IFIP AICT 313

Christian Piguet
Ricardo Reis
Dimitrios Soudris
(Eds.)

VLSI-SoC: Design Methodologies for SoC and SiP

16th IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2008
Rhodes Island, Greece, October 2008
Revised Selected Papers

Springer

IFIP Advances in Information and Communication Technology 313

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.

Christian Piguet Ricardo Reis
Dimitrios Soudris (Eds.)

VLSI-SoC: Design Methodologies for SoC and SiP

16th IFIP WG 10.5/IEEE International Conference
on Very Large Scale Integration, VLSI-SoC 2008
Rhodes Island, Greece, October 13-15, 2008
Revised Selected Papers

Springer

Volume Editors

Christian Piguet
CSEM, Centre Suisse d'Electronique et de Microtechnique
Jaquet-Droz 1, Case Postale, 2002 Neuchâtel, Switzerland
E-mail: christian.piguet@csem.ch

Ricardo Reis
Universidade Federal do Rio Grande do Sul, Instituto de Informática
Porto Alegre, Brazil
E-mail: reis@inf.ufrgs.br

Dimitrios Soudris
National Technical University of Athens, Department of Computer Science
9 Heroon Polytechneiou, Zographou Campus, 15780 Athens, Greece
E-mail: dsoudris@microlab.ntua.gr

Library of Congress Control Number: 2010923483

CR Subject Classification (1998): B.7-8, C.0, F.2, J.2-3, C.2.1

ISSN	1868-4238
ISBN-10	3-642-12266-3 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-12266-8 Springer Berlin Heidelberg New York

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

This book contains extended and revised versions of the best papers that were presented during the 16th edition of the IFIP/IEEE WG10.5 International Conference on Very Large Scale Integration, a global System-on-a-Chip Design & CAD conference. The 16th conference was held at the Grand Hotel of Rhodes Island, Greece (October 13–15, 2008). Previous conferences have taken place in Edinburgh, Trondheim, Vancouver, Munich, Grenoble, Tokyo, Gramado, Lisbon, Montpellier, Darmstadt, Perth, Nice and Atlanta.

VLSI-SoC 2008 was the 16th in a series of international conferences sponsored by IFIP TC 10 Working Group 10.5 and IEEE CEDA that explores the state of the art and the new developments in the field of VLSI systems and their designs. The purpose of the conference was to provide a forum to exchange ideas and to present industrial and research results in the fields of VLSI/ULSI systems, embedded systems and microelectronic design and test.

The 2008 edition of VLSI-SoC maintained the traditional structure of the previous VLSI-SoC conferences. The quality of submissions (193 papers) made the selection process difficult, but finally 56 full papers and 42 posters were accepted for presentation at VLSI-SoC 2008. Out of the 56 full papers presented at the conference, 14 regular papers were chosen by a selection committee to have an extended and revised version included in this book. These selected papers have authors from France, Germany, Italy, Greece, and Switzerland. Additionally, the selection committee invited Eby Friedman, Rochester University, USA, keynote speaker at VLSI-SOC 2008, to contribute a special chapter about "3-D Integrated Technologies".

VLSI-SoC 2008 was the culmination of many dedicated volunteers: paper authors, reviewers, session chairs, invited speakers and various committee chairs, especially the local arrangements organizers. Also, special thanks to the VLSI-SOC 2008 sponsors. We thank them all for their contribution.

This book is intended for the entire VLSI community and in particular those who did not have a chance to take part in the VLSI-SoC 2008 Conference. The selected papers cover a wide variety of excellence in VLSI technology and describe advanced research in the area. We hope that the reader (professional, instructor, engineer, student, etc.) will find the book useful, constructive and enjoyable, and that the technical material presented will contribute to the continued progress of the VLSI community as a whole.

July 2009

Christian Piguet
Ricardo Reis
Dimitrios Soudris

Organization

The IFIP/IEEE International Conference on Very Large Scale Integration-System-on-Chip (VLSI-SoC) 2008 took place during October 13−15, 2008 in the Grand Hotel Rhodes, on Rhodes, Greece. VLSI-SoC 2008 was the 16th in a series of international conferences, sponsored by IFIP TC 10 Working Group 10.5 (VLSI) and IEEE CEDA.

General Chair

Dimitrios Soudris*	Democritus University of Thrace, Greece
	*National Technical University of Athens, Greece (from 08/2008)

Program Co-chairs

Christian Piguet	Centre Suisse d'Electronique et de Microtechnique, Switzerland
Thanos Stouraitis	University of Patras, Greece

Publicity Co-chairs

David Atienza	Complutense University of Madrid, Spain
Bernard Courtois	TIMA Labs, France

PhD Forum Co-chairs

Josef Haid	Infineon Technologies, Austria
Bernard Courtois	University of Patras, Greece

Keynote Speakers

O. Koufopavlou	University of Patras, Greece

Tutorials

V. Paliouras	University of Patras, Greece

Special Sessions

K. Pekmestzi	National Technical University of Athens, Greece
G. Economakos	National Technical University of Athens, Greece

Proceedings

Ricardo Reis	Universidade Federal do Rio Grande do Sul, Brazil
S. Nikolaidis	Aristotle University of Thessaloniki, Greece

Finance-Local Arrangements

G. Theodoridis	Aristotle University of Thessaloniki, Greece

Conference Secretariat/Web Design

Christos Baloukas	Democritus University of Thrace, Greece

Steering Committee

Manfred Glesner	Technische Universität Darmstadt, Germany
Salvador Mir	TIMA Labs, France
Ricardo Reis	Universidade Federal do Rio Grande do Sul, Brazil
Michel Robert	Université Montpellier, France
Luis Miguel Silveira	Instituto de Engenharia de Sistemas e Computadores Investigação, Portugal

Technical Program Committee

Amara Amara	Institut Supérieur d'Electronique de Paris, France
Ioannis Andreadis	Democritus University of Thrace, Greece
Federico Angiolini	Bologna University, Italy
Antonios Argyriou	Philips, The Netherlands
Nadine Azermad	Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, France
Magdy Bayoumi	University of Louisiana at Lafayette, USA
Juergen Becker	Universität Karlsruhe, Germany
Cécile Belleudy	University Sophia-Antipolis Nice, France
Mladen Berekovic	Technische Universität Braunschweig, Germany
Swarup Bhunia	Case Western Reserve University, USA
Holger Blume	Rheinisch-Westfälische Technische Hochschule Aachen, Germany
Joao Cardoso	Instituto de Engenharia de Sistemas e Computadores Investigação, Portugal
Wim Dehaene	Katholieke Universiteit Leuven, Belgium
Yunsi Fei	University of Connecticut, USA
Joan Figueras	Universitat Politecnica de Catalunya, Spain
Georgi N. Gaydadjiev	Delft University of Technology, The Netherlands
Dimitris Gizopoulos	University of Piraeus, Greece
Carlo Guardiani	PDF Solutions, Inc., Italy

Frank Kagan Gurkaynak	Ecole Polytechnique Fédérale de Lausanne, Switzerland
Josef Haid	Infineon Technologies, Austria
Alkis Hatzopoulos	Aristotle University of Thessaloniki, Greece
Domenik Helms	OFFIS, Germany
Ahmed Hemani	KTH - Royal Institute of Technology, Sweden
Tang Hua	University of Minnesota, USA
Nathalie Julien	Université de Bretagne-Sud, France
Srinivas Katkoori	University of South Florida, USA
Avinoam Kolodny	Technion-Israel Institute of Technology, Israel
Hsien-Hsin S. Lee	Georgia Institute of Technology, USA
Jean-Didier Legat	University of Louvain-la-Neuve, Belgium
Yung-Hsiang Lu	University of Purdue, USA
Alberto Macii	Politecnico di Torino, Italy
Stylianos Mamagkakis	IMEC, Belgium
Salvador Mir	Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, France
Vincent J. Mooney III	Georgia Institute of Technology, USA
Srinivasan Murali	Ecole Polytechnique Fédérale de Lausanne, Switzerland
Alex Orailoglu	University of San Diego, USA
Vassilis Paliouras	University of Patras, Greece
Marios Papaefthymiou	University of Michigan, USA
Antonis Paschalis	University of Athens, Greece
Zebo Peng	Linkoping University, Sweden
Dionisios N. Pnevmatikatos	Technical University of Crete, Greece
Massimo Poncino	Politecnico di Torino, Italy
Ricardo Reis	Universidade Federal do Rio Grande do Sul, Brazil
Marcos Sanchez-Elez	Complutense University of Madrid, Spain
Dimitrios Serpanos	University of Patras, Greece
Cristina Silvano	Politecnico di Milano, Italy
Stylianos Siskos	Aristotle University of Thessaloniki, Greece
Thanos Skodras	Hellenic Open University, Greece
Konstantinos Tatas	Frederick University, Cyprus
Yiorgos Tsiatouhas	University of Ioannina, Greece
Dimitris Velenis	University of Rochester, USA
Flavio R. Wagner	Universidade Federal do Rio Grande do Sul, Brazil
Miroslav Velev	University of Illinois, Chicago, USA
Shiyan Hu	Michigan Technological University, USA

Table of Contents

Physical Design Issues in 3-D Integrated Technologies

Vasilis F. Pavlidis[1] and Eby G. Friedman[2]

[1] LSI EPFL, 1015 Lausanne, Switzerland
[2] Department of Electrical and Computer Engineering, University of Rochester, Rochester, New York 14627, USA
vasileios.pavlidis@epfl.ch, friedman@ece.rochester.edu

Abstract. Design techniques for three-dimensional (3-D) ICs considerably lag the significant strides achieved in 3-D manufacturing technologies. Advanced design methodologies for 2-D circuits are not sufficient to manage the added complexity caused by the third dimension. Consequently, design methodologies that efficiently handle the added complexity and inherent heterogeneity of 3-D circuits are necessary. These 3-D design methodologies should support robust and reliable 3-D circuits, while considering different forms of vertical integration, such as systems-in-package and 3-D ICs with fine grain vertical interconnections. The techniques described in this chapter address important physical design issues and fundamental interconnect structures in the 3-D design process.

1 Introduction

Technology scaling and CMOS technologies have steadily supported an increase in the performance of integrated circuits (ICs) over the past several decades. These driving forces are expected, however, to lose momentum as the fabrication of nanoscale devices at gigascale densities become increasingly difficult and economically infeasible [1]. Three-dimensional (3-D) integration is a novel design paradigm with great potential to fundamentally advance the computational power and functionality of modern integrated systems [2].

The inherent advantage of 3-D integration is the drastic decrease in interconnect length, particularly the long global interconnects, which directly results in increased speed [3], [4], [5]. The interconnect power is also reduced as the capacitance of the wires decreases [6]. Another characteristic of 3-D ICs of even greater importance than the decrease in the interconnect length is the ability of these systems to include disparate technologies, greatly extending the capabilities of modern systems-on-chip (SoC) [7].

This defining feature of 3-D ICs offers unique opportunities for highly heterogeneous and sophisticated systems [8]. This heterogeneity, however, greatly complicates the interconnect design process within a multi-plane system, as potential design methodologies need to manage the diverse interconnect impedance characteristics and process variations caused by the different fabrication processes and technologies employed in the multiple physical planes [9]. Additional primary challenges in 3-D circuits include the development of methodologies at the front end

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 1–21, 2010.

of the design process [10], [11], multi-plane functional testing [12], thermal management techniques [13], and maturing manufacturing technologies [14].

Physical and interconnect design techniques for 3-D circuits are the main focus of this chapter. A short description of vertical interconnects in 3-D circuits is offered in the following section, demonstrating the diverse characteristics of this revolutionary design paradigm. These traits, in turn, pose new constraints and requirements on the physical and interconnect design process. The primary physical design issues, namely floorplanning, placement, and routing for 3-D ICs, are discussed in Sections 3, 4, and 5, respectively. An approach to place the vertical interconnects to decrease the interconnect delay is described in Section 6. The important task of synchronization is considered in Section 7. Experimental results from a 3-D test circuit are also presented. The concept of 3-D NoC for improving the communication throughput within a system-on-chip while reducing interconnect design complexity is presented in Section 8. Several topologies and related improvements in the speed and power consumed by these global interconnects are also described in this section. The key points of this chapter are summarized in Section 9.

2 Vertical Interconnects

There are multiple ways to vertically interconnect 3-D circuits. The characteristics of the different vertical interconnects and the requirements associated with this type of interconnect structure are discussed in this section. To exemplify the role of these interconnects, consider the 3-D circuit shown in Fig. 1. Two different types of interconnect can be distinguished in Fig. 1. The horizontal or intraplane interconnects connecting circuits located within the same plane and the interplane interconnects connecting circuits located on different planes. The interplane wires comprise horizontal and vertical segments.

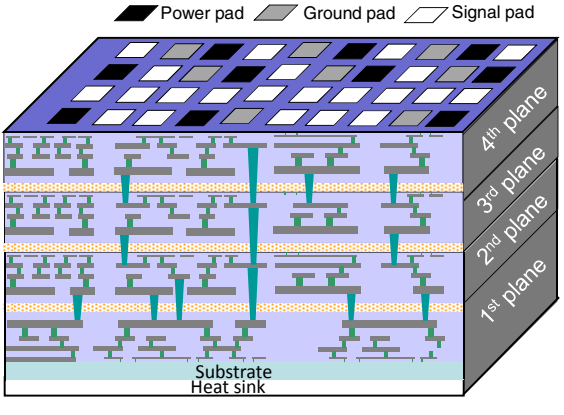


Fig. 1. Schematic of a 3-D IC consisting of four planes

The interconnects through the *z-axis* (*i.e.,* vertical) can be implemented with several means, such as solder balls, wire bonds, and vertical interconnects that are

etched through the silicon substrate. The latter type of interconnects is typically called a through silicon via (TSV) [15]-[18]. The density of the vertical interconnect dictates the granularity of the interconnected planes of the 3-D system, directly affecting the interplane communication bandwidth.

Coarsely interconnected 3-D systems include several either bare or packaged dice connected along the third dimension which are typically described as a system-in-package (SiP) [17]. The predominant benefits of SiP are the increased packaging efficiency as compared to 2-D integrated systems and shorter off-chip interconnects. The deleterious effects of the long on-chip interconnects, however, are not mitigated. These issues are effectively resolved by another form of vertical integration, called simply (and somewhat abstractly) 3-D ICs.

Three-dimensional circuits can be conceptualized as the bonding of multiple wafers or bare dice. The distinctive difference between an SiP and a 3-D IC is the granularity of the vertical interconnects. Examples of 3-D systems connected with different means are illustrated in Fig. 2. In addition to the different types of vertical interconnects, several bonding styles for 3-D ICs are also possible: front-to-front, back-to-front, and back-to-back are some of these styles which are also depicted in Fig. 2.

Other important criteria related to manufacturing TSVs include the reliability and cost of these structures. A high TSV aspect ratio, the ratio of the diameter of the top edge to the length of the via, may also be required for certain types of 3-D circuits. The effect of forming the TSVs on the performance and reliability of neighboring active devices should also be negligible.

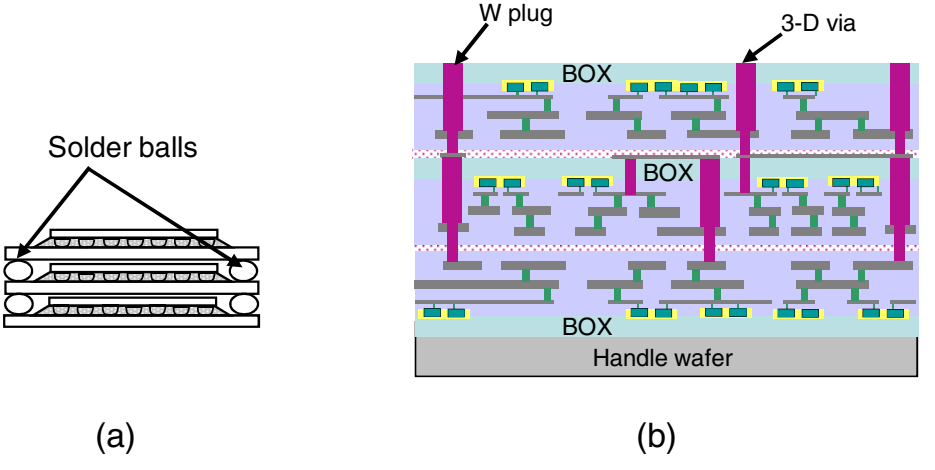


Fig. 2. Different forms of 3-D integration (not to scale), (a) system-in-package (SiP) [17] and (b) a 3-D circuit with dense through silicon vias [20], [21]. Two different bonding styles, front-to-front and front-to-back, are illustrated. The W plug is composed of tungsten.

Furthermore, producing TSVs with low impedance characteristics is another primary goal of 3-D manufacturing technologies since these characteristics can degrade the performance benefits that stem from the decreased wirelength in 3-D

circuits. Finally, not properly characterizing the contribution of the TSVs to the overall delay of the critical interplane interconnect can result in significant inaccuracy in estimating the performance of a 3-D system [19]. Consequently, these structures must be carefully considered during the 3-D physical design process. Examples of a TSV used in CMOS and SOI circuits are illustrated in Figs. 3a and 3b, respectively. The impedance and physical characteristics of these structures are listed in Table 1. A pitch equal to twice the diameter of the TSV is assumed where this dimension is not provided.

The thermal traits of the TSVs are also significant as these vias can affect the thermal behavior of a 3-D IC. TSVs can be used to provide high thermal conductivity paths to facilitate the flow of heat from the upper planes to the plane attached to the heat sink, maintaining the temperature of a 3-D circuit within acceptable levels. Since the vertical interconnects affect the performance of 3-D systems, the treatment of these interconnects is central to the development of 3-D physical and interconnect design techniques. The most important steps of the 3-D design process and related design methodologies are discussed in the remainder of this chapter.

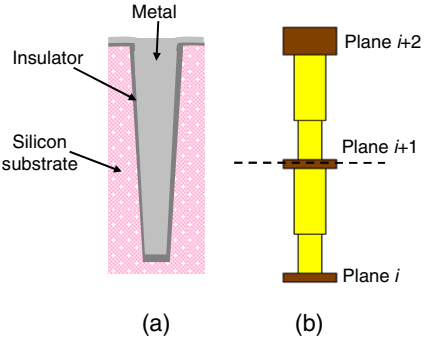


Fig. 3. Examples of a through silicon via (not to scale) used in (a) SiP and 3-D CMOS technologies [14], [22] and (b) 3-D SOI processes [21]

Table 1. Impedance and physical characteristics of TSVs

Process	*Depth [μm]*	*Diameter [μm]*	*Total resistance [mΩ]*	*Density [1/mm^2]*
[22]	25	4	140	~1.6×10^4
[11]	30	1.2	<350	~1.7×10^5
[23]	90	75	2.4	~44
[21], [24]	~12	1.75	148	~8.2×10^4

3 Floorplanning for 3-D Circuits

The predominant design objective for floorplanning a circuit has traditionally been to achieve the minimum area or, alternatively, the maximum packing density while interconnecting these blocks with minimum length wires. Most floorplanning algorithms can be classified as either slicing [25] or non-slicing [26]. Floorplanning techniques belonging to both of these categories have been proposed for 3-D circuits

[27]-[29]. An efficient floorplanning technique for 3-D circuits should adequately handle two important issues; representation of the third dimension and the related increase in the solution space. Floorplanning techniques for 3-D circuits that address these issues are discussed in this section. Multi-objective techniques are also reviewed.

Notating the location (*i.e.,* the *x, y, z,* coordinates) and dimensions (*i.e.,* width, length, and height) of the circuit cells in a volumetric system typically requires a considerable amount of storage. A 3-D circuit, however, consists of a limited number of planes. Consequently, such a system can be described as an array of two-dimensional planes, where circuit cells are treated as rectangles that can be placed on any of the planes within a 3-D system [13], [28], [29]. The second challenge for 3-D floorplanning is to effectively explore the solution space, where a hierarchical approach can often be more efficient for floorplanning 3-D circuits than a flat approach.

In non-hierarchical floorplanning algorithms, the floorplanning process proceeds by assigning the cells to the planes of the stack followed by simultaneous intraplane and interplane cell swapping, potentially exploring the entire solution space. Interplane moves, however, result in a formidable increase in the solution space, directly affecting the computational time of a flat floorplanning algorithm.

Alternatively, a hierarchical approach can be used to significantly reduce the number of candidate solutions, where a two step solution to the floorplanning problem is followed. Initially, the circuit cells are assigned to the physical planes. In the second step, a simulated annealing based engine simultaneously generates the floorplan of each of the planes by only permitting intraplane moves, considerably decreasing the search space for the optimal floorplan [28]. An example of the increase in the solution space due to the third dimension is illustrated in Figs. 4a and 4b.

The partitioning scheme adopted in the initial step of the hierarchical approach plays a crucial role in determining the compactness of a particular floorplan, as interplane moves are not allowed when floorplanning the planes. Different partitions correspond to different subsets of the solution space which may exclude the optimal solution(s). The criterion for partitioning should therefore be carefully selected. Partitioning can, for example, be based on minimizing the estimated total wirelength of the system [30] and/or the number of vertical interconnects [31]. Application of a hierarchical approach to the MCNC and GSRC benchmark suites [32] demonstrates a small reduction, on the order of 3%, in the number of vertical vias and a significant 14% reduction in wirelength, as compared to non-hierarchical 3-D floorplanning techniques [13], [30], [31].

The complexity of three-dimensional integration requires several dissimilar metrics for producing efficient floorplans for 3-D circuits beyond the use of traditional area and wirelength metrics. These metrics can consider, for example, communication throughput among the circuit blocks [33] or the number of interplane vias [13]. Techniques that include a thermal objective have also been developed [13]. The thermal objective typically aims at producing a uniform temperature distribution across each plane while peak temperatures are maintained sufficiently low. A multi-objective cost function inevitably increases the total computational runtime. A significant portion of this time is attributed to thermal profiling the 3-D circuit each time a candidate floorplan is generated. To reduce this time, simple thermal models are utilized, slightly degrading the quality of the solution [13].

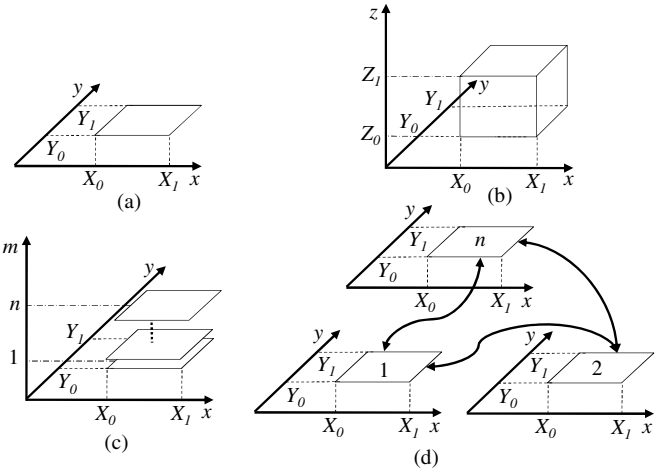


Fig. 4. Example of physical design solution space for floorplanning 2-D and 3-D circuits, (a) available area for floorplanning a planar circuit, (b) available volume for floorplanning a 3-D circuit, (c) a finite number of planes is considered to reduce the solution space, and (d) the floorplan of the planes is generated after the circuit cells are assigned to each plane. The arrows represent global constraints among planes that guide the floorplan of a 3-D system.

4 Placement for 3-D Circuits

Placement algorithms have traditionally targeted minimizing the area of a circuit and the interconnect length among the cells, while reserving space for routing the interconnect. In vertical 3-D integration, a "placement dilemma" arises in deciding whether two circuit cells sharing a large number of interconnects can be more closely placed within the same plane or placed on adjacent physical planes, decreasing the interconnection length. Placing the circuit blocks on adjacent planes can often produce a line with the shortest wirelength to connect these blocks. An exception is the case of small blocks within an SiP where the length of the interplane vias is greater than 100 μm [34]. Placement methodologies have also been discussed where other objectives, such as thermal gradients among the physical planes and the temperature of the planes [35], are considered.

Several approaches have been adopted for placing circuit cells within a volume [29], [36]. Different types of circuit cells for various 3-D technologies have been investigated in [37]. Layout algorithms for these cells have also been devised, demonstrating the benefits of 3-D integration. Since TSVs consume silicon area, possibly increasing the length of some interconnects, an upper bound on this type of interconnect resource is necessary. Alternatively, sparse utilization of the vertical interconnects can result in an insignificant savings in wirelength. To consider the effect of the vertical interconnects, a weighting factor can be used to increase the distance in the vertical direction, controlling the decision as to where to insert the interplane vias [38]. This weight essentially behaves as a controlling parameter that

favors the placement of highly interconnected cells within the same or adjacent physical planes.

Alternatively, TSVs can be treated as circuit cells since these interconnects occupy silicon area [39] and are included in the individual cell placement process within each plane as illustrated in Fig. 5. Although these approaches consider the location of the TSV, the fundamental objective is to decrease the interconnect length. The maximum achievable reduction in the interconnect length for the longest on-chip interconnect is proportional to $\sqrt{n}$ where n is the number of planes constituting a 3-D system [6]. Any further improvement in the performance of the interplane interconnects can be obtained by considering the electrical characteristics of the TSV. A placement methodology that exploits these characteristics is discussed in Section 6.

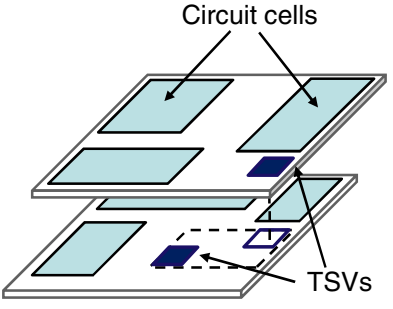


Fig. 5. Treating the TSVs as circuit cells on different planes can result in two different locations for placing a TSV. These locations define a region in which the TSV can be placed to satisfy different design objectives.

As with floorplanning, multi-objective placement techniques for 3-D circuits are necessary. Additional objectives that affect both the cell placement and wirelength are simultaneously considered. The force directed method is a well known technique used for cell placement [40], where repulsive or attractive forces are placed on the cells as if these cells are connected through a system of springs. The force directed method has been extended to incorporate the thermal objective during the placement process [41]. In this approach, repulsive forces are applied to those blocks that exhibit high temperatures (*i.e.,* "hot blocks") to ensure that the high temperature blocks are placed at a greater distance from each other. The efficiency of this force directed placement technique has been evaluated on the MCNC [42] and IBM-PLACE benchmarks [43], demonstrating a 1.3% decrease in the average temperature, a 12% reduction in the maximum temperature, and a 17% reduction in the average thermal gradient. The total wirelength, however, increases by 5.5%.

Alternatively, additional TSVs that do not function as a signal path can be utilized to further enhance the heat transfer process. The design objective is to identify those regions where thermal vias are most needed (the hot spots) and place thermal vias within those regions at the appropriate density. Such an assignment, however, is mainly restricted by two factors; the routing blockage caused by these vias and the size of the unoccupied regions or white space that exist within each plane. Although

thermal via insertion can be applied as a post placement step, integrated techniques produce a more efficient distribution of the thermal TSVs for the same temperature constraint [30]. The integrated technique requires 16% fewer thermal vias for the same temperature constraint, with a 21% increase in computational time and an almost 3% reduction in total area.

5 Routing for 3-D Circuits

Routing is the most complex and least developed of the physical design techniques used in 3-D circuits. The multiple metal layers available for routing on each physical plane exacerbate the difficulty in routing a net connecting several cells located on different planes. As these interconnects also compete with the transistors for silicon area, routing is a formidable task for 3-D circuits. Early results on routing 3-D circuits demonstrated several issues related to this physical design task [44]. Consequently, several heuristics have been developed that address routing in the third dimension [45], [46].

An effective approach for routing 3-D circuits is to convert the routing interplane interconnect problem into a 2-D channel routing task, as the 2-D channel routing problem has been efficiently solved [47]. A number of methods can be applied to transform the problem of routing the interplane interconnects into a 2-D routing task, which requires utilizing a portion of the available routing resources for interplane routing (usually the top metal layers).

Alternatively, multi-level algorithmic techniques [48] have been applied to route 3-D circuits. The advantages of multi-level routing are the lower computational time and higher completion rates as compared to flat and hierarchical routers. Multi-level routing can be treated as a three stage process, as illustrated in Fig. 6; a coarsening phase, an initial solution generation at the coarsest level (level p) of the grid, and a subsequent refinement process until the finest level of the grid is reached. Before the coarsening phase is initiated, the routing resources in each unit block of the grid are determined by a weighted area sum model. The routing resources are allocated during each coarsening step. The resources for the local nets within a block are transferred at each coarsening step. At the coarsest level, an initial routing tree is generated. This initial routing task commences with a minimum spanning tree for each multi-terminal net. A Steiner tree heuristic and a maze searching algorithm generate a 3-D Steiner tree for each of these interconnects. Additionally, the TSVs are estimated for each block. During the last phase, the initial routing tree is refined until the finest level is reached. In this refinement phase, the signal (and thermal) TSVs are successively assigned and distributed within each block. The routing of the wires follows the refinement of the TSVs. At the finest level, a detailed router completes the routing of the circuit [48].

Although this technique offers a routing solution for standard cell and gate array circuits, alternative techniques that support different forms of vertical integration, for example, systems-on-package (SOP), are also required. In an SOP, the routing problem can be described as connecting the I/O terminals of the blocks located on the planes of the SOP through interconnect and pin layers. For systems where the routing resources, such as the number of pin distribution layers, are limited, multi-objective routing is required to achieve a sufficiently small form factor [46].

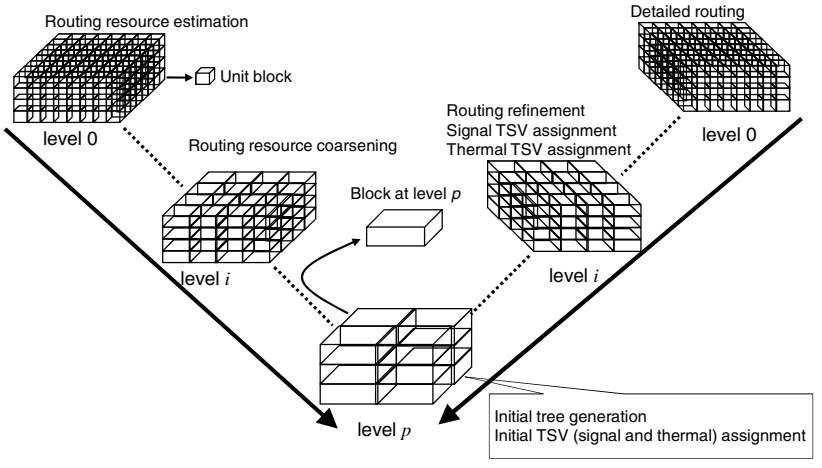


Fig. 6. Multi-level routing for 3-D circuits. The technique can be adapted to include multiple objectives for routing a 3-D circuit [48].

Multi-level routing for 3-D ICs has been extended to include the thermal objective [49]. In addition to routing resources, the power density within each block of the grid is determined at each coarsening step. An initial TSV assignment to each block is implemented during the coarser step along with generation of an initial routing tree. The TSV assignment includes both signal and thermal TSVs, with priority given to the signal TSVs. Alternatively, thermal TSVs are assigned to a block after insertion of the signal TSVs without exceeding the maximum TSV capacity of the block.

6 Timing Optimization of Interplane Interconnects

Three-dimensional integration demonstrates many opportunities for heterogeneous SoCs [9]. Integrating circuits from diverse fabrication processes into a single multi-plane system can result in substantially different interconnect impedance characteristics of each physical plane within a 3-D circuit. By considering the disparate interconnect impedance characteristics of 3-D circuits, the performance of the interplane interconnects can be significantly improved. An efficient technique to decrease the delay of interplane interconnects by optimally placing the TSVs is discussed in this section.

The interplane interconnects are modeled as an assembly of horizontal interconnect segments with different impedance characteristics connected by interplane vias where each segment is modeled as a distributed *RC* line. A schematic of an interplane interconnect connecting two circuits located *n* planes apart is illustrated in Fig. 7. The horizontal segments of the line are connected through the vias, which can traverse more than one plane where each via is placed within a certain physical interval. The via placement is constrained,

$$0 \le x_j \le \Delta x_j, \tag{1}$$

where Δx_j is the length of the interval where the via connecting planes j and $j+1$ can be placed. This interval length is called the "allowed interval" here for clarity. x_j is the distance of the via location from the edge of the allowed interval.

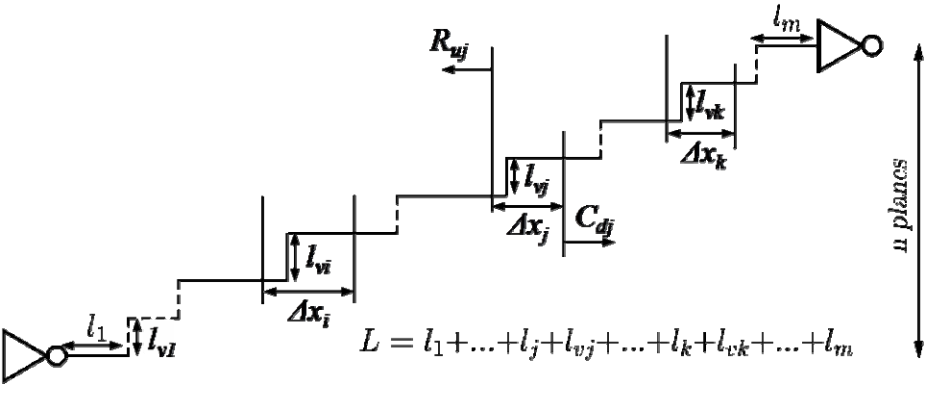


Fig. 7. Interplane interconnect connecting two circuits located n planes apart

A heuristic for near-optimal interplane via placement of two-terminal nets that include several TSVs has been developed [50]. Based on the Elmore delay model [51], the key concept in the heuristic is that the optimum via placement depends primarily upon the size of the allowed interval (that is estimated or known after an initial placement) rather than the exact location of the via.

This heuristic has been used to implement an algorithm that exhibits an optimal or near-optimal TSV placement for two-terminal interplane interconnects in 3-D ICs and has been applied to relatively short interconnects (< 2 mm) [50]. For these wires, SPICE delay simulations demonstrate an average improvement of 8.9% as compared to the case where the vias are placed at the center of the allowed intervals and 14.1% as compared to random via placement, respectively. The two-terminal via placement algorithm is also compared both in terms of optimality and efficiency to an optimization solver, YALMIP [52]. The algorithm exhibits high accuracy as compared to YALMIP independent of the number of planes that comprise the 3-D interconnect and exhibits a maximum error of 0.01%. Furthermore, the algorithm is approximately two orders of magnitude faster than YALMIP while the complexity of the algorithm exhibits an almost linear dependence on the number of interplane vias.

The two-terminal heuristic can also be adapted to the important class of multi-terminal nets. A simple interplane interconnect tree (also called an interconnect tree for simplicity) is illustrated in Fig. 8. The leaves of the tree are located on different physical planes within a 3-D stack. Sub-trees not directly connected to the interplane vias which do not contain any interplane vias (*i.e.,* intraplane trees) are also shown. The weighted summation of the distributed Elmore delay of the branches of an interconnect tree is considered as the objective function,

$$T_w = \sum_{\forall s_{pq}} w_{s_{pq}} T_{s_{pq}}, \tag{2}$$

where w_{spq} and T_{spq} are the weight and distributed Elmore delay of sink s_{pq}, respectively. Weights are assigned to the sinks according to the relative criticality of the sinks. The constrained optimization problem for placing a via within an interplane interconnect tree can be described as

$$\text{(P1) } \textbf{minimize } T_w, \textbf{ subject to } (1), \forall \text{ via } v_j. \tag{3}$$

The heuristic and related algorithm that solve (3) have been applied to interconnect trees for two different 3-D technologies. These case studies include a 3-D IC technology based on [20] where the TSV length is $l_{v3\text{-}D}$ = 10 μm and an SiP technology where the TSV length is l_{vSiP} = 70 μm [23]. The impedance characteristics of the TSVs are $r_{v3\text{-}D}$ = 22 Ω/mm and $c_{v3\text{-}D}$ = 210 fF/mm and r_{vSiP} = 22 Ω/mm and c_{vSiP} = 6 pF/mm for the 3-D IC and SiP technology, respectively. The savings in delay achieved by optimally placing the vias is listed in Table 2 for different via placement scenarios.

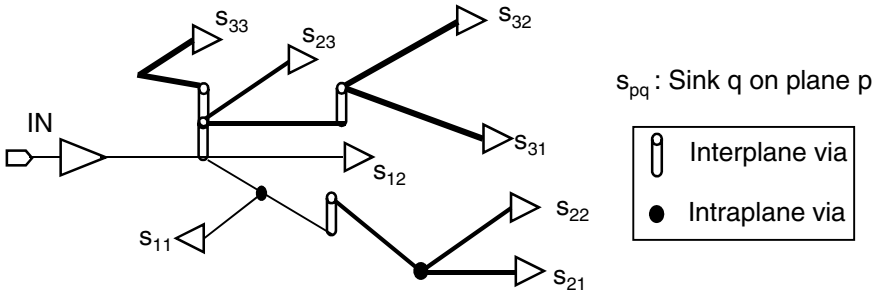


Fig. 8. An example of an interplane interconnect tree

The improvement in delay of the interconnect trees is listed in columns 6 through 9 of Table 2. The results are compared to the case where the vias are initially placed at the center of the allowed interval (*i.e.*, $x_i = \Delta x_i/2$) and the case where the vias are placed at the lower edge of the allowed interval (*i.e.*, $x_i = 0$). The improvement in delay depends upon the length of the allowed interval. Note that the improvement in delay achieved by optimally placing the TSVs in a 3-D IC is substantially greater than the improvement for an SiP technology. This difference is due to the significantly longer length and larger impedance characteristics of the TSVs utilized in an SiP. Manufacturing processes that provide short vertical interconnects with low parasitic impedances are therefore necessary; otherwise, the performance benefits due to the reduction in interconnect length will decrease since the TSVs contribute significantly to the overall interconnect delay.

From these results, exploiting the non-uniform impedance characteristics of the interplane interconnects when placing the vias can improve the delay of multi-plane lines. This improvement in delay can decrease the number of repeaters required to drive a global line or eliminate the need for repeaters in semi-global (medium length) lines. In addition, wire sizing can be avoided, thereby saving significant power. Decreasing the number of repeaters and avoiding wide lines reduce the overall power consumption, which is a particularly important issue in 3-D circuits.

Table 2. Delay of various interplane interconnect trees for different number of sinks, physical planes *n*, and 3-D technologies

n	*Technology*	*Number of sinks*	*Avg. branch length [μm]*	*Δx$_i$'s [μm]*	*Delay improvement [%]* $x_i^* = \Delta x_i/2$		$x_i^* = 0$		*Instances*
					Avg	*Max*	*Avg*	*Max*	
3	3-D IC	4	216	50	1.31	7.11	5.33	13.00	10000
4	3-D IC	8	407	50	1.47	6.88	6.83	13.22	10212
3	3-D IC	4	815	150	1.15	5.74	4.42	10.02	11000
4	3-D IC	8	909	150	1.29	4.98	5.70	9.48	10219
3	SiP	4	216	50	1.21	4.99	1.78	5.58	10000
4	SiP	8	407	50	0.90	3.54	1.98	5.72	10212
3	SiP	4	815	150	1.31	4.10	1.98	5.68	11000
4	SiP	8	909	150	1.04	3.28	2.34	5.71	10219

7 Synchronization in 3-D Circuits

An omnipresent and challenging issue for synchronous digital circuits is the reliable distribution of the clock signal to the many thousands of sequential elements distributed throughout a synchronous circuit [53], [54]. The complexity is further increased in 3-D ICs as sequential elements belonging to the same clock domain (*i.e.,* synchronized by the same clock signal) can be located on different planes. Another important issue in the design of clock distribution networks is low power consumption, since the clock network dissipates a significant portion of the total power consumed by a synchronous circuit [55]. This demand is stricter for 3-D ICs due to the increased power density and related thermal limitations.

In 2-D circuits, symmetric interconnect structures, such as H- and X-trees, are widely utilized to distribute the clock signal across a circuit [54]. The symmetry of these structures permits the clock signal to arrive at the leaves of the tree at the same time, resulting in synchronous data processing. Maintaining this symmetry within a 3-D circuit, however, is a difficult task. Consequently, asymmetric structures are useful candidates for distributing the clock signal within a 3-D circuit. Issues related to the distribution of the clock signal within a 3-D system are discussed in this section. Experimental results of a 3-D test circuit manufactured by MIT Lincoln Laboratories composed of several different 3-D clock network architectures are also described.

To evaluate the specific requirements of a 3-D clock network, consider a traditional H-tree topology. At each branch point of an H-tree, two branches emanate with the same length. An extension of an H-tree to three dimensions does not guarantee equidistant interconnect paths from the root to the leaves of the tree. Note that the vertical interconnects are of significantly different length as compared to the horizontal branches and exhibit different impedance characteristics.

A test circuit exploring four different clock network topologies for 3-D circuits has been designed, manufactured, and measured. The test circuit is based on a 3-D fully depleted silicon-on-insulator (FDSOI) fabrication technology recently developed by MIT Lincoln Laboratories (MITLL) [20]. The MITLL process is a wafer level 3-D integration technology with up to three FDSOI wafers bonded to form a 3-D circuit. The minimum feature size of the devices is 180 nm, with one polysilicon layer and three metal layers interconnecting the devices on each wafer. A backside metal layer

also exists on the upper two planes, providing the starting and landing pads for the TSVs, and the I/O, power supply, and ground pads for the entire 3-D circuit.

Each block contains the same logic circuit with different clock distribution networks. The off-chip clock signal is received by the clock driver through an RF pad located at the middle of each block. Additional RF pads are placed at different locations on the topmost plane of each block for probing. The fabricated test circuit is depicted in Fig. 9, where the RF and DC pads on the back side metal layer of the third plane are shown.

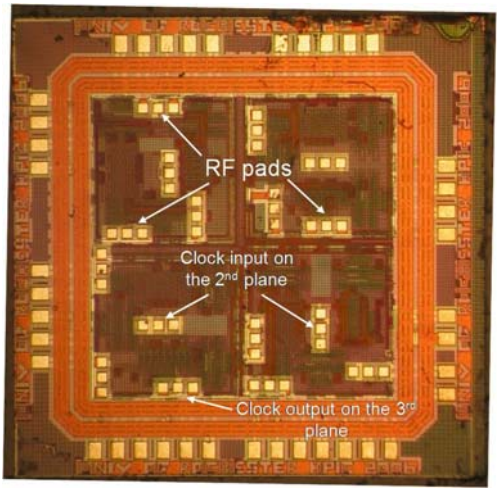
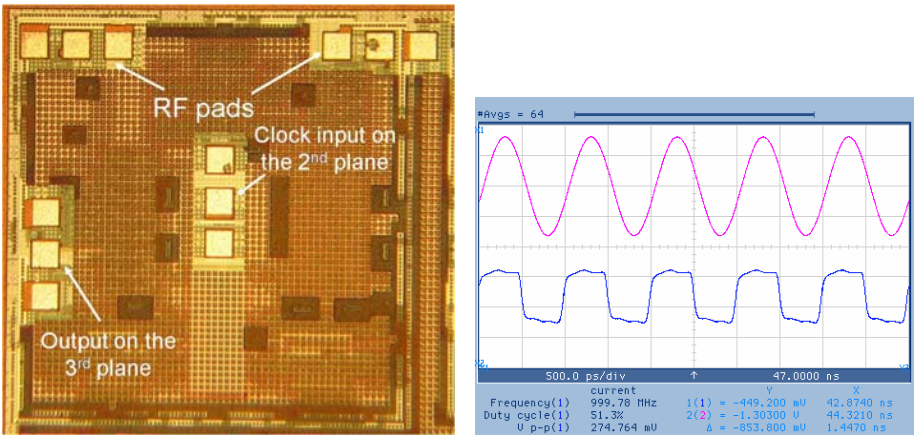


Fig. 9. Fabricated 3-D test circuit. The total area is 3 mm × 3 mm. There are four different blocks, with one input and three output RF pads for each block. The area of each block is approximately 1 mm^2.

The clock distribution networks combine commonly used networks such as H-trees, meshes, and rings. These clock network topologies range from highly symmetric topologies, such as H-trees, as the block shown in Fig. 10a, to fully asymmetric topologies, such as a trunk-based topology. The clock input is a 1.5 V peak-to-peak sinusoidal signal with 0.75 volt DC offset. The clock driver is implemented with a traditional chain of tapered buffers [56], which produces a square waveform at the root of the clock distribution network. The clock distribution network of the block illustrated in Fig. 10a contains a four level H-tree (*i.e.,* equivalent to 16 leaves) with identical interconnect characteristics in each plane. All of the H-trees are connected through a group of interplane vias. Note that the H-tree on the second plane is rotated by 90^o with respect to the H-trees on the other two planes. This rotation effectively eliminates inductive coupling between the H-trees. The second plane is front-to-front bonded with the first plane and both of the H-trees are implemented on the third metal layer. The vertical distance between these clock networks is approximately 2 μm. All of the H-trees are shielded with two parallel lines connected to ground. The waveform shown in Fig. 10b is the clock signal at a leaf of the H-tree on the third plane, demonstrating operation of the circuit at 1 GHz. Experiments

demonstrate that a clock distribution network that combines an H-tree on the second plane and meshes on the other two planes exhibits moderate skew, within 10% of the clock period, and the lowest power consumption [57], [58]. The superior performance of this topology is due to the symmetry of the H-tree and the balancing characteristic of the meshes.



(a) (b)

Fig. 10. Experimental results of the fabricated 3-D circuit, (a) tested circuit block and (b) clock signal waveform from the H-tree on the third plane operating at 1 GHz

8 Communication Centric 3-D Architectures

A promising design paradigm to appease foreseen interconnect problems is networks-on-chip (NoC) [59], where information is communicated among circuits within packets in an internet-like fashion. The synergy between these two design paradigms, namely NoC and 3-D ICs, can be exploited to significantly improve the performance and decrease the power consumption of communications limited systems. Several interesting topologies that emerge by incorporating the third dimension in networks-on-chip are discussed in this section.

On-chip networks differ from traditional interconnection networks in that communication among the network elements is implemented through the on-chip routing layers rather than the metal tracks of the package or printed circuit board. NoC provide communication among a variety of processing elements (PE), such as processor and DSP cores, memory blocks, FPGAs, and dedicated hardware [60], [61]. Furthermore, the length of the communication channel is primarily determined by the area of the PE, which is typically unaffected by the network structure. Mesh structures have been a popular network topology for conventional 2-D NoC, as illustrated in Fig. 11a, where each processing element (PE) is connected to the network through a router [59].

Integration in the third dimension introduces a variety of topological choices for NoCs. For a 3-D NoC, as shown in Fig. 11b, the total number of nodes is $N = n_1 \times n_2$

$\times n_3$, where n_1, n_2, and n_3 is the number of network nodes in the *x, y,* and *z* direction, respectively. In this topology, each PE is on a single, yet possibly different physical plane (2-D IC – 3-D NoC). In other words, a PE can be implemented on only one of the n_3 physical planes of the system and, therefore, the 3-D system contains $n_1 \times n_2$ PEs on each of the n_3 physical planes, where the total number of nodes is N [62]. A 3-D topology is illustrated in Fig. 11c, where the interconnect network is contained within one physical plane (*i.e.,* $n_3 = 1$), while each PE is integrated on multiple planes, notated as n_p (3-D IC – 2-D NoC). Finally, a hybrid 3-D NoC based on the two previous topologies is depicted in Fig. 11d. In this NoC topology, both the interconnect network and the PEs can span more than one physical plane of the stack (3-D IC – 3-D NoC).

