                   | 15       |