Analytic models of the zero-load latency and power consumption with delay constraints of these networks capturing the effects of the topology on the performance of 3-D NoC have been developed. The overall zero-load network latency for a 3-D NoC is [63]

$$T_{network} = hops(t_a + t_s) + hops_{2-D}t_h + hops_{3-D}t_v + \frac{L_p}{w_c}t_h, \tag{4}$$

where t_a, t_s, t_v, and t_h are the delay of the arbiter, crossbar switch, and vertical and horizontal channels, respectively. *hops*, $hops_{2\text{-}D}$, and $hops_{3\text{-}D}$ denote the average number of hops within the two dimensions n_1 and n_2, and within the third dimension n_3, respectively (see Fig. 11). L_p and w_c denote, respectively, the length of a data packet and the width of the interconnect buss connecting adjacent network routers. L_v denotes the length of the vertical buss, which is equal to one or more TSV lengths.

These models do not incorporate the effects of the routing scheme and traffic load. Since minimum distance paths and no contention are implicitly assumed in these expressions, non-minimal path routing schemes and heavy traffic loads will increase both the latency and power consumption of the network. These models can therefore be treated as lower bounds for both the latency and the power consumption of the network. Alternatively, these expressions provide the maximum improvement in the performance of a conventional NoC that can be achieved with vertical integration.

The resulting decrease in network latency as compared to a standard 2-D IC – 2-D NoC is illustrated in Fig. 12a for increasing network size where the area of each PE is denoted by A_{PE}. The 2-D IC – 3-D NoC topology decreases the number of hops while the interconnect buss delay remains constant. With a 3-D IC – 2-D NoC, the buss delay is smaller but the number of hops remains unchanged. With a 3-D IC – 3-D NoC, all of the latency components can be decreased by assigning a portion of the available physical planes to the network while the remaining planes of the stack are used for the PEs. A decrease in latency of 31.5% and 29.7% can be observed for N = 128 and N = 256 nodes, respectively, with A_{PE} = 1 mm^2. Note that the 3-D IC – 3-D NoC topology achieves the greatest savings in latency by optimally balancing n_3 with n_p. Consequently, the tradeoff between the number of hops and the buss length for various 3-D topologies can be exploited to improve the performance of a network-on-chip.

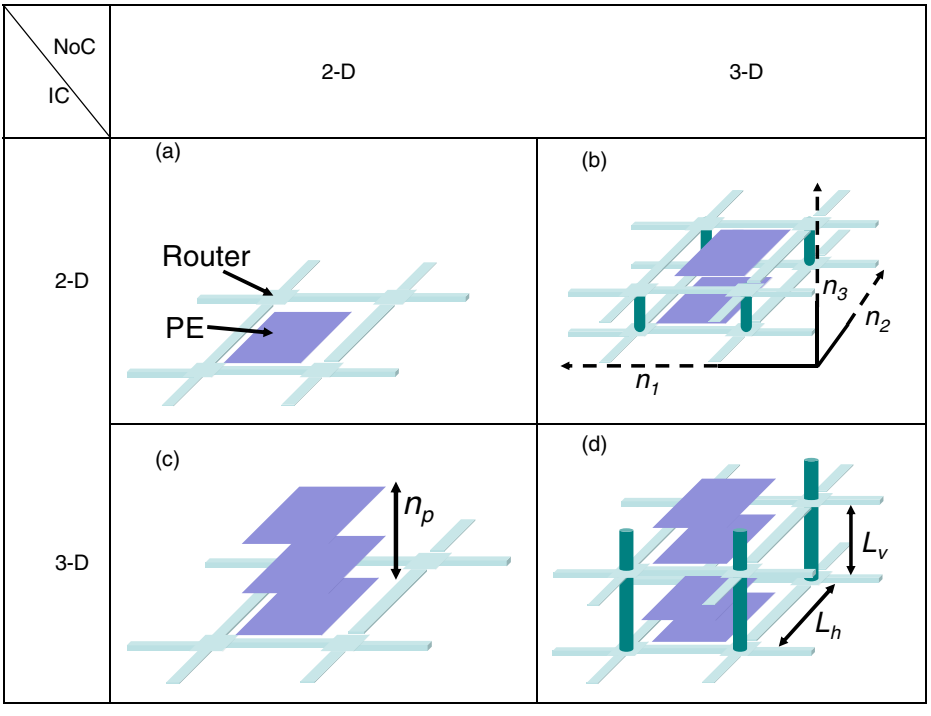
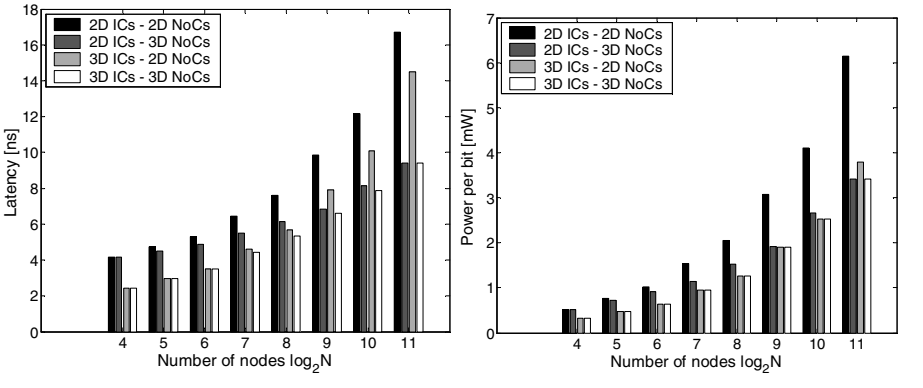


Fig. 11. Various NoC topologies (not to scale), (a) 2-D IC – 2-D NoC, (b) 2-D IC – 3-D NoC, (c) 3-D IC – 2-D NoC, and (d) 3-D IC – 3-D NoC



(a) (b)

Fig. 12. Performance of 3-D NoC topologies for a range of network sizes where A_{PE} = 1 mm^2; (a) zero-load latency and (b) power consumption with delay constraints

As with the zero-load latency, each topology affects the power consumption of a network in a different way. The power consumption can be reduced by either decreasing the number of hops that a packet travels or by decreasing the buss length.

In Fig. 12b, the power consumption of a 2-D NoC topology is compared to the three-dimensional topologies previously discussed. A power savings of 38.4% is achieved for $N = 128$ with $A_{PE} = 1$ mm^2. Allowing the available physical planes to be utilized either for the third dimension of the network or for the PEs, the 3-D IC - 3-D NoC scheme achieves the greatest savings in power in addition to the minimum delay.

Note that these topologies emphasize the latency and power consumption of a network, neglecting the performance requirements of the individual PEs. If the performance of the individual PEs is important, only one 3-D topology may be available; however, despite this constraint, a significant savings in latency and power can be achieved since in almost every case the network latency and power consumption are lower than for the 2-D IC – 2-D NoC topology.

9 Conclusions

Developing a design flow for 3-D ICs is a complicated task with many ramifications. Design methodologies at the front end and mature manufacturing processes at the back end are required to effectively provide large scale 3-D systems. A variety of floorplanning, placement, and routing techniques and algorithms for 3-D circuits have been described that consider the unique characteristics of 3-D circuits. In these techniques, the discrete nature of the third dimension is exploited to decrease the number of candidate solutions and, consequently, the computational time required to design a 3-D circuit.

In addition, due to increased power densities and greater distances between the circuits on the upper planes and the heat sink, physical design techniques that embody a thermal objective can be a useful mechanism to manage thermal issues in 3-D ICs. Design techniques can reduce thermal gradients and temperatures in 3-D circuits by redistributing the blocks among and within the planes of a 3-D circuit. Alternatively, thermal vias can be utilized in 3-D circuits to convey heat to the heat sink.

Significant performance improvements can be achieved by optimally placing interplane vias in 3-D circuits. Algorithms for determining the minimum delay of the interplane interconnects are an integral element of the physical design process for 3-D circuits. Interplane interconnect impedances of 3-D circuits vary considerably from 2-D interconnect impedances. This difference is due to several reasons, such as the heterogeneity of 3-D circuits, diverse fabrication technologies, and the variety of bonding styles.

Another requirement for maximizing the speed of 3-D circuits is to reliably distribute the clock signal. A 3-D clock distribution network, however, cannot be directly extended from a 2-D circuit due to the asymmetry of a multi-plane 3-D circuit and the effect of the interplane via impedances. Several clock distribution networks have been developed to investigate synchronization issues in 3-D systems. These clock distribution networks within a three plane 3-D circuit exhibit low clock skew while operating into the gigahertz regime.

In addition to higher performance, 3-D integration offers significant opportunities for designing highly diverse and complex systems. On-chip networks can be a useful solution to provide sufficient communication throughput among the components of these 3-D systems. 3-D NoC are a natural evolution of 2-D NoC, exhibiting superior

performance. These topologies decrease the latency and power consumption by reducing both the number of hops per packet and the length of the communications channels. These 3-D topologies demonstrate the tradeoff between the number of planes required to implement a network and those planes required to implement the PEs. Consequently and not surprisingly, the 3-D IC – 3-D NoC topology achieves the greatest improvement in latency and power consumption by most effectively exploiting the third dimension.

References

[1] International Technology Roadmap for Semiconductors (ITRS), Semiconductor Industry Association (2007)
[2] Pavlidis, V.F., Friedman, E.G.: Three Dimensional Integrated Circuit Design. Morgan Kaufmann Publishers, San Francisco (2009)
[3] Joyner, J.W., et al.: Impact of Three-Dimensional Architectures on Interconnects in Gigascale Integration. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 9(6), 922–928 (2001)
[4] Rahman, A., Reif, R.: System Level Performance Evaluation of Three-Dimensional Integrated Circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 8(6), 671–678 (2000)
[5] Stroobandt, D., Van Campenhout, J.: Accurate Interconnection Lengths in Three-Dimensional Computer Systems. IEICE Transactions on Information and System, Special Issue on Physical Design in Deep Submicron 10(1), 99–105 (2000)
[6] Joyner, J.W., Meindl, J.D.: Opportunities for Reduced Power Distribution Using Three-Dimensional Integration. In: Proceedings of the IEEE International Interconnect Technology Conference, June 2002, pp. 148–150 (2002)
[7] Banerjee, K., Souri, S.K., Kapour, P., Saraswat, K.C.: 3-D ICs: A Novel Chip Design Paradigm for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration. Proceedings of the IEEE 89(5), 602–633 (2001)
[8] Koyanagi, M., et al.: Future System-on-Silicon LSI Chips. IEEE Micro 18(4), 17–22 (1998)
[9] Pavlidis, V.F., Friedman, E.G.: Interconnect-Based Design Methodologies for Three-Dimensional Integrated Circuits. Proceedings of the IEEE, Special Issue on 3-D Integration Technology 97(1), 123–140 (2009)
[10] Bernstein, K., et al.: Interconnects in the Third Dimension: Design Challenges for 3-D ICs. In: Proceedings of the IEEE/ACM Design Automation Conference, June 2007, pp. 562–567 (2007)
[11] Patti, R.S.: Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs. Proceedings of the IEEE 94(6), 1214–1224 (2006)
[12] Lewis, D.L., Lee, H.-H.S.: A Scan-Island Based Design Enabling Pre-Bond Testability in Die-Stacked Microprocessors. In: Proceedings of the IEEE International Test Conference, October 2007, pp. 1–8 (2007)
[13] Cong, J., Wei, J., Zhang, Y.: A Thermal-Driven Floorplanning Algorithm for 3-D ICs. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 2004, pp. 306–313 (2004)
[14] Henry, D., et al.: Low Electrical Resistance Silicon Through Vias: Technology and Characterization. In: Proceedings of the IEEE International Electronic Components and Technology Conference, June 2006, pp. 1360–1365 (2006)

[15] Garrou, P.: Future ICs Go Vertical. Semiconductor International (February 2005)
[16] Karnezos, M.: 3-D Packaging: Where All Technologies Come Together. In: Proceedings of IEEE/SEMI International Electronics Manufacturing Technology Symposium, July 2004, pp. 64–67 (2004)
[17] Miettinen, J., Mantysalo, M., Kaija, K., Ristolainen, E.O.: System Design Issues for 3D System-in-Package (SiP). In: Proceedings of the IEEE International Electronic Components and Technology Conference, June 2004, pp. 610–615 (2004)
[18] Beyne, E.: The Rise of the 3rd Dimension for System Integration. In: Proceedings of the IEEE International Interconnect Technology Conference, June 2006, pp. 1–5 (2006)
[19] Pavlidis, V.F., Friedman, E.G.: Interconnect Delay Minimization through Interlayer Via Placement in 3-D ICs. In: Proceedings of the ACM Great Lakes Symposium on VLSI, April 2005, pp. 20–25 (2005)
[20] FDSOI Design Guide, MIT Lincoln Laboratories, Cambridge (2006)
[21] Burns, J.A., et al.: A Wafer-Scale 3-D Circuit Integration Technology. IEEE Transactions on Electron Devices 53(10), 2507–2515 (2006)
[22] Bower, C.A., et al.: High Density Vertical Interconnect for 3-D Integration of Silicon Integrated Circuits. In: Proceedings of the IEEE International Electronic Components and Technology Conference, June 2006, pp. 399–403 (2006)
[23] Jang, D.M., et al.: Development and Evaluation of 3-D SiP with Vertically Interconnected Through Silicon Vias (TSV). In: Proceedings of the IEEE International Electronic Components and Technology Conference, June 2007, pp. 847–850 (2007)
[24] Savidis, I., Friedman, E.G.: Electrical Modeling and Characterization of 3-D Vias. In: Proceedings of the IEEE International Symposium on Circuits and Systems, May 2008, pp. 784–787 (2008)
[25] Otten, R.H.J.M.: Automatic Floorplan Design. In: Proceedings of the IEEE/ACM Design Automation Conference, June 1982, pp. 261–267 (1982)
[26] Yong, E.F.Y., Chu, C.C.N., Zion, C.S.: Twin Binary Sequences: A Non-Redundant Representation for General Non-Slicing Floorplan. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22(4), 457–469 (2003)
[27] Cheng, L., Deng, L., Wong, D.F.: Floorplanning for 3-D VLSI Design. In: Proceedings of the IEEE International Asia and South Pacific Design Automation Conference, January 2005, pp. 405–411 (2005)
[28] Li, Z., et al.: Hierarchical 3-D Floorplanning Algorithm for Wirelength Optimization. IEEE Transactions on Circuits and Systems I: Regular Papers 53(12), 2637–2646 (2006)
[29] Deng, Y., Maly, W.P.: Interconnect Characteristics of 2.5-D System Integration Scheme. In: Proceedings of the IEEE International Symposium on Physical Design, April 2001, pp. 341–345 (2001)
[30] Li, Z., et al.: Efficient Thermal Via Planning Approach and its Application in 3-D Floorplanning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26(4), 645–658 (2007)
[31] Yan, T., Dong, Q., Takashima, Y., Kajitani, Y.: How Does Partitioning Matter for 3D Floorplanning. In: Proceedings of the ACM International Great Lakes Symposium on VLSI, April/May 2006, pp. 73–76 (2006)
[32] http://www.cse.ucsc.edu/research/surf/GSRC/progress.html
[33] Healy, M., et al.: Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26(1), 38–52 (2007)

[34] Lo, W.-C., et al.: An Innovative Chip-to-Wafer and Wafer-to-Wafer Stacking. In: Proceedings of the IEEE International Electronic Components and Technology Conference, June 2006, pp. 409–414 (2006)
[35] Goplen, B., Sapatnekar, S.: Placement of Thermal Vias in 3-D ICs Using Various Thermal Objectives. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(4), 692–709 (2006)
[36] Obenaus, S.T., Szymanski, T.H.: Gravity: Fast Placement for 3-D VLSI. ACM Transactions on Design Automation of Electronic Systems 8(3), 298–315 (2003)
[37] Harter, A.: Three-Dimensional Integrated Circuit Layout. Cambridge University Press, Cambridge (1991)
[38] Kaya, I., Olbrich, M., Barke, E.: 3-D Placement Considering Vertical Interconnects. In: Proceedings of the IEEE International SOC Conference, September 2003, pp. 257–258 (2003)
[39] Davis, W.R., et al.: Demystifying 3D ICs: the Pros and Cons of Going Vertical. IEEE Design and Test of Computers 22(6) (November/December 2005)
[40] Eisenmann, H., Johannnes, F.M.: Generic Global Placement and Floorplanning. In: Proceedings of the IEEE/ACM Design Automation Conference, June 1998, pp. 269–274 (1998)
[41] Goplen, B., Sapatnekar, S.: Efficient Thermal Placement of Standard Cells in 3-D ICs using a Force Directed Approach. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 2003, pp. 86–89 (2003)
[42] `http://er.cs.ucla.edu/benchmarks/ibm-place`
[43] `http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LayoutSynth92`
[44] Enbody, R.J., Lynn, G., Tan, K.H.: Routing the 3-D Chip. In: Proceedings of the IEEE/ACM Design Automation Conference, June 1991, pp. 132–137 (1991)
[45] Tong, C.C., Wu, C.-L.: Routing in a Three-Dimensional Chip. IEEE Transactions on Computers 44(1), 106–117 (1995)
[46] Minz, J., Lim, S.K.: Block-Level 3-D Global Routing With an Application to 3-D Packaging. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(10), 2248–2257 (2006)
[47] Ohtsuki, T. (ed.): Advances in CAD for VLSI. Layout Design and Verification, vol. 4. Elsevier, Amsterdam (1986)
[48] Cong, J., Xie, M., Zhang, Y.: An Enhanced Multilevel Routing System. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 2002, pp. 51–58 (2002)
[49] Cong, J., Zhang, Y.: Thermal Driven Multilevel Routing for 3-D ICs. In: Proceedings of the IEEE Asia and South Pacific Design Automation Conference, June 2005, pp. 121–126 (2005)
[50] Pavlidis, V.F., Friedman, E.G.: Timing Driven Via Placement Heuristics in 3-D ICs. Integration, the VLSI Journal 41(4), 489–508 (2008)
[51] Boese, K.D., et al.: Fidelity and Near-Optimality of Elmore-Based Routing Constructions. In: Proceedings of the IEEE International Conference on Computer Design, October 1993, pp. 81–84 (1993)
[52] Löfberg, J.: YALMIP: A Toolbox for Modeling and Optimization in MATLAB. In: Proceedings of the IEEE International Symposium on Computer-Aided Control Systems Design, September 2004, pp. 284–289 (2004)
[53] Friedman, E.G. (ed.): Clock Distribution Networks in VLSI Circuits and Systems. IEEE Press, New Jersey (1995)

[54] Friedman, E.G.: Clock Distribution Networks in Synchronous Digital Integrated Circuits. Proceedings of the IEEE 89(5), 665–692 (2001)
[55] Xanthopoulos, T., et al.: The Design and Analysis of the Clock Distribution Network for a 1.2 GHz Alpha Microprocessor. In: Proceedings of the IEEE International Solid-State Circuits Conference, February 2001, pp. 402–403 (2001)
[56] Prunty, C., Gal, L.: Optimum Tapered Buffer. IEEE Journal of Solid-State Circuits 27(1), 1005–1008 (1992)
[57] Pavlidis, V.F., Savidis, I., Friedman, E.G.: Clock Distribution Networks for 3-D Integrated Circuits. In: Proceedings of the IEEE International Conference on Custom Integrated Circuits, September 2008, pp. 651–654 (2008)
[58] Pavlidis, V.F., Savidis, I., Friedman, E.G.: Clock Distribution Architectures for 3-D SOI Integrated Circuits. In: Proceedings of the IEEE International Silicon-on-Insulator Conference, October 2008, pp. 111–112 (2008)
[59] Jantsch, A., Tenhunen, H.: Networks on Chip. Kluwer Academic Publishers, Dordrecht (2003)
[60] Benini, L., De Micheli, G.: Networks on Chip: A New SoC Paradigm. IEEE Computer 31(1), 70–78 (2002)
[61] Kumar, S., et al.: A Network on Chip Architecture and Design Methodology. In: Proceedings of the International IEEE Annual Symposium on VLSI, April 2002, pp. 105–112 (2002)
[62] Li, F., et al.: Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In: Proceedings of the IEEE International Symposium on Computer Architecture, June 2006, pp. 130–142 (2006)
[63] Pavlidis, V.F., Friedman, E.G.: 3-D Topologies for Networks-on-Chip. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 15(10), 1081–1090 (2007)

Universal Methodology to Handle Differential Pairs during Pin Assignment

Tilo Meister[1], Jens Lienig[1], and Gisbert Thomke[2]

[1] Dresden University of Technology, Dresden, Germany
[2] IBM Research and Development, Böblingen, Germany

Abstract. Differential signaling has been a major challenge in design automation. The routing of differential pairs requires a suitable pin assignment of the respective nets. However, current automatic pin assignment algorithms lack the ability to consider differential pairs. We present a methodology to include differential pairs during pin assignment. Our solution can be applied to automatic or manual pin assignment processes without changing the methodologies already in place. This universality is achieved by using any established pin assignment approach as a black box, which is extended by pre and post processing steps. Extensive studies in industrial design flows show that our differential pair methodology does not compromise pin assignment quality with the added benefit of effective differential pair allocations.

1 Introduction

Differential pairs are a common challenge during digital and analog/mixed-signal layout generation of modern electronic devices. The challenge is to route as closely together as possible a pair of wiring paths (the so-called *differential pair*) in order to improve the routing solution. The resulting routing geometry provides significantly better electrical characteristics than single ended signaling. For example, interference identically captured by both routing paths is filtered out. However, the routing of differential pairs requires an adequate pin assignment that has to be generated beforehand.

The pin assignment of a component, such as a chip, is the assignment of its I/O signals to its I/O pins, often referred to as pads (Fig. 1). Usually, this pin assignment is created after components are placed on the wiring substrate, such as a printed circuit board (PCB) or a multi chip module (MCM). Optimizing this pin assignment is a crucial stage because the routability of the substrate largely depends on both pin assignment and component placement. Due to rising I/O counts, the routability challenge has continued to increase rapidly in recent years, which puts enormous pressure on a well-performed pin assignment.

Furthermore, there has been a growing demand for differential pairs, which have to be considered during this stage. This is mostly due to more stringent electrical requirements of signals in modern applications. However, to the best of our knowledge, none of the published pin assignment approaches considers the implementation of differential pairs.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 22–42, 2010.

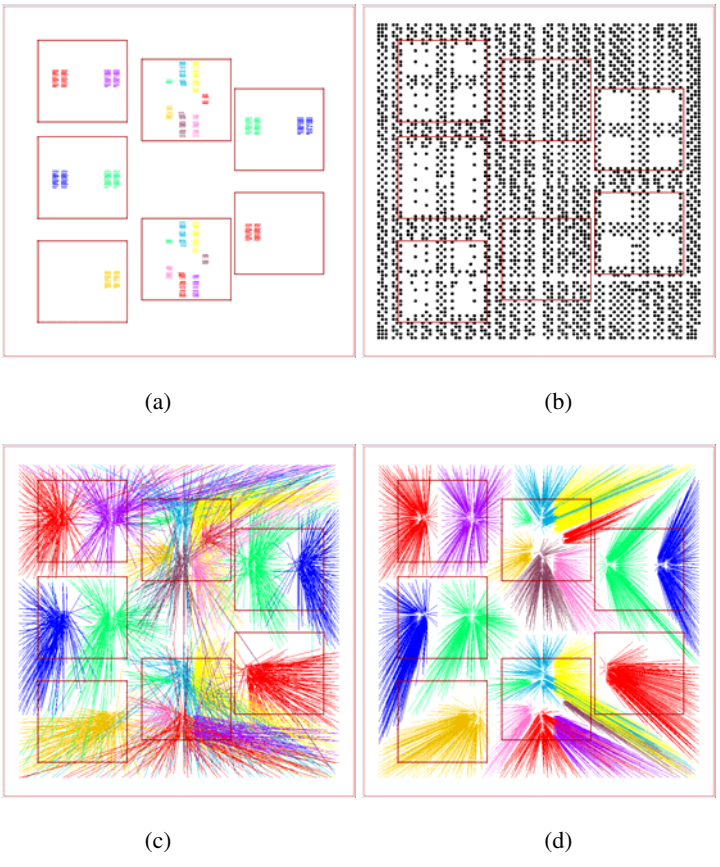
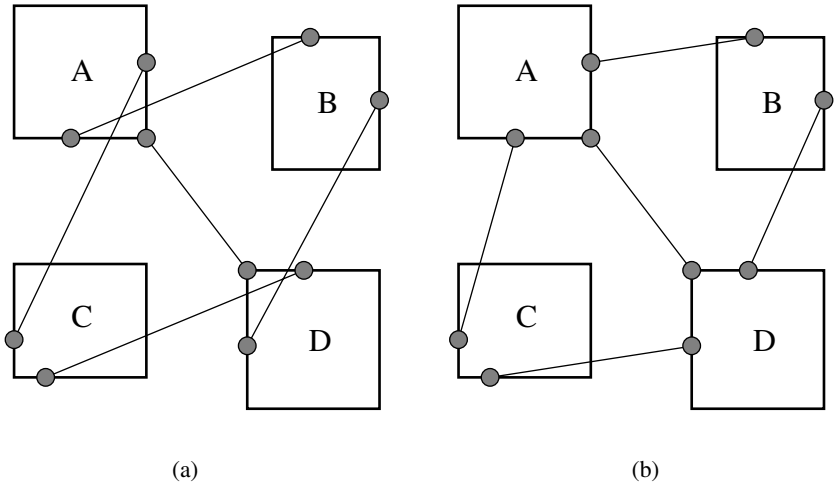


Fig. 1. Illustration of pin assignment with the I/O pins of seven chips (a) to be assigned to signals that connect a wiring substrate such as an MCM (b). The pin assignment based on the shortest Manhattan distances of the individual connections is depicted in (c) whereas (d) illustrates the pin assignment with minimum overall length of all Euclidean distances (*flylines*).

This chapter presents a universal methodology to extend pin assignment algorithms to consider differential pairs. This methodology requires no significant changes to the basic pin assignment algorithm, thereby respecting any individual pin assignment routines already in use. As shown below, this add-on approach has almost no impact on the quality of the created pin assignments while at the same time efficiently considering all differential pair requirements. Furthermore, the algorithm can be used for any given percentage (from zero to 100%) of differential pairs among the nets to be considered during pin assignment. As such, it allows a flexible inclusion of differential pair requirements in digital and analog/mixed-signal real-world design flows.

The remainder of this chapter is organized as follows. Pin assignment and differential pairs are introduced in the following two sections. The differential pair methodology is proposed in the section thereafter. The effectiveness of the proposed methodology is proven in the section presenting experimental results. At the end of this chapter, we present limitations of our approach, an outlook, and conclusions.



(a) (b)

Fig. 2. Example with four components A-D to illustrate the influence of pin assignment on the routability. Pins are marked by circles ● on the outline of the components. Nets are shown as flylines. If no wiring is allowed below components A through D, the design with pin assignment (a) is not routable in one layer, whereas (b) allows single layer routing.

2 The Pin Assignment Problem

During logic design, logical pins are defined to be the signal interface between the different components of a design. During the subsequently performed layout synthesis, these logical pins, and thus the associated signals, have to be mapped to real, physical pins, which serve as the actual electrical joints between the components.

This mapping of logical pins (signals) to physical pins is called *pin assignment* and has great influence on the routability, electrical characteristics and the cost of the design (see example in Fig. 2). Hence, the objective of pin assignment is to assign signals to physical pins such that these circuit characteristics are fulfilled best for the individual designs.

Pin assignment has been studied for all system levels such as digital and analog/mixed-signal circuits (ICs), MCMs and PCBs. For ICs, the pin assignment of macro blocks is usually optimized with regard to routability during placement [3][4], buffer planning [5] or routing [6]. Pin assignment approaches for PCBs and MCMs can be found in [7][8][9][10].

2.1 Context

Pin assignment is closely related to both component placement and routing. All three design steps have in common that their individual optimal solutions depend on each other. Finding the overall optimal solution for these design steps would require incorporating them into one optimization task. Due to complexity and the related NP-hardness of physical design, this is unfeasible. Hence, the repeated sequential execution of these design steps is currently the only accomplishable approach to physical design.

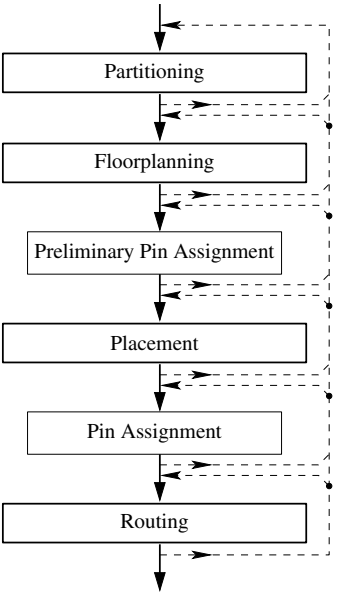


Fig. 3. Simplified physical design flow. The dashed arrows indicate iterations over the individual design steps.

Fig. 3 shows the major steps of the physical design flow [14][17], including pin assignment steps, of the design of electronic devices. Of these steps, placement is the first stage that requires an exact pin assignment, because the target function of placement depends on wire lengths and routing congestion. At this point a typical dilemma of physical design becomes obvious. The objectives of pin assignment are a minimal wire length and minimal routing congestion, which cannot be computed before determining the component placement. At the same time, placement depends on the chosen pin assignment. To come around this paradox either pin assignment has to be incorporated into placement [15] or a preliminary pin assignment has to be chosen before component placement is being optimized. Such a preliminary pin assignment is usually based on heuristics and the experience of designers and allows computing an optimized component placement.

Having optimized the placement for a specific preliminary pin assignment, it is then possible to improve the pin assignment for this optimized placement. To further improve design quality, it is possible to go back (one or more iterations) and revise the placement solution based on the optimized pin assignment (see Fig. 3).

A similar interdependency exists for pin assignment and routing. Routing largely depends on placement and pin assignment. Unfortunately, only after routing has been completed, which is extremely time consuming, it is possible to ultimately judge the quality of placement and pin assignment. Therefore, good estimates of the routability are essential for effective pin assignment algorithms.

It is further possible to integrate pin assignment into the global [16] and/or detailed routing phase. By integrating pin assignment into global routing, it can be adapted to

the global requirements of routing, whereas a combination with detailed routing would support local adjustments of the pin assignment.

2.2 Pin Assignment Algorithms Used in This Work

We use four pin assignment algorithms to evaluate the differential pair methodology presented. Three of them are heuristics, which either reduce signal intersections or balance the lengths of nets within a bus. The fourth algorithm analytically minimizes net lengths and the number of signal intersections. All four algorithms assume that pin assignment is done in-between placement and routing (see Fig. 3). The details of the four pin assignment algorithms are described in [7].

By using these four algorithms in various configurations, we obtain seven different pin assignment procedures in order to evaluate the presented differential pair methodology. Specifically, the analytical algorithm can be utilized with different parameters to its cost function. Also, one of the heuristic algorithms can be used to modify pin assignment results of the remaining three algorithms.

3 Differential Pairs

A differential pair are two wires which are routed close together, have matched electrical characteristics, and are used to transmit one signal. This signal is encoded in the voltage difference between both wires. Differential pairs are essential for many electronic devices, because differential signaling has superior electrical characteristics to single ended signaling [1][2]. In particular, differential signaling leads to lower cross-talk and lower electromagnetic interference. Both noise emission and noise acceptance are minimized by differential pairs if both (1) the distance between the two routing paths is minimal and (2) the lengths and electrical characteristics of both paths are matched.

The basic functional principle of a differential pair is shown in Fig. 4. The differential sender encodes the signal $S = u(t)$ into the difference of two complementary signals $a \cdot S = u_p(t)$ and $-a \cdot S = u_n(t)$ propagating along the two routing paths n and p. Where a is the gain of the differential sender. If both routing paths have the same electrical characteristics and are routed close together, captured noise A can be presumed to be identical for both signals $u_{pA}(t) = u_p(t) + A$ and $u_{nA}(t) = u_n(t) + A$. The signals $u_{pA}(t)$ and $u_{nA}(t)$ are then translated back to the original signal S by subtracting $u_{pA}(t) - u_{nA}(t) = S_{rcv} = 2 \cdot a \cdot S$ which at this point filters out any noise A identically captured along both paths.

In case routing paths n and p are not routed close together and/or have different electrical properties, both tracks capture noise differently A_n and A_p leaving the received signal $S_{rcv} = 2 \cdot a \cdot S + (A_p - A_n)$ distorted with noise $(A_p - A_n)$. Additionally, if electrical characteristics of the tracks differ, propagation delays of the n- and p-signal may be different, resulting in a distortion of the transmitted signal as shown in Fig. 5.

Assuming a signal frequency of 3GHz, a timing difference between signals n and p of only 167 ps shifts signals by half a clock. In a FR4 printed circuit board that timing difference is equal to a difference in wiring lengths of roughly 2.5cm. That is, the tolerance for propagation delays and wiring length differences for a differential pair at 3 GHz is in the domain of picoseconds and millimeters respectively.

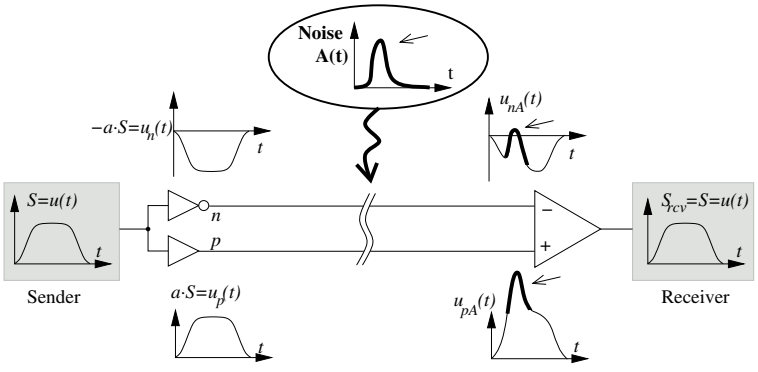


Fig. 4. Functional principle of a differential pair

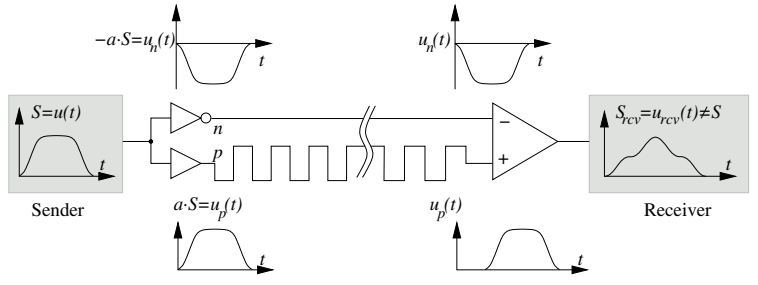


Fig. 5. Effect of unmatched propagation delays of a differential pair

The special wiring geometry of differential pairs requires suitable pin assignments. Specifically, for the two nets *n* and *p* of a differential pair, the pin assignment has to be chosen such that each pin of the routing path *n* has a so-called *parallel pin* at the same distance from the sender in the routing path *p* and vice versa. The distance between those parallel pins must not exceed a maximum distance d_{max}. This parameter is technology-dependent and for MCMs and PCBs usually ranges from one to two times the pin grid. For the sake of simplicity, we call the parallel pins of a differential pair a ***d**ifferential **p**in **p**air* (***DPP***). If the distance between the pins of the DPP is not greater than $d_{max,}$ we call it a *valid* DPP, else it is labeled an *invalid* DPP.

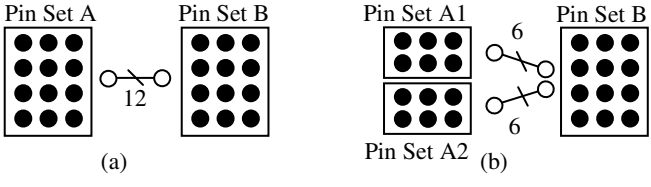


Fig. 6. Two pin assignment tasks for 12 two terminal nets. (a) 12 nets need an assignment to pins of sets A and B. (b) The nets of set B are to be assigned to two sets A1 and A2 and vice versa.

4 Differential Pair Methodology

In this section, we present our novel methodology to handle differential pairs during pin assignment. Our approach is used as an extension of any automatic or manual procedure in place that solves the pin assignment problem (Fig. 6 shows two example problems). The underlying basic pin assignment procedures, which are to be extended, are labeled ***PAA*** (**p**in **a**ssignment **a**lgorithm) throughout this paper.

4.1 Overview of the Algorithm

Our approach can be summarized in five steps (Fig. 7).

1. First, a transformation is applied to the original pins. This transformation embeds data about valid DPPs. We call the transformed pins *fat pins*.
2. Second, the PAA in place is applied to these fat pins.
3. Third, the pin assignment for the fat pins (*fat pin assignment*) is split up to the original pins. This back transformation returns a pin assignment only for a subset of the pins and nets.
4. Therefore in the fourth step, a pin assignment without differential pairs is created for the remaining unassigned nets with the same PAA as applied in the second step.
5. Finally, the two interim pin assignments created in steps (3) and (4) are merged into one final pin assignment, which respects all constraints of both the pin assignment problem and differential pairs.

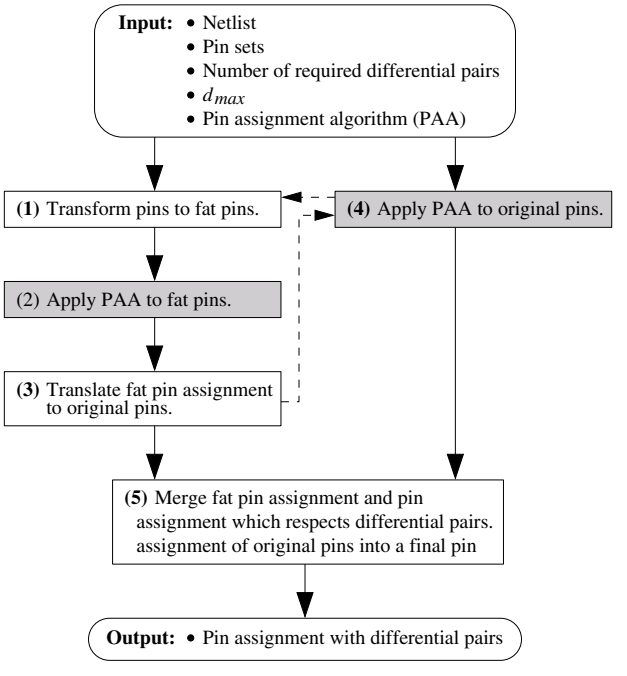


Fig. 7. Overview of our differential pair methodology

This methodology is a framework that allows considering any number of differential pairs by utilizing any existing pin assignment algorithm (see above) without the need to modify the existing pin assignment algorithm itself. Steps (1), (3) and (5) are pre and post processing steps (white boxes in Fig. 7), while any already existing pin assignment procedure PAA can be plugged-in at steps (2) and (4) (gray boxes in Fig. 7).

The inputs for this framework are the netlist, the sets of pins, and an existing pin assignment algorithm. In addition, the designer specifies d_{max} for each set of pins and the number of differential pairs. The output is a pin assignment for all nets, which respects the constraints for as many differential pairs as specified by the designer. This pin assignment is topologically very similar to a pin assignment created by the basic pin assignment algorithm (PAA) alone.

The individual steps as well as the indicated interactions (dashed arrows in Fig. 7) are presented in the following three subsections.

4.2 Combine Pin Pairs to Fat Pins

In order to generate the so-called fat pins (Step 1 in Fig. 7), valid DPPs are automatically determined among the original pins. This automatic selection of DPPs may be controlled by the designer by manually specifying an arbitrary number of DPPs. As outlined in this subsection, pins that cannot be combined to a valid DPP either ignored or are paired to invalid DPPs. As described above, these so-called invalid pin pairs cannot be used for differential signals in the final pin assignment. Nonetheless, allowing invalid DPPs at this point has a significant impact on the quality of the final pin assignment with differential pairs. The section presenting the experimental results (see below) shows the influence of invalid DPPs on the final pin assignment.

A maximum weighted matching (as shown in [11]) has to be calculated to find automatically as many DPPs as possible, with the least distance between the pins of the individual pairs. The implementation presented in [12] has a complexity of $O(p^3)$ (p number of pins). However, components with differential pairs have well-suited pin configurations such that DPPs can be determined effectively by heuristic, greedy algorithms. Therefore, we have developed two greedy algorithms, which are more time efficient than the slower optimal algorithms presented in [11][12].

The first algorithm (MOST_PAIRS) creates as many pin pairs as possible. The second algorithm (PREFERRED_PAIRS) focuses on pairs whose two pins are closest. The complexity of both algorithms is defined by the sorting algorithm, which is used to sort pins according to their distance to so-called *partner pins* and by the number of partner pins, respectively. Thereby, partner pins of one pin are those that are no further away than d_{max}. We use insertion sort, which has a complexity of $O(p^2)$ in the worst case. Still, the practical efficiency is much better since many pin pairs are of the same distance and most pins have the same number of partner pins.

Both algorithms first locate the next pin to be paired. In MOST_PAIRS, this is the pin with the least number of valid partner pins (distance $\leq d_{max}$) but at least one partner pin. In PREFFERED_PAIRS, it is the pin that has a valid partner pin that is closest amongst all possible pairs of pins. The located pin and its closest partner pin are then paired. This is repeated until no more pins can be paired. Fig. 8 (a) shows the automatically selected pin pairs for a small area array component.

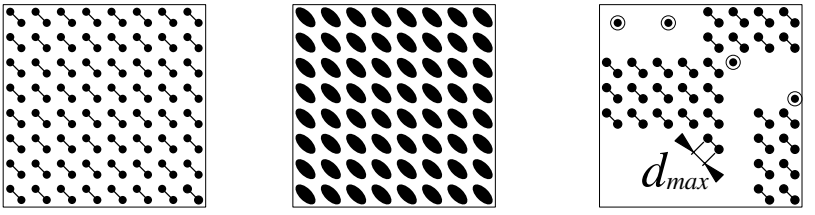


Fig. 8. (a) Pin pairs of a small area array component. The black dots denote pins, the line between two pins represents a pin pair. (b) Fat pins created from the selected pin pairs. (c) Pin configuration which prevents some pins (circled) to be used for differential pairs.

As depicted in Fig. 8 (c), there may be pins, which cannot be paired to valid DPPs. Either those pins are ignored or they are paired to invalid DPPs by the same two strategies described above thereby ignoring d_{max}. Thus, we can create four different selections of pin pairs, which eventually lead to different pin assignments with differential pairs:

- PREFERRED_PAIRS with only valid DPPs
- PREFERRED_PAIRS with valid and invalid DPPs
- MOST_PAIRS with only valid DPPs
- MOST_PAIRS with valid and invalid DPPs

Which of the four variants are used depends on the number of differential pairs required (see Section Integrating Fat Pin Assignment with PAA).

Next, a *fat pin* is created for each computed pin pair, regardless whether it is valid or invalid. The coordinate of a fat pin is the arithmetic mean of the coordinates of its original two pins (see Fig. 8 b). Except for its coordinates, the new fat pin inherits all characteristics, such as design rules, from the two original pins. At the same time, specific nets are combined in order to ensure an identical number of nets and fat pins.

4.3 Fat Pin Assignment

Following fat pin creation, all fat pins are treated just like conventional pins and are fed to any PAA that solves the pin assignment problem (Step 2 in Fig. 7). The resulting fat pin assignment is consequently transformed back to specify the assignment for the individual pins (Step 3 in Fig. 7).

The transformations illustrated in Fig. 9 are applied to each pin pair: Fig. 9 (a) shows the pin assignment task for two nets (lines) with two pins each (ending dots). A_1, A_2, B_1 and B_2 are the pins that are arranged in two separate sets. A_1 and A_2 are in the pin set named “From”. B_1 and B_2 are in the pin set named “To”. In Fig. 9 (b) pins A_1, A_2, B_1, and B_2 are transformed to fat pins A and B. Thus, only one of the two nets remains. Fig. 9 (c) shows the fat pin assignment by applying a PAA to the fat pin sets. Fig. 9 (d1) and (d2) denotes the two possibilities for the subsequent inverse transformation. Either pins A_1 and B_1 (Fig. 9.d1) or pins A_1 and B_2 (Fig. 9.d2) are assigned to the same net. We select the configuration with the smaller difference in the individual

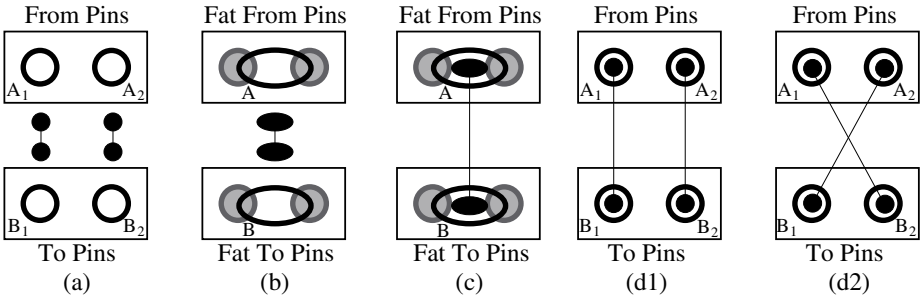


Fig. 9. Fat pin transformation and inverse transformation. (a) Pin assignment task for two nets. (b) Transformation from pins to fat pins. (c) Fat pin assignment. (d1) First alternative for inverse transformation. (d2) Second alternative for inverse transformation.

lengths and the shortest overall length of the flylines of both nets (which is (d1) in this example). This choice supports the matching of the net lengths of a differential pair.

If fat pins A and B are valid fat pins (A_1 and A_2, as well as B_1 and B_2, respectively, are no further apart than d_{max}), the two nets *can* be used for *either* a differential pair *or* for two single ended signals. Consequently, the number of nets which have all their pins assigned to valid fat pins defines the number of *possible differential pairs* in the final pin assignment because they can, *but need not*, be used as differential pairs.

4.4 Integrating Fat Pin Assignment with PAA

All unpaired pins and dropped nets are ignored and do not receive a pin assignment during fat pin assignment (Steps 1–3 in Fig. 7 and as described in the previous two subsections). To find the pin assignment for those pins and nets, the basic PAA is applied to original pins and nets (Step 4). The thus created pin assignment is integrated with the fat pin assignment to determine the final pin assignment with differential pairs (Step 5).

We propose two methods to integrate the two interim pin assignments. *Aggressive blending* creates more possible differential pairs than *defensive blending*, yet the results of defensive blending are better with respect to the objective function of the underlying PAA. Both methods can be used in combination with any of the four different methods to select pin pairs (see above), all together resulting in eight different pin assignments with differential pairs.

Aggressive Blending. To determine the pin assignment for all pins that did not receive a fat pin assignment (Fig. 10 c), the basic PAA is applied to these pins and nets (Fig. 10 d). The final pin assignment (Fig. 10 m) with differential pairs results from the combination of the fat pin assignment (see above and Fig. 10 k) with the pin assignment created by applying the PAA to the leftover pins and nets (Fig. 10 d). For aggressive blending, the fat pin assignment is applied to all pins that were paired, while the basic PAA is limited to the remaining pins and nets and is not aware of the fat pin assignment already created. Fig. 7 shows the flow of this algorithm. The limitation of the basic PAA to pins without a fat pin assignment is indicated by the

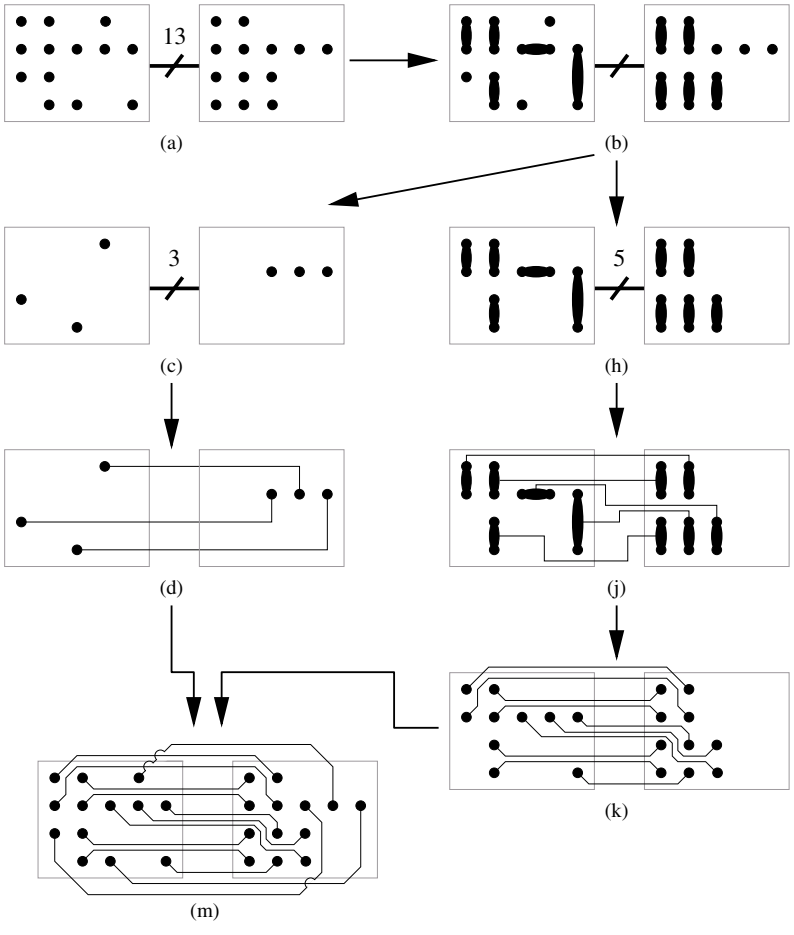
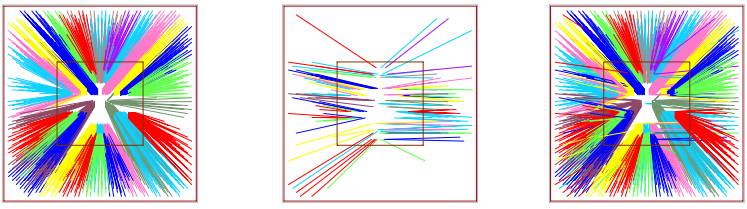


Fig. 10. Example pin assignment procedure with differential pairs using aggressive blending. (a) Pin assignment task with differential pairs for 13 nets. (b) Automatically selected fat pins. (c) Pins that were not paired to fat pins during (b). (d) Basic pin assignment for unpaired pins. (h) Automatically selected fat pins (leftover pins omitted). (j) Fat pin assignment. (k) Back-transformation of fat pin assignment to original pins. (m) The final pin assignment with differential pairs is the combination of the basic pin assignment (d) and the fat pin assignment (k).

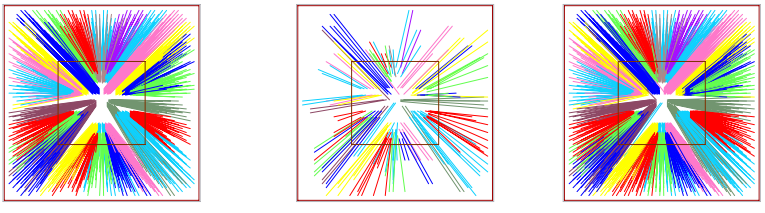
dashed arrow pointing from step 3 to step 4. A step-by-step example of pin assignment using aggressive blending is shown in Fig. 10.

Compared to defensive blending (described in the following subsection), the resulting pin assignment is of lower quality with respect to the objective function of the PAA, because the topologies of the two interim pin assignments differ in general. However, their better topological similarity during defensive blending results in fewer possible differential pairs, as shown in Figs. 11 and 12 and described in the following subsection.



(a) (b) (c)

Fig. 11. Pin assignment for a single chip module using aggressive blending. (a) Fat pin assignment. (b) Assignment of remaining pins and nets. (c) The final pin assignment with 468 possible differential pairs is the combination of (a) and (b).



(a) (b) (c)

Fig. 12. Pin assignment for a single chip module using defensive blending. (a) Fat pin assignment. (b) Assignment of remaining pins and nets. (c) The final pin assignment with 454 possible differential pairs is the combination of (a) and (b).

Defensive Blending. Defensive blending is an iterative process to improve the integratability of the fat pin assignment by *incrementally* adapting the selection of differential pin pairs. The advantage of defensive blending, in contrast to aggressive blending, is that all pins and nets are considered during the creation of the basic pin assignment. However, fewer pins are combined to fat pins. Compared to aggressive blending, this yields a better final pin assignment with respect to the basic objective function at the cost of decreasing the number of possible differential pairs in the final pin assignment.

In a first step, the basic PAA is applied to the original pins and nets (Fig. 13b). This pin assignment is then used as a reference throughout the following iterations. Next, pin pairs are selected as described above (Fig. 13c). Subsequently, all pins that have not been paired receive their pin assignment from the reference pin assignment of the first step (Fig. 10d). The pin assignment of those unpaired pins is final and is never changed again. For all remaining unassigned pins, the current selection of pairs is discarded and recreated (Fig. 13e) in order to optimize the selection. This process is repeated until all pins either received their final pin assignment or are paired (Fig. 10f and 10h).

All pins that are finally paired undergo fat pin assignment, and are then transformed back to their original pins (see section on fat pin assignment above, Fig. 13j and 13k). Hence, in defensive blending, the final pin assignment results from the

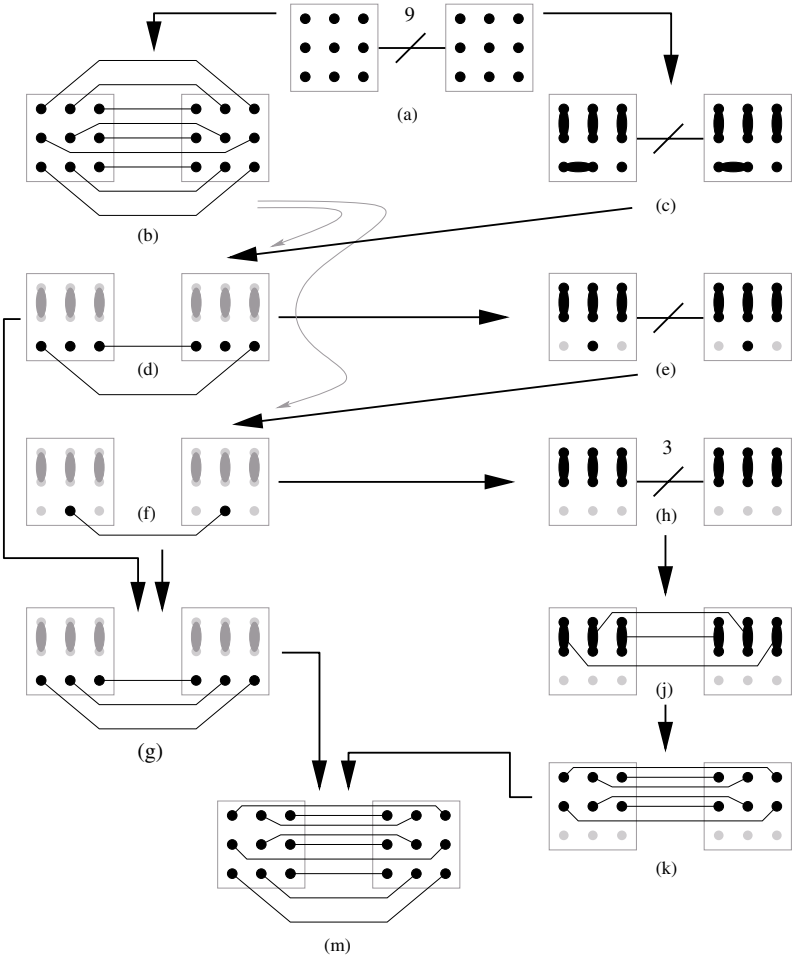


Fig. 13. Using defensive blending to obtain differential pairs of two 9x9 pin arrays. (a) Pin assignment task with differential pairs. (b) Basic pin assignment (PAA) without differential pairs. (c) Automatically selected fat pins. (d) Basic pin assignment for pins that were not paired to fat pins during (c). (e) Newly selected fat pins (pins with an assignment from (d) are ignored). (f) Basic pin assignment for pins that were not paired to fat pins during (e). (h) Newly selected fat pins (pins with an assignment from (d) or (f) are ignored). (j) Create fat pin assignment, since all remaining pins were paired to fat pins. (k) Back-transformation of fat pin assignment to original pins. (g) Pins with basic pin assignments from (d) and (f). (m) Final pin assignment with differential pairs is the combination of the basic pin assignment (g) and fat pin assignment (k). In this example the final pin assignment with differential pairs contains three pairs of nets usable for differential pairs, which were assigned during fat pin assignment (see (k)). It contains one more net pair that happens to be usable as differential pair, which was assigned during the basic PAA (see (g)).

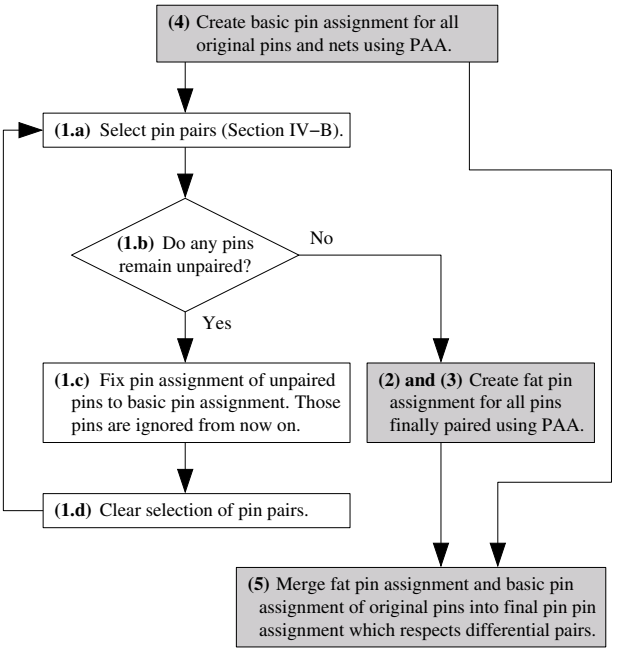


Fig. 14. Major steps of pin assignment with differential pairs using defensive blending. This figure extends Fig. 7 in which the interaction of the basic pin assignment and the process of selecting pin pairs is indicated as a dashed arrow from step 4 to step 1.

combination of the reference pin assignment for all finally unpaired pins and the back transformation of the fat pin assignment (Fig. 13m). Fig. 14 shows the flowchart of the defensive blending method.

Fig. 12 shows the two interim pin assignments (a) and (b) and the final pin assignment (c) created with defensive blending for the same single chip module as in Fig. 11.

Defensive blending and aggressive blending do not differ and give identical results in case all pins are paired to fat pins during the first iteration.

4.5 Summary

The eight possible combinations of methods for selecting fat pins and methods for integrating the interim pin assignments yield eight different pin assignments with differential pairs. They vary in the number of possible differential pairs and in the magnitude of changes compared to the basic pin assignment without differential pairs.

The number of possible differential pairs of each variant cannot be predicted exactly. Yet, experimental results show that the different variants can be ranked with respect to their quality and the number of possible differential pairs. In general, the quality of the pin assignment deteriorates with an increase in possible differential pairs. Therefore, the best pin assignment for a specific design is the one with just enough possible differential pairs. We find this pin assignment by sequentially applying the different variants starting with the one that creates best pin assignment results while providing the least differential pairs. Subsequently, pin assignment variants

with more and more differential pairs are created, until the best pin assignment for the design is found.

5 Experimental Results

The effectiveness of the presented methodology is proven by comparing pin assignments with differential pairs to those without differential pairs. First, results from PAAs (without differential pairs) applied to industrial designs are reported. Next, these PAAs are extended by the fat pin methodology to include differential pairs. The pin assignments are compared by means of *SHPWL*, *HPWL MATCH*, *AVG Flylines*, *STD Dev*, and the number of signal intersections.

If (x_{ai}, y_{ai}) and (x_{bi}, y_{bi}) are the coordinates of the two pins of net i, p is the number of nets in the pin assignment task and $dx_i = |x_{ai} - x_{bi}|$, $dy_i = |y_{ai} - y_{bi}|$, then the measurement metrics are defined as follows:

- *SHPWL*: The sum of the HPWLs (half perimeter wire lengths) of all nets.

$$SHPWL = \sum_{i}^{p} dx_i + dy_i$$

- *HPWL MATCH*: The additional length necessary to match the HPWL routing length of all nets. A lower value of *HPWL MATCH* indicates less routing effort, especially for busses.

$$HPWLMATCH = p \cdot max(dx_1 + dy_1, \ldots, dx_p + dy_p) - SHPWL$$

- *AVG Flylines*: The average net length in Euclidean geometry.

$$AVG\ \ Flylines\ \ = \frac{1}{p} \sum_{i}^{p} \sqrt{dx_i^{\,2} + dy_i^{\,2}}$$

- *STD Dev*: The standard deviation of the net lengths in Euclidean geometry, which similarly to *HPWL MATCH* evaluates the expected wiring effort necessary to match wiring lengths.

$$STD\ \ Dev\ \ = \frac{1}{p-1} \sqrt{\sum_{i}^{p} \left(AVG\ \ Flylines\ \ - \sqrt{dx_i^{\,2} + dy_i^{\,2}} \right)^2}$$

- The number of signal intersections is calculated as the number of intersections within the flylines of all nets.

In the following subsection, the differential pair methodology is compared with regular PAAs using the above metrics. In the subsection after the following, an investigation of the four proposed fat pin variants and the two proposed merging strategies is presented.

5.1 Quality of Fat Pin Methodology

The results presented in Table 1 are taken from a commercially fabricated IBM single chip module (SCM) that carries one die on top and is covered with a regular array of pins on the bottom side (1058 signal pins, 1058 power/ground pins). The pin assignment

algorithms are extended by our fat pin methodology and used to create an assignment with differential pairs of die signal pins to bottom signal pins.

The used PAAs have the following objectives (a detailed description of these algorithms can be found in [7]):

1. Heuristic to minimize *HPWL MATCH* and *STD Dev*
2. Heuristic to minimize signal intersections within busses for a specified direction of fanout.
3. Same as 1. with subsequent removal of signal intersections.
4. Same as 2. with subsequent removal of signal intersections.
5. Minimum *AVG Flylines*.
6. Minimum *SHPWL*.
7. Concurrent minimization of *SHPWL* and signal intersections.

All pins of design SCM are transformed to valid fat pins by the PREFERRED_PAIRS algorithm accordingly to Fig. 8 (a, b). As a result, the final pin assignment is completely defined by the fat pin assignment and no merging of interim pin assignments is necessary. In addition, the creation of fat pins by the MOST_PAIRS algorithm returns identical results. Hence, there are two relevant pin assignments with differential pairs for each PAA. Firstly, the PAA unintentionally allows for a significant number of differential pairs. Those pairs result from *parallel pins* (see section on differential pairs) with a distance smaller than d_{max} (d_{max} is equal to the diagonal pin grid in our experiments, Fig. 8 c). Secondly, the pin assignment created by fat pins allows all nets to be used as differential pairs.

For each PAA 1–7, Table 1 compares the pin assignment created by the basic PAA and its differential pair extension. Absolute values are given for the number of possible differential pairs (#Diff Pairs), intersections of flylines and the runtime. For measures *SHPWL*, *HPWL MATCH*, *AVG Flylines* and *STD Dev* the pin assignment results with differential pairs are given as a the percentaged difference (Δ) to the respective result of the basic PAA. Table 1 shows that the impact of fat pins on the objectives of the basic PAAs is marginal. One exception are signal intersections estimated as intersections of flylines, which increased considerably. Yet, closer inspection shows that intersections are introduced in places where they can easily be resolved by the router

Table 1. Experimental pin assignment results of design SCM without and with differential pairs (/o | w/ DP) using the seven PAAs 1–7 with different objectives, as listed in the text. Percentage values denote the difference between the basic PAA and its differential pair extension with positive percentages indicating an increase of the respective value.

PAAs	#Diff Pairs (/o | w/ DP)	Δ*SHPWL*	Δ*HPWL MATCH*	Δ *AVG Flylines*	Δ *STD Dev*	Intersect. of Flylines (/o | w/ DP)	Runtime in s (/o | w/ DP)
1.	367 | 529	+0.17%	+1.3%	+0.17%	+0.34 %	6159 | 7000	<1 |<1
2.	160 | 529	-0.52%	-6.4%	+0.70%	-2.4 %	44686 | 46331	<1 |<1
3.	313 | 529	+0.17%	-6.7%	+0.18%	+1.0 %	0 | 1633	1 |<1
4.	294 | 529	+0.00%	-3.2%	+0.01%	-0.57 %	0 | 1551	<1 |<1
5.	283 | 529	+0.13%	+1.6%	+0.12%	-0.28 %	80 | 1619	8 | 1
6.	93 | 529	+0.28%	+5.2%	-0.02%	-0.92 %	27955 | 27212	3 |<1
7.	298 | 529	+0.13%	-6.1%	+0.09%	-0.29 %	0 | 1564	10 | 1
Absolute Average	258 | 529	0.20%	4.4%	0.18%	0.83 %	11269 | 12416	3 |<1

because they are either close to the endpoints of nets or the intersecting nets are almost parallel, thereby not affecting routability.

5.2 Comparison of Fat Pin Variants

In order to compare the four different variants of selecting pin pairs (PREFFERED_PAIRS without invalid DPPs, PREFFERED_PAIRS with invalid DPPs, MOST_PAIRS without invalid DPPs, and MOST_PAIRS with invalid DPPs) and the two merging strategies (aggressive blending and defensive blending), the results of a multi chip module (MCM) of an IBM industrial design are presented. This MCM has seven dies on top, 2930 signal pins and 2112 power/ground pins (see Fig. 1). The arrangement of the pins is irregular such that 68 of the signal pins cannot be used as differential pin pair because no other signal pin is closer than d_{max}. Additional 190 signal pins are not usable for differential pin pairs because these pins are in 190 "islands of pins" (which are further apart than d_{max}) with each having an odd number of pins (see Fig. 8 c).

The PAA 5, which minimizes the overall length of the flylines, is used to create the assignment of bottom signal pins (Fig. 1 b) to die signal pins (Fig. 1 a). The eight variants of the fat pin methodology (#1–#8 in Table 2) and the basic PAA alone (#0 in Table 2) deliver nine pin assignments with differential pairs. The results (Table 2, Figs. 15 and 16) show that along with an increasing number of available differential pairs, the length of the flylines, which is the objective of the used PAA 5, slightly increases. In six out of eight cases, the increase stayed below 0.25% (with no measurable increase in routing lengths when comparing the actual routing results with and without differential pairs). For variants #7 and #8 (see # in Table 2) the increase in lengths are 5.4% and 1.9%, which resulted in a similar increase in actual final routing length (Cadence SPECCTRA autorouter).

The results show that aggressive blending (#5–#8) yields more differential pairs than defensive blending (#1–#4). Furthermore, the selection of pin pairs by MOST_PAIRS generally gives more differential pairs than PREFERRED_PAIRS.

Table 2. Results of differential pair pin assignment of the eight different fat pin variants (#1–#8) and of the PAA 5 (#0) for design MCM. The used PAA 5 minimizes the overall length of the flylines. The names of the algorithms PREFERRED_PAIRS and MOST_PAIRS are abbreviated as PREF_P and MOST_P, respectively.

#	Blending Method	Invalid DPPs	Selection of Pin Pairs	Number of Diff. Pairs	*SHPWL*	*AVG Flylines*	*STD Dev*	Intersections of Flylines	Runtime in Sec
#0	n/a	n/a	*None*	376	46321	**11.84**	8.49	209	243
#1	defensive	no	PREF_P	237	46351	11.85	8.48	2365	245
#2			MOST_P	243	46350	11.85	8.48	2437	244
#3		yes	PREF_P	1127	46452	11.87	8.46	4120	277
#4			MOST_P	1081	46430	11.86	8.52	3661	266
#5	aggressive	yes	PREF_P	1143	46410	11.86	8.48	3808	34
#6			MOST_P	1217	46440	11.86	8.51	4052	32
#7		no	PREF_P	1251	48182	12.48	9.25	17008	20
#8			MOST_P	1336	47110	12.06	8.54	9750	23

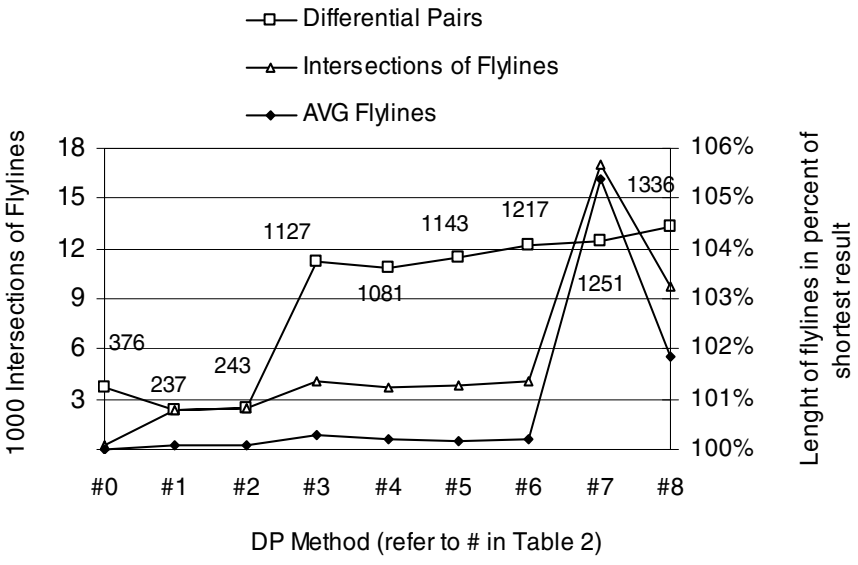
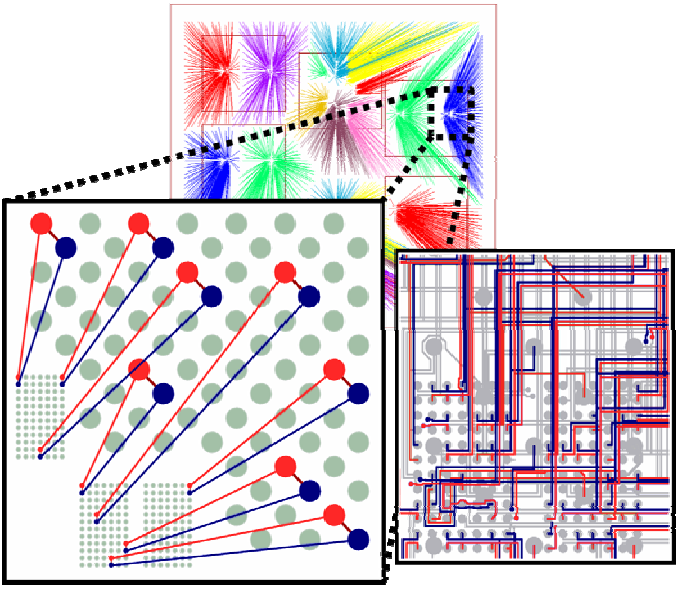


Fig. 15. The impact of the eight different fat pin variants (#1–#8) on the number of differential pairs, flyline intersections and overall flyline lengths (results of design MCM, see also Table II)



(a) (b)

Fig. 16. Details of a differential pair pin assignment. As illustrated by the shown subset of flylines in (a), each differential pair is assigned adjoining chip and MCM pins (smaller and larger dots) with distances of less than d_{max}. The final routing result of differential pairs is shown in (b). Note that (a) contains only a small subset of differential pairs, non-differential pairs are omitted for simplicity.

The effect of invalid DPPs depends on the method of blending. For defensive blending, the number of created differential pairs is drastically increased by using invalid DPPs (variants #3 and #4), while the quality with respect to the basic objective slightly decreases. (Without invalid DPPs, many pins are not transformed to fat pins and receive their basic pin assignment, while only paired pins are treated via fat pin assignment.)

For aggressive blending, invalid DPPs (#5 and #6) decrease the number of created differential pairs, while improving the quality with respect to the basic objective. This is because each pair of nets that is assigned at least one invalid DPP cannot be used for a differential pair. However, more pins are considered during fat pin assignment, hence, the overall pin assignment quality is better.

The number of created differential pairs by each variant is not predicable. Therefore, we sequentially apply variant #0 (pin assignment with the best quality and least possible differential pairs) followed by variants #3 through #8 (pin assignment with the least quality and the most possible differential pairs) until the pin assignment with enough differential pairs and the best quality achievable for the specific design is found. This methodology has been proven effective in numerous industrial examples.

6 Limitations and Outlook

Conventional pin assignment algorithms that minimize the overall lengths of flylines, the overall Manhattan lengths and the standard deviation of those lengths can easily be combined with the fat pin methodology without solution degradation. Pin assignment algorithms with the objective of minimum signal intersections have a limited compatibility to the fat pin methodology. This is due to the difference in coordinates of the fat pin and its two original pins that can lead to intersections near the end of the routing path. However, these additional intersections are in places where they are easily resolved by the final router and thus, do not affect routability.

The presented algorithms to select pin pairs are based on the distances of pins. Pin pairs have been specified manually if specific DPP patterns are needed (e.g., for specialized differential pair connectors). In the future, pairing algorithms must include more complex constraints than only one spacing rule. Due to the modularity of our fat pin methodology, the presented pairing algorithms can easily be replaced with any other extended method for pairing.

One example for future design challenges is the signaling method presented in [13]. It requires four nets and their pins to be handled in one group. Our methodology can easily be modified for pin assignments suitable for this signaling method by selecting groups of four pins that are to be represented by one fat pin.

7 Conclusions

In this chapter, a universal differential pair methodology that is applicable to all algorithms or manual processes that solve the pin assignment problem has been presented. This is the first algorithmic approach that includes differential pair constraints during pin assignment. It has been shown that it has only a minor effect on the quality of the

underlying basic pin assignment algorithm (PAA). This has been verified not only during pin assignment but also by considering the actual routing results.

The fundamental principle of the presented solution is that two nets, which *can* be used for a differential pair (because their parallel pin pairs meet the spacing rules), do *not need* to be used for a differential pair. Instead, they can also be used for any two single ended nets. Based on this observation, the fat pin transformation approximately halves the number of pins that have to be considered. Thereby, the complexity of the pin assignment problem is significantly reduced, while still allowing for near optimal solutions. A pin assignment of differential pairs can be retrieved from this reduced number of pins by PAAs that originally do not respect differential pairs.

The methodology consists of different algorithms for selecting differential pin pairs (DPPs) and integrating the fat pin assignment. They can be used in different combinations to produce similar pin assignments with different numbers of possible differential pairs. The number of created possible differential pairs by the variants cannot be predicted exactly. Yet, specific variants create more differential pairs than others while in general the quality of the pin assignment decreases with an increasing number of differential pairs. Therefore, in order to find the best pin assignment with differential pairs for a specific design, the variants are executed sequentially starting with the one producing the least differential pairs, until the first pin assignment with sufficient differential pairs is found.

Based on this add-on methodology, any present or future algorithms for the pin assignment problem can easily be extended to include differential pairs. The presented differential pair methodology is in use in the industrial design flow at IBM. Here it has shown its robustness and quality combined with a minimum of interference with the design flow that had already been established.

References

1. Nakagawa, K., Watanabe, M., et al.: Giga-hertz electrical characteristics of flip-chip BGA package exceeding 2,000 pin counts. In: 54th Conference Proceedings of Electronic Components and Technology, vol. 1, pp. 334–341 (2004)
2. Yuan, W., Pang, K.H., et al.: Electrical analysis and design of differential pairs used in high-speed flip-chip BGA packages. In: 17th International Zurich Symposium on Electromagnetic Compatibility 2006, pp. 578–581 (2006)
3. Westra, J., Groeneveld, P.: Post-placement pin optimization. In: IEEE Computer Society Annual Symposium on VLSI, pp. 238–243 (2005)
4. Westra, J., Groeneveld, P.: Towards integration of quadratic placement and pin assignment. In: Proceedings of IEEE Computer Society Annual Symposium on VLSI 2005, pp. 284–286 (2005)
5. Xiang, H., Tang, X., Wong, D.F.: An algorithm for integrated pin assignment and buffer planning. In: Proceedings of 39th Design Automation Conference, pp. 584–589 (2002)
6. Xiang, H., Tang, X., Wong, D.F.: Min-cost flow-based algorithm for simultaneous pin assignment and routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22(7), 870–878 (2003)
7. Meister, T., Lienig, J., Thomke, G.: Novel Pin Assignment Algorithms for Components with Very High Pin Counts. In: Proceedings of Design, Automation and Test in Europe, DATE 2008, pp. 837–842 (2008)

8. Chen, S., Tseng, W., Yan, J., Chen, S.: Printed circuit board routing and package layout codesign. In: APCCAS 2002, vol. 1, pp. 155–158 (2002)
9. Kubo, Y., Takahashi, A.: A global routing method for 2-layer ball grid array packages. In: Proceedings of the 2005 ISPD, pp. 36–43 (2005)
10. Yu, M., Dai, W.W.-M.: Pin assignment and routing on a single-layer pin grid array. In: Proceedings of the ASP-DAC 1995/CHDL 1995/VLSI 1995, pp. 203–208 (1995)
11. Galil, Z.: Efficient algorithms for finding maximum matching in graphs. ACM Comput. Surv. 18(1), 23–38 (1986)
12. Gabow, H.N.: An efficient implementation of Edmonds algorithm for maximum matching on graphs. Journal of the ACM 23(2), 221–234 (1976)
13. Choi, S., Lee, H., Park, H.: A three-data differential signaling over four conductors with pre-emphasis and equalization: a CMOS current mode implementation Solid-State Circuits. IEEE Journal of Solid-State Circuits 41, 633–641 (2006)
14. Sherwani, N.A.: Algorithms for VLSI Physcial Design Automation. Kluwer Academic Publishers, Dordrecht (1998)
15. Westra, J., Groeneveld, P.: Towards integration of quadratic placement and pin assignment. In: IEEE Computer Society Annual Symposium on VLSI, Proceedings, pp. 284–286 (2005)
16. Xiang, H., Tang, X., Wong, M.: Min-cost flow-based algorithm for simultaneous pin assignment and routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22, 870–878 (2003)
17. Lienig, J.: Layoutsynthese elektronischer Schaltungen (Algorithms in Physical Design). Springer, Heidelberg (2006)

Analysis and Design of Charge Pumps for Telecommunication Applications

Vassilis Kalenteridis, Konstantinos Papathanasiou, and Stylianos Siskos

Electronics Laboratory, Physics Department,
Aristotle University of Thessaloniki 54124 Thessaloniki, Greece

Abstract. This chapter addresses modern telecommunication integrated circuits from the synthesizer focal point; in particular it concentrates at the analysis and the design of integrated charge pump circuit blocks. It presents an overview of charge pump topologies in addition to a coherent analysis of the associated benefits and shortcomings of all circuit alternatives. Moreover a novel favorable charge pump combining current steering techniques with well utilized unity gain buffers in a novel, noiseless feedback scheme, is introduced to improve on switching speed, inherent charge pump ac noise, dead-zone interval, therefore overall steady state aliased loop noise; while on the other hand this charge pump exhibits superb DC matching characteristics in a wide output voltage range. Furthermore a well documented estimation of the active devices that contributes mostly to the overall charge pump noise performance is presented. Also an associated mathematical analysis concerning the frequency content of the charge pump noise current is given. This proposed topology manifests its applicability to charge pump alternatives, as it is demonstrated by the associated simulation results from a 0.18μm design. Because of the low-noise and accurate properties of this improved charge pump, it is ideally suited to modern telecommunication standards synthesizer realizations.

1 Introduction

Fully monolithic Phased-Locked Loops (PLLs) are essential building blocks, widely used in modern communication or complex digital systems [1-8]. A PLL based on a charge pump is often preferred over other synthesizer alternatives, because it exhibits a wide capture range with no systematic phase offset and arguably provides one of the simplest and most effective design platforms [9-14]. The Charge Pump based PLL also provides flexible design tradeoffs by decoupling various design parameters such as the loop bandwidth, damping factor and lock range [22]. Figure 1 shows a typical implementation of a charge pump based PLL. It consists of a Phase/Frequency Detector (PFD), a Charge Pump (CP), a Loop Filter (LF), a Voltage Controlled Oscillator (VCO) and a divider. The most widely used PFD generates a pair of digital pulses corresponding to the phase/frequency error between the reference clock f_{ref} and the VCO output, by comparing the positive (or negative) edges of the two inputs. The CP circuit converts the digital pulses into an analogue current which is consequently integrated producing a voltage on the passive (or active) loop filter. This voltage drives

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 43–60, 2010.

the VCO circuit block which in turn produces the synthesized frequency of operation as it is demanded by the system specification.

However, some non-idealities of the CP such as DC mismatch of the charging/discharging currents and glitches degrade the performance of the overall loop. Moreover the noise of the charge pump is the dominant close-in phase noise contributor in a PLL [15]. Several charge pump implementations have been proposed in the associated literature [16-18]. In [16, 17] an opamp has been used in order to keep the dc mismatch current, and hence the resultant phase offset at a minimum level and improve the overall performance. This in effect adds significant noise contribution at the output of the proposed charge pump due to the increased gain introduced by the opamp. Others [18] assume that the Up and Dn signals from the PFD that drive the charge pump switches could not be simultaneously high, to avoid the dc mismatch between the pump-up and pump-down currents. This is a fallacy because at lock both the Up and Dn signals are high for a given short time to ensure the elimination of the PLL dead-zone, which if present will degrade significantly the in-band noise suppression characteristic function of the PLL.

The objective of this chapter is the design of an improved single-ended low noise charge pump with low dc mismatch current, high voltage output range and programmable gain. The second section depicts some typical charge pump architectures either for single-ended or differential design, along with the advantages and disadvantages of each category. In the third section a detailed analysis of the improved charge pump is presented and compared to other alternative designs. Also the noise contribution of the improved charge pump active devices to the total output noise is given with the appropriate mathematical noise analysis. In the fourth section the simulation results from three alternative methods (DC, PSS and Pnoise) are presented, over temperature and process corners for the charge-pump key specifications to signify the applicability of the overall approach. Finally the key concluding remarks of this chapter are given in the last section.

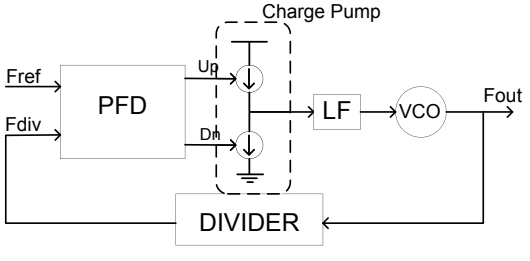


Fig. 1. Block level diagram of a charge pump based PLL

2 CMOS Charge Pump Architectures

2.1 Single-Ended Charge Pump Architectures

Single-ended charge pump circuits are an elegant approach to system flexibility, low-power consumption, minimization of pads and external components, or area. The output

current of the charge pump can be as high as 4.5mA [23] at lock to provide better spur performance thus less leakage current and to have high SNR for low noise contribution to the PLL, while this current can be significantly more while the PLL is in the tracking period, to improve on settling time. By using tri-state operation, the current consumption of the charge pump is limited to a few hundred μA depending on the reference clock frequency and the delay of the PFD. Figure 2 shows some typical single-ended charge pump topologies.

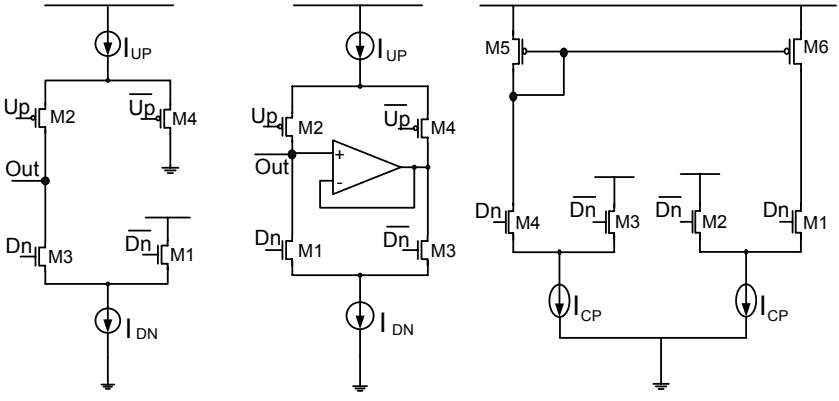


Fig. 2. Single-ended charge pump architectures: a) with current steering switch, b) with unity gain active amplifier and c) with NMOS switch only

Figure 2a shows a charge pump utilizing a current steering switch. This structure provides high speed switching for a single-ended charge pump, since the switching time is improved by the current steering properties of the associated switching pair (M1-M3 and M2-M4). Another charge pump approach utilizing current steering with an active amplifier [24-25] is shown in figure 2b. This unity gain amplifier, buffers the voltage at the output node forcing the drain voltage of the current sources I_{DN} and I_{UP} to be the same when M1 and M2 are on or when they are off. This reduces the charge sharing effect, when the switch is turned on. This architecture ensures fast transient response through current steering, reduces the effect of any parasitic capacitance, at the expense of extra current. Finally, in figure 2c the inherent mismatch of pmos and nmos transistor is avoided by using only nmos switches [26]. Since the current does not flow in the current mirror, (M5 and M6), when the UP switch is turned off, the current mirrors still limit the performance unless large current is used [3].

2.2 Differential Charge Pumps

A fully differential charge pump has several advantages over the conventional single-ended charge pump [27-28]. Firstly, the switch mismatches between nmos and pmos transistors do not substantially affect the overall performance. This relaxes the matching requirement between the two type of transistors. Secondly, the differential charge pump has only nmos switching transistors thus the inverter delays for the Up and Dn signals are fully symmetric and therefore do not generate any offset. Thirdly, this

configuration doubles the range of the output voltage compliance compared to the single-ended charge pump. This is a significant advantage for low voltage operation, since the limited output voltage range of the charge pump makes it difficult for the VCO to meet the specified tuning range. Fourthly, the differential output stage is less sensitive to the leakage current, since the leakage current behaves as a common-mode offset at the dual output stages. Lastly, the use of two on-chip loop filters provides better immunity to the supply, ground and substrate noise, while the lack of bond wire inductors facilitates faster switching speeds and reduces transient oscillations. However, these advantages can only be achieved at the cost of extra area due to the use of two loop filters, common-mode feedback circuitry [3], higher noise levels and power consumption imposed by the potential introduction of an active filter and most importantly the flexibility of altering the overall PLL loop characteristics by changing the loop filter should this prove desirable.

3 Improved Charge Pump Design

The improved accurate low noise charge pump is shown in figure 2. It is a single-ended tri-state charge pump with programmable gain. This topology exhibits improved switching speed, since all nodes are precharged to the resultant operating points and the current is either steered to the output or to the unity-gain buffer. The two opamps O_{P1} and O_{P2} are used in order to minimize the DC mismatch current that will be introduced by the output voltage variation. The O_{P1} and O_{P2} inputs are connected at the drains of the corresponding transistors as it is shown in figure 3, while the opamp outputs drive the gates of transistors M3 and P1 respectively.

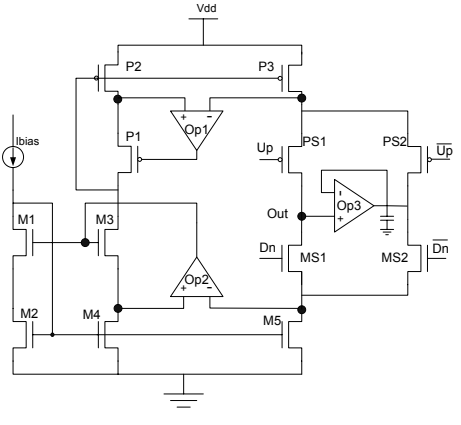


Fig. 3. Improved Charge Pump Circuit

3.1 Characteristics of the Improved Charge Pump Approach

In the improved circuit transistors M1, M2, M3 and M4 compose a cascode current mirror with increased output resistance and minimized channel length modulation.

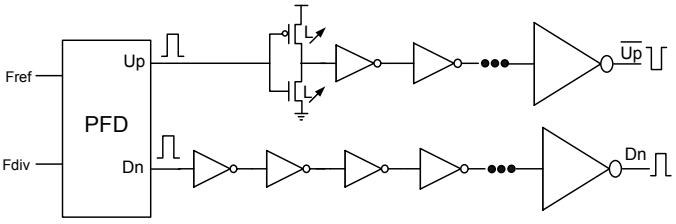


Fig. 4. Buffer chain between PFD and CP

This cascode connection offers the advantage of low voltage operation for the charge pump. The bias current (Ibias) from the input branch is mirrored to the output branch where P3 and M5 act as current sources. The current source mirroring ratio is 4, which means that the output current is four times larger than Ibias. MS1 and PS1 are the transistor switches which are driven by the Dn and Up signals from the PFD. When the Up signal is low, the PS1 switch is turned on and the current Iup from P3 charges the loop filter capacitor, increasing the output voltage. On the other hand when the Dn signal is high, the MS1 switch is turned on and the output voltage is decreased by the discharging current Idn that flows through M5. Transistors MS2 and PS2 are the switches which are driven by the complementary Up and Dn signals, providing a constant current flow path when the switches MS1 and PS1 are off. This implies a fast switching operation at the expense of increased power consumption.

When the loop is locked, both switches are on for a small fraction of the time. At lock both MS1 and PS1 have to switch on and off simultaneously to reduce the noise introduced in the loop and the magnitude of the $2 \cdot f_{ref}$ and consequent spurs. For this reason a buffer with a timing synchronization scheme which constitutes from two chains is used; the first chain is used to generate the Up signal and the second to generate the Dn signal. This buffer placed between PFD and CP, as shown in figure 4. The scaling ratio for the inverters is chosen to be close to 4 [19], in order to achieve the best power, speed and area trade-off. Also the channel length L, of the nmos and pmos transistor in the first inverter of the Up signal is increased to equalize the delay between the two timing control signals introduced by the asymmetry of the two chains [7]. Synchronization can also be achieved by using the two paths of the chain, where the first one includes an extra inverter compared to the second one and introduce an active resistor (a transmission gate adequately dimensioned) in the second path. In addition to that the dimensions of the switches must be properly sized, in order to turn on and off simultaneously.

An important advantage of the improved CP circuit is the low DC mismatch between the pump up and pump down currents. The two opamps O_{P1} and O_{P2} are used in order to minimize this DC offset current. As it is shown in figure 3, the two inputs of O_{P1} and O_{P2} are connected to the drains of P2-P3 and M4-M5 transistors respectively, forming a closed loop. If the output voltage increases to lock at a higher frequency, then the voltage at the drain of M5 increases as well. Because of the O_{P2} the same voltage is forced on the drain of M4. Likewise O_{P1} forces the voltage to the drains of P2 and P3 to be almost the same. As a result, the same amount of current flows between the two branches, for a wide output voltage dynamic range.

The dimensions W/L of the current source transistors that is M5 and P3 are chosen in such a way to minimize the current mirroring mismatch from the input to the output branch. Also the systematic current variation due to any residual in the V_{DS} despite the presence of the opamps has been remedied by choosing large gate lengths for the current source transistors. In addition the PS1 and MS1 transistor switches, which operate in the linear region, have been designed with minimum gate length, in order to achieve maximum output voltage range.

Moreover the unity gain amplifier O_{P3} plays an important role, since it sets the voltage at the drain of the switches PS1 and MS1 at the output node. Thus the charge sharing effect becomes minimal when the switches turned on. It also increases the switching-speed of the charge pump due to current constant flow from P3 to M5, even when the PLL is locked. Hence the charge and discharge of the parasitic capacitances at the drain node of P3 and M5 is avoided. This increases switching speed, therefore the dead zone and the resultant noise contribution is reduced during the lock condition at the expense of a small current consumption introduced by O_{P3}, since it only needs to source or sink a small P and N mismatch current. Finally a compensation capacitance has been added at the output of the amplifier, to increase the phase margin, as denoted by an associated stability analysis. Moreover it acts as a charge reservoir during switching transitions.

3.2 Comparison to Alternative Approaches

The significant improvement compared to similar, alternative charge pump implementations [16, 17] is in essence due to the fact that the output of the opamps drives the gates of the cascode transistors and not the gates of the current source transistors directly. In the improved approach P1 is connected as a source follower, resulting to a significant reduction of the O_{P1} noise contribution at the charge pump output.

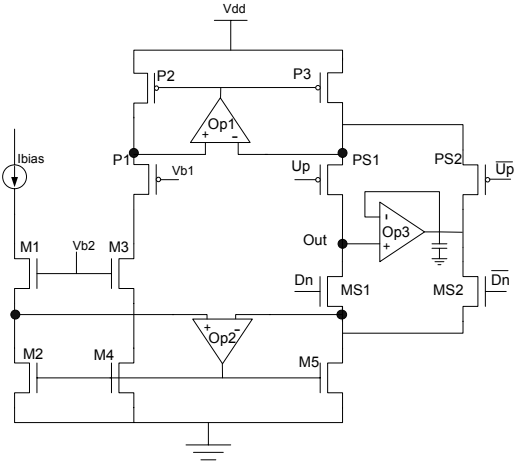


Fig. 5. Opamp drives directly the gates of P2 and P3

For example in the case where the O_{P1} drives directly the gates of P3 and P2 transistors, as shown in figure 5, the noise current i^*_{out1} generated at the output of P3 is given by the following equation:

$$i^*_{out1}{}^2 = gm^2_{P3} \cdot V_n^{*2} \tag{1}$$

where Vn^{*2} represents the output referred voltage noise of the opamp O_{P1} (both flicker and thermal) and gm is the transconductance of the current sinking transistor. From the above equation it can be seen that the noise current is the product of the noise voltage and the transconductance of P3. In our case, as shown in figure 3, the output of O_{p1} is connected to the gate of P1 instead of P3. Taking into account that P2 acts as a current source with a finite large output resistance r_o, the noise current of P2 is equal to:

$$i^*_{n2}{}^2 = \frac{V_n^{*2}}{r_o^2} \tag{2}$$

This current produces a noise voltage at the gate of P2 equal to

$$v^*_{n2}{}^2 = \frac{i^*_{n2}{}^2}{gm^2_{P2}} \tag{3}$$

Thus the noise current that appears to the output of P3 is given by the equation:

$$i^*_{out2}{}^2 = gm^2_{P3} \cdot v^*_{n2}{}^2 = \frac{gm^2_{P3} \cdot V_n^{*2}}{gm^2_{P2} \cdot r_o^2} \tag{4}$$

Therefore the ratio of the improved charge-pump over the one in [16] is given by the equation:

$$\frac{i^*_{out2}{}^2}{i^*_{out1}{}^2} = \frac{1}{gm^2_{P2} \cdot r_o^2} \tag{5}$$

For example if common modern transistor values r_o=72kΩ and g_m=2.84mS are subsidized in the above equation, a significant reduction by 45dB, of the Op1 induced noise at the output is obtained.

3.3 Analysis and Estimation of Noise Contributors of the Improved Charge Pump

The analytical estimation of the noise contribution, from the charge pump transistors is presented in this section. As it is well known the flicker (1/f) and thermal (white) noise from the active devices are the dominant noise sources that affect the overall noise performance of the charge pump. The noise plot of an active device (MOS or Bipolar transistor) is shown in figure 6, which has only two distinctive regions; thermal noise and 1/f region. The 1/f noise corner is in the vicinity of 500kHz to 1MHz

for a sub-micron CMOS technology and it is in the vicinity of 1kHz to 10kHz for bi-polar transistor [21].

There are three different combinations for the charge pump switching conditions which are given in the following table:

Table 1.

Signals	MS1	PS1
Up(low), Dn(high)	On	On
Up(low), Dn(low)	Off	On
Up(high), Dn(high)	On	Off

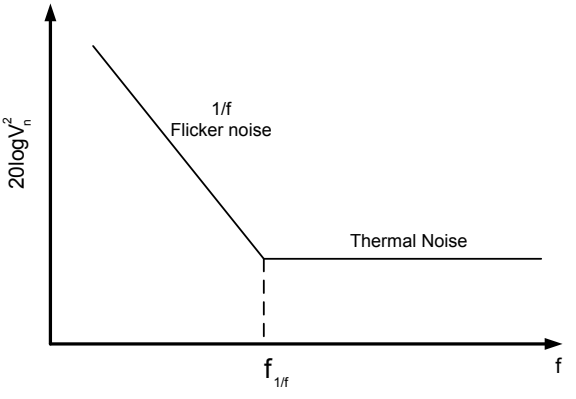


Fig. 6. Noise characteristics of a MOS transistor at a fixed bias voltage

1. PS1 on and MS1 on

The first condition is when the two transistor switches are both on for a small fraction of time corresponding to the locked condition of the loop.

For the flicker noise estimation a noise voltage source is placed at the appropriate gate device and the resultant noise current is calculated at the output of the transistor. So in this condition the transistors that affect the total noise of the charge pump are: M2, M4, M5, P2, P3. g_{mni} and g_{mpi} are the transconductances for nmos and pmos transistors respectively, where index i indicates the number of the corresponding transistor. For the noise calculation the flicker noise is easily modeled as an equivalent voltage source $V^{*}{}_{n}^{2}$ in series with the gate of a MOS transistor and roughly given by the following equation

$$V^{*}{}_{n}^{2} = \frac{K_f}{C_{ox} \cdot W \cdot L} \cdot \frac{1}{f} \tag{6}$$

where K_f is a process-dependent constant on the order of 10^{-25} V^2F, C_{ox} is the oxide capacitance, W and L are the width and length of the transistor respectively. The inverse dependence of (7) on W, L suggests that to decrease 1/f noise, the device area must be increased.

Taking into account that $W_{M2}=W_{M4}$, $W_{P2}=4\cdot W_{M4}$ and $W_{P3}=4\cdot W_{P2}=16\cdot W_{M2}$ the total output noise current is given by the following expression:

$$\begin{aligned} i^*_{n,out}{}^2 &= g_{mp3}{}^2\cdot V^*_{n4}{}^2 + g_{mp3}{}^2\cdot V^*_{np3}{}^2 + g_{mp3}{}^2\cdot V^*_{np2}{}^2 + g_{mn5}{}^2\cdot V^*_{n5}{}^2 = \\ &= g_{mp3}{}^2\cdot V^*_{n4}{}^2 + g_{mp3}{}^2\cdot V^*_{np3}{}^2 + g_{mp3}{}^2\cdot V^*_{np2}{}^2 + g_{mn5}{}^2\cdot V^*_{n5}{}^2 = \\ &= i^*_{n2}{}^2 + \frac{1}{4}i^*_{n2}{}^2 + \frac{1}{4}i^*_{n2}{}^2 + \frac{1}{16}i^*_{n2}{}^2 \Rightarrow i^*_{n,out}{}^2 = \frac{25}{16}i^*_{n2}{}^2 \end{aligned} \quad (7)$$

where $i^*_{n2}{}^2$ is the output reffered noise current of transistor M2. It should be noted that the first two terms which are the summation of the noise current from M2 transistor are cancelled by the third term; the negative sign of this third term comes from the fact that the two noise currents $g_{mn5}{}^2\cdot V^*_{n2}{}^2$ and $g_{mp3}{}^2\cdot V^*_{n2}{}^2$ are fully correlated with a phase difference of 180 degrees to each other (for the actual CP switching frequencies). This is because the P3 transistor sources current while M5 transistor sinks the same noise current. Moreover these transistors have equal transconductances, since the ratio of their mobilites is equal to the ratio of the dimensions W/L for the same current.

2. PS1 Off and MS1 On

In the second condition only M2 and M5 transistors are taken into account since they affect the charge pump noise and the total output noise current is given by the following expression:

$$i^*_{n,out}{}^2 = g_{mn2}{}^2\cdot V^*_{n2} + g_{mn5}\cdot V^*_{n5}{}^2 = i^*_{n2}{}^2 + \frac{1}{4}i^*_{n2}{}^2 = \frac{5}{4}i^*_{n2}{}^2 \quad (8)$$

The diode connected M2 produces a noise current $i^*_{n2}{}^2$ which is mirrored at the output of the charge pump. M5 acts as a current sink producing also a noise current which is four times smaller than the noise current of M2, because its width is four times larger than the width of M2, as depicted in (7).

3. PS1 On and MS1 Off

In the last operating condition the output noise current consists of the noise currents of the M2, M4, P2 and P3 transistors. M5 does not contribute any noise at the output because the MS1 switch is in the off state. Taking into account that $W_{M2}=W_{M4}$, $W_{P2}=4\cdot W_{M4}$ and $W_{P3}=4\cdot W_{P2}=16\cdot W_{M2}$ the total output noise current is given by the following expression:

$$i^*_{n,out}{}^2 = g_{mn2}{}^2 \cdot V^*_{n2} + g_{mn4} \cdot V^*_{n4}{}^2 + g_{mp2}{}^* \cdot V^*_{np2}{}^2 + g_{mp3}{}^2 \cdot V^*_{np3}{}^2 = \\ = i^*_{n2}{}^2 + i^*_{n2}{}^2 + \frac{1}{4} i^*_{n2}{}^2 + \frac{1}{16} i^*_{n2}{}^2 = \frac{37}{16} i^*_{n2}{}^2 \quad (9)$$

Comparing the results from the three different operating conditions, a significant conclusion is obtained. In the first case, though both the switches MS1 and PS1 are on, the circuit does not exhibit higher noise. This is because the noise current generated by the M2 transistor is fully correlated in both the pmos and nmos branch and therefore cancelled at the output of the charge pump. The most noisy operation state is the last one where the PS1 switch is on and the MS1 switch is off. In the third section of the chapter these noise calculations will be confirmed by the associated simulation results.

3.4 Spectral Components of the Charge Pump Output Signal

In this section an attempt to calculate the spectral components of the output signal Iout, as a function of the phase error $\Delta\theta$, between f_{ref} and f_{div}, is presented. In the following analysis it is assumed that the output of the charge pump consists of current pulses of amplitude I_{cp}. It is also assumed that there is no mismatch between the current sources (M5, P3) of figure 3. The duty cycle of the output pulse is equal to τ/T_{ref}, where τ is the active time of the charge pump output current and T_{ref} is the period of the reference signal. From the signal processing theory [20] it is known that the Fourier series expansion for a periodic train of pulses of amplitude I_{cp} and duration τ is:

$$I_{out}(t) = \frac{I_{cp}\tau}{T_{ref}} + 2\frac{I_{cp}\tau}{T_{ref}} \sum_{n=1}^{\infty} \frac{\sin(n\pi\tau/T_{ref})}{n\pi\tau/T_{ref}} \cos(\frac{2\pi nt}{T_{ref}}) \quad (10)$$

The equation (11) can be expressed as a function of the phase error $\Delta\theta$, taking into account that the ratio τ/T_{ref} is proportional to $\Delta\theta/2\pi$:

$$I_{out}(t) = \frac{I_{cp}\Delta\theta}{2\pi} + 2\frac{I_{cp}\Delta\theta}{2\pi} \sum_{n=1}^{\infty} \frac{\sin(n\pi\frac{\Delta\theta}{2\pi})}{n\pi\frac{\Delta\theta}{2\pi}} \cos(\frac{2\pi nt}{T_{ref}}) \quad (11)$$

If the duty cycle δ_{cp} equals to τ/T_{ref} and for small values of the δ_{cp}, the sinc function $\sin(n\pi\tau/T_{ref})/(n\pi\tau/T_{ref})$ can be approximated as unity. This results in the following expression for I_{out}:

$$I_{out}(t) = I_{cp}\delta_{cp} + 2I_{cp}\delta_{cp} \sum_{n=1}^{\infty} \cos(2\pi n f_{ref} t) \quad (12)$$

which shows that the amplitude of the spectral components of I_{out} are twice as large as its dc value $I_{cp}\delta_{cp}$. Therefore, if $\delta_{cp}=\Delta\theta/2\pi$ equals to zero the charge pump output ideally contains no dc or ac signal components.

The next step is to study the effect of mismatch in current sources. Mismatch originates in the different type of devices used to implement the n-type current sink, which sinks current from the output node to ground and the p-type current source

which sources current from the supply to the output node. Moreover the nominal current supplied by the n-type and p-type current sources is likely to be a function of the voltage at the output node of the charge pump. This is filtered by the loop filter producing the tuning voltage V_{tune} to the oscillator, and therefore it is a function of the output frequency of the entire loop. If $V_{mismatch}(n \cdot f_{ref})$ is the magnitude of the ripple voltage at the fundamental and harmonics of the reference frequency, then the equation which relates the above voltage with the current-source mismatch is given below:

$$V_{mismatch}(n \cdot f_{ref}) = I_{out}(n \cdot f_{ref}) \cdot \left| Z(j2\pi n f_{ref}) \right| \tag{13}$$

where $\left| Z(j2\pi n f_{ref}) \right|$ is the magnitude of the transimpedance function of the loop filter and n ranging from 1 to ∞.

It is common to express the magnitude of the undesired signal components with respect to the magnitude of the carrier frequency f_{LO}. From the standard modulation theory [20] the relationship of the peak phase deviation $\theta_p(f_m)$ to the peak frequency deviation $\Delta f(f_m)$ and the modulation frequency f_m is given by

$$\theta_p(f_m) = \frac{\Delta f(f_m)}{f_m} \tag{14}$$

The peak frequency deviation is the product of the magnitude of the spectral components of the mismatch voltage $V_{mismatch}(n \cdot f_{ref})$ with the gain K_{VCO}(V/Hz) of the VCO:

$$\Delta f(f_m) = V_{mismatch}(n \cdot f_{ref}) \cdot K_{VCO} \tag{15}$$

Combining (14) and (16) and substituting into (15) we get the peak phase deviation due to each of the spurious frequency components $n \cdot f_{ref}$

$$\theta_p(n \cdot f_{ref}) = \frac{I_{out}(n \cdot f_{ref}) \left| Z(j2\pi n f_{ref}) \right| \cdot K_{VCO}}{n \cdot f_{ref}} \tag{16}$$

Each of the baseband modulation frequencies $n \cdot f_{ref}$ generates two RF spurious signals, which are located at offset frequencies $\pm n \cdot f_{ref}$ from the carrier frequency f_{LO}. The amplitude of each spurious signal A_{sp} is related to the magnitude of the carrier A_{LO} and to the peak phase deviation θ_p by

$$A_{sp}(f_{LO} \pm n \cdot f_{ref}) = A_{LO} \frac{\vartheta_p(n \cdot f_{ref})}{2} \tag{17}$$

Substituting (17) into the numerator of (18) the following expression in dB is obtained

$$\left[\frac{A_{sp}(f_{LO} \pm n \cdot f_{ref})}{A_{LO}} \right]_{dBc} = 20 \cdot \log \frac{I_{out}(n \cdot f_{ref}) \left| Z(j2\pi n f_{ref}) \right| \cdot K_{VCO}}{2 \cdot n \cdot f_{ref}} [dBc] \tag{18}$$

An important conclusion that can be drawn from (19) is that the relative amplitude of the spurious signals is independent on the absolute value of loop bandwidth or on the

nominal charge-pump current I_{cp}. Instead, they are determined by the transimpedance of the loop filter, by the magnitude of the reference spurious components, by the VCO gain and by the value of the reference frequency.

3.5 Noise Performance of Charge Pump

An ideal CP-PLL with zero phase error neither sources current to, nor sinks current from, the loop filter. However, PLLs with zero phase error are insensitive to small loop-phase deviations due to finite rise times in the PFD and charge pump which is also called the "dead zone". A commonly employed solution to "dead-zone" is to use an artificial phase offset so that CP pumps/sink current when PLL is locked. When the PLL is locked the average output current flowing into the filter is zero. Both the currents from the P3 and M2 transistors in figure 3, are on for the duration of the "dead zone" pulse. Even though the average current is zero, noise is injected from both transistors currents for the duration of the "dead zone" pulse. The charge pump noise current injected to the loop filter under lock condition can be calculated as,

$$i^{*}{}_{n,out}{}^{2} = 2 \cdot \left(\frac{\delta_{cp}}{T_{ref}} \right) \cdot I^{2}{}_{n,CP} \tag{19}$$

where constant factor 2 is used to account for the source and sink current pulses. $I^{2}{}_{n,CP}$ is the noise current of the CP in A^2/Hz. δ_{CP} is the dead zone pulse width and T_{ref} is the reference period signal. The above equation suggests that if the reference frequency is increased then more noise will be injected into the loop filter and in consequence to the VCO circuit block. Moreover the PLL close-in phase noise will increase with the reference frequency by a factor proportional to $10 \cdot \log (f_{ref})$. Also a "dead zone" pulse with large duration leads to an increased noise as depicted by equation (20).

4 Simulation Results

The improved charge pump was designed using 0.18um CMOS technology. The supply voltage was 2.5Volt for the charge pump and inverters in the buffer chain. Simulations were obtained by using Cadence design framework with spectre device models from UMC 0.18um and 0.35um devices. The pump up and pump down currents are 1mA from a 2.4V power supply. The CP design includes a programmable gain by a step of 250uA. Corner and temperature analysis has also been performed, in order to further the demonstrability, applicability and robustness of the improved circuit. The percentage of DC mismatch current over output voltage of the improved CP, for three different process and temperature worst cases (typical─$0 \cdot \sigma$ @ 27°C, slow─ $-3 \cdot \sigma$ @ 85° C, fast─$+3 \cdot \sigma$ @ -45°C) as shown in figure 7.

The circuit is able to operate with a mismatch current less than 2.25% at the typical case and 2.5% at the extreme process/temperature conditions. The output voltage ranges from 300mV to 2.2V. Beyond these limits, transistors close to the supply rails (P3, M5), leave the saturation and enter to the linear region of operation, which results to an increased mismatch current. The power consumption of the improved charge pump including the three opamp consumption is roughly 6.65mW for a 2.4 power supply voltage.

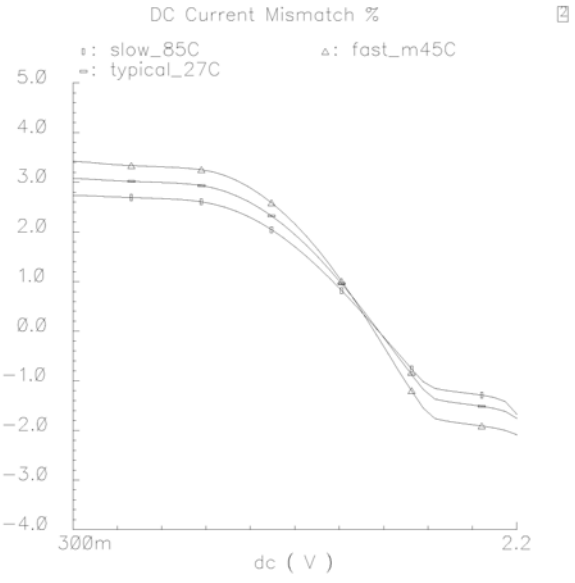


Fig. 7. The dc mismatch of the output current vs output voltage

A periodic noise analysis (Pnoise) has also been performed in order to obtain the noise contribution of the devices at the output of the charge pump. Figure 8 illustrates the noise power spectral density of the output current, when the loop is locked, which means that the transistors are on for a small fraction of time that is equal to the reset delay of the PFD. This time delay is 500ps, which results in a duty cycle of 0.5%. Because of the fast switching characteristics of the improved charge-pump even smaller duty cycles can be used, further reducing the close in noise contribution of the charge pump by as much as 6dB!

Due to the delay added in the reset path of the PFD, the current sources (P3, M5) are on for small or zero phase errors. The dumped charge on the capacitor of the loop filter as a function of the phase error is illustrated in figure 9. The X axis represents the phase error in time. A maximum 180 degrees phase deviation corresponds to 50ns.

In addition to that a second periodic noise analysis has been performed to confirm the validity of equation (20). The three plots in figure 10 represent the power spectral density of the charge pump noise current for three different duty cycles. As it can be observed, increased duty cycle leads to an increased noise to the charge pump output.

To verify the theoretical noise analysis and highlight the active noise contributors of the charge pump, an ac noise simulation has been done for the circuit. Three different cases have been simulated for typical model transistor and room temperature (27C) and the results are given in the following tables. At the first column is given the corresponding noise contributor transistor and at the second column is the simulated current noise at a specific spot frequency (f=1Hz for this case).

Table 2.

PS1 on MS1 on	
Noise Contributors	Noise Current (A^2/Hz)
M4	2.3e-16
M5	5.8e-17
P2	3.15e-17
P3	7.95e-18

Table 3.

PS1 off MS1 on	
Noise Contributors	Noise Current (A^2/Hz)
M2	1.875e-16
M5	5.61e-17

Table 4.

PS1 on MS1 off	
Noise Contributors	Noise Current (A^2/Hz)
M4	8.4e-16
M5	2.3e-16
P2	5.25e-17
P3	13.15e-18

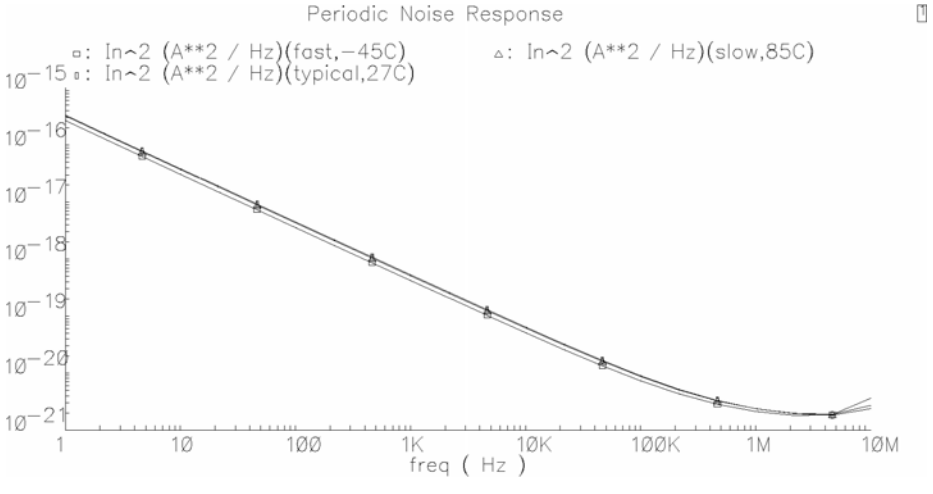


Fig. 8. Noise power spectral density from Pnoise analysis

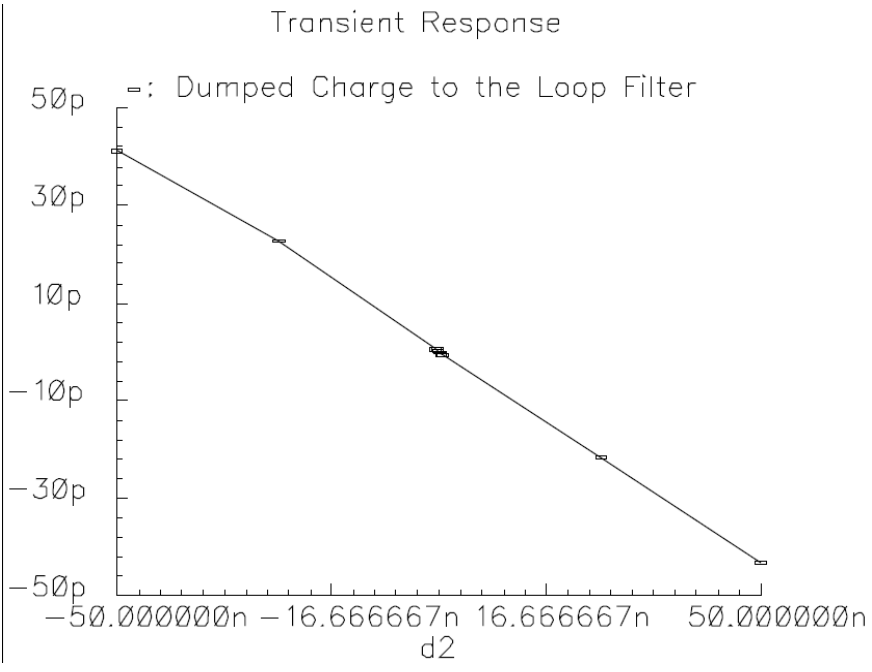


Fig. 9. Dumped charge as a function of phase error

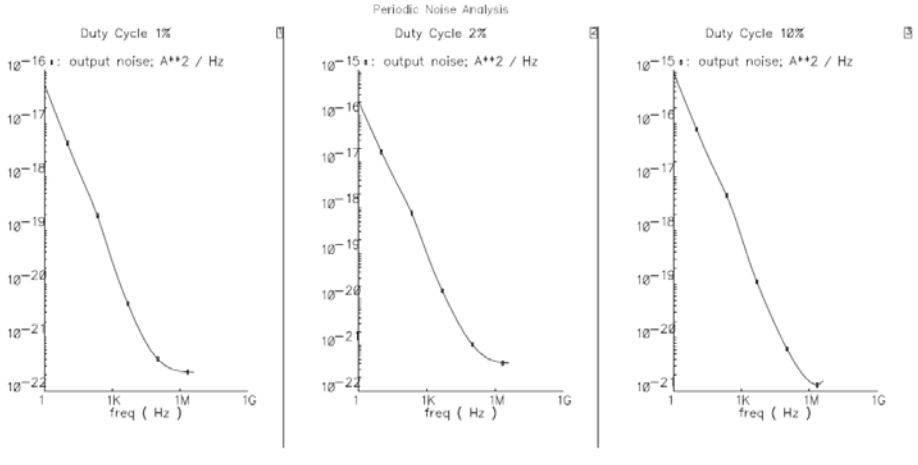


Fig. 10. PSD of CP Current Noise for three different Duty Cycle

The most significant noise contributors have been obtained for each case in consistency with the theoretical analysis, which is given in the previous section. It is worthy to note that M2 in table 2 does not produce any noise contribution, due to the correlation of the current noise components which cancel each other at the output. Moreover nmos transistors contribute higher amount of noise current than pmos transistor due to their higher mobility.

5 Conclusions

The design and analysis of a low noise charge pump has been presented in this chapter. Low noise charge pumps are essential to modern telecommunication systems, because they dominate the close-in noise of the associated synthesizer. In modern GSM, CDMA or OFDM standards the noise of the PLL is defined by the in-band noise, since the VCO noise can be reduced by simply opening the loop bandwidth to hit the VCO phase noise characteristic in a more attenuated level at a higher frequency which is especially true nowadays with the evolution of the fractional-N synthesizers. Therefore the low-noise properties of the charge pump are becoming increasingly more essential.

The improved charge-pump (figure 3) performs better than older alternatives, because it uses the current steering technique to switch on and off, therefore minimizes delays. Furthermore the introduction of O_{P3} at figure 3 ensures that all nodes are precharged to their final levels, and therefore less time is needed for the circuit to settle, furthering the initial speed improvement. Fast switching speed in essence demands a smaller dead-zone time and thus minimizes the noise contribution of the charge pump at lock (and non-lock) condition.

Inherent noise of the charge-pump is reduced by adding the stabilizing opamp circuits in an improved fashion compared to other alternatives [16, 17, 18]. By adding this novel feedback approach it is possible though to improve on output matching, without increasing noise. The improved charge-pump accuracy over the full output range, expressed by an excellent P/N mismatch, ensures that there is a limited systematic DC offset and therefore the spurious content is smaller, thus making it easier to meet the modern $2 \cdot f_{ref}$ spurious content specifications, which are steadily decreasing in size.

In table 5 alternative advanced CP families are compared, to provide a good perception of the improvement introduced by the circuits presented in the current chapter.

Table 5.

CP version	Switching Speed	Noise Performance	DC mismatch
Cheng et al.	Very Good	Good	Good
Rapinoja et al.	Medium	Good	Good
Chang et al.	Good	Medium	Good
This approach	Very Good	Very Good	Very Good

References

[1] Parker, J.F., Ray, D.: A 1.6-GHz CMOS PLL with on-chip loop filter. IEEE Journal of Solid State Circuits 33(3), 337–343 (1998)

[2] Craninckx, J., Steyaert, M.S.J.: A fully integrated CMOS DCS-1800 frequency synthesizer. IEEE Journal of Solid-State Circuits 33(12), 2054–2065 (1998)

[3] Rhee, W.: Design of high-performance CMOS charge pumps in phase-locked loops. In: IEEE International Symposium on Circuits and Systems (ISCAS), vol. 2, pp. 545–548 (1999)

[4] Rategh, H.R., Samavati, H., Lee, T.H.: A CMOS frequency synthesizer with an injection-locked frequency divider for a 5-GHz wireless LAN receiver. IEEE Journal of Solid-State Circuits 35(5), 780–787 (2000)

[5] Rhee, W., Song, B.S., Ali, A.: A 1.1-GHz CMOS fractional-N frequency synthesizer with a 3-b third-order ΔΣ modulator. IEEE Journal of Solid-State Circuits 35(10), 1453–1460 (2000)

[6] Hungand, C.-M., Kenneth, K.O.: A fully integrated 1.5-5.5-GHz CMOS phase-locked loop. IEEE Journal of Solid-State Circuits 37(4), 521–525 (2002)

[7] De Muerand, B., Steyaert, M.S.J.: A CMOS monolithic $\delta\sigma$-controlled fractional-N frequency synthesizer for DCS-1800. IEEE Journal of Solid-State Circuits 37(7), 835–844 (2002)

[8] Loke, A.L.S., Barnes, R.K., Wee, T.T., Oshima, M.M., Moore, C.E., Kennedy, R.R., Gilsdorf, M.J.: A versatile 90-nm CMOS charge-pump PLL for SerDes transmitter clocking. IEEE Journal of Solid-State Circuits 41(8), 1894–1907 (2006)

[9] Perrott, M.H., Trott, M.D., Sodini, C.G.: A modeling approach for sigma-delta fractional-N frequency synthesizers allowing straightforward noise analysis. IEEE Journal of Solid-State Circuits 37(8), 1028–1038 (2002)

[10] Shu, K., Sanchez-Sinencio, E., Silva-Martinez, J., Embabi, S.H.K.: A 2.4-GHz monolithic fractional-N frequency synthesizer with robust phase-switching prescaler and loop capacitance multiplier. IEEE Journal of Solid-State Circuits 38(6), 866–874 (2003)

[11] Arora, H., Klemmer, N., Morizio, J.C., Wolf, P.D.: Enhanced phase noise modeling of fractional-N frequency synthesizers. IEEE Transactions on Circuits and Systems I: Regular Papers 52(2), 379–395 (2005)

[12] Huh, H., Koo, Y., Lee, K.-Y., Ok, Y., Lee, S., Kwon, D., Lee, J., Park, J., Lee, K., Jeong, D.K., Kim, W.: Comparison frequency doubling and charge pump matching techniques for dual-band delta sigma fractional-N frequency synthesizer. IEEE Journal of Solid-State Circuits 40(11), 2228–2236 (2005)

[13] Woo, K., Liu, Y., Nam, E., Ham, D.: Fast-lock hybrid PLL combining fractional-n and integer-n modes of differing bandwidths. IEEE Journal of Solid-State Circuits 43, 379–389 (2008)

[14] Mitomo, T., Fujimoto, R., Ono, N., Tachibana, R., Hoshino, H., Yoshihara, Y., Tsutsumi, Y., Seto, I.: A 60-GHz CMOS receiver front-end with frequency synthesizer. IEEE Journal of Solid-State Circuits 43, 1030–1037 (2008)

[15] Fahimand, A.M., Elmasry, M.I.: A low-power CMOS frequency synthesizer design methodology for wireless applications. In: The International Symposium on Circuits and Systems (ISCAS), vol. 2, pp. 115–119 (1999)

[16] Cheng, S., Tong, H., Silva Martinez, J., Karsilayan, A.I.: Design and analysis of an ultra high-speed glitch-free fully differential charge pump with minimum output current variation and accurate matching. IEEE Transactions on Circuits and Systems II: Express Briefs 53, 843–847 (2006)

[17] Rapinoja, T., Stadius, K., Halonen, K.: A low-power phase-locked loop for uwb applications. Analog Integrated Circuits and Signal Processing 54, 95–103 (2007)
[18] Chang, R.C., Kuo, L.-C.: A new low-voltage charge pump circuit for PLL. In: The International Symposium on Circuits and Systems (ISCAS), vol. 5, pp. 701–704 (2000)
[19] Rabaey, B.N.J.M., Chandrakasan, A.: Digital Integrated Circuits, A Design Perspective, 2nd edn. Prentice Hall, Englewood Cliffs (2002)
[20] Taub, H., Schilling, D.L.: Principles of Communication Systems, 2nd edn. McGraw-Hill, New York (1986)
[21] Jones, D.A., Martin, K.: Analog Integrated Circuit Design, 1st edn. Wiley, Chichester (1996)
[22] Hanumolu, P.K., Brownlee, M., Mayaram, K., Moon, U.-K.: Analysis of Charge Pump Phase Locked Loops. IEEE Transactions on Circuits and Systems I: Regular Papers 51(9), 1665–1674 (2004)
[23] LMX2330A, National Datasheet
[24] Johnson, M., Hudson, E.: A variable delay line PLL for CPU-coprocessor synchronization. IEEE Journal of Solid-State Circuits 23(10), 1218–1223 (1988)
[25] Young, I.A., Greason, J.K., Wong, K.L.: A PLL Clock Generator with 5 to 110MHz of Lock Range for Microprocessors. IEEE Journal of Solid-State Circuits 27(11), 1599–1607 (1992)
[26] Maneatis, J.: Low-Jitter and Process-Independent DLL and PLL Based on Self-Biased Techniques, ISSCC Digest of Technical Papers (1996)
[27] Soyuer, M., Ewen, J.F., Chuang, H.L.: A Fully Monolithic 1.25GHz CMOS Frequency Synthesizer. In: Symposium on VLSI Circuits, Digest of Technical Papers, vol. 6, pp. 127–128 (1994)
[28] Razavi, B.: Monolithic Phase-Locked Loops and Clock Recovery Circuits, pp. 1–39. IEEE Press, Los Alamitos (1996)

Comparison of Two Autonomous AC-DC Converters for Piezoelectric Energy Scavenging Systems

Enrico Dallago[1], Daniele Miatton[1], Giuseppe Venchi[1], Valeria Bottarel[2], Giovanni Frattini[2], Giulio Ricotti[2], and Monica Schipani[2]

[1] Department of Electrical Engineering, University of Pavia, 27100 Pavia, Italy
[2] STMicroelectronics 20010 Cornaredo, Milan, Italy

Abstract. Piezoelectric Energy Scavenging Systems (PESS) are used to convert the energy of mechanical vibrations into electrical energy exploiting the piezoelectric effect. Their output is a voltage which strongly varies in time; to obtain a suitable supply source an AC-DC conversion of the output voltage of these transducers is needed. Since the output power level of the energy transducer can be very low, the conversion should be as efficient as possible.

The paper describes an active AC-DC converter, based on the voltage doubler topology, in which two different driving circuitries have been implemented.

The proposed solutions are fully autonomous, i.e. they are supplied by the energy that they harvest. To reduce and to optimize their power consumption a bias circuit has been designed to make the total bias current supply independent.

A test chip has been diffused using STMicroelectronics 5V CMOS technology. The performances of the two solutions were compared with the ones of a passive Schottky based voltage doubler. The figures of merit were the average power supplied by the piezoelectric transducer and the average power delivered on a load resistance. Furthermore, the significance of such parameters is also discussed, and a more general figure of merit is defined with the advantage of also weighting the ability of the converter to harvest all the power available at the transducer terminals.

Keywords: Piezoelectric energy harvesting, active AC-DC converter, low-power circuits, environmental vibrations.

1 Introduction

Energy scavenging systems are used to harvest the normally lost environmental energy (associated to vibrations, thermal gradients, solar radiation, pressure gradients) and to convert it into electrical energy. This solution can be very attractive to supply portable or low power electronic devices where batteries are a bottleneck for the whole system (i.e. they have a finite life time and their replacement or recharge is not feasible or too expensive). An energy scavenging system, instead, is a theoretically endless energy source. For this reason they are rapidly gaining popularity in the scientific and the industrial community. Typical applications are wireless sensing (both single spot and multi-nodes systems), biomedical (patient monitoring, hearing aids, etc.), automotive (using self supplied, wireless devices can help reducing the

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 61–80, 2010.

cable length on board). The challenge is twofold: on one side the power requirements of the supplied electronics should be as low as possible, on the other, given the very low energetic content associated to environmental energy, the interface circuit has to be as effective as possible in managing the harvested power.

In literature many papers can be found which describe methodologies to realize an energy-scavenger [1], [6], [8], [10]. Each case has different requirements depending on the environmental energy source considered, the characteristics of the transducer, the typical frequency of the process. No universal solution exists: each scavenger interface electronics has to be tailored to the specific transducer and application.

Many of the above mentioned works are focused on the conversion of the energy associated to mechanical vibrations since they can be easily found in many environments [1], [7]. Among the available vibration transducers this paper considers a system based on a piezoelectric transducer since it is one of the most efficient which can be used [1]-[2]. When exposed to vibrations coming from the real world the electrical energy at the output of this transducer is a strong and irregular function of time [1]-[4], [9], hence, to realize a DC supply source, an AC-DC conversion is needed.

Two active AC-DC converters based on the voltage doubler topology (see Figure 1) are presented. The reasons behind the choice of this topology are the fact that is it simple and that it is able to increase the value of the input voltage by approximately a factor two. This last aspect is quite important: as it will be shown later, the piezoelectric transducer exhibits a resonant behaviour; at resonance the output voltage for a given acceleration can be quite high, but out of the resonance the same acceleration will produce a voltage much lower. In perspective of an actual application, the harvester will be exposed to an acceleration whose spectrum is spread over a range, rather than being tuned at a particular frequency, so being able to step-up the input voltage is a desired feature.

The idea of implementing an active rectifier, rather than a passive one, is basically inspired by the observation that the voltage drop across the switches of the rectifier causes power dissipation and, most of all, has to be subtracted from the output

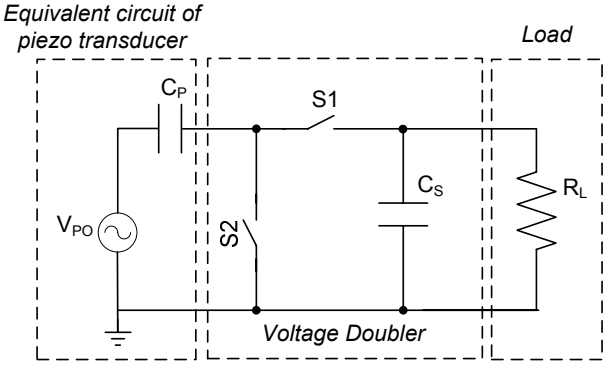


Fig. 1. Piezoelectric energy harvesting system

voltage, decreasing it and consequently the energy that can be stored at the output (for example across a capacitor as shown in Fig. 1). Of course the active circuitry comes at the price of using a part of the harvested energy to supply the active circuitry itself: active converters can be more efficient than passive solution [3-4], [11], but they are more difficult to be designed because their whole power consumption has to be very low.

To this purpose, and for other reasons which will be illustrated in the following, two different driving circuitries were considered to command switches S1 and S2 (Figure 2). In particular, the topology shown in Figure 2a, which will be called "hard", uses comparators (CMP1-2), while in the one of Figure 2b, which will be called "soft", the switches are driven with operational amplifiers (OP1-2). Both proposed solutions use only a fraction of the harvested energy to supply the active circuitry and make the energy scavenging system autonomous. In particular, at the output a capacitance (C_S) of 1 μF is used to store the harvested energy and no external power source is required, neither during start-up nor in normal operation[1].

Since the voltage across the storage capacitor Cs increases as the energy is harvested, a supply independent bias circuitry is used in both cases: it makes the current consumption of the active part supply independent.

Both circuits were diffused as test chips exploiting a 5V CMOS STMicroelectronics technology and were extensively tested. They were compared in terms of efficiency of the converter and of the average output power, evaluated under the same input conditions while varying the load. The significance of such parameters is also discussed, and a more general figure of merit is defined with the advantage of also weighting the ability of the converter to harvest all the power available at the transducer terminals.

Section 2 deals with the design of the two considered AC-DC driving circuitries highlighting their pro and cons, while Section 3 shows the experimental results compared with the simulated ones.

2 Piezoelectric Energy Scavenging System

2.1 Mechanical Aspects

The considered piezoelectric transducer is a cantilever which works in 31-mode when it is excited by the mechanical vibrations, as shown in Figure 3. This means that the cantilever is exposed to a strain in the direction of axis 1 while the resulting electrical displacement is along axis 3.

To have a maximally efficient conversion of the mechanical vibrations into electrical energy the cantilever should be excited at its resonant frequency which can

[1] The presence of the output capacitor is not only required to supply the control circuitry or to filter out the voltage ripple, but it often works as an energy tank. In fact, in most cases the supplied electronics works with a duty cycle. For example a wireless sensor that transmits one measurement every 10 seconds will be active only for a few hundreds of milliseconds, while it will switch in a low consumption or standby mode for the rest of the time. During this interval the scavenger is expected to accumulate the energy required during the burst of operation.

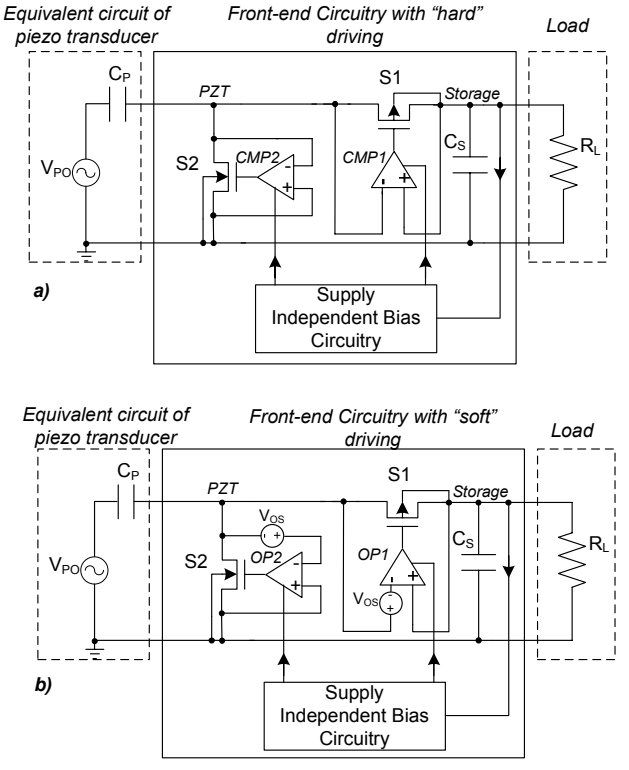


Fig. 2. Schematics of the proposed ESS. Voltage doubler with "hard" driving circuitry (a) and with "soft" driving circuitry (b).

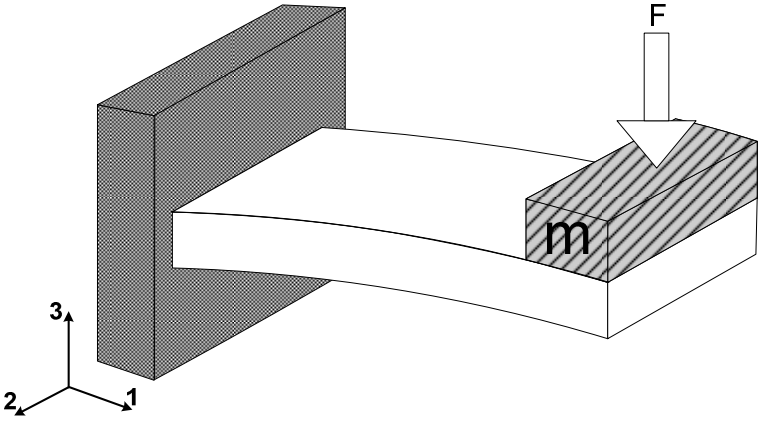


Fig. 3. Piezoelectric energy transducer working in 31-mode

be varied adding a mass (m) on its free end [1]. This allows to move the resonance frequency of the energy transducer around the frequency which is more likely in the application. The experimental measurements shown in Table 1 and [1] show that the frequency range of mechanical vibrations existing into civil environments is approximately (10-380) Hz.

Table 1. Experimental measurements of environmental vibration sources

Environmental Vibration Source	$a_{rms}[m/s^2]$	Bandwidth [Hz]
CD reader of a desktop PC	0,30	80
Pocket of a walking man	2,16	2
Man walking onto stairs	3,53	2
Into a woman bag	6,37	10
Vibracall of a mobile phone	3,83	90
Car dashboard [city street @ 20km/h]	0,78	30
Car dashboard tortuous street @ 50km/h]	0,88	20
Car front glass [primary street @ 50 km/h]	0,98	15
Guard rail near a busy street	0,10	50
Bridge over a railway	0,10	100
Bus stop near a busy street	0,19	10
Fun	1,40	100
Web server	0,20	12

The piezoelectric transducer can be modeled at resonance by the equivalent circuit shown in Figure 1, [1]-[2]. Generator V_{PO} is a sinusoidal voltage source whose frequency is equal to the transducer resonance frequency and whose amplitude is equal to the open circuit output voltage, while C_P is the electrical capacitance of the piezoelectric cantilever.

The presented results are based on a piezoelectric transducer with geometrical dimensions equal to (30x15x0.2) mm^3 (LxWxH). The piezoelectric material used was a soft Lead Titanate Zirconate (PZT) and it was sputtered onto a Nichel alloy support. The piezoelectric capacitance C_P was measured to be 19 nF.

Figure 4 shows the experimental set-up which was used to shake the piezoelectric transducer with a controlled acceleration. It is composed of an electrodynamic shaker (Bruel&Kjiaer 4810), which moves the piezoelectric transducer, and by a triaxial MEMS accelerometer (LIS3L02AS4 of STMicroelectronics); it was screwed to the shaker movable table so to measure the acceleration given by the shaker itself and to guarantee the repeatability of the measurements.

Figure 5 shows the experimental frequency response of the piezoelectric transducer when it was excited with a sinusoidal acceleration with a peak to peak amplitude

equal to 0.8 g. It is possible to see that its resonance frequency is equal about to 130 Hz.

For example, the measured resonance frequency of the cantilever is about 64 Hz if a lead mass of about 0.8 grams is added.

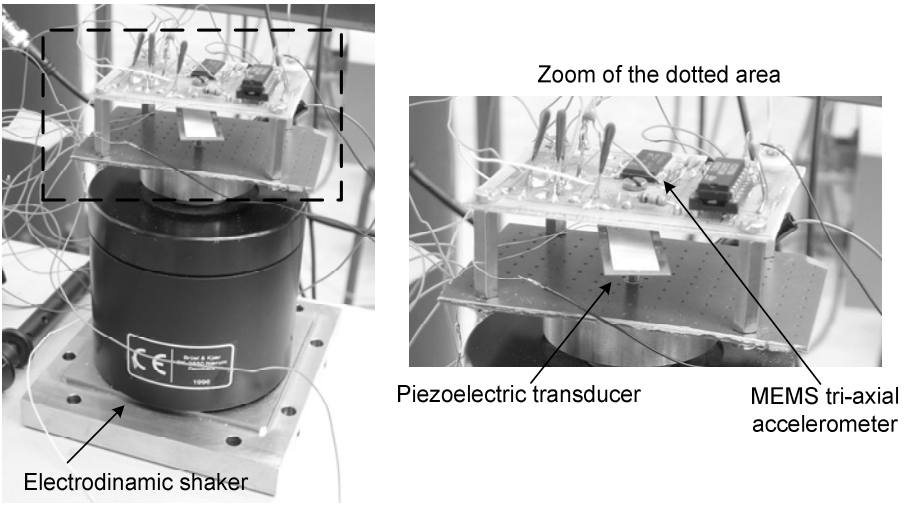


Fig. 4. Picture of the measurement set-up

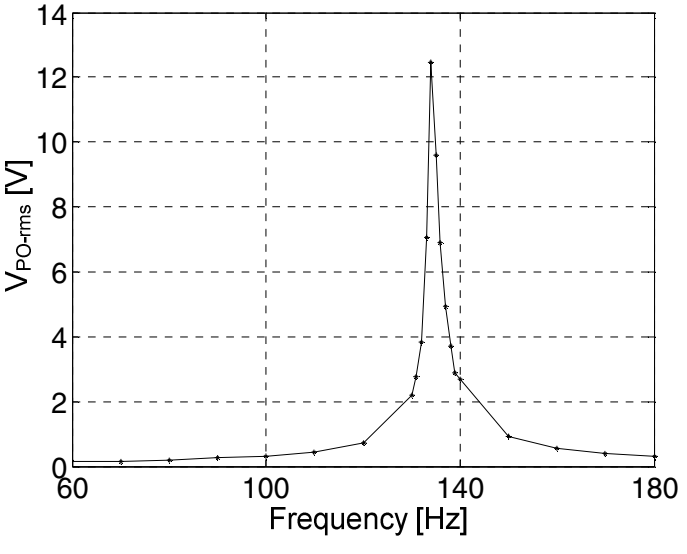


Fig. 5. Frequency response of the piezoelectric transducer

2.2 Design of the Proposed Front-End Circuitries: Common Parts

During the start-up the voltage across C_S is too low to supply the active devices. In this case, for both solutions, the operation of the converter is guaranteed by the body-drain diode of MOSFETs S1 and S2 which implement a standard, passive diode voltage doubler rectifier. Hence, the proposed rectifiers can be seen as a parallel of two AC-DC converters: an high efficiency active one and a lower efficiency passive one, the latter working only during start-up. As soon as the output voltage is sufficient the active one comes alive and flawlessly takes over the passive one; this switch does not need any dedicated control circuitry.

Furthermore, after the active part starts working, the voltage across C_S varies with time as the energy is harvested. Since this voltage is used as the supply for the active devices, it is advantageous to make their current consumption independent on the supply voltage itself.

Figure 6 shows the circuital topology used to implement a supply independent bias; it is modified with respect to [5] by introducing a diode connected MOSFET (M4). In fact, since the system requires supply currents in the range of tens of nanoamperes, in the scheme without M4 the resistance R1 would be in the order of tens of megaohms, which is too area expensive for an integrated solution. The effect of M4 is to reduce the voltage drop across R1 lowering its value for a given current. Furthermore, a start up circuit is needed. This was obtained with dummy MOSFETs ML1 and ML2: the leakage of their body-drain diode has been exploited to inject a current into nodes A and B so to have the start-up aid. This solution allows us to avoid additional start-up circuitry, reducing total power consumption.

Figure 7 shows the simulated behaviour of the two bias voltages versus the supply voltage. In particular the circuitry starts to regulate when the supply voltage is higher than about 680 mV. Above this value, the voltages V_{BiasP} and V_{BiasN} can be used to mirror a supply independent current.

The current consumption of the whole bias circuitry is about 100nA.

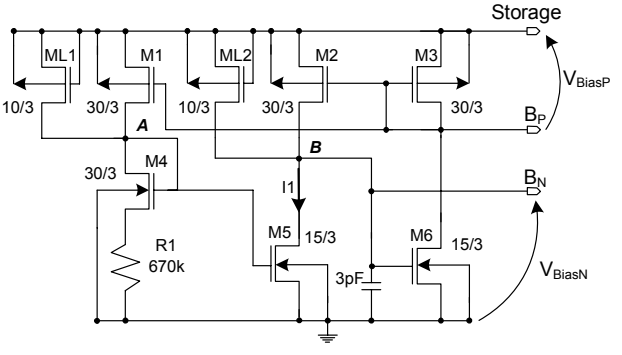


Fig. 6. Schematic of the proposed supply independent bias circuitry

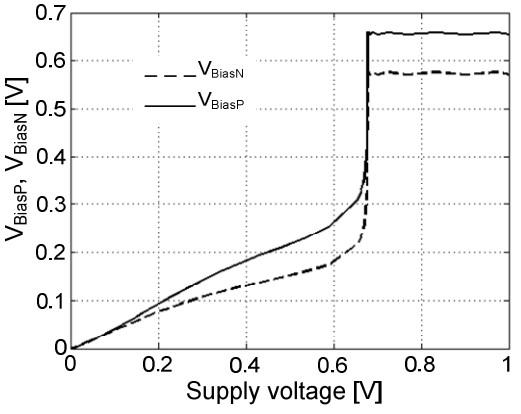


Fig. 7. Simulation results of the supply independent bias circuitry, bias voltages (V_{BiasN} and V_{BiasP}) versus supply voltage (in the figure supply voltage has been limited to 1V for clarity. The behaviour has been verified for supply voltages up to 5V).

2.3 Working Principle of "Hard" Driving and Design Methodology

Figure 2a shows the active topology of the ac-dc converter that uses comparators CMP1 and CMP2 to drive switches S1 and S2. The comparators sense the drain to source voltages of the two MOSFETs and properly turn them on or off [11]. This is the most straightforward way to implement an active version of the voltage doubler converter, since the switches and their driving emulate some ideal diode with a very low, almost zero, forward voltage drop, but the design of the two comparators is critical because a very accurate choice of the threshold voltages is required. In principle one could use a single threshold for each comparator equal to zero; in practice the spread due to process mismatches has to be taken into account. In fact if this voltage varies, during the diffusion process, with respect to the ideal value a negative effect, (i.e. the oscillation of the driving signal), can take place [4]. To reduce the probability to fall into this case the comparators has be designed with an hysteretic transfer characteristic, with an hysteresis wide enough to accommodate the expected variance of threshold. Among the two, the most critical threshold voltage, for both comparators, is the one which switches S1 or S2 off, called "Th-OFF". To prevent the above mentioned oscillations the designer has to guarantee it is positive for CMP1 and negative for CMP2; but, at the same time, to minimize the unwanted discharge of the piezoelectric or of the storage capacitance caused by a delayed switching [4], the "Th-OFF" should be as small as possible. These two requirements are, of course, contrasting.

Figure 8a and Figure 8b show the electrical schemes of the two designed comparators that are supply and ground compatible respectively, while Figure 8c shows a picture of the diffused circuit. The bias voltages B_N and B_P are provided by

the supply independent bias circuitry. Referring, for example, to CMP1 (similar considerations apply to CMP2) it is possible to see that the current in M_4 depends on the output voltage of CMP1 itself. This allows to obtain the hysteretic transfer function for the comparator. In fact, when output voltage of CMP1 is low, (S1 is into its ON condition) the current in M4 is equal to the current in M5; on the contrary, when the output of CMP1 is high (S1 in OFF condition), an extra current, given by M3, flows in M4. As a consequence, the two threshold voltages are different from each other. To obtain a positive "Th-OFF" threshold voltage, a mismatch between the aspect ratio of the MOSFET M4 and M5 was introduced. In order to assure that the two "Th-OFF" voltages have the correct sign, 300 MonteCarlo simulations have been performed. The resulting distribution diagrams of the critical "Th-OFF" threshold voltages are shown in Figure 9 for both comparators. The chosen values are +9 mV and -9 mV for CMP1 and CMP2 respectively.

The static current consumption of each comparator is about 110 nA and their minimum operating supply voltage is about 650 mV.

The structures indicated with “OD” into Figure 8c are two common source configuration MOSFETs whose gate terminals are connected to the output node of CMP1 and CMP2 respectively and whose drain terminals are directly connected to two output pads. These MOSFETs were added to allow the comparator output voltages to be measured, so to verify their functionality, without adding a loading capacitance too high to their output nodes.

2.4 Working Principle of "Soft" Driving and Design Methodology

Figure 2b shows the active topology of the voltage doubler converter where the driving circuitry is realized with operational amplifiers OP1 and OP2.

To introduce its working principle let us consider Figure 10 which shows the case of OP1. It is possible to apply Kirchhoff Voltage Law (KVL) to the external mesh:

$$V_{IN} + V_{SD} - V_{OS} = 0 \tag{1}$$

If the operational amplifier has a DC gain equal to A the voltage on the gate of S1 is:

$$V_G = A \cdot V_{IN} \tag{2}$$

Equations (1) and (2) can be solved as a function of V_G:

$$V_G = A \cdot (V_{OS} - V_{SD}) \tag{3}$$

In the ideal case the DC gain A of the operational amplifier is infinite; as a consequence the difference in equation (3) has to vanish in order to have a finite value of the gate voltage. Thus a regulation loop modulates the gate voltage V_G so to keep, for each value of the drain current, the source to drain voltage is equal to Vos.

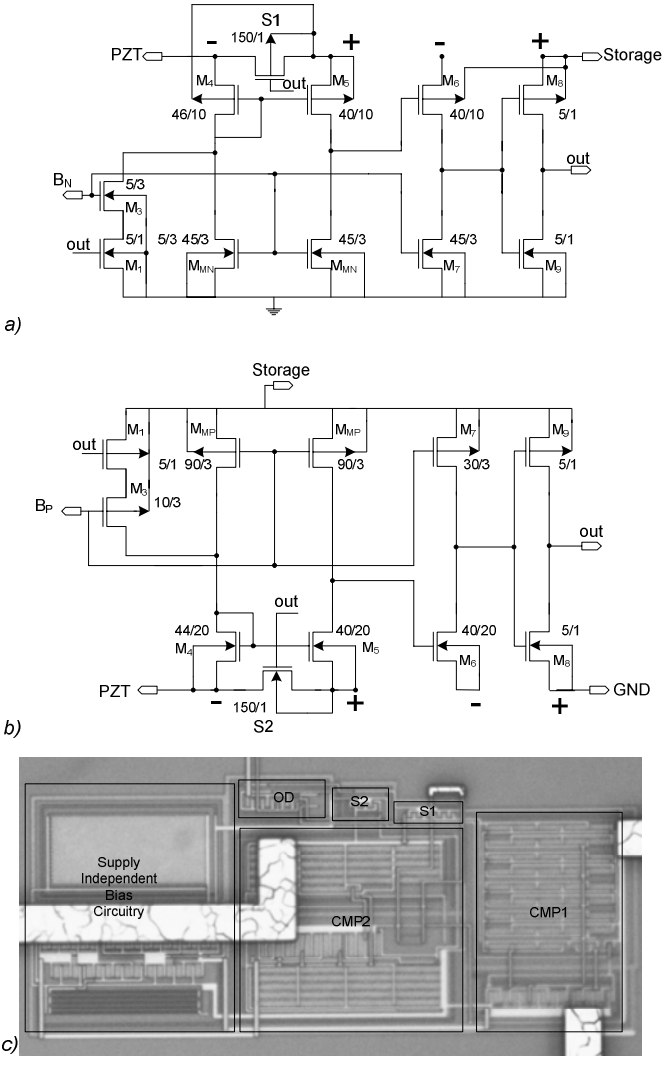


Fig. 8. a), b) Designed circuital topologies of CMP1 and CMP2 respectively; c) picture of the diffused "hard" AC-DC converter

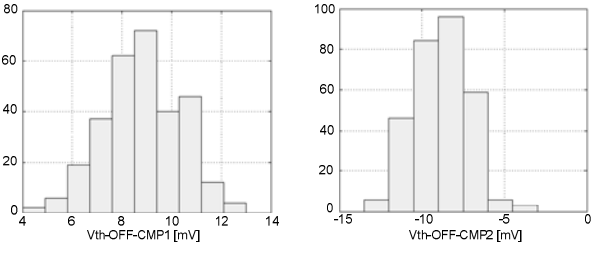


Fig. 9. MonteCarlo simulations of the most critical threshold voltage of the designed comparators

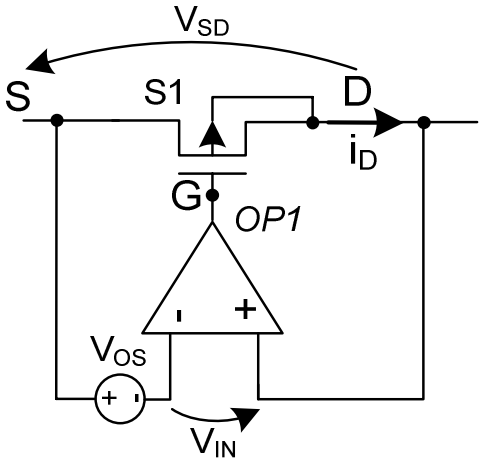


Fig. 10. Regulation loop composed by the operational amplifier OP1 and MOSFET S1

In the ideal case, when drain current is positive the regulation loop sets the working point of the MOSFET at the intersection between its characteristics and the offset voltage. When the current decreases, the loop moves the working point of the MOSFET at lower values of its source to gate voltage, until the current is equal to zero. At this point the regulation loop turns the MOSFET off: since there is no intersection between the MOSFET characteristics and the offset voltage, negative values of the current are not allowed and the regulation loop holds the transistor off. This principle intrinsically guarantees that, differently from the "hard" topology, no oscillations of the driving signal can take place. Furthermore, a negative current is impeded and the discharge of the capacitances is prevented. Symmetrical considerations apply to OP2. Differently from the "hard" approach, the value of Vos is not critical because it has simply to be far enough from zero so that the process mismatches will not change its sign. While it is true that an higher offset gives higher losses on the switch, they are still negligible for practical offset values (which can be easily designed in the tens of millivolts range).

In practice, the real operational amplifier has a finite DC gain; this means that the value of the voltage across the MOSFET is slightly different from the theoretical one. Nevertheless it can be demonstrated that, with the previously described choice of the offset, this will not affect the operation of the circuit.

Figure 11a shows the designed operational amplifiers while Figure 11b contains a picture of the diffused structure. Because of the level of their input voltages, OP1 and OP2 have to be supply compatible and ground compatible respectively.

The bias of the operational amplifiers is given by the supply independent bias circuitry. The operational amplifiers have been designed so that they are able to work with the minimum possible value of the voltage supply. In this way the active rectifier takes over the passive one as soon as a very low energy has been stored into capacitance C_S. A 5pF capacitance has been introduced to compensate the regulation loop.

The offset voltage was obtained mismatching the aspect ratio of the input MOSFETs M_A and M_B: the values of the obtained offset voltages are equal to 26mV and 21mV for OP1 and OP2 respectively. Figure 12 presents the simulation results of 500 MonteCarlo iterations, showing the possible spread of these voltages. It is possible to see that this spread is sufficiently small to have the correct sign also with process mismatches.

Aspect ratio of the MOSFETs S1 and S2 has to be chosen to avoid the loop saturation. The expression of the drain current when the MOSFET is turned on is, in a first approximation, equal to:

$$i_D = k\frac{W}{L}(V_{GS} - V_{th})V_{DS} \tag{4}$$

where V_{th} is the threshold voltage of the MOSFET, W/L is its aspect ratio and k is its characteristic constant.

The term into the parenthesis is the overdrive voltage: its value is modulated by the regulation loop which, for each drain current, varies the gate voltage.

If the overdrive voltage is enough the source to drain voltage is equal to V_{OS}, that is:

$$i_D = k\frac{W}{L}(V_{GS} - V_{th})V_{OS} \tag{5}$$

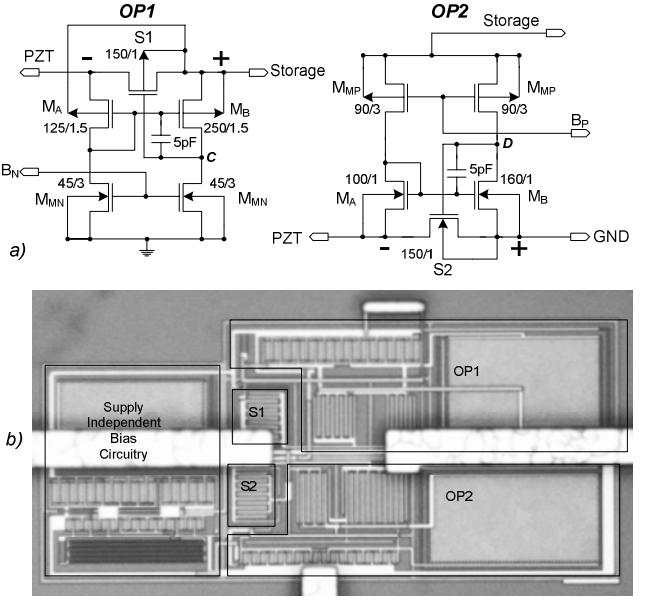


Fig. 11. a) Schematic of the proposed operational amplifiers; b) picture of the diffused "soft" AC-DC converter

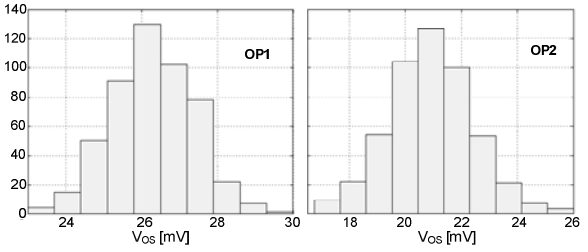


Fig. 12. MonteCarlo simulations of the offset voltages of the designed operational amplifiers

On the other hand if, for a given current, the overdrive is not enough (which means that the op-amp is saturated) the regulation loop does not work and the source to drain voltage of S1 or S2 is not equal to offset voltage.

The aspect ratio has to be designed to accommodate the maximum expected current.

Tab. 2 resumes the characteristics obtained for the amplifiers: in particular the current consumption is about 200 nA for each amplifier. $V_{dd\text{-}min}$ is the minimum supply voltage required by the operational amplifiers to work.

Table 2. Characteristics of the designed operational Amplifiers

	DC-GAIN [dB]	Band-Width [Hz]	Vdd-min [mV]	Voffset [V]	Current Consumption [nA]
OP1	50.37	2500	730	26e-3	200
OP2	49.78	2610	675	21e-3	200

3 Experimental and Simulated Results

Before evaluating the performances of the proposed solutions an experimental characterization of the supply independent bias circuit was realized. This was done using a stand alone version of the circuit depicted in Figure 13b: three MOSFETs Ma, Mb and Mc were added to amplify the bias current by a factor 30. This current is then converted into a voltage drop by means of an external resistance, equal to 2 MΩ. Experimental characterization was done supplying the bias circuit with a triangular waveform in the range 0-5 V. Figure 13a shows that, as soon as the supply voltage is higher than 680 mV, the circuit generates a constant voltage drop across the 2 MΩ resistance which corresponds to a constant bias current, as it was predicted by the simulations.

The performances of the proposed solutions have been evaluated at different values of the load resistance R_L; in particular, the considered figures of merit were the average power at the output of the piezoelectric transducer (P_P) and the average power delivered to R_L itself (P_L). The load resistance was varied between 50 kΩ to 550 kΩ.

A passive voltage doubler, realized with BAT-86 Schottky diodes that have a threshold voltage equal to 0.2 V, was considered as a benchmark. This comparison is quite significant since the selected diodes as a forward voltage drop way lower than a standard diode and the passive solution does not have any additional consumption due to the control circuitry.

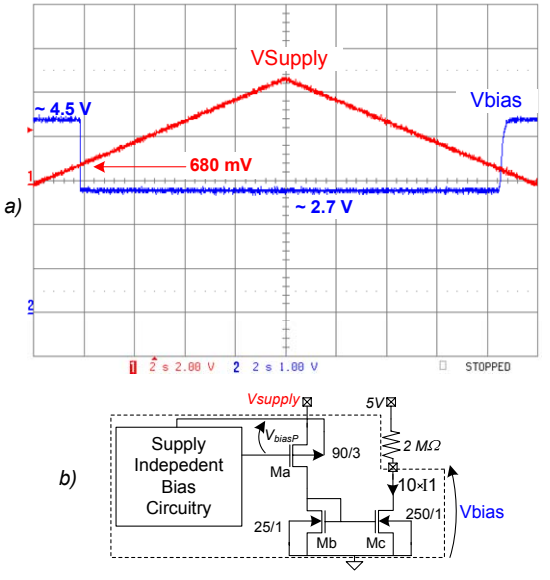


Fig. 13. a) Experimental results of the supply independent bias circuitry; b) circuitry used to do the characterization

Figure 14 compares the experimental results of the "hard" solution with those of the passive converter. The piezoelectric transducer was mechanically excited at 130 Hz and the acceleration was regulated so that the peak value of V_{PO} was equal to 1.5 V.

The powers drawn from the piezoelectric transducer are comparable between the two cases. On the contrary, the output power of the "hard" solution is heavily affected at low load resistance values. This is due to the fact that at values lower than 100 kΩ the energy stored into C_S, and hence the voltage across it, is not enough to switch the driving circuitry on and, as a consequence, only the well-source diodes of the passive path are exploited for the conversion. Conversely, at higher load values the converter works in the active way and the output power is higher with respect to the passive solution. This confirms that the lower forward voltage drop across the switches has positive effects on the output power, consistent enough to compensate for the power consumption of the control circuitry.

Furthermore, the accuracy of the design has guaranteed that, in spite of the fact S1 and S2 are switched off with a delay due to comparator threshold voltage, this does not affect the energy harvesting and makes the “hard” active converter a competitive solution with respect to the passive one.

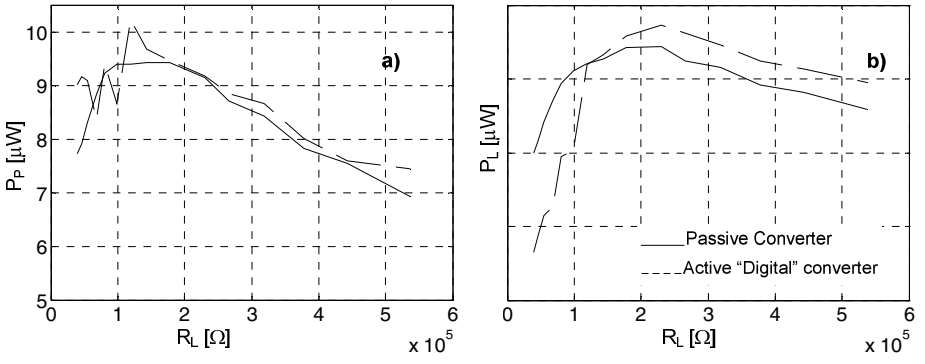


Fig. 14. a, b) Experimental results about input and output powers (P_P and P_L) respectively. Comparison between active "digital" converter and passive converter, this was realized with Schottky BAT86 diodes.

The performances of the "soft" solutions were evaluated at different values of the mechanical acceleration so to obtain different values of V_{PO} (i.e., 1.5, 2 and 2.5 V); Figure 15 a, b show the obtained results, comparing them with the corresponding behaviour of the passive converter, as usual. It is possible to see that under any condition the "soft" converter draws a lower power from the piezoelectric transducer than the passive one. On an absolute scale this means that the combination of the piezoelectric transducer and the "soft" converter makes a less effective energy harvester.

Returning to the interpretation of the experimental results the "soft" converter is generally less capable of extracting energy from the transducer with respect to both the "hard" rectifier and the passive one.

At this point one should expect that also the power delivered to the load resistance P_L is lower, but the experimental results of Figure 15 show that the two rectifiers are equivalent when V_{PO} is 1.5V, that the active rectifier performs better at lower load resistance values for V_{PO}=2, and that finally it outperforms the diode converter on a wide load range for V_{PO}=2.5V.

This means that the "soft" converter has an efficiency higher than the passive one and its advantage increases as V_{PO} is higher. The reason is that the power consumption of the driving circuitry weights most at lower values of P_P, that is at the lower values of input voltage.

Coming to a comparison between the two proposed converters, when the threshold voltages have the correct sign, the performances of the "hard" solution (ref. Figure 14b) are better with respect to both "soft" and passive converter.

Efficiency (η) of an energy conversion system is defined as the ratio between its output power with respect to its input power. This is usually used as figure of merit which allows to compare different solutions which work at the same output power; this means that the system which requires the lower input power is the better one, under the hypothesis that the input source could be able to supply an infinite input power.

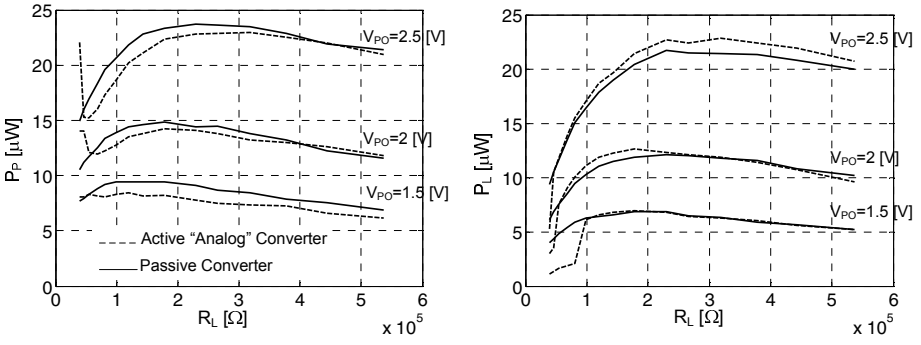


Fig. 15. Measured input (P_P) and output (P_L) powers at different values of V_{PO}, comparison between passive Schottky based converter and "analog" AC-DC converter

Comparing the experimental results obtained at V_{PO}=1.5 V (see Figure 16) for the three different AC-DC converters it is possible to see that their η can not be used as figure of merit to evaluate the performances of the energy harvesting system. In fact, the "soft" converter shows the better performance in terms of efficiency with respect any other solution, but it is equivalent at the passive one in terms of output power. Furthermore, "hard" solution outperforms any other converter in terms of output power. This is due to the fact that the input energy source, that is the piezoelectric transducer, is not an infinite energy source and its matching with the front-end circuitry is an important aspect which makes effectiveness the energy harvesting system. Furthermore, none of the systems is controlling the output voltage, hence the output power is not fixed, but it is the natural result of the behaviour of the entire system. All these considerations give strength to the choice of focusing on a different figure of merit (FoM) to compare the converters; for example this could be the ratio of the harvested power with respect to the maximum power theoretically available from the transducer. This last quantity can be defined studying the behaviour of a piezoelectric transducer, at no load condition, when exposed to a sinusoidal acceleration at frequency *f* which strains the transducer as shown in Figure 17.

Starting from its rest condition at t_0, that is no strain and no electrical charges on its plates, the cantilever reaches its maximum positive bending (and strain) at t_1. At this point the collected electrical charge is Q_P and the piezoelectric output voltage is equal to V_{PO}. If an ideal front-end circuitry could recover all of this charge, completely discharging the transducer, when the cantilever reaches its maximum negative deflection at t_3 it will collect a negative charge two times higher ($-2Q_P$) because the deflection is doubled; in fact, at t_2 the collected charge is already equal to $-Q_P$. As a consequence the piezoelectric peak voltage at t_3 is equal to $-2V_{PO}$ and the transduced energy is:

$$E_{t_1-t_3} = \frac{1}{2} C_P \left(-2V_{PO}\right)^2 = 2C_P V_{PO}^2 \tag{6}$$

Now imagine that at t_3, again ideally, capacitor C_P is newly, fully discharged while, at the same time, the associated energy is stored. Since the excursion in the t_3-t_4 interval

starts with no charge on the capacitor plates, it evolves as a replica of what happened in the t_1-t_3 interval, only the sign of the voltage is inverted. This means that the available energy at t_4 is equal to:

$$E_{t_3-t_4} = \frac{1}{2} C_P (2V_{PO})^2 = 2C_P V_{PO}^2 \tag{7}$$

From this moment on the phenomenon continues to repeat itself with a periodicity equal to t_4-t_1.

Summarizing the total energy that can be harvested during one cycle becomes:

$$E_{cycle} = E_{t_1-t_3} + E_{t_3-t_4} = 4C_P V_{PO}^2 \tag{8}$$

Finally, the maximum power theoretically available from the transducer is:

$$P_{max} = 4C_P V_{PO}^2 f \tag{9}$$

The quantity expressed by (9) will be used to weight the power actually delivered to the load by the proposed rectifiers. This approach allows to compare the various circuits, but is also allows to quantify how effective a given circuit is in harvesting the power made available by the transducer.

The figure of merit, which is function of R_L, can be defined as:

$$F.o.M. = \frac{P_L}{P_{max}} \tag{10}$$

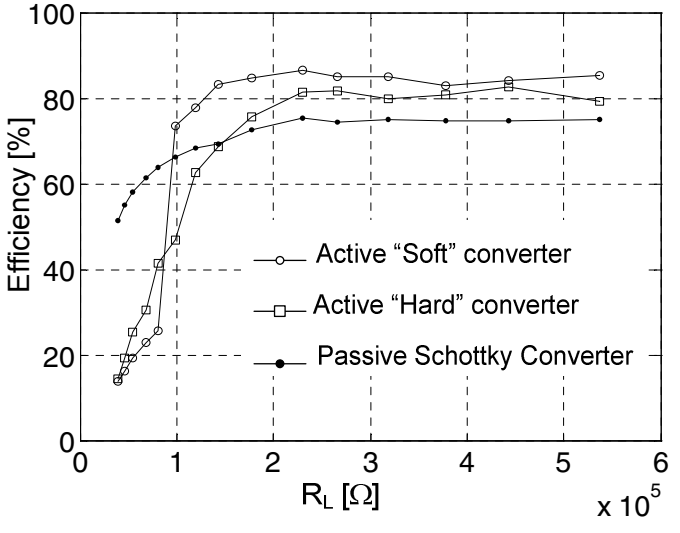


Fig. 16. Efficiency of the AC-DC converters at V_{PO}=1.5V against load resistance

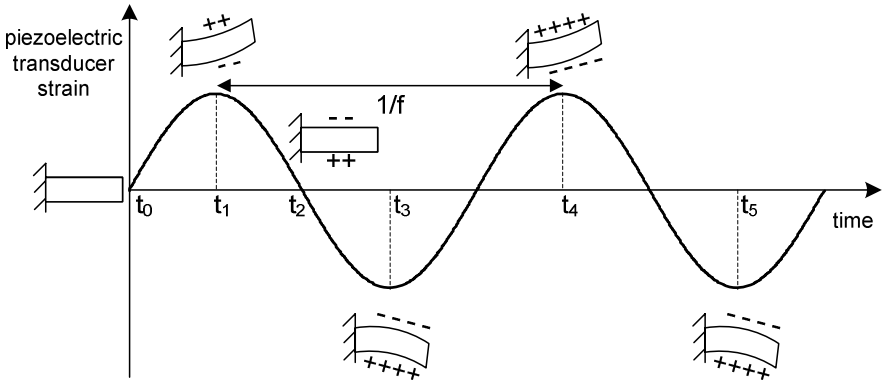


Fig. 17. Mechanical strain of piezoelectric trasnducer when it is excited by a sinusoidal mechanical acceleration

Figure 18 shows this figure of merit for the presented AC-DC converters in case of V_{PO} equal to 1.5V. It is possible to see that "hard" topology outperforms both "soft" and passive ones but, at the same time, none of the converters is able to harvest all the power the transducer could deliver.

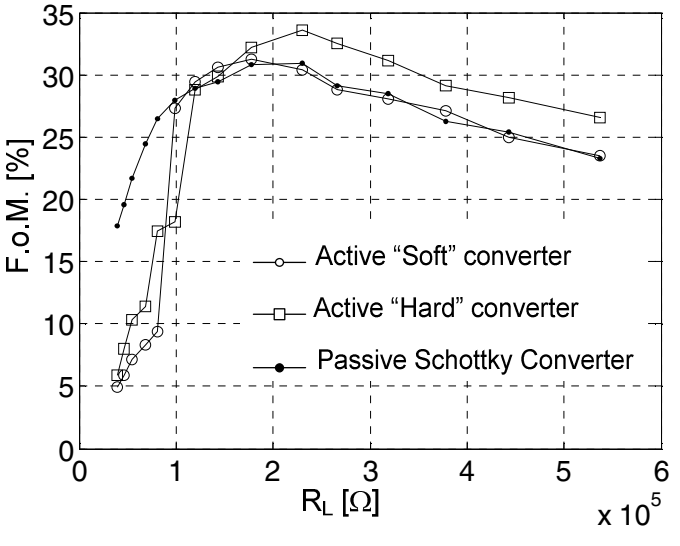


Fig. 18. Power harvested by the AC-DC converters with respect to the maximum defined in Equation 7

4 Conclusions

This paper presents two actives AC-DC converters which can be used into piezoelectric energy scavenging systems. The basic circuit is the voltage doubler

topology, where the switches are actively driven. The two considered driving circuitries are realized with comparators and operational amplifier respectively.

It was shown that the first configuration, called “hard” is slightly more difficult to be designed than the second one, called “soft”. In particular the comparators need to have an hysteretic characteristic and if the value of the thresholds is different from the ideal one the efficiency of the converter could be significantly deteriorated. Hence a lot of effort is needed to minimize the process dependence of the thresholds. On the contrary, the “soft” configuration drives the switches with a regulation loop approach, which is far less sensitive to process uncertainties.

A test chip has been diffused in STMicroelectronics 5V CMOS technology and experimental results are presented. The tests were realized using a piezoelectric transducer working in 31-mode, excited by a electro-dynamic shaker.

The performances of the proposed active solutions were evaluated at different values of a load resistance and they were compared with the ones of a passive Schottky-based voltage doubler. The average power supplied by the piezoelectric transducer and the power delivered on a load resistance were measured. The experimental results show that, under the same mechanical excitation, the "hard" solution outperforms the other two converters in terms of power delivered to the resistive load. The "soft" solution was less effective because the delivered average power by the piezoelectric transducer was lower; this was probably due to a different matching between the piezoelectric transducer and the converter itself which is better in the "hard solution". The efficiencies of the three converters were compared at different load conditions. The "soft" solution, despite its lower effectiveness, has the higher efficiency values with respect to the other two converters: this means that the converter efficiency can not be used as the only figure of merit for an energy harvesting system, it is necessary to evaluate also the effectiveness of the converter in terms of average power harvested from the transducer and delivered to the load.

To do this a dedicated figure of merit was introduced which compares the output power with the power that the transducer can theoretically deliver rather than with the power it actually delivers.

References

[1] Roundy, S.: Energy Scavenging for Wireless Sensor Nodes with a focus on Vibration to Electricity Conversion. PhD Thesis, The University of California Berkeley (Spring 2003)

[2] Roundy, S., et al.: Improving Power Output for Vibration Based Energy Scavengers. Pervasive Computing, 28–36 (January/March 2005); Published by the IEEE and IEEE ComSoc.

[3] Le Jifeng Han, T.T., von Jouanne, A., Mayaram, K., Fiez, T.S.: Piezoelectric Micro Power Generation Interface Circuits. IEEE Journal of Solid State Circuits 41(6), 1411–1420 (2006)

[4] Dallago, E., Frattini, G., Miatton, D., Ricotti, G., Venchi, G.: Self Supplied Integrable High Efficiency AC-DC Converter for Piezoelectric Energy Scavenging Systems. In: International Sympoium on Circuits and Systems, ISCAS 2007, New Orleans, LO, May 27-30, pp. 1633–1636 (2007)

[5] Dong, Z., Allen, P.E.: Low Voltage, Supply Independent CMOS Bias Circuit. In: The 2002 45th Midwest Symposium on Circuits and Systems, August 4-7, vol. 3, pp. 568–570 (2002)
[6] Peano, F., Tambosso, T.: Design and Optimization of a MEMS Electret Based Capacitive Energy Scavenger. Journal of Microelectromechanical Systems 14(3), 429–435 (2005)
[7] Renaud, M., Sterken, T., Fiorini, P., Puers, R., Baert, K., van Hoof, C.: Scavenging Energy from Human Body, Design of a Piezoelectric Transducer. In: The 13th International Conference on Solid State Sensors, Actuators and Mycrosystems, Seoul, Korea, June 5-9, pp. 784–787 (2005)
[8] Stark, I.: Thermal Energy Harvesting with Thermo Life®. In: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, BSN 2006, pp. 19–22. IEEE Computer Society, Los Alamitos (2006)
[9] Han, J., von Jouanne, A., Le, T., Mayaram, K., Fiez, T.S.: Novel power conditioning Circuits for Piezoelectric Micro Power Generators. In: APEC 2004, February 2004, vol. 3, pp. 1541–1546 (2004)
[10] Sterken, T., Fiorini, P., Baert, K., Puers, R., Borghs, G.: An Electret-Based Electrostatic μ-generator. In: The 12th IEEE International Conference on Solid State Sensors, Actuators and Microsystems, Boston, June 8-12, vol. 2, pp. 1291–1294 (2003)
[11] Dallago, E., Frattini, G., Miatton, D., Ricotti, G., Venchi, G.: Integrable High Efficiency AC-DC converter for Piezoelectric Energy Scavenging System. In: IEEE International Conference on Portable Information Devices, Orlando (FL), USA, March 25-29, pp. 1–5 (2007)

Trapping Biological Species in a Lab-on-Chip Microsystem: Micro Inductor Optimization Design and SU8 Process

Christophe Escriba[1,2], Rémy Fulcrand[1,2], Philippe Artillan[1,2], David Jugieu[1,2], Aurélien Bancaud[1,2], Ali Boukabache[1,2], Anne-Marie Gue[1,2], and Jean-Yves Fourniols[1,2]

[1] LAAS-CNRS ; Université de Toulouse ; 7, avenue du Colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse; UPS, INSA, INP, ISAE

Abstract. Micro-spiral inductor dedicated to microbeads manipulation in a fluidic channel had been optimized by analytical modelling correlated to multi physics fem numerical Maxwell®3D L software. Main advantage of the analytical model described below is time analysis calculus decrease of and the capability offered to optimize geometrical and electrical parameters of the inductor. First experimental results show a good correlation between simulation and realized integrating micro-devices in a fluidic channel.

Keywords: Analytical model, Finite element method, Magnetic field / force, Magnetic actuators, magnetic bead separation.

1 Introduction

Lab-On-Chip micro-systems with embedded analysis are the two major concepts of MEMS dedicated to fluidic applications. In order to realize biochemical analysis or pharmacological screening, separation of biological species, we propose magnetic bead as a very useful technique. In most cases, the magnetic activation is macroscopic and positioned outside of the system. As these approaches limits strongly the MEMS integrating process, we develop in LAAS-CNRS lab, a multi-functional magnetic source in order to realize a complete fluidic micro-system for handling paramagnetic beads.

By reporting here first works dedicated to model the behaviour of magnetic sources and the forces generated on microbeads, we present two methods allowing to determine magnetic field generated by a planar inductor: one entirely analytical and other by using the multi-physics software Maxwell®3D. Comparisons are discussed in terms of performances, accuracy, and CPU time computation. As a complementary part, we present first technological results of research (in progress) in order to integrate a family of inductors in channels. These preliminary devices confirm the feasibility of our approach.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 81–96, 2010.

2 Determination of Magnetic Field Distribution in a 3D Space. Analytical Modeling

The magnetostatic field induced by a DC current flowing in an electric conductor is given by the following two Maxwell's equations:

$$\nabla \times \vec{H} = \vec{J} \tag{1}$$

$$\nabla \cdot \vec{B} = 0 \tag{2}$$

With the following constitutive (material) relationship being also applicable:

$$\vec{B} = \mu_0 \mu_r \vec{H} + \mu_r \overrightarrow{M_p} \tag{3}$$

- $H(x, y, z)$ is the magnetic field strength
- $B(x, y, z)$ is the magnetic flux density.
- $J(x, y, z)$ is the conduction current density.
- $Mp(x, y, z)$ is the permanent magnetization.
- $\mu_0 = 4 * \pi * 10^{-7} H.m^{-1}$ is the permeability of vacuum.
- μ_r is the relative permeability.

First, we consider all the segments constituting the inductor which coordinates are $[\alpha_i\ \omega_i]$ in $\Re$ frame (Figure 1).

Geometrical and electrical characteristics are:

- L_1 : length to the first segment of the inductor (axis Ox)
- L_2 : length to the second segment of the inductor (Oy)
- N : number of spires,
- s : inter-coil distance,
- I : current in all segments

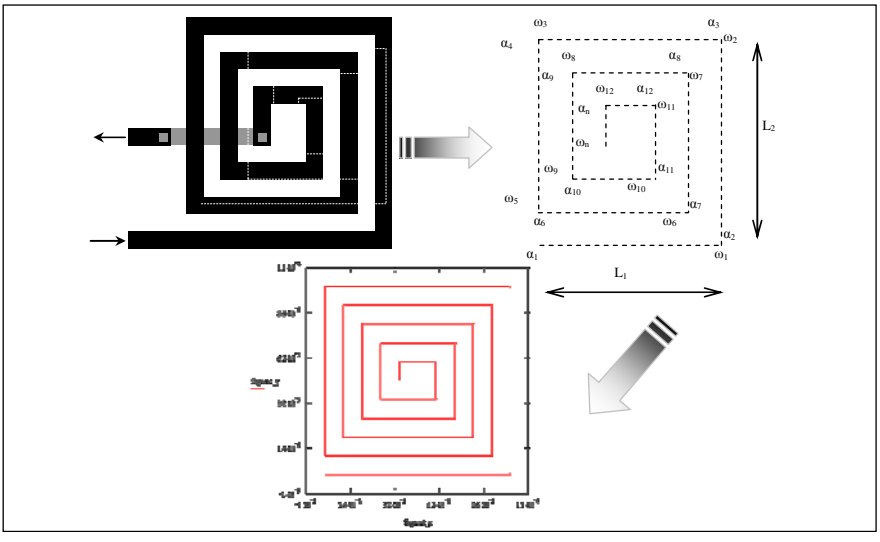


Fig. 1. Equivalent topology of planar inductor and display results for five spires inductor

In order to evaluate quickly magnetic field generated by the induction coil, we consider the inductor in frame $\Re$ like a set of elementary segments where the magnetic field B is first obtained for each segment [α_i ω_i], and the total magnetic field will be determined by summation of each elementary field calculated in the frame $\Re'$

A. STEP1: Magnetic field expression $\vec{B}(x',y',z')$ created by a single segment in $\Re'$

We consider M point, included in the (x0y) area where we define [α, ω] segment. We define the orthonormal frame $\Re'$ with α origin, and first axis [α, ω], Figure 2.

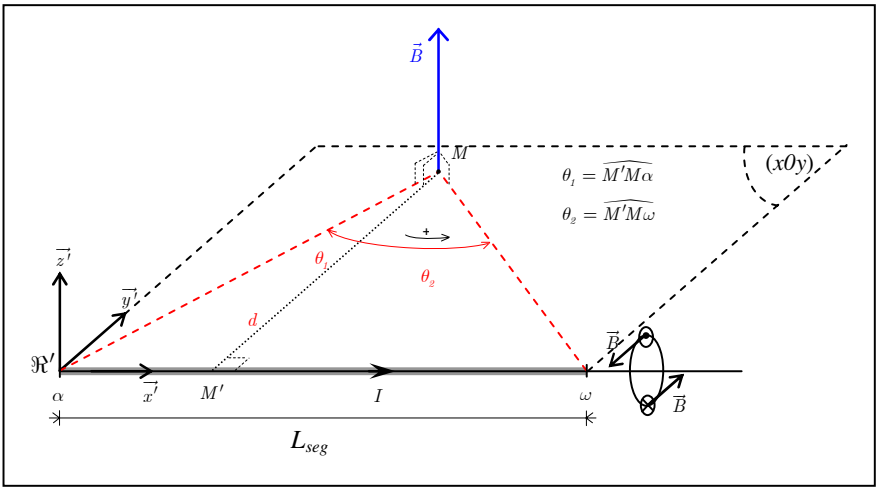


Fig. 2. Magnetic field created by the [α ω] segment at point M

Distance d between point M and [α, ω] segment is evaluated by the equation:

$$d = \sqrt{y^2 + z^2} \tag{4}$$

Using θ_2, θ_1 angles defined in Figure 2, the magnetic field $\vec{B}(x',y',z')|_{\Re'}$ in $\Re'$ frame is:

$$B(x',y',z')|_{\Re'} = \frac{\mu_0 I}{4\pi d}\left[\sin\theta_1 - \sin\theta_2\right]\vec{z'} \tag{5}$$

where $\theta_1 = -tan^{-1}\left(\frac{x}{d}\right)$ and $\theta_2 = tan^{-1}\left(\frac{L_{seg} - x}{d}\right)$

B. STEP2: Summation of all magnetic field $\vec{B}$ segments

The $\Re'$ frame is obtained by a translation of T vector and a linear transformation through [P] matrix (Figure 3).

$$\overrightarrow{OM}\,|_{\Re} = \vec{T}\,|_{\Re} + \left[P\right]\cdot \overrightarrow{\alpha M}\,|_{\Re'} \tag{6}$$

With:

$$\vec{T} = \overrightarrow{O\alpha}\ ,\ \vec{x'} = \frac{\overrightarrow{\alpha\omega}}{\left\|\overrightarrow{\alpha\omega}\right\|}\ ,\ \vec{z'} = \frac{\overrightarrow{\alpha\omega}\times\overrightarrow{\alpha M}}{\left\|\overrightarrow{\alpha\omega}\times\overrightarrow{\alpha M}\right\|}\ ,\ \vec{y'} = \vec{z'}\times\vec{x'} \tag{7}$$

Where ($\times$ is vector product).

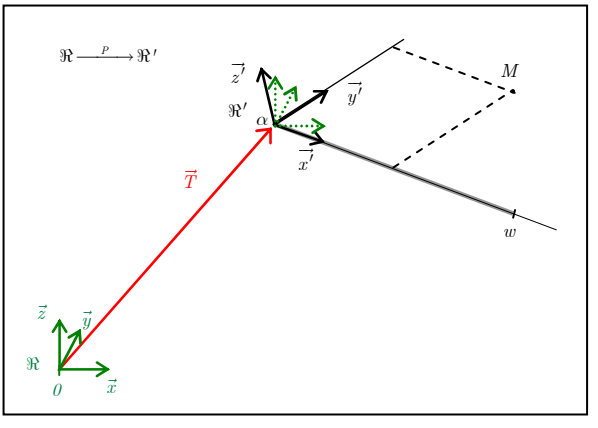


Fig. 3. $\Re \leftrightarrow \Re'$ frame definitions

Combining M coordinates with α and ω points, we obtain scalar elements of [P] matrix by concatenation of $\vec{x'}$, $\vec{y'}$, $\vec{z'}$:

$$P = \left[\vec{x'}\,|\,\vec{y'}\,|\,\vec{z'}\right] \tag{8}$$

M point coordinates are showing in $\Re$ frame. For calculus of every components of B_i, it's required to transfer this coordinates in $\Re'$ according to (Figure 4):

$$\overrightarrow{\alpha_i M}\,|_{\Re_i'} = P^T \cdot \left(\overrightarrow{OM}\,|_{\Re} - \vec{T}\,|_{\Re}\right) \tag{9}$$

Considering, $\vec{B_i}\,|_{\Re} = P\cdot \vec{B_i}\,|_{\Re_i'}$ magnetic field associated to $[\alpha_i, \omega_i]$ segment, the total magnetic field is obtain by summation of every B_i components:

$$\overrightarrow{B_{tot}}\,|_{\Re} = \sum_{i=1}^{N_{seg}} \vec{B_i}\,|_{\Re} \tag{10}$$

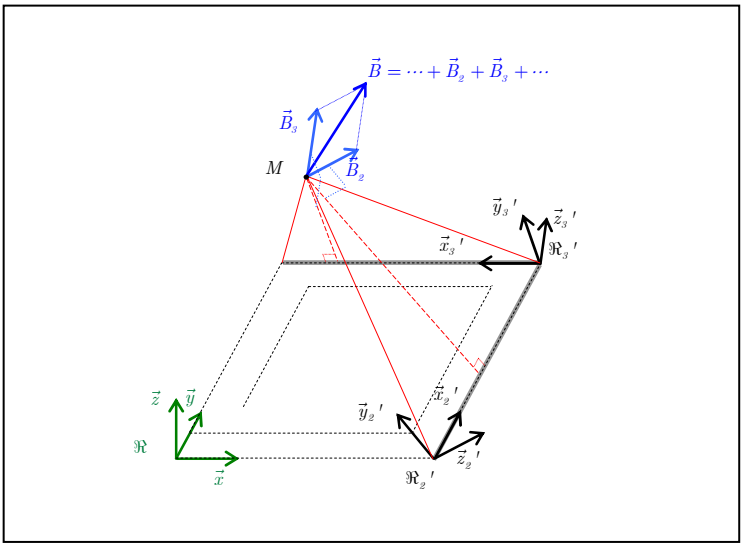


Fig. 4. Coordinates systems describing the micro-inductor and the global magnetic field at M point

C. Estimation of magnetic field of planar induction

Figure 5, represents the magnetic field computed with our algorithm, for a inductor with 5 spires, side 100 μm by 100 μm covered by an electrical current I = 5 mA at high z = 50 μm.

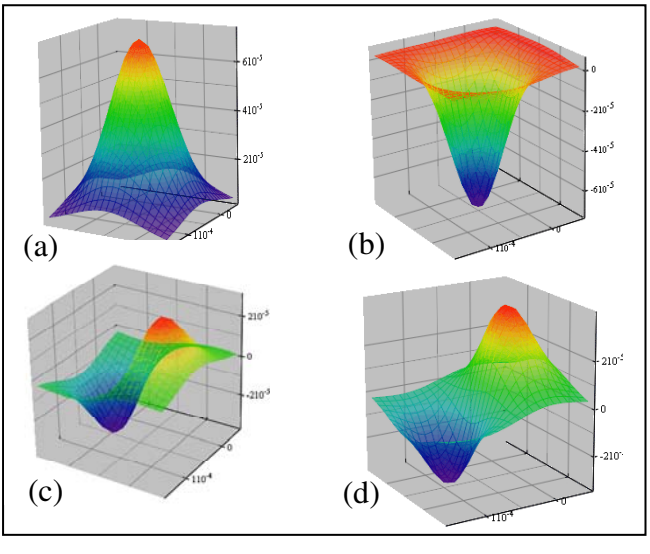


Fig. 5. Magnetic field computed: (a) modulus; (b) along Oz axis, (c) along Ox axis, (d) along Oy axis

We can observe that the z component present an extrema above the center of the inductor and a value close to zero along Ox and Oy.

3 Validation of the Analytical Approach

In order to check the validity of the previous analytical modelling, we compare results obtained width the Maxwell®3D software and hence a FEM approach with those obtained with our model.

The electric field is restricted to the objects modelled as real (non ideal) conductors and is totally decoupled from the magnetic field. We consider that there no time variation effects and objects are considered to be stationary. So previously, the magnetostatic field solution verifies the Maxwell's equations. The MAXWELL3D magnetostatic solver considers the magnetic field H with the following components:

$$\overrightarrow{H} = \overrightarrow{H_p} + \nabla\varphi + \overrightarrow{H_c} \tag{11}$$

Where φ is the magnetic scalar potential, $\overrightarrow{H_p}$ is a particular solution constructed by assigning values to all the edges in the mesh in such a way that Ampere's law holds on all contours of all tetrahedral faces in the mesh, and $\overrightarrow{H_c}$ accounts for permanent magnet if any. Thus, the Degrees Of Freedom (DOFs) are the nodal values of the magnetic scalar potential with ten values per tetrahedron at each of the four vertices and all six mid edge nodes, ensuring a quadratic approximation inside each finite element.

There are major advantages of this formulation over other existing ones, including using considerably fewer computational resources (due to the scalar nature of the DOFs), not requiring a gauge due to excellent numerical stability, significantly reducing cancellation errors, and capably of automatically multiplying connected iron regions. The magnetostatic solver handles both 3D linear and nonlinear problems. The magnetostatic solver calculates the magnetic field distribution produced by a combination of known DC current density vector distribution (Figure 6).

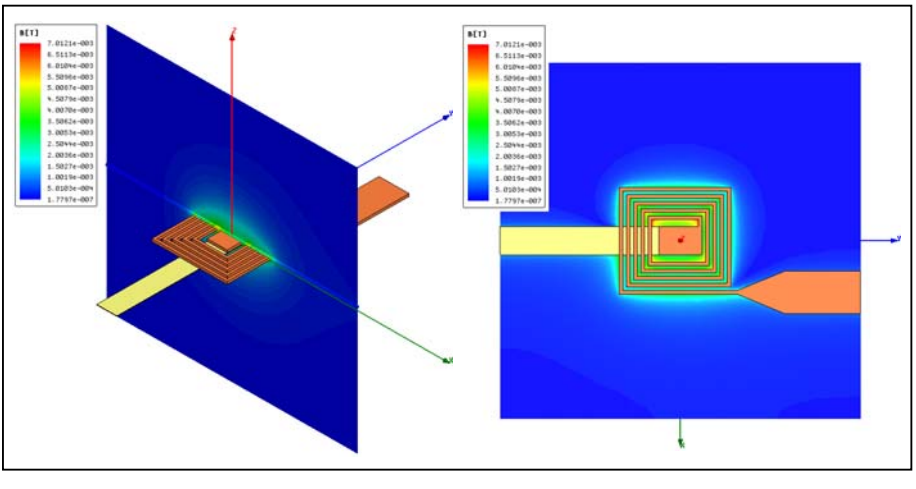


Fig. 6. 3D representation of the magnetic field. a) in (X, Z) plan and b) in (X, Y) plan

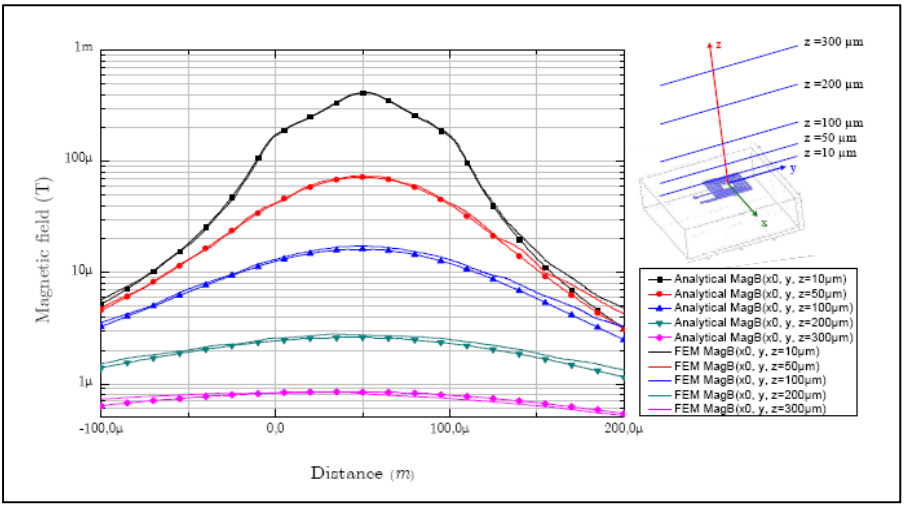


Fig. 7. Magnetic field calculated by analytical and finite element methods

Various coil designs have been simulated with the use of MAXWELL3D. For example, Figure 6 shows the magnetic field calculated for a micro-coil with 5 turns, a conductor square section of 5µm × 5µm and a 50mA current.

The comparison between analytical and FEM modelling of magnetic field $\vec{B}$ is detailed in Figure 7, where we have reported the values of B calculated at different heights z from the coil.

These results confirm the good correlation between the two methods and fully validate the efficiency of the analytical model. Thus we can assume that the analytical model developed is powerful for parametric optimization.

4 Application: Magnetic Beads Trapping

Using our analytical model, we have optimized the design of an integrated magnetic actuator dedicated to the manipulation of microbeads in lab on chip devices. The criterion of optimization was to obtain magnetic force allowing the trapping super paramagnetic microbeads in micro-channels. The magnetic forces have been calculated starting from the simulation results. For this purpose, we have integrated some micro-inductors in a micro-fluidic device made entirely with a biocompatible SU-8 polymer. This prototype will undergo a series of characterization and validation.

5 Magnetic Force Exerted on a Magnetic Microbead

The force $\vec{F}_p^m$ exerted by a magnetic field on a particle is represented by the gradient of the magnetic interaction energy of the particle immersed in the magnetic field.

$$\vec{F}_p^m = -\nabla \vec{U}_p^m \tag{12}$$

The magnetic energy can be expressed by:

$$\vec{U}_p^m = -\frac{1}{2}\mu_0 \int \vec{M}_p . \vec{H} . dv \tag{13}$$

Where $\mu_0 = 4 * \pi * 10^{-7} H.m^{-1}$ is the vacuum permeability and $\vec{M}_p$ the magnetizing of this particle by the magnetic field of excitation $\vec{H}$.

In the case of a particle with very small dimension, the integration of (13) is then replaced by the value of the field in the center of the particle multiplied by the volume V_p of this particle:

$$\vec{U}_p^m = -\frac{1}{2} V_p \mu_0 \vec{M}_p . \vec{H} \tag{14}$$

In air or the vacuum, a diamagnetic or paramagnetic micro particle acquires a magnetizing $\vec{M}_p$ which a function of the magnetic susceptibility χ_p, the field of excitation $\vec{H}$ and the demagnetization coefficient D (D = 1/3 for a sphere, D=1 for a parallelepiped, D = 0 long bar).

$$\vec{M}_p = \frac{\chi_p}{1+\chi_p . D} \vec{H} \tag{15}$$

It is considered that $1 + \chi_p . D \approx 1$ and equation (15) can be simplified:

$$\vec{M}_p = \chi_p \vec{H} \tag{16}$$

Equation (14) becomes then:

$$\vec{U}_p^m = -\frac{1}{2} V_p \mu_0 \chi_p |H|^2 \tag{17}$$

The expression of the magnetic force deduced from (12) and (17) is given by:

$$\vec{F}_p^m = \mu_0 V_p \chi_p (\vec{H} . \nabla) \vec{H} \tag{18}$$

This expression shows that direction of the force does not depend on the field sign but of the product of field with its gradient. Its orientation is also related to the susceptibility of the particle. The total magnetic force exerted on the particle is given by:

$$\vec{F}_{MagTtl} = \vec{F}_p^m + \vec{F}_f^m = \mu_0 V_p (\chi_p - \chi_f)(\vec{H} . \nabla) \vec{H} \tag{19}$$

Because of very low value of susceptibility of the immersing medium $(\chi_f = -9.048 * 10^{-6} \text{ for water})$ we assume that the magnetic field induced by the medium can be neglected and that:

$$\vec{B} = \vec{B}_{coil} = \mu_0 \vec{H} \tag{20}$$

The equation of the total magnetic force becomes:

$$\vec{F}_{MagTtl} = \frac{V_p \chi_p}{\mu_0} (\vec{B}_{coil} . \nabla) \vec{B}_{coil} \tag{21}$$

The components of the magnetic force in a Cartesian reference mark are expressed by:

$$\vec{F}_{MagTtl} = \frac{V_p}{\mu_0} (\chi_p) (B_x \quad B_y \quad B_z) \begin{pmatrix} \frac{d}{dx} \\ \frac{d}{dy} \\ \frac{d}{dz} \end{pmatrix} (B_x \quad B_y \quad B_z) \tag{22}$$

Thus in the case of a super paramagnetic particle in a conveying fluid like water, the particle is attracted towards magnetic field maxima.

By convention, the magnetic actuator (coils) being in (X, Z) plan, the force is taken into account only in the (X, Z) plan (Figure 8) and:

$$F_{MagTtlx}(x, z) = \frac{V_p}{\mu_0} (\chi_p) \left(B_x \frac{dB_x}{dx} + B_z \frac{dB_x}{dz} \right) \tag{23}$$

$$F_{MagTtlz}(x, z) = \frac{V_p}{\mu_0} (\chi_p) \left(B_x \frac{dB_z}{dx} + B_z \frac{dB_z}{dz} \right) \tag{24}$$

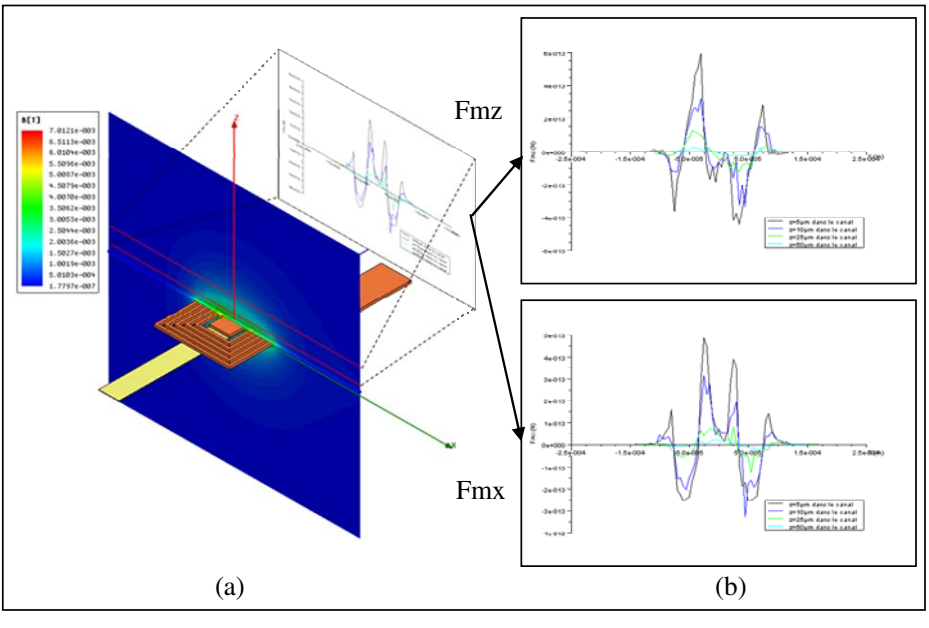


Fig. 8. (a) Sight 3D of the device and the plan (X, Z) where is represented the distribution of the magnetic field B. In red, the fluidic channel where calculations of the forces for different height are localized. (b) Graphic of the magnetic forces according to X and Z obtained by the post-processing.

6 Technological Realizations

6.1 Process Flow for the Fabrication of Integrated Microsystem

One of the main tasks of this work was to develop a generic polymer based technology in order to manufacture a microfluidic device able to trap functionalized micro beads. As it will be shown below, this technology combines standard electroplating technique and lamination technologies in order to achieve flexible and transparent full polymer systems with the precision and reliability of "stat-of-the-art" microsystems. The requirements for the integrated microsystem are:

→ accurate alignment between the fluidic levels
→ high reliability process
→ flexible microsystem
→ optically transparent material
→ biocompatibility.

According to these criteria, we have chose to utilize the polyethylene terephthalate layer (PET sheet, for the flexibility) as a "sacrificial" material, the epoxy-based negative photoresist SU-8 on the structural material for the microfluidic network (transparency, biocompatibility, very high aspect ratio…) and copper for the electroplating process of the micro-coil. The technological process flow can be described in Figure 9.

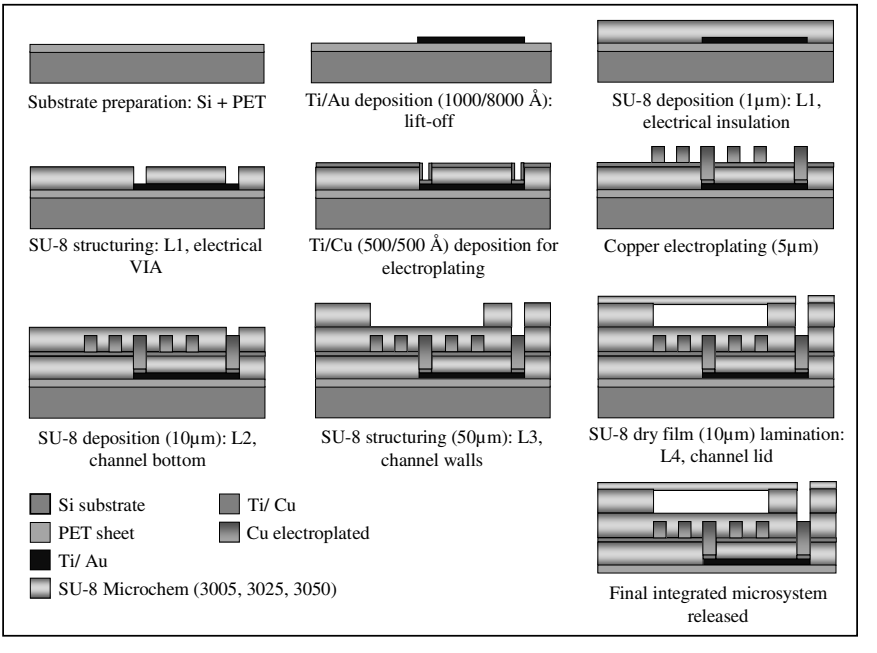


Fig. 9. Process flowchart for the fabrication of the flexible integrated microsystem

As a first step, a PET sheet (50µm Polyethylene Terephthalate film) is laminated on the top of a silicon wafer. The PET was chosen for its poor adhesion properties and transparency. Then, conductive tracks are patterned (Ti/Au) by lift-off processing (Figure 10 (a)). A layer of SU-8 (Microchem 3005) is deposited on top of the structure in order to perform electrical insulation between conductive tracks and coils, and to create electrical VIA (Figure 10 (b)). A seed layer of Ti/Cu (500/500 Å) is deposited followed by the deposition and patterning of a positive photoresist (AZ4562) in order to create the electroplating mold. Then, copper coils are electroplated (5µm) into the resist mould (Figure 10 (c)) and the photoresist and seed layers (Ti/Cu) are removed.

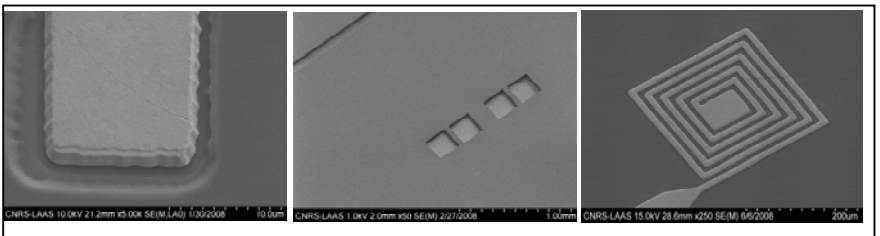


Fig. 10. SEM pictures of the different steps processing

A layer of SU-8 negative photoresist (Microchem 3005, 10µm) is spin-coated and patterned to form the channel bottom and insulate microcoils from flowing liquids. Microfluidic network is then fabricated: channel height and width are respectively 50 and 500µm, and thickness of the channel lid is 10µm. The SU-8 microfluidic network is optically transparent (n = 1.8) as demonstrate on Figure 11 and allows therefore optical detection and characterization of the microsystem. The resulting device was finally released from the substrate thanks to the poor adhesion of the PET layer.

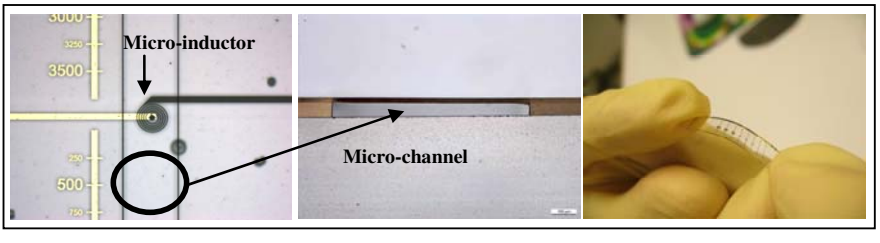


Fig. 11. Photographs of the flexible integrated microsystem with SU-8 microfluidic network

6.2 Magnetic Microbeads Trapping

Magnetic micro and nano-beads have proven to be very interesting and reliable tool in biological and chemical analysis in recent years. Separation or purification are often

practiced using magnetic labeled beads in biological laboratory and sometimes for biomedical diagnosis [1, 2]. The experimental procedure for the realization of an experimental bead separation is illustrated in Figure 12. In general, bead based bioanalysis protocols use permanent magnet [3, 4] and repeat a specific sequence manipulation.

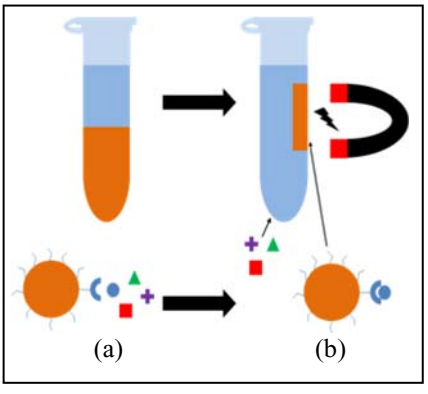


Fig. 12. Procedural sequences for magnetic bead separation. a) Magnetic bead labeled incubation with antigen target. b) Separation of magnetic bead to the solution.

In a first step, magnetic microbeads are functionalized with a probe (antibody) specific of the targeted molecule. Then, the target (antigen) contain in the sample is recognized by the probe and adsorbs specifically on the microbead surface. Finally, microbeads are trapped by an permanent magnet. The goal of our study was to implement such kind of protocol in a lab on chip device integrating a microelectromagnet. Compared to permanent magnets, electromagnets offer a higher flexibility and a higher control of magnetic field. Since small dimensions are able to be manufactured in microfluidic system, the combination of microelectromagnets and microfluidic network offers an interesting approach to this integrated microsystem [5, 6, 7 and 8].

6.3 Real Time Experimentation and Validation

The choice of the type of magnetic microbeads depends on the application specifications. In this study, we choose to use Dynabeads® M500 Subcellular microbeads and Dynabeads® M270 Carboxylic Acid with diameter of respectively 5 µm and 2.8µm. Figure 13 shows a SEM photograph of Dynabeads® M270 and an optical picture of microbeads in microfluidic channel.

In order to validate our technological concept, chips integrating microelectromagnets in a microfluidic channel have been fabricated. Magnetic bead separation has been performed using these systems. Figure 14 shows a schematic overview (a) and cross-section (b) of the system.

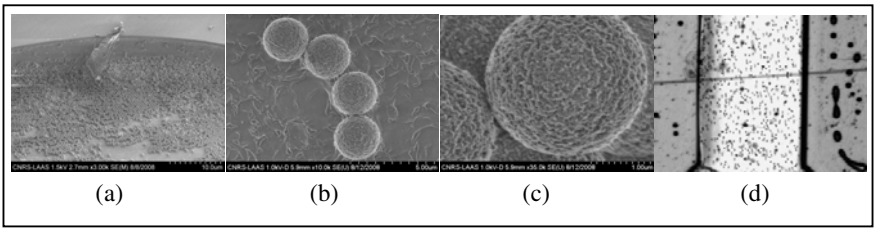


Fig. 13. SEM photographs of Dynabeads M270 (a, b and c), and optical picture of flow microbeads in micro-channels (d)

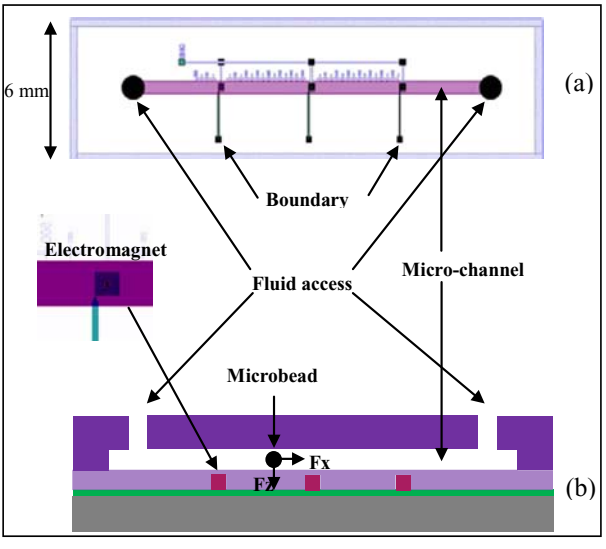


Fig. 14. The complete microsystem. a) An overview of the chips with three microelectromagnets, the micro-channel with flow inlet and outlet. b) A cross-section of the chip

The experimental setup is shown in Figure 15. The instrumental bench includes a CCD camera for recording of bead trapping, a syringe for dispensing the liquid sample and a DC power supply.

A driving current of 50mA was applied to the inductor. Experimentation were performed using a square micro-coils with 5 turns and a square section (5µm*5µm).

The video frame is constituting by 12 frames per second. The main steps of trapping are reported on Figure 16:

- ⇨ Firsr we present the magnetic microbead contained in the microfluidic channel before supplying current to the microelectromagnet (0s).
- ⇨ Then, current was applied to the microelectromagnet and microbeads are trapped at the micro-channel bottom just above the micro-coils (transient analysis from 10s to 100s).

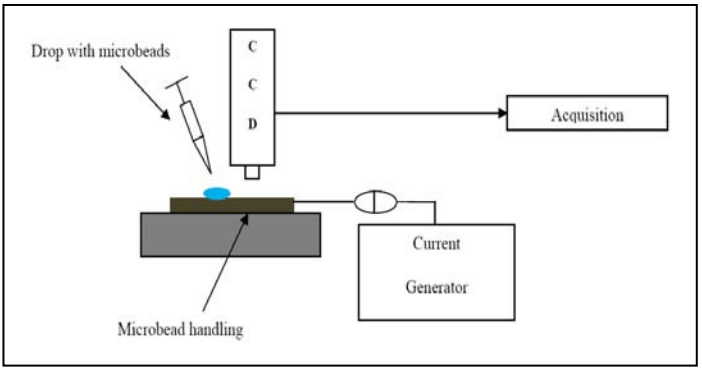


Fig. 15. Experimental setup to test and demonstrate our microbead handling

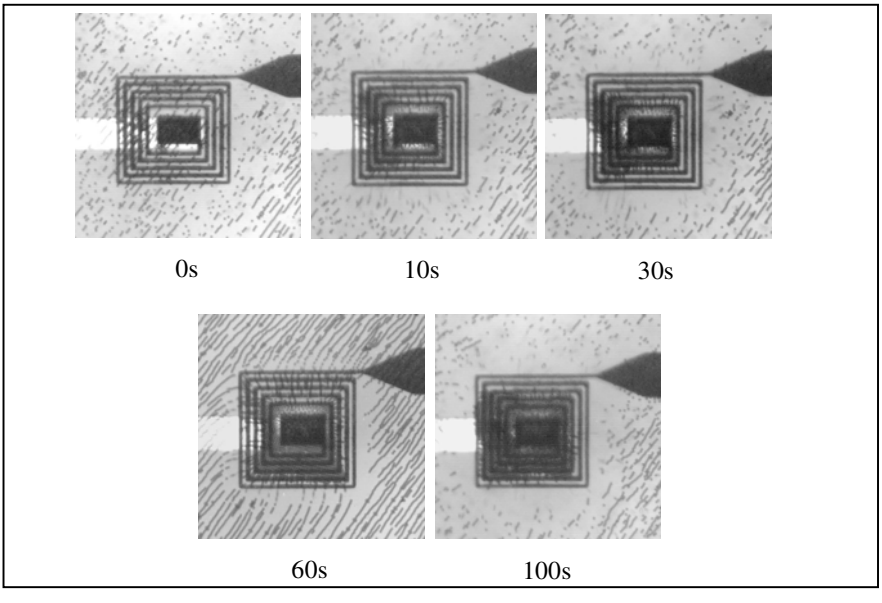


Fig. 16. Magnetic beads trapping using our microelectromagnets. (0s) without excitation applied. (10s to 100s) after excitation (I=50mA) applied to the microelectromagnet: trapping beads.

Figure 17 shows a comparison between real time magnetic beads trapping and modeling to magnetic field distribution around the microelectromagnets.

This example put the stress on the good correlation between our Finite Element approach and our experimental realization. Future work will be done to further improve the trapping and sorting efficiency of the flexible integrated microsystem by a design optimization of our microelectromagnets and microfluidic network.

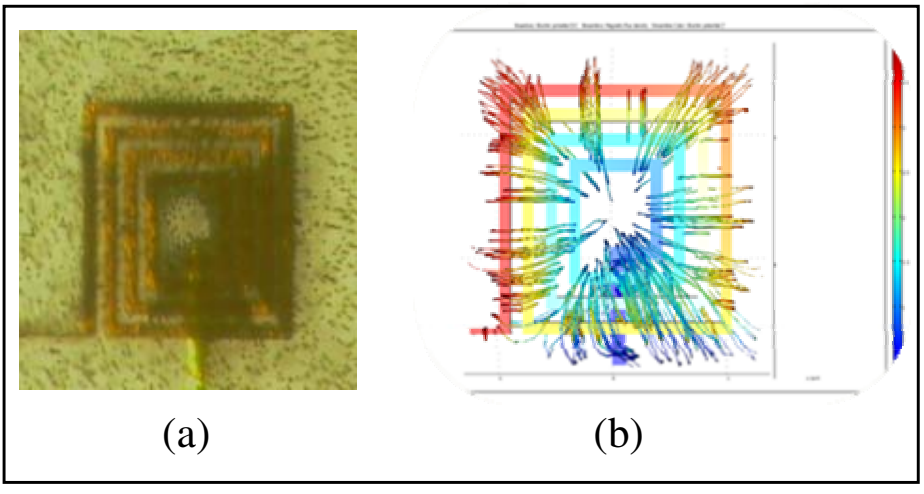


Fig. 17. Comparison between our real time experimentation and modeling: magnetic field repartition around the microelectromagnet

7 Conclusion

In this study, we describe magnetic field modelling created by a planar micro-inductor. A 3D model of the micro-spiral inductor was been realized by using our own analytical model and by using a second method based on COMSOL Multiphysics software. These two approaches allow us to check the magnetic field distribution generated by the inductor and the force exerted on magnetic bead. All these studies show that many parameters can influence the inter-action between the bead and the micro-spiral inductor. The model offer the behaviour response to any electrical current and can optimized geometrical coil parameters respect to microbeads and fluid characteristics. The first experimental results with a fluidic structure fabricated entirely in a biocompatible polymer give to our approach a novel dynamic and validate the concept and its analytical model used for actuating the biological species.

References

[1] Norgall, S., Papoutsi, M., Rôssler, J., Schweigerer, L., Wilting, J., Weich, H.A.: Elevated expression of VEGFR-3 in lymphatic endothelial cells from lymphangiomas. BMC Cancer 7, 105 (2007)

[2] Safarik, I., Safarikova, M.: Magnetic techniques for the isolation and purification of proteins. Biomagnetic Research and Technology 2, 7 (2004)

[3] Pamme, N., Eijkela, J.C.T., Manza, A.: On-chip free-flow magnetophoresis: Separation and detection of mixtures of magnetic particles in continuous flow. Journal of Magnetism and Magnetic Materials 307, 237–244 (2006)

[4] Kim, Y.H., Hong, S., Kim, B., Yun, S., Kang, Y.R., Paek, K.K., Lee, J.W., Lee, S.H., Ju, B.K.: Droplet-based Magnetically Activated Cell Separation. In: Proceedings of the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, September 1-5 (2004)
[5] Smistrup, K., Lund-Olesen, T., Hansen, M.F.: Microfluidic magnetic separator using an array of soft magnetic elements. Journal of Applied Physics 99, 08P102 (2006)
[6] Deng, T., Whitesides, G.M.: Manipulation of magnetic microbeads in suspension using micromagnetic systems fabricated with soft lithography. Applied Physics Letters 78, 1775–1777 (2001)
[7] Ramadan, Q., Samper, V., Poenar, D., Yu, C.: On-chip micro-electromagnets for magnetic-based bio-molecules separation. Journal of Magnetism and Magnetic Materials 281, 150–172 (2004)
[8] Rida, A., Fernandez, V., Gijs, M.A.M.: Long-range transport of magnetic microbeads using simple planar coils placed in a uniform magnetostatic field. Applied Physics Letters 83, 2396 (2003)

Fine-Grain Reconfigurable Logic Cells Based on Double-Gate MOSFETs

Ian O'Connor, Ilham Hassoune, and David Navarro

University of Lyon, Lyon Institute of Nanotechnology UMR 5270
Ecole Centrale de Lyon
36 avenue Guy de Collongue, F-69134 Ecully cedex, France
ian.oconnor@ec-lyon.fr

Abstract. This work presents a new style of gate-level reconfigurable cells based on the double-gate (DG) MOSFET device. The proposed dynamic- and static-logic cells demonstrate significant gate area reductions compared to conventional CMOS lookup table (LUT) techniques (between 80-95%) while configuration memory requirements are also reduced (up to 60%). Simulation results show that it can be used either in low power reconfigurable applications (up to 90% power reduction is possible) or for speeds comparable to those of CMOS-LUTs.

1 Introduction

The necessary structuring of the projected tens of billions of elementary, unreliable, nanometric devices to achieve the computing capacities necessary for future software applications will lead to the emergence of reconfigurable platforms as the principal computing fabric before the end of the next decade. The reconfigurable approach allows volume manufacturing and reduces the impact of the evolution of mask costs, projected to move above the $10M mark in 2010 [1]; can efficiently cover a broad range of applications while exceeding performance levels of programmable systems; and couples naturally to fault-tolerant design techniques for robust architectures.

Gate-level, or fine-grain, reconfigurability enables benefits in terms of silicon real estate, since it makes it possible to reduce the number of logic cells necessary to implement a given switching function (in comparison to the implementation of these functions with conventional logic). It also makes it possible to simplify the interconnect network, reducing area and the parasitic capacitances due to routing. It can thus be expected to reduce dynamic power dissipation and improve speed. These two performance metrics are often the weak points of the various types of reconfigurable circuits (FPGA, coarse-grain reconfigurable systems) compared to "full-custom" solutions.

While CMOS device scaling has led to increasingly better performances, higher packing density and lower cost per device, short channel effects have become difficult to control [1]. To pursue performance improvement in conventional planar bulk CMOS devices, channel doping will have to be increased with scaling to almost impossibly high values, which will cause a reduction in mobility and high leakage current (and static power dissipation) due to band-to-band tunneling between the drain

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 97–113, 2010.

and the bulk. Also, the total number of dopants in the channel for very small MOSFETs is increasingly low, resulting in extremely high fluctuataions in the number of dopants, and hence unacceptably large statistical variation of the threshold voltage. These difficulties, especially power dissipation and variability, have introduced the need for new device architectures and the emergence of structures with improved and more flexible electrostatic control of the channel.

Ultra-thin body, fully depleted (UTB FD) SOI MOSFETs represent one solution where channel doping is relatively low; in these devices, the threshold voltage can be set by adjusting the work function of the gate electrode, rather than by doping the channel as in planar bulk MOSFETs. Metal gate electrodes with work functions tunable within a few hundred meV above and below midgap should be used to set the threshold voltage to the desired values. Single gate SOI MOSFETs are projected for 2010 for high-performance logic. Multiple-gate, ultra-thin body, fully depleted MOSFETs, in both planar (DG MOSFET) and vertical dispositions (FinFET), are both more complex and more scalable, and are projected to be implemented in 2011 for high-performance logic.

The Double-Gate (DG) MOSFET on FD SOI technology is known as a promising advanced device which, thanks to the double-gate structure, is expected to overcome drawbacks of the conventional MOSFET in nanometric technologies. Compared to its counterpart single gate FD SOI MOSFET, the DG SOI MOSFET reduces the short channel effects and improves the sub-threshold slope and drive current [2][3][4] while benefiting from the advantages of FD SOI technology. These include reduced latch-up, reduced parasitic source and drain capacitances, smaller sensitivity to temperature variation and reduced leakage current [5]. The double-gate structure allows independent switching of the gates or dynamic adjusting of the threshold voltage. In more conventional (single gate) device structures, a dynamic Vth variation can be achieved by varying the body or back gate voltage for bulk and fully depleted SOI devices respectively. However, since all devices share the same well or substrate (for bulk devices) or the same back-gate (for single gate FD SOI devices), dynamic Vth variation for individual transistors is either highly impractical or impossible to achieve.

Partially depleted SOI devices are better suited to dynamic adjustment of the threshold voltage since the body is isolated and can thus be contacted to a separate bias potential per device. However, double gate SOI MOSFETs offer the same flexibility as PD SOI single gate MOSFETs with regard to this dynamic Vth adjustment. Moreover, it has been demonstrated that an independent control of front and back gates can be exploited to reduce both dynamic power and sensing delay in a sense amplifier design [6]. It can also be used to merge parallel transistors [7] and thus reduce dynamic power through the reduction of parasitic capacitance, as well as static power. Furthermore, designers can choose between different types (symmetric or asymmetric [3]) of DG MOSFET devices, in order to make the threshold voltage tailored to the requirements of circuit operation. This makes it well-suited for some leakage power management circuit techniques commonly used in digital circuit design. Furthermore, it allows consideration of new design approaches. However, all these advantages come at the expense of a higher switching gate capacitance (in the

case of the connected gates scheme) and a die area penalty compared to the single gate device.

Such devices enable designers to achieve improved density, power and speed metrics [3][4][7][8] in logic cells. Further, with four accessible terminals, these devices also offer the opportunity to design novel building blocks exploiting the additional terminal for reconfigurability purposes [9]. In this work, we cover the principles of the design of *m*-input DG MOSFET reconfigurable cells in both dynamic- and static-logic forms. These principles are applied to the design of 2-input cells, and the simulated results are then compared to those of conventional CMOS LUT techniques.

2 Generic *m*-Input Reconfigurable Cell

The main tenet of our approach lies in the construction of cells containing n- and p-networks for which the data-switching properties can be modified with control voltages applied to the back gates of DGMOS transistors. This dynamically modifies the threshold voltage of individual devices. The behavior of an n-type DGMOS device according to the applied back gate voltage can be roughly described as follows:

- when a sufficiently positive voltage V^+ is applied, the device is always on (regardless of front gate voltage). In other terms, the threshold voltage is lowered to below the lowest voltage applied to the front gate (e.g. logic "0").
- when 0V is applied, normal operation is achieved, i.e. device switching depends on the front gate voltage.
- when a sufficiently negative voltage V^- is applied, the device is always off (regardless of front gate voltage). In other terms, the threshold voltage is raised to above the highest voltage applied to the front gate (e.g. logic "1").

This behavior is shown in simulations for both n-type and p-type devices in Figure 1. These simulations are for individual devices with W/L=0.25µm/0.13µm with 1.2nm front- and back-gate oxide thicknesses, and use a double-gate FD-SOI/CMOS technology model implemented in Verilog-A. This explicit analytical charge-based compact model of independent double gate MOSFET devices is based on Poisson and field continuity equations and demonstrates <2% drain current value error with respect to Atlas simulations over all regions of operation and for both long and short channel devices. It has also been extensively validated against experimental device characteristics. Further details of the model are outside the scope of this work and can be found in [10].

For certain branches it is necessary to use asymmetric devices to achieve a dominant influence of the control voltage on the transistor behavior. Previous work in this field [9] has been inconclusive since only symmetric devices were used, resulting in a circuit structure with limited functionality and unsatisfactory performance. Asymmetric devices provide additional degrees of freedom and can be achieved with different oxide thicknesses or different gate workfunctions (i.e. with different gate

metals). Our work is based on the former approach, with front-gate oxide thickness T_{oxf}=2.5nm or 5nm (depending on the degree of asymmetric control required – increasing the oxide thickness also increases leakage current) and back-gate oxide thickness T_{oxb}=1.2nm. Simulated I_{ds}-V_{gs} characteristics show (for an N-type device of the previously cited dimensions with T_{oxf}=2.5nm, V_{bgn}=0V) a slight (15%) increase in I_{off}, and a more significant (45%) decrease in I_{on}. Again, the model used has been extensively corroborated against technology simulations.

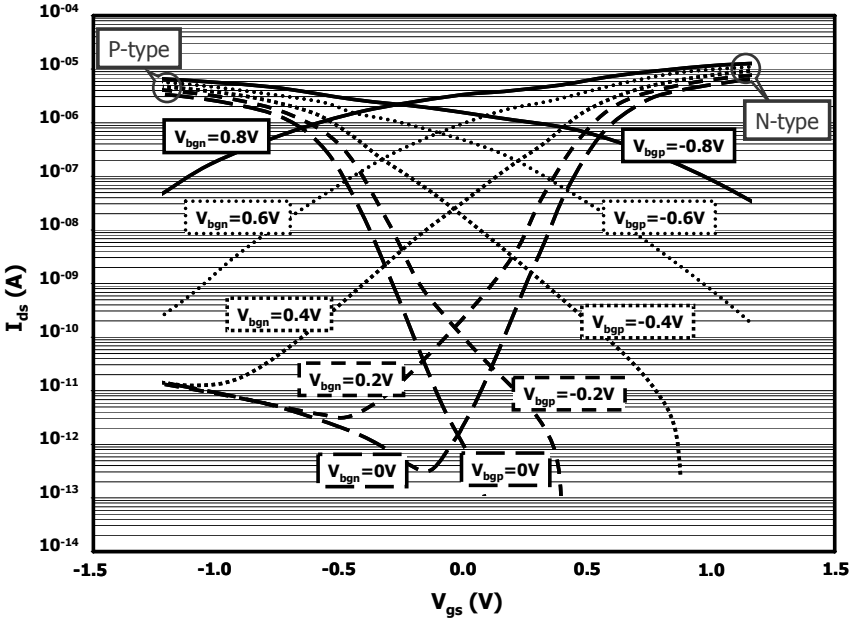


Fig. 1. Log I_{ds}-V_{gs} plot of n-type and p-type DGMOS model with independent gate control

2.1 Dynamic-Logic Reconfigurable Cell *DG-DLRC*

The general principle for building an *m*-bit dynamic-logic reconfigurable cell (DG-DLRC) is shown in Figure 2. This novel structure uses n-type dynamic logic, where a switching network composed of n-type devices is sandwiched between clocked precharge and evaluation switches (M_{pc} and M_{ev} respectively), and allows conditional discharge of the output node F during the evaluation phase. The n-type device network that realizes the logic functions is composed of:

- one branch containing a stack of *m* symmetric DG MOSFETs
- *(m-1)* branches each containing a single asymmetric DG MOSFET (with $T_{oxf} > T_{oxb}$).

The front gates of these devices are controlled by the *m* logic inputs. *2(m-1)* control signals are applied to the back gates to configure the logic function dynamically.

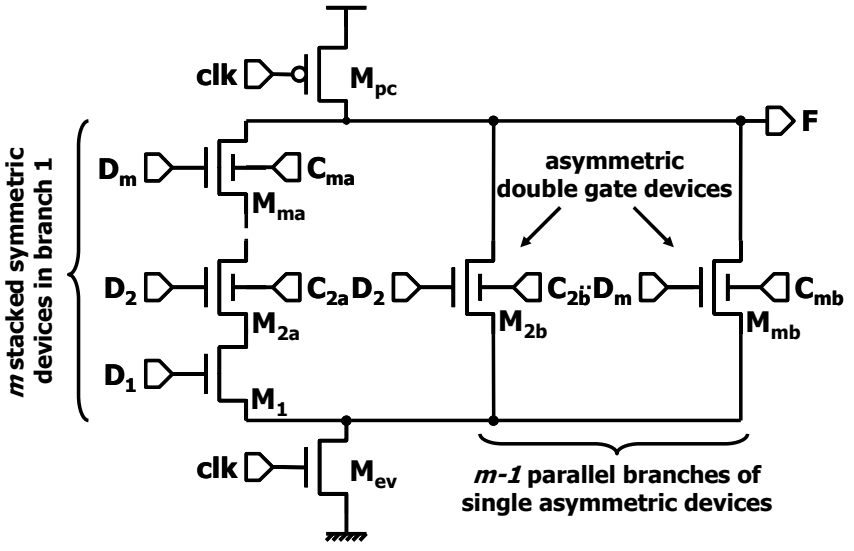


Fig. 2. Generic DGMOS dynamic-logic reconfigurable cell (DG-DLRC)

Dynamic logic is generally more compact (in terms of device count) than static complementary logic when implementing complex logic functions, since it does not require a complementary p-device network and thus demonstrates reduced total parasitic capacitance and silicon area, in particular for cells with a large number of inputs. However, this approach requires clock lines and imposes more stringent constraints on device off currents, since leakage leads to a deterioration of calculated results.

A simple set of configuration codes (i.e. back-gate voltage sets) can be applied to configure an *m*-input reconfigurable cell to a particular logic function from those available (NAND, NOR, INV). After having identified the type of function, the presence of each input D_x ($\forall x \in \{2,m\}$) is evaluated, enabling the corresponding configuration codes $\{C_{xa},C_{xb}\}$ to be extracted from Table 1.

Table 1. General configuration code table for *m*-input DG-DLRC

Function	D_x present in expression		D_x absent from expression	
	C_{xa}	C_{xb}	C_{xa}	C_{xb}
NAND	0	V^-	V^+	V^-
NOR	V^- [1]	0	V^- [1]	V^-
INV	V^-	0	V^- [1]	V^-

For the **NAND**-configuration, 0V is applied to C_{xa} such that transistor M_{xa} operates as a normal n-transistor (i.e. on or off for D_x equal to logic "1" or "0" respectively) when D_x is present in the expression. If D_x is not in the expression, then transistor M_{xa} is turned completely on with $C_{xa}=V^+$. Independently of the presence of D_x in the expression, V^- is applied to C_{xb} to turn transistor M_{xb} off (regardless of the logic value

[1] Unless D1 is present in the expression (in this case, $C_{xa}=V^+$).

of D_x). An asymmetric device must be used for M_{xb} to increase the front-gate threshold voltage and thus enable complete turn-off in the NAND-configuration. The value of V^- must be chosen with respect to the gate breakdown voltage limitation. The resulting effective threshold voltage is chosen such that the functionality of the NAND-configuration is met without affecting that of the NOR or INV configurations.

In the **NOR**-configuration and when D_1 is present in the expression, V^+ is applied to C_{xa} in order to significantly decrease the threshold voltage of M_{xa} and turn it completely on, regardless of the logic state of the signal at the front gate. If D_1 is not in the expression, then transistor M_{xa} must be turned off with $C_{xa}=V^-$. If D_x is in the expression, then 0V is applied to C_{xb} for normal operation of transistor M_{xb}, otherwise M_{xb} is turned off with $C_{xb}=V^-$.

In the **INV**-configurations, a single branch is activated to switch with D_1 only (by turning M_{xa} completely on with $C_{xa}=V^+$, and turning M_{xb} completely off with $C_{xb}=V^-$) or with D_x only (by turning M_{xa} completely off with $C_{xa}=V^-$, and selecting normal operation with M_{xb} by applying $C_{xb}=0$). For the latter operation, it is also possible to use $C_{xa}=0V$ to include M_{xa} in switching with D_x (all other control voltages in this branch should then be set to V^+), but this results in non-deterministic timing behavior (since the drive strength depends on the state of D_1).

2.2 Static-Logic Reconfigurable Cell DG-SLRC

Static logic styles generally feature better noise immunity than dynamic logic, and thus are well-suited to applications that require resistance to harsh environments. The novel *m*-bit static-logic reconfigurable cell structure (DG-SLRC) is shown in Figure 3. In

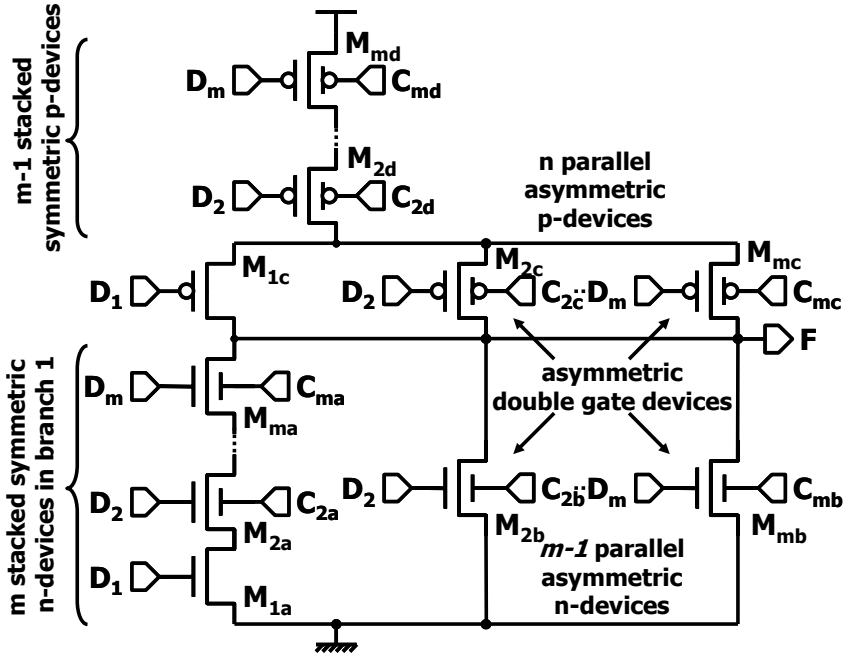


Fig. 3. Generic DGMOS static-logic reconfigurable cell (DG-SLRC)

addition to the n-device branch described in the previous section, DG-SLRC requires a p-device branch composed of:

- one network containing *m* parallel asymmetric DG MOSFETs (with $T_{oxf} > T_{oxb}$)
- a stack of *(m-1)* symmetric DG MOSFETs.

As before, the front gates of these devices are controlled by the *m* logic inputs. *4(m-1)* control signals are applied to the back gates in order to configure the logic function at the output dynamically. The configuration codes $\{C_{xa},C_{xb}\}$ to be extracted for the various functions are given in Table 2.

Table 2. General configuration code table for *m*-input DG-SLRC

Function	D_x present in expression				D_x absent from expression			
	C_{xa}	C_{xb}	C_{xc}	C_{xd}	C_{xa}	C_{xb}	C_{xc}	C_{xd}
NAND	0	V^-	V^+	0	V^+	V^-	V^+	0
NOR	V^- [2]	0	V^+	V^+	V^- [2]	V^-	0 [2]	0
INV	V^-	0	0	V^+	V^- [2]	V^-	V^+	0

In the **NAND**-configuration and when D_x is present in the expression, 0V is applied to C_{xa} for normal operation of transistor M_{xa}, while V^- is applied to C_{xb}. Since M_{xb} is asymmetric, this device is turned completely off, regardless of the value of D_x. V^+ is applied to C_{xc} for normal operation of M_{xc}, while 0V is applied to C_{xd} in order to turn M_{xd} on regardless of the value of D_x. If D_x is not present in the expression, then M_{xa} is turned completely on ($C_{xa}=V^+$) and M_{xb} completely off ($C_{xa}=V^-$). Here again, 0V is applied to C_{xd} in order to turn M_{xd} on regardless of the value of D_x. V^+ is applied to C_{xc}; this voltage does not ensure that the p-type DG MOSFET M_{xc} is switched completely off, although the off-state is nearly reached due to the asymmetric structure of M_{xc} with high T_{oxf}. However, this situation means that power performance is likely to be poor, and the output logic "0" level is degraded (simulation results in the 2-input case show 60mV). To avoid this, the strength (i.e. W/L) of M_{xc} must be reduced.

In the **NOR**-configuration and when D_1 is present in the expression, V^+ is applied to C_{xa} in order to ensure M_{xa} is always on, while 0V is applied to C_{xb} for normal operation of transistor M_{xb} (if D_x is in the expression; otherwise M_{xb} is turned off with $C_{xb}=V^-$). If D_1 is not in the expression, then transistor M_{xa} must be turned off and V^- is applied to C_{xa}. In the p-device network, V^+ is applied to C_{xc} to approach the off-state of M_{xc} if D_1 is in the expression (C_{xc}=0V and M_{xc} is completely on if not) and V^+ is applied to C_{xd} for normal operation of M_{xd} (if D_x is in the expression; otherwise M_{xd} is turned on with C_{xd}=0V).

In the **INV**-configurations, a single branch is activated to switch with D_1 only or with D_x only. In the first case, in the n-device network M_{xa} is turned completely on with $C_{xa}=V^+$, and M_{xb} completely off with $C_{xb}=V^-$; while in the p-device network we apply C_{xd}=0V to turn M_{xd} completely on, and $C_{xc}=V^+$ to approach the off-state for M_{xc}. In the second case, we turn M_{xa} completely off with $C_{xa}=V^-$, and select normal operation for M_{xb} by applying C_{xb}=0V in the n-device network; while in the p-device network we set $C_{xd}=V^+$ to achieve D_x-dependent switching, and C_{xc}=0V to turn M_{xc} completely on.

[2] Unless D_1 is present in the expression (in this case, $C_{xa}=V^+$).

3 Tests with Two-Input DG-XLRC

In this section we consider the implementation, in both dynamic- and static-logic forms, of the previously presented reconfigurable cell in its 2-input form. The design of the cells was based on the double-gate FD-SOI/CMOS technology model mentioned earlier, and simulations were performed throughout with a calculation rate of 50Mbit/s (i.e. data period=20ns), using signal rise and fall times of 40ps. The load capacitance considered was 5fF.

3.1 Two-Input DG-DLRC

Figure 4 illustrates the 2-input reconfigurable cell (with logic inputs D_1=A and D_2=B), implemented with DG devices and based on dynamic logic [11]. Transistors M_1, M_4, and M_5 depict symmetric DG devices (i.e. symmetric oxide thicknesses and workfunctions for the front and back gates) with connected front and back gates. Transistor M_2 depicts a symmetric DG device, while transistor M_3 depicts an asymmetric DG device. Both M_2 and M_3 use independent gate control. For this mixed (symmetric and asymmetric devices) cell denoted DG-DLRC_mixed, asymmetric biasing is used with $\{V^+,V^-\}=\{1.0V,-0.5V\}$.

Another variant of the cell, DG-DLRC_asymm, uses only asymmetric DG MOSFETs. The use of the same device type in the cell can be more convenient since it eases circuit fabrication. In this case, V_{dd}=0.6V to achieve symmetric gate biasing on C_{2a} and C_{2b} where $\{V^+,V^-\}=\{+0.6V,-0.6V\}$ while using a maximum absolute gate-source and gate-drain voltage value of 1.2V .

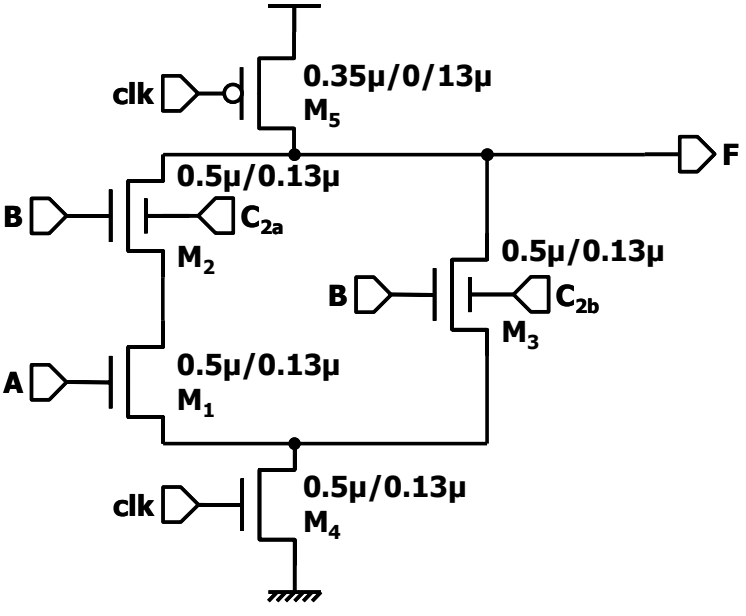


Fig. 4. Two-input DG-DLRC

Table 3 shows the logic state of the cell output (node F) with respect to the applied back gate voltages on the C_{2a} and C_{2b} terminals, as a 2-input implementation of Table 1. As can be observed from Table 3, the cell can implement the NAND, NOR, INV and unconditional '1' and '0' logic functions.

Table 3. Truth table of two-input DG-DLRC

C_{2a}	C_{2b}	Function
0	V^-	NAND(A,B)
V^+	0	NOR(A,B)
V^+	V^-	INV(A)
{0, V^-}	0	INV(B)
V^-	V^-	1
X	V^+	0

This cell has been evaluated using the simulation conditions described previously. The simulation waveforms for DG-DLRC_mixed are shown in Figure 5, where the switching from one configuration to another, as can be observed at the output F,

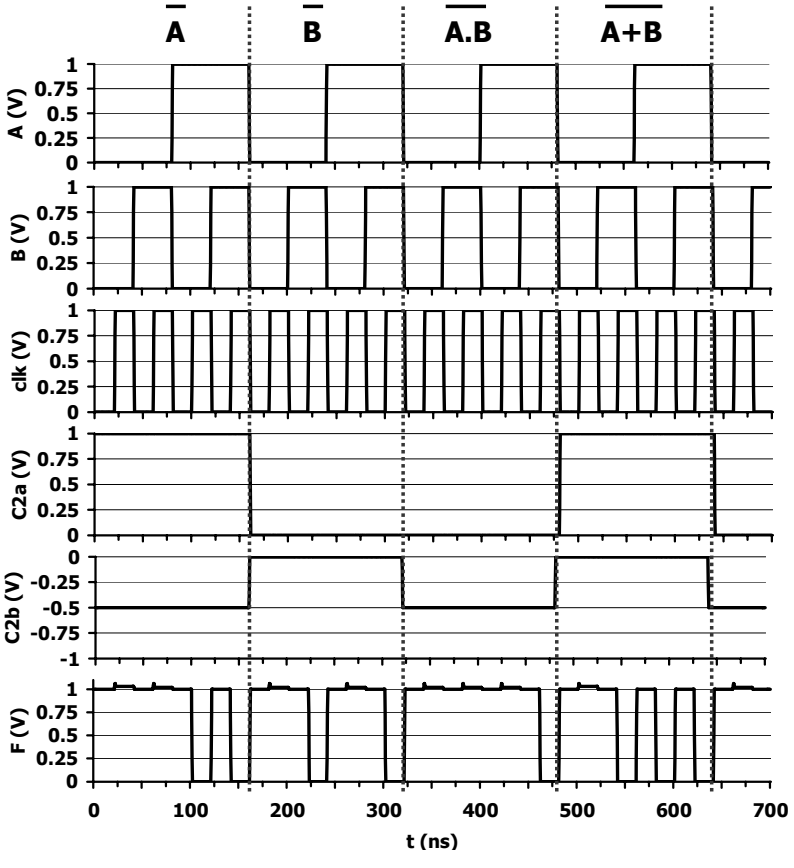


Fig. 5. Simulated configuration of the two-input DG-DLRC to NAND, NOR and INV functions

is obtained through the dynamic configuration of signals C_{2a} and C_{2b}. Similar waveforms are obtained with DG-DLRC_asymm.

The power and delay performance characteristics of both variants are summarized in Table 4 for each function configuration. The reconfigurable cell performance (average power and worst case delay) depends not only on the activity factor, the total switched capacitance and device number lying on the critical path, but also on the different back gate biasing used in each configuration. These factors affect, differently from one configuration to another, the total drive current, sub-threshold and gate leakages, and consequently the total power and the worst case delay.

Table 4. Simulated performance figures for the two-input DG-DLRC

Function	DG-DLRC_mixed			DG-DLRC_asymm		
	Av. power (nW)	Worst-case delay (ps)	PDP (fJ)	Av. power (nW)	Worst-case delay (ps)	PDP (fJ)
NAND(A,B)	256	140.6	0.04	33.2	540	0.02
NOR(A,B)	476.7	740.6	0.35	96.2	2590	0.25
INV(A)	361	140.6	0.05	71.9	2590	0.19
INV(B)	476.3	667	0.32	46.8	2440	0.11

3.2 Two-Input DG-SLRC

The 2-input static reconfigurable cell (with logic inputs D_1=A and D_2=B) is shown in Figure 6. The truth table of the cell is shown in Table 5, as a 2-input implementation

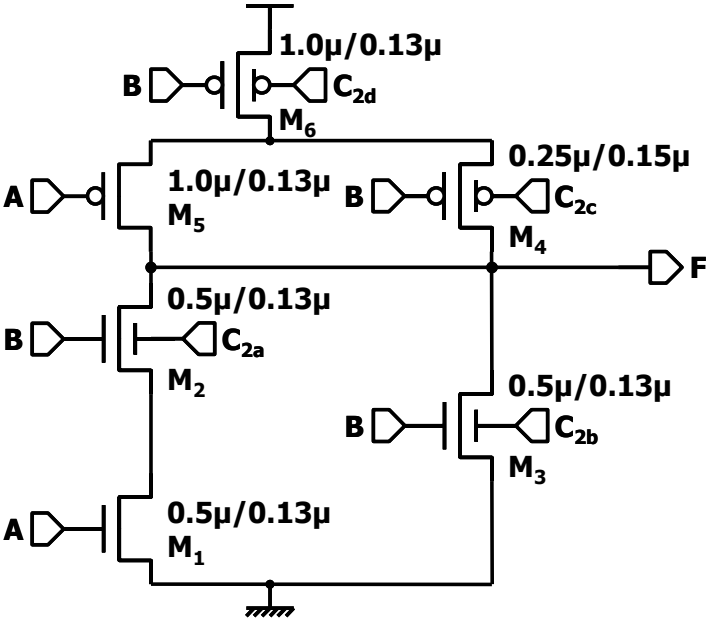


Fig. 6. Two-input DG-SLRC

Table 5. Truth table of two-input DG-SLRC

C_{2a}	C_{2b}	C_{2c}	C_{2d}	F
0	V^-	V^+	0	NAND(A,B)
V^+	0	V^+	V^+	NOR(A,B)
V^+	V^-	V^+	0	INV(A)
{0, V^-}	0	0	V^+	INV(B)
V^-	V^-	0	0	1
X	V^+	V^+	V^+	0

of Table 2. As with DG-DLRC, this cell implements the NAND, NOR and INV functions. Each logic function is obtained by applying the relevant configuration codes in terms of back gate biases $C_{2a\text{-}d}$, as shown in Table 5.

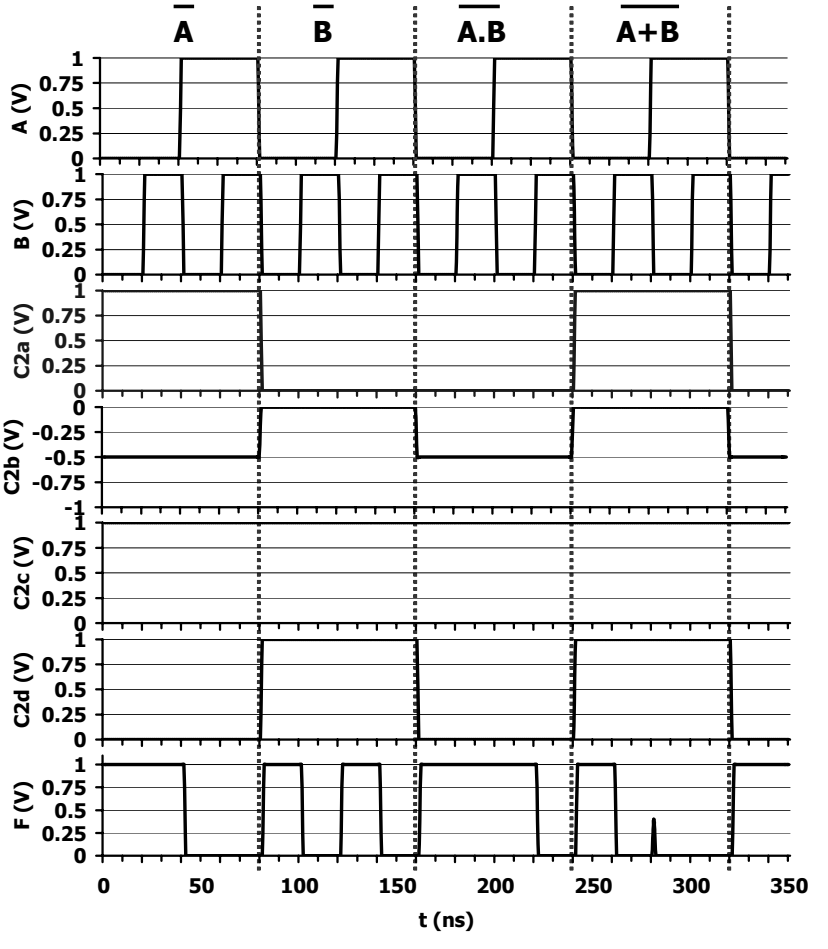


Fig. 7. Simulated configuration of the two-input DG-SLRC to NAND, NOR and INV functions

As with DG-DLRC, the static variant can also be implemented using all asymmetric DG devices (as DG-SLRC_asymm). In this case V_{dd}=0.6V and symmetric biasing is used with $\{V^+,V^-\}=\{+0.6V,-0.6V\}$. Correct functionality is observed with this cell due to:

- the reduced V_{dd}/V_{th} ratio (V_{th} ≈0.4V with V_{bg}=0V), thus allowing cut-off of transistor M_3 when V^- is applied to its back gate
- the small W/L ratio for transistor M_4 combined with the small V_{dd}/V_{th} ratio.

The simulation results obtained from both DG-SLRC_mixed and DG-SLRC_asymm are shown in Figure 7. The power and delay performance characteristics of both variants are summarized in Table 6 for each function configuration.

Table 6. Simulated performance figures for the two-input DG-SLRC

Function	DG-SLRC_mixed			DG-SLRC_asymm		
	Av. power (nW)	Worst-case delay (ps)	PDP (fJ)	Av. power (nW)	Worst-case delay (ps)	PDP (fJ)
NAND(A,B)	1189	590.5	0.70	126	1620	0.20
NOR(A,B)	182	309.4	0.06	197	660	0.13
INV(A)	2800	111.5	0.31	331	660	0.22
INV(B)	397	295.3	0.12	102	1740	0.18

4 Comparison to Conventional LUT and Discussion

We have carried out experiments to evaluate the performance gain of DG-xLRC with respect to conventional solutions. While this technique can be considered to open up many possibilities for new system-level programming paradigms, it is also possible to

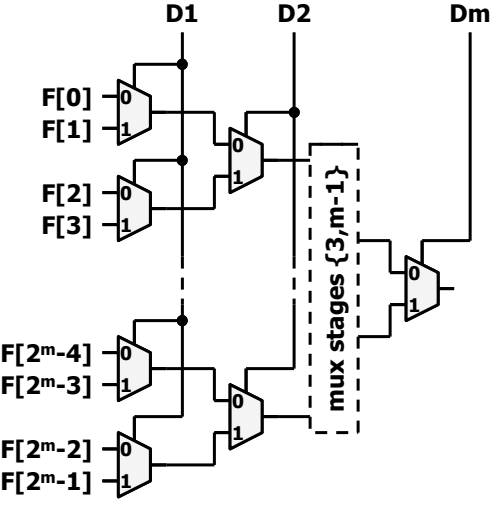


Fig. 8. Reference *m*-bit MUX-based look-up table

consider the cell family to be a set of incomplete look-up tables (LUTs) [12] and make a direct comparison to conventional *m*-bit MUX-based LUTs (the reference structure of which is shown in Figure 8). It should be noted that the aim of this section is to provide an objective comparison at the circuit level using individual characteristics, rather than a system-level comparison where the impact of cell characteristics is not so clear.

4.1 Gate Area and Memory Requirements

For 2-, 3- and 4-input LUTs and DGMOS-based reconfigurable cells, we evaluated the gate area (i.e. channel dimensions only), and the required number of memory cells to retain the configuration codes (Figure 9).

The gate area results reflect the exponential and linear growth of transistor count in LUTs and DG-xLRCs respectively. For LUTs, the transistor count grows with $N_{mux}*(2m-1)$ (where m represents the number of inputs and N_{mux} represents the 2-1 MUX transistor count, usually equal to 12), while transistor count grows with $3+2(m-1)$ and $2+4(m-1)$ for DG-DLRC and DG-SLRC respectively. Total area comparisons incorporate extra interconnect requirements (an extra -V power line, precharge and evaluation lines for the dynamic cell, double inputs) with some reduction in configuration lines for certain variants. The complete layouts (including all routing but excluding configuration memory cells and precharge/evaluate logic buffers) for the 2-input dynamic and static cells show area reduction factors of 6.5 and 4.7 respectively as compared to the CMOS-LUT, instead of 7.8 and 4.9 considering gate area only. The precharge/evaluate logic and signal distribution tree was not included in the analysis.

The number of memory cells required has an impact not only on auxiliary hardware requirements, but also on configuration time. For LUTs, this increases with $2m$, while for DG-DLRC and DG-SLRC it is equal to $2(m-1)$ and $4(m-1)$ respectively.

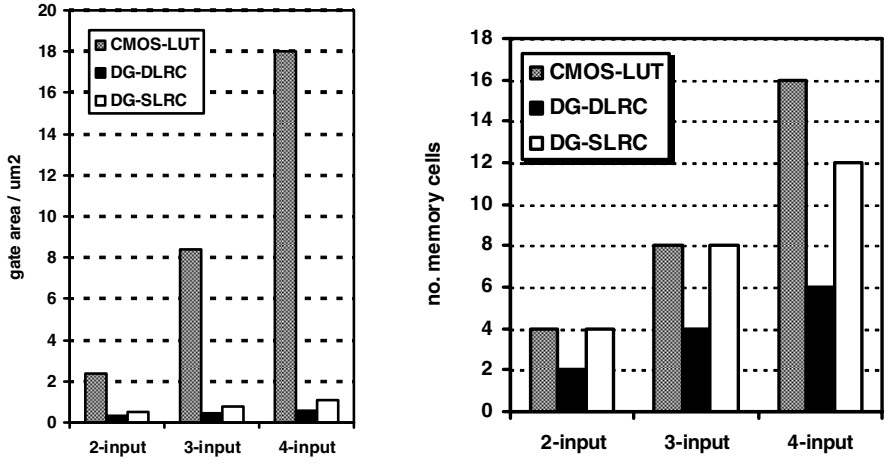


Fig. 9. Comparison of m-input DG-xLRC and CMOS-LUT characteristics (a) gate area (b) no. memory cells

It should of course however be borne in mind that while a LUT potentially offers configurations, DG-xLRC offers rather less. The number of available functions corresponding to *m*-input cells is plotted in Figure 10 and compared to the figures for CMOS-LUTs for values of *m* ranging from 2 to 6. In practice, the number of inputs that the reconfigurable cells can reasonably handle is 4. Beyond this figure, the number of series devices in a stack becomes too high.

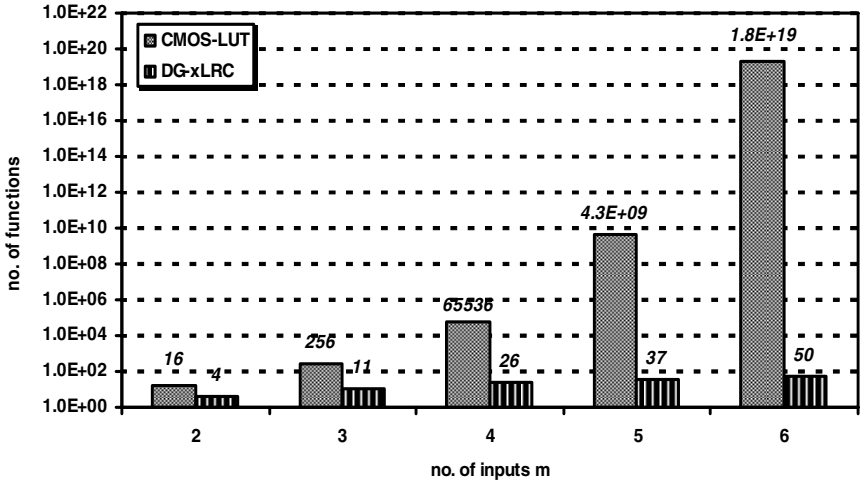


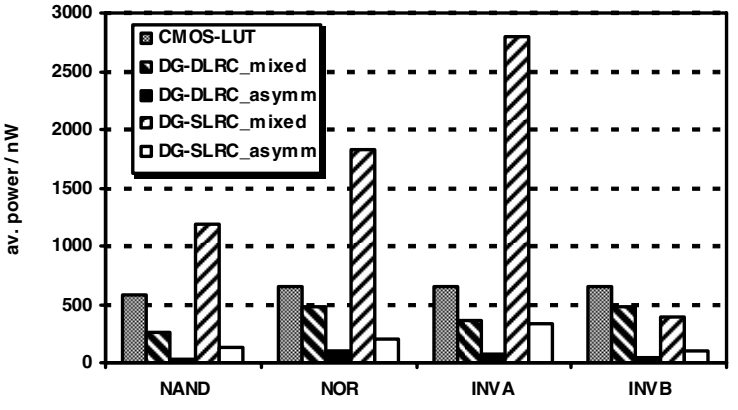
Fig. 10. Number of available functions for m-input reconfigurable cells

It is clear from this figure that a direct transposition of this cell as a LUT in conventional configurable logic blocks will result in limited flexibility. For this reason we believe that further work must be carried out to explore new programming paradigms to benefit from the cell performance (and in particular its reduced configuration memory requirements for easier dynamic reconfiguration) at system level.

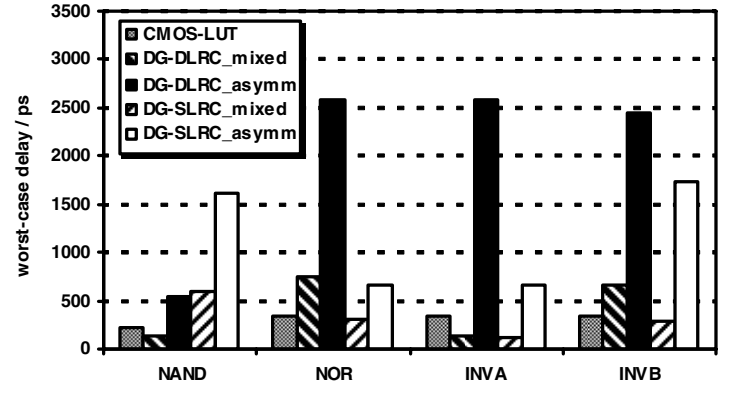
4.2 Average Power and Worst-Case Delay

We have also carried out detailed simulations of the 2-input solutions to compare average power and worst-case delay performance metrics. To this end we have simulated the LUTs in a 65nm CMOS technology. The choice of this reference technology was based on a comparison of oxide thicknesses, doping levels, mobility parameters and gate metal types. However, since the reconfigurable cells use 130nm gate lengths (for reasons of model validity and lack of technological maturity) the 65nm CMOS standard cell transistor dimensions were scaled to match the gate lengths and achieve comparable parasitic capacitance values and a fair basis for comparison.

Identical simulation conditions were used, i.e. 50Mbit/s calculation rate, load capacitance C_L=5fF, 40ps rise and fall times on inputs. Comparisons of the CMOS-based LUT figures were carried out for the four operational functions with respect to both DG-DLRC and DG-SLRC, in mixed and all-asymmetric variants (Figure 11).



(a)



(b)

Fig. 11. Comparison of two-input DG-xLRC and CMOS-LUT characteristics (a) average power (b) worst-case delay

These figures clearly show that, apart from the mixed implementation of DG-SLRC, total power performance is systematically better with DG-xLRC solutions, in one case achieving an average of over 90% reduction in power over the four function configurations. For static power, it should be noted that all configurations of both static and dynamic logic cells (except the unconditional '0' configuration) bias N-type and P-type DGMOS back gates to 0V / V^- and V^+ respectively (leading to either the same or lower I_{off} as with connected gates), except when an unconditional short is required – in which case the off current of the branch is defined by another transistor in the equivalent configuration to that of a connected gates transistor. This means that the low I_{off} values of DGMOS transistors are exploited in the proposed cells (our simulations show I_{off} = 0.7pA for an N-type DGMOS of 0.5µm/0.13µm with t_{oxf}=2.5nm and t_{oxb}=1.2nm) – however the I_{on} values are lower than a connected-gates equivalent transistor since the back gate is set to 0V or V^+ for N- and P-type

conducting transistors respectively. The peak current value for a transistor with the previously cited characteristics (i.e. for $V_{fg}=V^+$ and $V_{bg}=0V$) is around 50µA. This has an impact on the cell drive current, and the consequences are visible in Figure 11 in terms of the mediocre worst-case delay comparison. The best achievement is an overall 20-30% delay penalty for the mixed solutions. Additional technology and circuit optimization should enable some tradeoff between power and speed through the improvement of cell drive current (by increasing device width) – but clearly the power, gate area and I_{off} will all deteriorate by such a strategy.

Overall recommendations are that (i) the all-asymmetric device reconfigurable logic cells using $\{V^+,V^-\}=\{0.6V,-0.6V\}$ are best-suited to low power reconfigurable circuits operating with moderate speed, while (ii) the mixed-device reconfigurable cells using $\{V^+,V^-\}=\{1.0V,-0.5V\}$ can operate at comparable speeds to CMOS-LUTs but only the dynamic-logic variant shows benefits in terms of power.

5 Conclusion

In this paper, we have presented a new style of reconfigurable cell dedicated to programmable logic applications and based on the DG MOSFET device, particularly exploiting those with asymmetric oxide thicknesses for the front and back gates and independently controlled gates. Significant gate area reductions are possible compared to conventional CMOS LUT techniques (between 80-95%) while configuration memory requirements are also reduced (up to 60%). The 2-input reconfigurable cell used as a benchmark was implemented in both static and dynamic logic styles. Simulation results in DG FD SOI/CMOS technology of the proposed cell have shown that it can be used either as an all-asymmetric device variant with low V_{dd} (0.6V) in low power reconfigurable applications (up to 90% power reduction is possible) or as a mixed-device variant with a higher V_{dd} (1V) to achieve comparable speeds to CMOS-LUTs (20-30% penalty).

References

1. International Technology Roadmap for Semiconductors, 2007 edn.
2. Fossum, J.-G., Kim, K., Chong, Y.: Extremely Scaled Double-Gate CMOS Performance Projections, Including GIDL-Controled Off-State Current. IEEE Trans. on Electron Devices 46(11), 2195–2199 (1999)
3. Roy, K., Mahmoodi, H., Mukhopadhyay, S., Ananthan, H., Bansal, A., Cakici, T.: Double-Gate SOI Devices for Low-Power and High-Performance Applications. In: International Conference on Computer Aided Design (2005)
4. Dao, T.: Advanced double-gate fully-depleted silicon-on-insulator (DG-FDSOI) device and device impact on circuit design & power management. In: Proc. IEEE International Conference on Integrated Circuit Design and Technology, pp. 99–103 (2004)
5. Colinge, J.-P.: Silicon-on-Insulator Technology – Materials to VLSI, 3rd edn. Kluwer Academic Publishers, Dordrecht (2004)
6. Mukhopadhyay, S., Mahmoodi, H., Roy, K.: Design of High Performance Sense Amplifier Using Independent Gate Control in sub-50nm Double-Gate MOSFET. In: Proc. IEEE ISQUED 2005 (2005)

7. Chiang, M.-H., Kim, K., Tretz, C., Chuang, C.-T.: Novel High-Density Low-Power High-Performance Double-Gate Logic Techniques. In: Proc. IEEE Int. SOI Conference (2004)
8. Hassoune, I., O'Connor, I., Navarro, D.: On the performance of Double-Gate MOSFET circuit applications. In: Proc. IEEE NEWCAS (2007)
9. Beckett, P.: A fine-grained reconfigurable logic array based on double gate transistors. In: Proc. IEEE International Conference on Field-Programmable Technology (FPT), pp. 260–267 (2002)
10. Reyboz, M., et al.: Explicit short channel compact model of independent double gate MOSFET. In: Proc. Workshop on Compact Modeling (2007)
11. Hassoune, I., O'Connor, I.: Double-Gate MOSFET Based Reconfigurable Cells. Electronics Letters 43(23), 1273–1274 (2007)
12. Hutton, M., Schleicher, J., Lewis, D., Pedersen, B., Yuan, R., Kaptanoglu, S., Baeckler, G., Ratchev, B., Padalia, K., Bourgeault, M., Lee, A., Kim, H., Saini, R.: Improving FPGA Performance and Area Using an Adaptable Logic Module. In: Becker, J., Platzner, M., Vernalde, S. (eds.) FPL 2004. LNCS, vol. 3203, pp. 135–144. Springer, Heidelberg (2004)

Timed Coloured Petri Nets for Performance Evaluation of DSP Applications: The 3GPP LTE Case Study

Laura Frigerio[1], Kellie Marks[2], and Argy Krikelis[2]

[1] Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milano, Italy
laura.frigerio@polimi.it
[2] European Technology Center
Altera Corporation
High Wycombe, United Kingdom
{kmarks,akrikeli}@altera.com

Abstract. In this paper we propose the use of Timed Coloured Petri Nets for the Performance Evaluation of Hardware/Software systems for DSP applications. Complex systems on chip, composed by hardware and software parts, are often required to meet strict timing constraints, both in terms of throughput and latency. However, the verification of the suitability of a system configuration can usually be performed only after the integration of the hardware and software components, when design modifications and optimizations are particularly expensive. This article proposes a framework to evaluate the performance of HW/SW systems in which Timed Coloured Petri Nets can be exploited in the early phases of the design. The framework is tested by modelling the Physical Uplink Shared Channel (PUSCH) bit-rate receiver portion of 3GPP (3rd Generation Partnership Project) LTE (Long Term Evolution) standard, the next generation of 3G wireless systems.

1 Introduction

Complex DSP systems nowadays include heterogeneous hardware and software components, like multithreaded CPUs, hardware accelerators and fast interconnections. Due to the systems complexity and to the time-to-market pressure, IP (Intellectual Property) based methodologies are often used to reduce the development time and enhance module reuse [1].

Embedded systems for DSP applications have strict constraints on performance that designers try to meet by efficiently combine pre-verified IPs and ad-hoc implementations. However, the evaluation of the system performance, in order to verify the system throughput and latency, is usually very difficult due to both the high degree of concurrency and the heterogeneity of modules. The system verification can therefore be completed only in final stages of the development, when the hardware and software modules are integrated into the system. At this stage however, very little flexibility is left for optimizations and problems in meeting the required constraints lead to expensive and time consuming modifications of the systems.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 114–132, 2010.

For this reason this paper presents a framework for the early evaluation of the system performance, that can be used to tune and improve the system design before the actual integration of the components takes place.

Different methods can be used to evaluate the performance of a system and can generally be classified in 1) simulation techniques and 2) formal models. Simulation techniques provide information on the system behaviour by tracing the results obtained when applying stimuli to a system model. Pure simulative approaches using for example the SystemC Library have been applied in [2] and [3]. However, simulation approaches alone, cannot provide information on system properties like the absence of deadlocks or system bottlenecks.

Formal models describe the system in a mathematical form and can provide accurate information on its behaviour. Example of formal models used for performance evaluation are Markov processes [4], Queuing Networks [5] and Timed Petri Nets [6]. In this paper, we consider Timed Petri Nets since they are especially suited for describing HW/SW systems in general and DSP applications in particular. First of all, Petri Nets are an intuitive and powerful way to define concurrent and asynchronous processing, useful to describe HW/SW systems. Moreover, with respect to other methods that consider Stochastic timing models only, Petri Nets allow to consider both Deterministic and Stochastic timing models. In DSP applications, where IP blocks are often used in the design, the availability of Deterministic times allows to build accurate models, since the exact timing required to process input data is often available (e.g. number of clock cycles of an hardware module) and can be considered in the model. Finally, several tools are provided to support both the extraction of analytical properties and the simulation of Petri Nets models.

Petri Nets have been used to model a broad range of applications (refer to [7] for examples of industrial use). They have also been used to model digital hardware (many references can be found in [8]). In the context of SoC, the description of communication infrastructure [9] and the formal verification of the implementation [10] have been considered. In this paper, Timed Coloured Petri Nets (TCPN) are used to evaluate the Performance of Hardware/Software systems in early phases of the system design. The use of Coloured Petri Nets enriches the timing description with high level elements, such as complex data types and hierarchy decomposition.

Differently from other works on HW/SW systems based on Petri Nets (like [11], [12], [13]), Petri Nets are not used to support the system design or partitioning, but to perform a rapid Performance Evaluation at IP-blocks granularity, by seamlessly integrate HW and SW models.

The rest of the paper is organized as follows. The next Section provides the formal definition of the Timed Coloured Petri Nets (TCPN) used thorough the paper. A framework to model a HW/SW system with Petri Nets is then introduced. The following Section describes the 3GPP LTE application, used as a case study to verify the proposed framework. The reference HW/SW platform is then presented followed by the description of the mapping of the LTE application on it, using the Petri Net framework previously introduced. The following Section compares the results obtained by applying the Petri Net model to those obtained by the integration of the hardware and software systems. Finally, last Section concludes the work.

2 Formal Definitions

Coloured Petri Nets are an extension of classical Petri Nets, introduced by C. A. Petri [14]. A Coloured Petri Net is defined [15] as a nine-tuple $CPN=(\Sigma, P, T, A, N, C, G, E, S)$, where:

1. Σ is a set of non-empty types, also called colour sets.
2. P is a finite non-empty set of places.
3. T is a finite non-empty set of transitions
4. A is a finite non-empty set of arcs such that: $P \cap T = P \cap A = T \cap A = \emptyset$
5. N is a node function, defined from A into $P \times T \cup T \times P$, which maps each arc into a tuple where the first element is the source node and the second element is the destination node.
6. C is a colour function, defined from P into Σ, which means that C maps the place p to a colour set;
7. G is a boolean expressions, called guard function, which maps the transition t to the Boolean function needed to be evaluated as "true" to enable the transition;
8. E is an arc expression function. A transition is enabled if there exists, for each input arc, a token in the input place bounded to that arc.
9. S is the initialization function, which specifies the initial state of the Petri Net.

The initial marking M_0 assigns to the place the initial (coloured) Tokens. For a formal definition, refer to [15]. In Timed Coloured Petri Nets, transition occurrences fire in 'real-time' associated with each occurrence of each transition. In this paper we consider deterministic (D-times) nets, where the times are deterministic [6]. Any enabled transition starts its firing in the same instant in which it becomes enabled.

Each firing can be considered as a three phase event; first, the (coloured) tokens are removed from the input places as indicated by the arc functions of the firing occurrence, the second phase is the firing time period, and when it is finished, (coloured) tokens are deposited to output places as indicated by the arc functions of the firing occurrence. If a transition occurrence becomes enabled while it is firing a new independent firing cycle begins.

Formally, a Timed Coloured Petri Nets is a couple *TCPN = (CPN, f)*, where:

1. *CPN* is a coloured Petri Net, $CPN=(\Sigma, P, T, A, N, C, G, E, S)$.
2. f is a firing-time function which assign the firing time to each occurrence colour of each transition of the net, $f: T \times C \to R^+$.

3 Modelling with Petri Nets

A typical DSP application, can be decomposed in a sequence of functions elaborating data. Each function is intended as an operation, or set of operations, to be applied to a data unit. A simple representation is given by the application graph (Figure 1a) where circles identify functions and arcs are used to specify their dependencies. If there are different data going through different paths, an extension of the previous representation is considered using different types of arcs (Figure 1b).

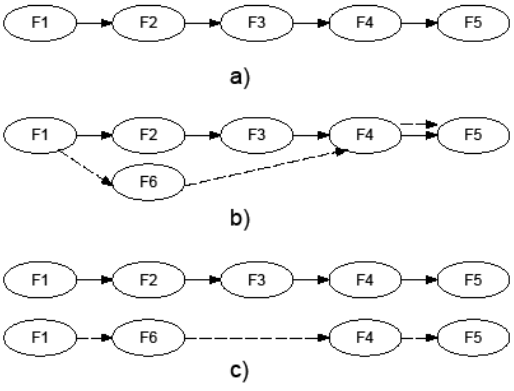


Fig. 1. Application graphs for a sequence of F_1; F_2; … ; F_m functions

However, this situation can be represented as the first one by considering two separated graphs like in Figure 1c. Each function is executed by an executor that can be a processor (if the function is implemented in software) or a hardware module (if the function is implemented in hardware). There can exist multiple instances of the same executor, in order to satisfy the performance requirements. In the following, we indicate as *resource class* (or in short *resource*) a set of identical executors, and as *availability* the number of instances of executors in the same resource class. For example, we can compute a DFT (function) by the use of a DFT hardware module, or a processor executing a DFT software algorithm (resources). Let us consider, for performance reasons, to include in the design two DFT hardware modules in order to be able to process two requests of the DFT function in parallel. In this case the availability of the resource DFT is equal to two.

More formally, given a set of functions F and a set of resources R we define for each function a mapping m on the resource on which it is executed, $m : F \rightarrow R$. The execution of a function f_i on a resource r_j requires a certain amount of time t_{ij} . Values t_{ij} are known if the design process is based on IPs (Intellectual Property) or can be estimated on the basis of previous and similar implementations. In case of variability a timing distribution or an average time value can be considered.

Starting from these definitions, we can generate the Timed Coloured Petri Net modelling the application, as represented in Figure 2 for a simple example. For each function and each resource we introduce respectively two places (an F-Place and an R-Place). We also add a Q-place for each function to represent the queue of data waiting to execute the function. The output transition of the F-Place is annotated with time t_{ij} and the initial marking of each R-Place is defined by its availability. F-Places are connected according to the application graph; if a resource is shared by different functions, a single R-Place is used and appropriate arcs are used to connect different F-Places to the same R-Place.

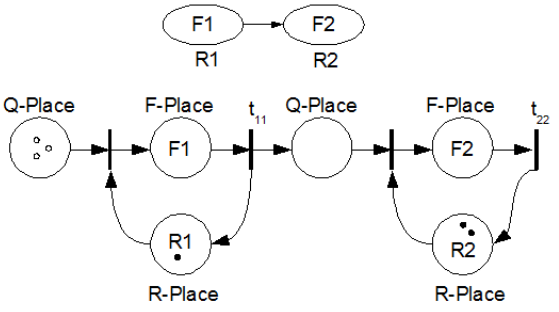


Fig. 2. TCPN generated from an application graph, where the availability of R1 is equal to 1, the availability of R2 is equal to 2. D-tokens in the Q-Place represent three data units waiting for the execution.

Coloured tokens are used to represent both the resource availability (R-Tokens contained in R-Places) and the data units (D-Tokens contained in F-Places and Q-places). The type of D-Tokens is defined according to the parameters needed to determine the system execution. For example, a D-Token can contain an integer value corresponding to the input dimension of the DFT function, that affects the time it takes to execute that function.

In the following we introduce some extension of the presented model.

3.1 Multiple Data Management

If there are data that take different paths, we could exploit the availability of expressions and bindings in CPN to represent this situation. For example, in Figure 3 the D-Token contains two fields, and the value of the first one is used to select the path the token follows in the net.

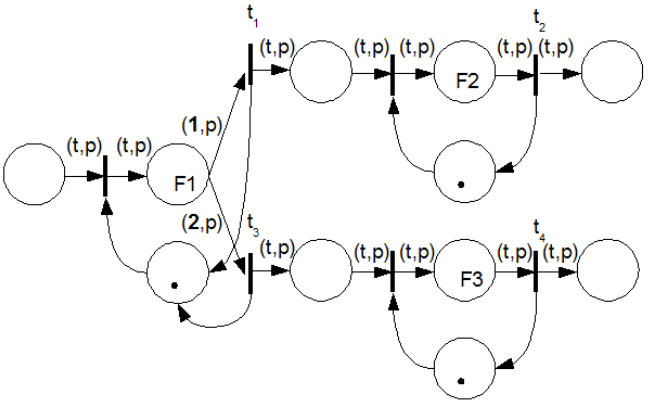


Fig. 3. TCPN for multiple data management

3.2 Pipeline Hardware

A pipelined hardware resource can accept a new data in input every clock cycle (stage time). The execution is completed after the total number of clock cycles required by the function. The execution on a pipelined resource is therefore characterized by two values: a time t_{ij} representing the total time to execute the function and a time s_{ij} representing the stage time. The stage time is equal to one considering the common definition of pipeline, but in a more generic module it is a value greater that one and smaller than t_{ij}.

An example is represented in Figure 4.

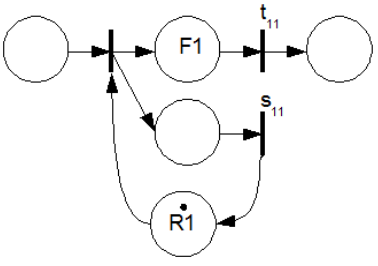


Fig. 4. TCPN to model a pipelined hardware resource

3.3 Data Ordering

If data ordering must be considered, we can exploit the use of FIFOs in CPN [16]. For example, Figure 5 represents two functions executing on the same resource, where the requests for the use of the resource are queued in a FIFO. An additional place, that contains one token of type FIFO, is introduced. Each time an element has to be added, the FIFO token is removed, and replaced with an updated version with the new element at the end. Each time an element has to be extracted, the FIFO-token is removed, and replaced with an updated version without the first element.

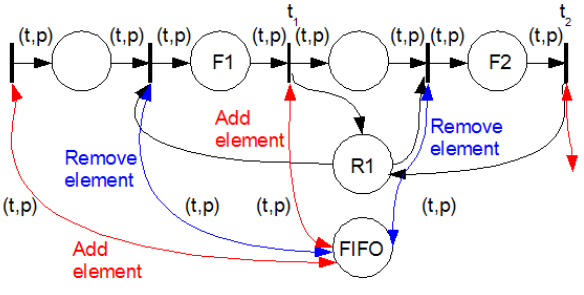


Fig. 5. TCPN to model data ordering

3.4 Design Granularity

The concepts of functions and resources are not necessary restricted to IPs granularity, but can be adapted according to the specific needs when modelling the application. We can model, for example, the internal behaviour of a hardware module or take into consideration the availability of memory space as an additional resource to perform an operation. Example of finer granularity are provided in the rest of the paper.

4 Introduction to the LTE Application

This Section gives an overview of the LTE application, and explains the reasons for needing to model the system using the Petri-net approach. After providing an introduction on the LTE main features, we highlight the LTE criticalities, in particular latency requirements and complexity. The presence of these criticalities constitutes a major obstacle in evaluating a HW/SW solution before its actual implementation. The following Sections show how Petri Nets can help doing this type of evaluation.

4.1 Application Description

3GPP Long Term Evolution (LTE) is next generation of 3G networks aimed at delivering lower latencies, with greater capacity and throughput. It is based on OFDM in the downlink and Single Carrier Frequency Division Multiple Access (SCFDMA) in the uplink.

LTE has evolved from previous 3GPP standards with each evolution providing greater network throughput and lower latencies. The first 3GPP standard, known as release 99, used a radio access technique called Wideband Code Division Multiple Access (WCDMA). It could provide data rates of up to 384kbps in the downlink and 384kbps in the Uplink with round-trip latencies of approximately 150ms. The next two 3GPP releases, High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA), improved the data rate to 14.4Mbps in the Downlink and 5.7Mbps in the Uplink. The round-trip time reduced from approximately 100ms to approximately 50ms respectively. In Release 7 or HSPA+ (High Speed Packet Access +) a new multiple antenna technique known as MIMO (Multiple Input Multiple Output) was introduced, which improved the data rates by using multiple transmit antennas to carry parallel streams of data which are then extracted separately in the receiver. This technique improved the data rates to 28/42Mbps (depending on the number of antennas used) in the downlink and 11Mbps in the uplink.

Release 8 or 3GPP LTE, represents a new generation of wireless techniques by moving away from WCDMA, and employed OFDMA (orthogonal frequency division multiple access) in the Downlink and SC-FDMA in the uplink. Higher data rates (up to 172Mbps in the downlink and 86Mbps in the Uplink) are achieved in LTE through the efficient use of the available spectrum by the use of higher order modulation schemes (up to 64QAM) and MIMO techniques. In addition to this, LTE provides greater flexibility for network operators allowing variable spectrum allocations up to 20MHz, and for the mobile devices, the use of SC-FDMA in the uplink means greater terminal or mobile efficiency and longer battery life.

The uplink bit-rate receiver portion (Physical Uplink Shared Channel - PUSCH) of the LTE system has been chosen to illustrate the use of Petri-net models to evaluate the performance of the system. An overview of the PUSCH may be found in [17], and a technical specification of the PUSCH including the processing steps required may be found in [18].

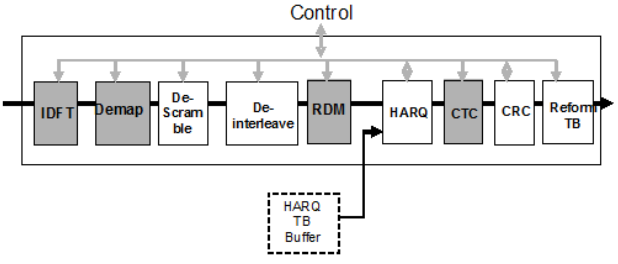


Fig. 6. Functions composing the LTE application

Figure 6 shows basic processing steps of the SC-FDMA uplink bit-rate receiver. The shaded blocks have been considered, for illustration purposes, to present the use of Petri-net models for complex DSP systems.

Table 1. Blocks and parameters of the LTE uplink application

Block	**Function**	**Parameters affecting functionality/latency**	**Ex. Parameter Ranges**
IIDFT	Transform Precoding	Number of resource Blocks	12-1296
Demapper	Demodulation	Modulation scheme	QPSK,16QAM, 64QAM
Rate De-Matcher	Channel coding	Code block size, Coding rate, Filter bits, Redundancy version	40 – 6144 1/9 – 5/6 0 – 64 1 – 4
CTC (Turbo)	Channel coding	Code Block size	40-6144

Blocks are characterized by parameters that can affect not only the functionality but also the latency of the block. Table I gives examples of the different parameters and the range of values the parameters may take.

Users are allocated a number of resource blocks for transmission. The modulation scheme and coding rate determine the number of data-bits transmitted during the slot.

Due to the low latency target for LTE, the uplink SC-FDMA link budget is in the order of 1ms. Meeting this latency target is a key requirement of the system, and requires careful analysis of the latency of the system. The number of possible parameter combinations and the interaction between the latency and throughput of each block in the system makes this a difficult task to perform without a tool to model these interactions.

In the following the main features of each block being modelled are summarized.

4.1.1 IDFT

In the transmitter the OFDM symbol is "orthogonally spread" onto the subcarriers using a Discrete Fourier Transform (DFT). The number of subcarriers that it is spread across represents the number of resource blocks allocated for the users transmission, and is equivalent to the DFT size.

In the receiver the IDFT is used to retrieve the DFT-Spread OFDM symbol transmitted across the air interface. The IDFT accepts a sequence of complex data samples and produces a complex output sequence of the same length.

4.1.2 Demapper

The symbol demapper (demapper) translates the complex data samples produced by the IDFT into soft-valued bits. Each bit in the symbol is given a log-likelihood ratio value based on the exact position of the received symbol in the IQ plane.

The soft-decision values depend on the modulation scheme used (and therefore the constellation pattern produced). Possible modulation schemes used for LTE Uplink Shared Data channel (PUSCH) include QPSK, 16QAM and 64QAM.

4.1.3 Rate De-Matcher

The rate dematcher maps the size of the data in the transport layer onto the appropriate physical layer resources by inserting or removing redundancy.

The rate dematcher takes soft-value bits as input from the symbol demapper and produces systematic (S) and parity bits (P1, P2) for the turbo.

4.1.4 CTC (Turbo Decoder)

The turbo decoder is used to perform forward error correction of the input data stream by utilizing the redundancy in the encoded data stream. Turbo codecs have become the coding technique of choice in many communication systems due to their near Shannon limit error correction capability.

The turbo decoder takes the systematic and parity bits produced by the rate dematcher and produces a stream of bits, representing the recovered data bits. The turbo block operates on the code blocks produced by the rate dematcher.

4.2 Understanding LTE Latency Requirements

Providing low network latency is a key network metric for LTE systems. Services such as voice over IP, video conferencing and network gaming applications are particularly sensitive to latency as it has a major impact on the user's experience of these services.

To provide this reduction in latency, LTE employs two main mechanisms [21]

1. Reducing the Transmission Time Interval (TTI). LTE will use a TTI of 1ms, 50% less than the previous generation wireless standard HSUPA (High Speed Uplink Packet Access).
2. Faster HARQ or retransmission processes for lost or damaged blocks of data. By providing faster feedback mechanisms, LTE will enable the transmitter to resend the lost blocks earlier, making the radio transmission more efficient.

The LTE user-plane latency is defined in [21] as: “the one-way transit time between a packet being available at the IP layer in either the UE/RAN edge node and the availability of this packet at IP layer in the RAN edge node/UE. The RAN edge node is the node providing the RAN interface towards the core network”, where UE stands for User Equipment, or the mobile device and RAN stands for Remote Access Network, referring to the eNB (Evolved Node B) or base station. The requirement for the LTE user-plane latency is 5ms.

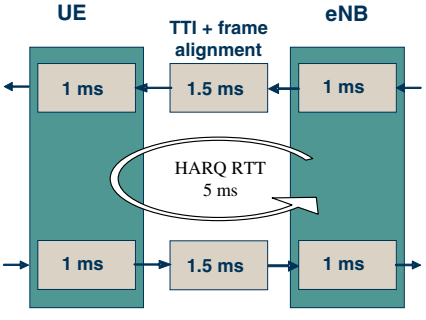


Fig. 7. User Plane Latency components in LTE[22]

This latency figure contains several identifiable latency components as shown in Figure 7. The times shown in the Figure are a lower bound as to what is achievable with LTE, as they assume that a single user system, transmitting small IP packets (0 byte payload with IP headers). This implies that the network is not loaded and that there are no delays due to queuing or scheduling. In addition to this, the HARQ round trip time must be 5ms.

For eNB providers, this means that providing these latency targets are met, a trade off may be made between the Uplink and Downlink processing times in the eNB. It may be possible for the provider to use only 0.6ms for the DL processing time leaving an extra 0.4ms for the UL processing. Since the UL processing is significantly more complex hat the DL processing such an analysis might prove valuable, requiring careful analysis of the latency in each individual components.

4.3 Understanding LTE Complexity

The LTE specification is characterized by an increase in the complexity of the signal processing over other OFDM systems, such as WIMAX. LTE requires high data rate forward error correction, Multiple Input, Multiple Output antenna techniques, and in the uplink, SC FDMA requires an extra stage of processing to transform from the frequency domain to the time domain. This additional complexity in the signal processing, is also matched by commensurate increase in complexity of the control required to manage the signal processing elements. Therefore the LTE system is highly suited for a HW/SW approach whereby the complex data processing is done in HW, and the control is performed in SW.

5 Reference Architecture

The reference architecture considered to implement the LTE system is based on the Altera Hardware/Software solution for high performance datapath applications [20]. The solution is based on a combination of a multithreaded soft processor and hardware accelerators.

The overall processing is based on an asynchronous execution paradigm triggered by task (i.e. software process) and event (i.e. hardware accelerated process) requests. The overall system is composed of the multithreaded processor with supporting control and interfaces that manage the communication with dedicated accelerator modules through buses and queues. The details of the Hardware/Software interaction and communication are hidden from the applications developer and Hardware/Software communication introduces an almost negligible latency of very few clock cycles.

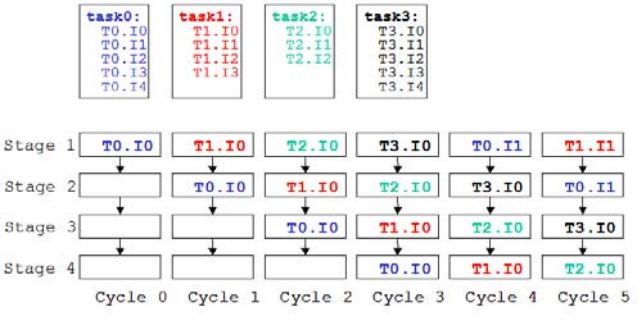


Fig. 8. Instruction interleaving

The soft processor can execute 8 threads simultaneously by means of a simultaneous multithreading. In a traditional multithreading, a new thread is executed when the previous thread stalls; however, in this design, instructions corresponding to 8 different threads are mixed (interleaved) in the pipeline. This allows to avoid the overhead for thread switching and pipeline stalls since whatever hazard in a given thread instruction is resolved before the next instruction of the same thread is executed. The execution scheme is depicted in Figure 8 for an exemplified pipeline with 4 stages. One of the advantages of this approach is that the software execution time becomes deterministic given an execution path, since all the sources of indeterminism are avoided. Hazards and context switching introduce no penalty, and no cache is used in the system (in the great majority of datapath applications data and program code are limited in size and can be stored directly on the chip).

For each independent flow a unique ID is assigned (PID). The number of PIDs is defined during the hardware synthesis of the soft processor and it can be adjusted to suit the application performance requirements.

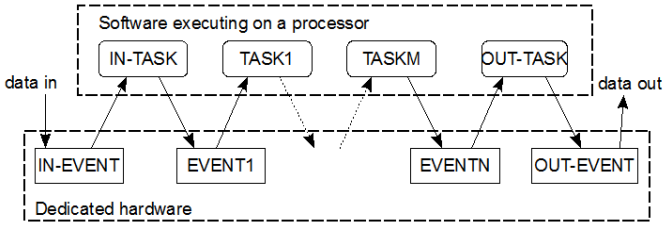


Fig. 9. Execution Flow on the Hardware/Software Altera architecture for datapath processing

A typical processing flow combines Tasks that are executed in software and Events executed by dedicated hardware blocks, as schematically depicted in Figure 9. The inherent parallelism of the multithreaded processor and the multiplicity of dedicated hardware blocks allows for several independent flows to be processed concurrently.

5.1 Architecture Modelling with TCPN

Since the hardware and software parts can generally run at two different frequencies F_{hard} and F_{soft}, we consider a reference frequency F_{ref}. The modelling of this architecture with a Petri Net can be done as following:

- Multithreading. The execution of eight threads on the same processor at frequency F_{soft}, with the instruction interleaving described in the previous Section, is functionally equivalent to the execution of eight threads on eight identical processors each one running at a frequency $F_{soft}/8$. The multithreaded processor is therefore represented with a resource class having availability equal to eight and frequency equal to $F_{soft}/8$.
- Timing. Both hardware and software times can be considered as deterministic. Each function f_i executing on a resource r_j is associated with the execution ticks t_{ij} computed as:
  - $t_{ij} = (Num.\ of\ instructions * F_{ref})/(F_{soft}/8)$ (SW).
  - $t_{ij} = (Num.\ of\ clock\ cycles * F_{ref})/(F_{hard})$ (HW).
- Number of PIDs. An additional place (PID-Place) is added, having as initial marking a number of R-Tokens equal to the number of PIDs. Each time a new block of data enters the system an R-Token in consumed from the PID-Place and is produced when the block of data exits the system. In a more generic architecture this place can be used to represent the maximum depth of queues for Hardware/Software communications.
- Communication. Since the overhead for the Hardware/Software communication is negligible, it is not modelled. In a more generic architecture, if the communication introduces substantial overhead, this can be represented exploiting the same framework used for the rest of the system (for example, a data transfer between two modules is the function and a bus is the resource).

6 Mapping of the LTE Application on the Platform

The implementation of the LTE application has been organized as follows. The majority of the complex DSP processing is done with hardware accelerators; this includes IDFT, Symbol Demapper, Rate De-matcher, and Turbo.

The control of the data flow through these blocks, and the configuration of the blocks with the relevant parameters (see Table I) is done using software running on the threads in the processor.

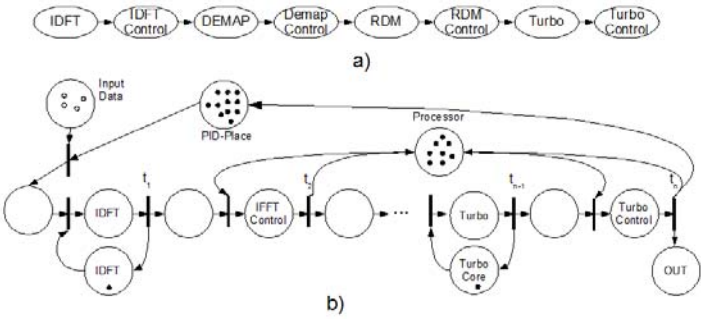


Fig. 10. a) Application Graph for the LTE application, b) Sketch of the TCPN associated to the LTE application graph

Figure 10 represents the application graph and a sketch of the correspondent TCPN.

Coloured tokens that flow into the net contain all the information needed to influence the system evolution, in particular timing and computational path.

The most important parameters are: pid, number of resource block, modulation scheme, coding block size, coding rate, filter bits, redundancy version and constitute the fields characterizing the tokens. Other parameters (like the number of subcarriers or the number of symbols per TTI) constitutes system settings and are therefore associated to the system model instead of being stored into the tokens.

The times associated to the transitions depend on the number of hardware clock cycles and software instructions required to process the functions. For each function composing the system appropriate timing has been considered, often dependent on the parameters cited before.

In the LTE system where there are multiple complex IP Blocks interacting, it is often necessary to buffer blocks of data before the processing. This may be because the function requires all data present in order to calculate the result or it may be done to achieve the required throughput of the system.

In order to obtain a more accurate model, the behaviour of the hardware modules have been described with a finer grain of detail, by decomposing the functions in more steps and considering additional resources like buffers and memories.

In the following, we present the Petri Nets schemes developed for the blocks of the LTE architecture, highlighting the strategies used to enhance the accuracy of the model.

6.1 IDFT

The IDFT is characterized by the loading, executing and unloading phases. For each phase, the computing time is function of the resource block size. The three phases must be completed for each data before the computing can start for a new one. The situation is represented in Figure 11 where the computing is decomposed in three steps and the R-Place associated with the IDFT core is connected respectively to the transition entering the first phase and the transition exiting the last phase.

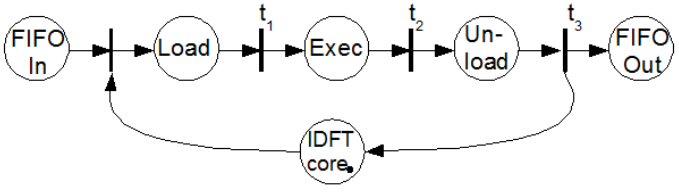


Fig. 11. Petri Net structure of the IDFT block

6.2 Demapper

The symbol demapper module is responsible for transforming the IQ samples representing the constellation points as dictated by the modulation scheme, to soft decision bits or LLR (log-likelihood ratio) values.

The Petri Net scheme correspondent to the SDM is represented in Figure 12. The hardware resource that implements the function is pipelined, therefore it is modelled as explained previously, by distinguish the time required for the stage (that is equal to one clock cycle in this case) and the time required for the computing.

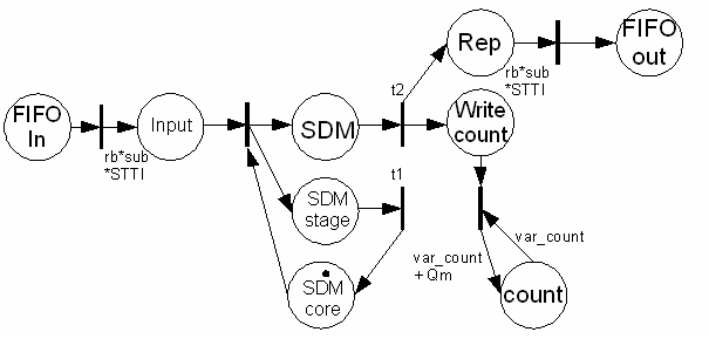


Fig. 12. Petri Net structure for the Symbol demapper

The module operates with the granularity of a “complex data sample”, that for each user is proportional to the number of resource blocks (in particular it is equal to the number of resource blocks multiplied by the subcarriers *sub* and symbols per TTI

STTI). The first transition therefore generates the tokens corresponding to the data samples that will be processed by the engine and put them in the input place. To generate the next software task the processing of all the tokens must be terminated, therefore a place representing a repository is used to activate the next software task when the processing is finished. Some extra checks, that for simplicity are not shown in the Figure, are used to guarantee the correct execution order.

Each sample generates a number of soft bits dependent on the modulation scheme (represented by parameter *Qm* in the Figure). For each processed sample, the total number of bits generated is updated, by the use of the place "count" that contains an integer value token. The token is withdrawn and put back with its value updated. This is an alternative to the use of many tokens representing the soft bits, that has been chosen in order to increase the model efficiency. Indeed, for the simulation engine, updating the value of a single token is simpler and quicker than maintaining all the information related to a large number of tokens.

6.3 Rate De-Matcher

The rate de-matcher is activated when a request is ready and the symbol demapper has produced enough bits to start the computation. Therefore, we use a condition on the module input transition that checks if enough bits are available to start the computation. In this case, the transition fires, with the effect that the number of bits is updated and the computation is started. Figure 13 represents the corresponding net.

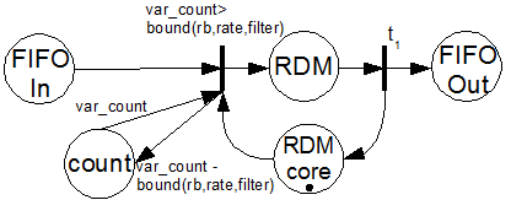


Fig. 13. Petri Net structure for the rate de-matcher

6.4 CTC (Turbo Decoder)

The Turbo model has two input buffers, a core execution module and two output buffers. The functioning is divided into 3 stages: loading, executing and unloading.

The corresponding PN is represented in Figure 14. The load operation can start when the input port and a input buffer are available. After that, data are ready to be processed by the core. The processing can start if an output buffer is available (to write the produced data) and the execution core is free. At the end of the execution the input buffer is freed and can be used to load new data. Finally data are unloaded when an output port is available and at the end the output buffer is freed.

The transitions timing depends on the parameters affecting the system, and on the configuration of the hardware module.

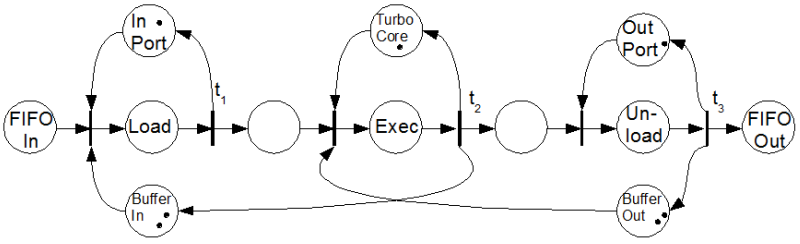


Fig. 14. Petri Net structure of the Turbo block

7 Experimental Results

In order to collect information about the application performance, the Petri Net model has been simulated using the CPNtool developed by CPN Group of University of Aarhus in Denmark [19]. The tool allows to describe a TCPN, to automate the simulations and to collect statistics. The results obtained from the model have been compared with accurate simulation results obtained by implementing the application on the reference architecture. These results have been collected by integrating the ISS simulator of the Altera multithreaded CPU with software models of the hardware event modules annotated with high level latencies.

In the following, we investigate different transmission scenarios. Each configuration specifies the number of users, and for each user the assigned number of Resource Blocks (RB), the coding rate (CR) and the modulation scheme. The number of users and resource blocks affect the number of blocks processed by the system. The coding rate and the modulation scheme affect the block dimension. In particular the block dimension increases with a lower coding rate and a modulation scheme with more constellation points.

The considered scenarios are the following:

1. 110 users, with 1 resource block each, coding rate 5/6, modulation 64 QAM.
2. 5 users with different spectrum allocations: (RB=10,CR= 5/6), (RB=36,CR=1/9), (RB=20,CR=1/9), (RB=4,CR=3/4), (RB=40, CR=1/4), modulation 64 QAM.
3. 2 users, with different spectrum allocations (RB=100,CR=5/6), (RB=10,CR=5/6), modulation 64 QAM.
4. 50 users with 1 resource block each, coding rate 1/3, modulation QPSK.
5. 18 users with 6 resource blocks each, coding rate 2/3, modulation 16 QAM.

Figure 15 shows the data chunks output times obtained for in the five scenarios, for both the simulations. The dimension and number of the data chunks elaborated for each user are computed according to the LTE specification [18].

The performance evaluation shows that the system composed of the shaded blocks represented in Figure 6 is able to support the strict timing performance (1ms for 110 resource blocks) in all the tested configurations. The use of the Petri Net model allows to obtain such evaluation in early stages of the system development, without requiring an actual hardware/software integration.

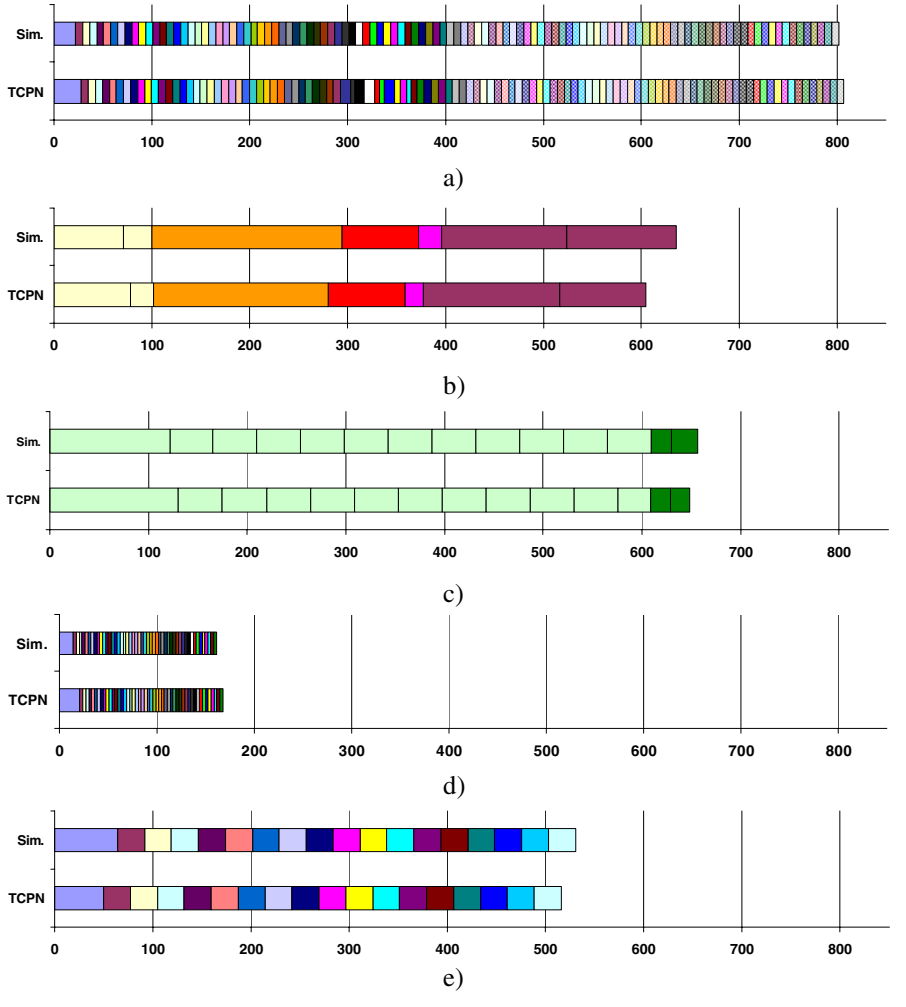


Fig. 15. Comparison between the system output times obtained through TCPN and the real system simulation for different scenarios. Each colour represents a user.

The comparison with the results obtained by combining the hardware and software modules shows that the TCPN model can provide a good accuracy. Figure 16 represents the errors in the arrival times for all the data chunks in all the five scenarios. The difference between the arrival times obtained using the Petri Net and the ones obtained simulating the system are always inferior to 35 microseconds, as shown by the left Y-axis in the graph. Normalizing the values with the greatest arrival time (first scenario) we obtain errors inferior to 5%, as shown by the right Y-axis in the graph. Considering for each scenario a normalization to the greatest arrival time of that scenario, we still obtain errors inferior to 5%.

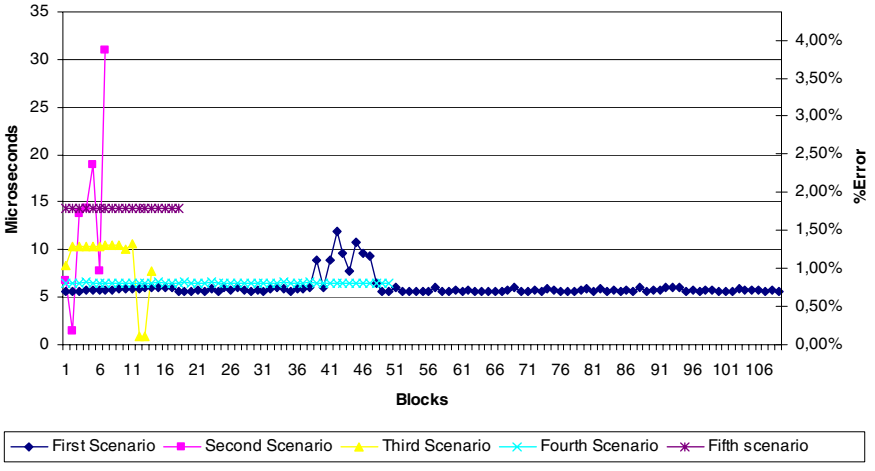


Fig. 16. Absolute and percentage errors of the blocks arrival times in each scenario

8 Conclusion

One of the main problems, when designing a DSP application, is the meeting of strict timing constraints; however, the verification of the system can usually be performed very late in design phase. This paper proposes the use of Timed Coloured Petri Net for the early evaluation of the system Performance of Hardware/Software DSP applications. We show how to model an application by generating a TCPN that considers the functions and the resources composing the system. The modelling of the 3GPP LTE application has been considered as case study. The experimental results are quite accurate when compared with hardware/software simulations and, as a substantial advantage, can be generated in early stages of the design, when modifications and improvements of the system are still possible. The proposed approach reduces the risk of highly expensive re-spins for the modification of the final system and provides room for the exploration of the solution space.

References

1. Gajski, D.D.: IP-based methodology. In: Proc. of 36th DAC (1999)
2. Haubelt, C., Falk, J., Keinert, J., Schlichter, T., Streubühr, M., Deyhle, A., Hadert, A., Teich, J.: A SystemC-based design methodology for digital signal processing systems. EURASIP J. Embedded Syst. 1 (2007)
3. Ueda, K., Sakanushi, K., Takeuchi, Y., Imai, M.: Architecture-level Performance Estimation for IP-based Embedded Systems. In: DATE 2004 (2004)
4. Papoulis, A.: Probability, Random Variables, and Stochastic Processes, pp. 515–553. McGraw-Hill, New York (1984)
5. Bunday, B.D.: An Introduction to Queueing Theory. Oxford University Press, Oxford (1996)

6. Zubereck, W.M.: Timed Petri Nets – definitions, properties and applications. Microelectronic and Reliability, 627–644 (1991)
7. Zurawski, R., Zhou, M.: Petri Nets and Industrial Applications: A Tutorial. IEEE Transactions on industrial electronics 41(6) (1994)
8. Yakovlev, A., Gomes, L.: Luciano Lavagno, Hardware Design and Petri Nets. Springer, Heidelberg (2000)
9. Blume, H., Von Sydow, T., Noll, T.G.: A case Study for the application of Deterministic and Stochastic Petri Nets in SoC communication Domain. Journal of VLSI Signal Processing (2006)
10. Zhan, J., Sang, N., Xiong, G.: Formal Co-verification for SoC Design with Colored Petri Net. LNCS. Springer, Heidelberg (2005)
11. Maciel, P., Barros, E., Rosenstiel, W.: A Petri Net Model for Hardware/Software Codesign. Design Automation for Embedded Systems Journal (1999)
12. Rust, C., Tacken, J., Böke, C.: Pr/T-Net Based Seamless Design of Embedded Real-Time Systems. In: ATPN 2001 (2001)
13. Gomes, L., Costa, A.: Petri nets as supporting formalism within Embedded Systems Co-design. In: Industrial Embedded Systems Symposium (2006)
14. Petri, C.A.: Communication with Automatas, PhD Dissertation, University of Bonn (1962) (in German)
15. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Basic Concepts. Monographs in Theoretical Computer Science, vol. 1. Springer, Heidelberg (1997)
16. Bourcerie, M., Morel, J.Y.: Algebraically structured colored Petri nets to model sequential processes. IEEE Transactions on Systems, Man, and Cybernetics 27(4), 681–686 (1997)
17. 3GPP TS 36.300 Technical Specification group radio Access Network, E-UTRA, E-UTRAM, Overall Description, Stage 2
18. 3GPP TS 36.201, 36.211-14
19. CPN Tools, Coloured Petri Net Group, University of Aarhus, Denmark, http://www.daimi.au.dk/CPnets/
20. Altera Corporation, http://www.altera.com
21. R2-061402, Concept evaluation of user plane latency in LTE, Ericsson
22. R2-072187, LS on LTE latency analysis, Ericsson

Real-Time Biologically-Inspired Image Exposure Correction

Vassilios Vonikakis, Chryssanthi Iakovidou, and Ioannis Andreadis

Democritus University of Thrace
Department of Electrical & Computer Engineering
Laboratory of Electronics
Xanthi, Greece

Abstract. This chapter presents a real-time FPGA implementation of a biologically-inspired image enhancement algorithm. The algorithm compensates for the under/over-exposed image regions, emerging when High Dynamic Range (HDR) scenes are captured by contemporary imaging devices. The transformations of the original algorithm, which are necessary in order to meet the requirements of an FPGA-based hardware system, are presented in detail. The proposed implementation, which is synthesized in Altera's Stratix II GX: EP2SGX130GF1508C5 FPGA device, features pipeline architecture, allowing the real-time rendering of color video sequences (25fps) with frame sizes up to 2.5Mpixels.

Keywords: FPGA, Real-Time Image Enhancement, Human Visual System, High Dynamic Range Imaging.

1 Introduction

Conventional Standard Dynamic Range (SDR) sensors, which are the case in most consumer-electronics cameras, fail to adequately reproduce HDR scenes, which can be common in outdoor capturing conditions. The main reason for this problem is the low dynamic range of the capturing device, compared to the dynamic range of the scene. As a result, the captured images usually suffer from under/over-exposed regions, in which, little or no information is available to the observer. Adjusting the exposure time is not a solution to the problem, since acceptable reproduction can only be achieved for the dark or the bright image regions but not for both. This is clearly depicted in Fig. 1.

A straight-forward solution to this problem is the use of HDR capturing devices instead of the conventional SDR ones. Nevertheless, HDR cameras cannot always provide a practical solution. Their increased cost has limited their use, while the majority of the existing vision systems are already designed for SDR cameras. Furthermore, an additional algorithm is required in order to perform the mapping of the HDR data to an SDR monitor. Another possible solution is to acquire an HDR image by combining multiple SDR images, captured with different exposures [1]. Nevertheless, this solution can be applied only to static scenes, since it is impossible to have multiple exposures of a moving object, always, at the same background.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 133–153, 2010.

Consequently, this approach cannot be applied to time-critical applications i.e. video. A third solution to the above problem is the use of an unsupervised tone-enhancement algorithm, which will compensate for the under/overexposed regions of SDR images, without affecting the correctly exposed ones. This approach overcomes the limitations of the previous two solutions: it does not increase considerably the total cost of the system and it can be used in video sequences as well.

Fig. 1. There is not a single exposure that can adequately capture both the dark and light regions of an HDR scene

Several enhancement algorithms have been proposed in this direction, yet, very few have been implemented in hardware. Some of the most important are the Retinex family of algorithms (Retinex: Retina + Cortex), among which, the more widespread are the "Multi Scale Retinex with Color Restoration" (MSRCR) [2] and the "Variational Framework for Retinex" [3]. The first has been implemented on a Digital Signal Processor (DSP) [4, 5], allowing the real-time, single-scale rendering of grayscale images, with sizes up to 256×256 pixels. A variation of the second has been implemented on an Application Specific Instruction-set Processor (ASIP) [6], allowing the processing of SXGA (1280×768 pixels) or WXGA (1366×768 pixels) still images in 1 sec, or the a real-time rendering of video frames with size 256×256 pixels and frame rates up to 29 frames per second (fps). Both implementations do not meet the VGA standard (color images of 640×480 pixels size and 25fps) in video rendering. Recently, an alternative enhancement algorithm, inspired by the shunting characteristics of the center-surround cells of the Human Visual System (HVS), has been presented in [7]. It exhibits low complexity, as well as the fastest execution times, compared to the algorithms of the previous two implementations. Consequently, it constitutes a good basis for a hardware implementation.

This chapter presents an FPGA implementation of this algorithm. It focuses on the transformations which are necessary in order to optimize the original algorithm for meeting the requirements of an FPGA-based system. The main objective is to implement a pipeline architecture, which will allow the real-time rendering of color video sequences, with sizes greater than the contemporary implementations (256×256

pixels). Two alternative architectures are presented, both synthesized in Altera's Stratix II GX: EP2SGX130GF1508C5 FPGA device. The first allows the real-time (25fps) rendering of color images, with sizes up to 640×480 pixels. The second architecture can render, in real-time, images with sizes up to 2.5MPixels. Both architectures are designed in a way that easily allows future improvements in the core of the algorithm. This, in addition to the FPGA platform, results into a low-cost, robust solution, which can be used to other vision systems as preprocessing, compensating for the low dynamic range of the SDR image sensor.

The chapter is organized as follows. In Section 2 the structure of the original algorithm is briefly described. In Section 3, the transformations of the original algorithm, which will make it suitable for hardware implementation, are presented. The proposed implementations are presented in Section 4, with gate level integrated circuits. The comparison between the hardware and the software are provided in Section 5. Finally, conclusions and a discussion are provided in Section 6.

2 Structure of the Algorithm

In this section the original algorithm, which compensates for the under/over-exposed regions, will be briefly described in order to underline the modifications made in the proposed hardware implementation. A detailed description of the algorithm is out of the scope of this chapter. Extensive details can be found in [7].

The block diagram of the original algorithm is depicted in Fig. 2. In order not to distort the colors of the image, the YCbCr color space is employed, which decorrelates the chromatic and achromatic information. The original method works only on the luminance component and comprises three different stages: a linear stretch of the luminance component, a parameter estimation block and the local enhancement stage. The core of the algorithm is depicted in the equations (1)-(7). Equation (1) stretches linearly the luminance values to the interval [0,*B*], in order to use the full range of the Y channel. *B* is the maximum value of the luminance data (255 for 8-bit values), $Y_{\min}$ and $Y_{\max}$ are the minimum and maximum luminance values of the image, while (*i*, *j*) denotes the spatial coordinates of the pixel.

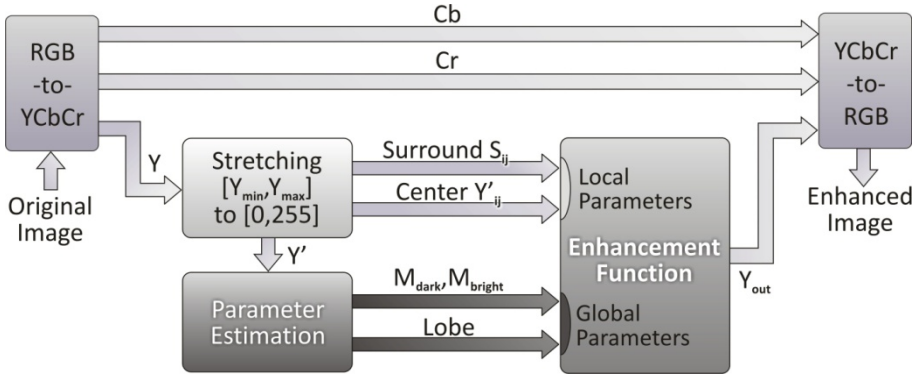


Fig. 2. The block diagram of the original algorithm used in the hardware implementation

$$Y'_{ij} = \frac{Y_{ij} - Y_{\min}}{Y_{\max} - Y_{\min}} \times B \tag{1}$$

$$Y^K_{out,ij}\left(Y', S\right) = \begin{cases} \dfrac{\left[B + A\left(S^K_{ij}\right)\right] \cdot Y'_{ij}}{A\left(S^K_{ij}\right) + Y'_{ij}} & \forall S^K_{ij} < \dfrac{B}{2} \\ \dfrac{A\left(S^K_{ij}\right) \cdot Y'_{ij}}{A\left(S^K_{ij}\right) + B - Y'_{ij}} & \forall S^K_{ij} \geq \dfrac{B}{2} \end{cases} \tag{2}$$

$$A(S) = \begin{cases} \left[M_{dark} + q(S)\right] \cdot d(S) & \forall S < \dfrac{B}{2} \\ \left[M_{bright} + q(B-S)\right] \cdot d(B-S) & \forall S \geq \dfrac{B}{2} \end{cases} \tag{3}$$

$$d(x) = \frac{B/2}{B/2 - x} \quad \forall x \in [0, B/2) \tag{4}$$

$$q(x) = \frac{x^2}{Lobe} \tag{5}$$

$$S^K_{ij} = \frac{1}{\left(2b_K + 1\right)^2} \sum_{y=i-b_K}^{i+b_K} \sum_{x=j-b_K}^{j+b_K} Y'_{yx} \tag{6}$$

$$Y_{out,ij} = \frac{Y^{K1}_{out,ij} + Y^{K2}_{out,ij} + Y^{K3}_{out,ij}}{3}, \text{ with } b_{K1} < b_{K2} < b_{K3} \tag{7}$$

Equations (2)-(5) describe the enhancement function of the algorithm. As it is clearly depicted in Fig. 2, the enhancement function uses two local and three global parameters. The local parameters, which depend on the local characteristics of the image, are the stretched luminance value Y'_{ij} of the pixel and the average luminance S^K_{ij} of its square surrounding region. K denotes the scale upon which the surround is calculated. Equation (6) describes the surround calculation. Three different surround sizes are employed, with b_K denoting the radius of the square surrounding region for scale K. The final corrected value $Y_{out,ij}$ for every pixel, is calculated by

equation (7) and it is the average between the corrected values $Y^{K}_{out,ij}$ of the three spatial scales. The global parameters, which depend on the global image statistics, are M_{dark}, M_{bright} and *Lobe*. Their exact calculation is described by the following equations.

$$bin\_low = \frac{\sum_{i=1}^{py}\sum_{j=1}^{px} u\left(\frac{B}{3} - Y'_{ij}\right)}{px \cdot py} \times 100 \tag{8}$$

$$M_{dark} = \frac{270}{100}\left(100 - bin\_low\right) + 30 \tag{9}$$

$$bin\_high = \frac{\sum_{i=1}^{py}\sum_{j=1}^{px} u\left(Y'_{ij} - 2\frac{B}{3}\right)}{px \cdot py} \times 100 \tag{10}$$

$$M_{bright} = \frac{270}{100}\left(100 - bin\_high\right) + 30 \tag{11}$$

$$bin\_middle = 100 - bin\_low - bin\_high \tag{12}$$

$$Lobe = \frac{29}{100}\left(100 - bin\_middle\right) + 1 \tag{13}$$

where *px*, *py* are the dimensions of the original image and $u(\cdot)$ is the unitary step function. *bin_low*, *bin_middle* and *bin_high* are the bins of a 3-bin normalized histogram (*bin_low*[0, *B*/3], *bin_middle*(*B*/3, 2*B*/3), *bin_high*[2*B*/3, *B*]) that divides the range of the stretched luminance channel Y' into 3 equal tone intervals: dark, medium and bright. For a detailed analysis on the characteristics of the global parameters, refer to [7].

3 Algorithm Optimization

3.1 Optimizing the Structure of the Algorithm

The straight-forward conversion of an algorithm's software implementation to hardware, usually leads to unsatisfactory results. Most of the times many transformations are necessary in order to optimize the algorithm for hardware implementation. This section

focuses on these optimizations. The original algorithm, as depicted in Figure 2, requires three different scans of the image, in order to get the final result. This is depicted in Figure 3.

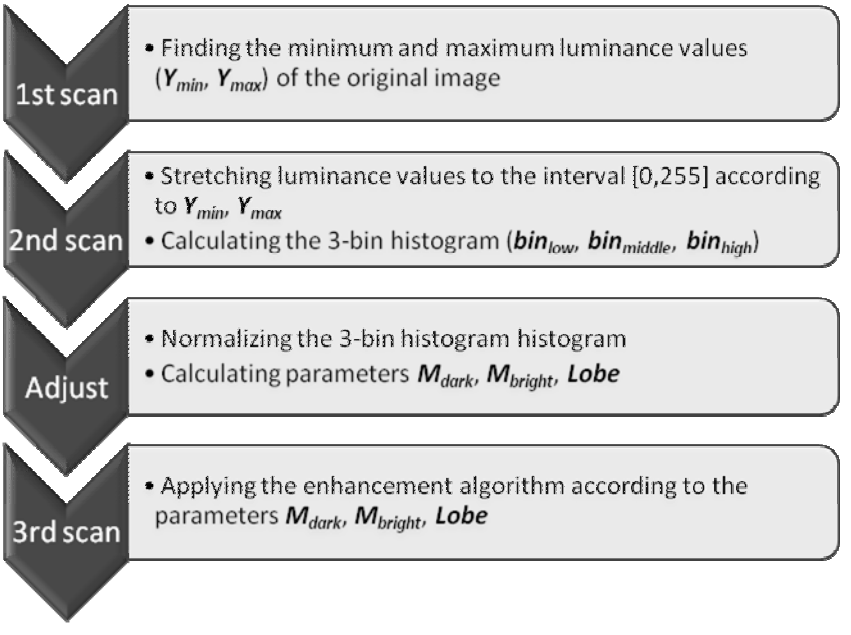


Fig. 3. The structure of the original software implementation

The first scan of the image is necessary in order to find Y_{min} and Y_{max}. In the second scan, equation (1) is applied to all pixels, stretching their luminance values to [0,255]. The second scan can be eliminated using a Look-Up-Table (LUT). Instead of applying the stretching transformation to all image pixels separately, equation (1) can be executed only 256 times, one for each luminance value. These precomputed values are stored in the "Stretching LUT". Feeding the original luminance value Y_{ij} of a pixel to the *StretchingLUT* module, will output its stretched luminance value Y'_{ij}, as equation (14) indicates.

$$Y'_{ij} = StretchingLUT\left[Y_{ij}\right] \tag{14}$$

This simple transformation reduces the required scans of the image to two, as Figure 4 indicates.

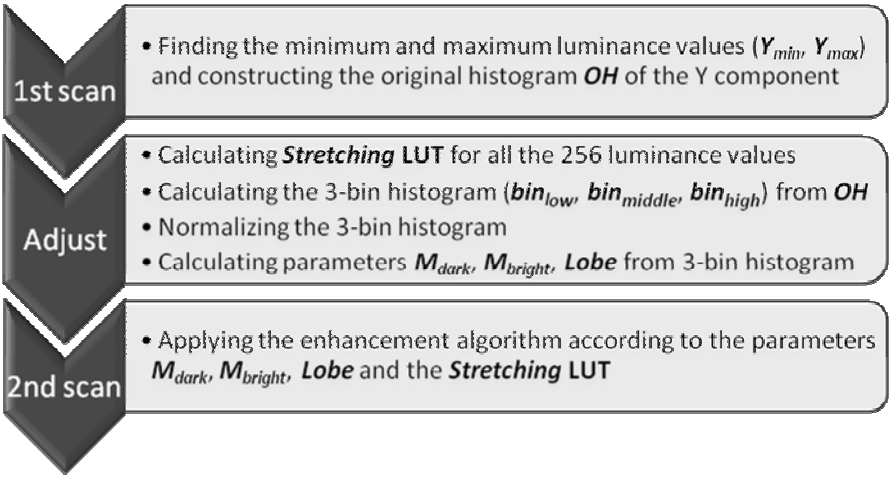


Fig. 4. The structure of the algorithm after the Stretching LUT transformation

3.2 Requirements for a Pipelined Architecture

Implementing a pipelined architecture is a primary objective, since it will allow a high throughput, which is essential for real-time applications. For this reason, two identical modules are needed for the original histogram generator (*OH1*, *OH2*), the stretching LUT (*StretchingLUT1*, *StretchingLUT2*) and the global parameters (*Parameters1*, *Parameters2*). The first modules process the odd frames, while the second modules process the even ones. This is depicted in Figure 5. When *frame k* is in the *adjust* state, the *1st scan* is performed for *frame k+1*. Consequently, *OH1* processes *frame k*,

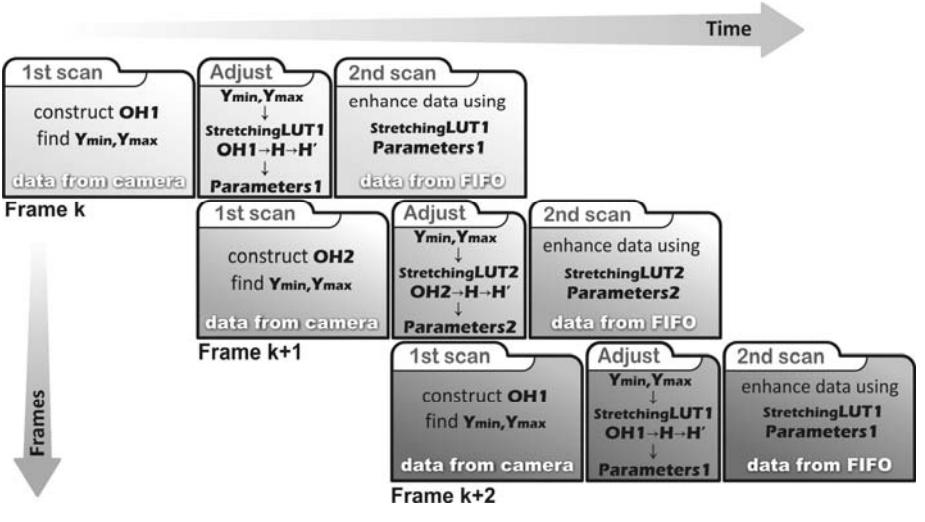


Fig. 5. Double components are necessary in order to achieve a pipelined architecture

while *OH2* generates the histogram of *frame k+1*. When *frame k* is in the *2nd scan*, *frame k+1* is in the *adjust* state. Then, *StretchingLUT1* processes *frame k*, while *StretchingLUT2* is being loaded with the stretching values for *frame k+1*. Similarly, when *frame k* is in the *2nd scan*, its image data are enhanced using the global parameters from module *Parameters1*. At the same time, the global parameters of *frame k+1* are calculated using the module *Parameters2*.

4 Hardware Implementation

This section presents the key stages of the proposed hardware implementation. Figure 6 depicts the block diagram of the system. We assume that the frames are fed into the FPGA sequentially, pixel by pixel, in the RGB color space format. Consequently, in order to create the second scan, the data are fed into a FIFO memory, after the transformation from RGB to YCbCr. This FIFO should have an appropriate length in order to introduce a delay to the data, equal to the execution time of both *1st scan* and *adjust* stages. Taking into consideration that the adjust stage requires 300 clock cycles, the FIFO length should comprise 1frame+300 memory elements. As a result, when the *2nd scan* is about to begin, the YCbCr data will be ready for processing.

As mentioned in the previous section, two *StretchingLUT* modules are required for a pipelined architecture: one for the odd and one for the even frames. Figure 6 however

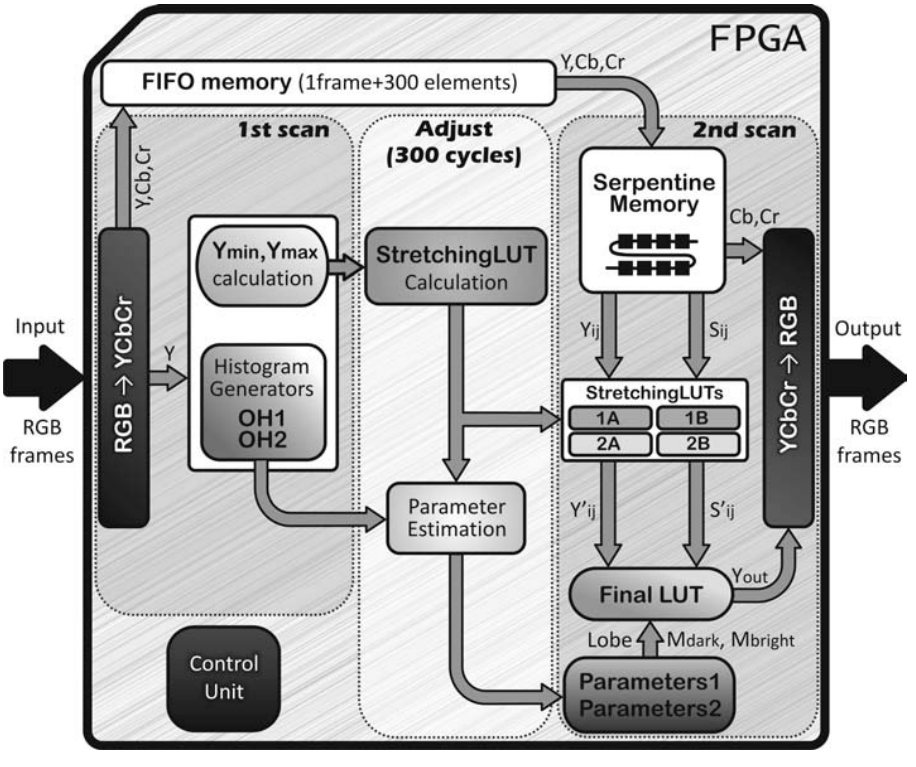


Fig. 6. The proposed hardware implementation

shows that two extra *StretchingLUT* modules are required: one for the luminance value of the pixel Y_{ij} and one for its surround luminance S_{ij}. This increases the total number of modules to four. *StretchingLUT* 1A and 1B are identical and are used simultaneously for the luminance values Y_{ij} and S_{ij} of the odd frames. Similarly, *StretchingLUT* 2A and 2B are also identical and are used simultaneously for the luminance values Y_{ij} and S_{ij} of the even frames. While the odd LUTs are used to transform the luminance values, the even LUTs are loaded with data. In the following frame, the even LUTs are used for transforming luminance values and the odd LUTs are stored with data.

4.1 Color Space Transforms

The transformations RGB→YCbCr and YCbCr→RGB are the first and last processing stages of the system. The original mathematical forms of the transformations comprise floating point arithmetic, which can be computationally intensive in a straight-forward implementation. In order to avoid the floating point operations, the transformations are altered as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.230R + 0.661G + 0.109B \\ -0.101R - 0.338G + 0.439B + 128 \\ 0.439R - 0.399G - 0.040B + 128 \end{bmatrix} = \\ = \frac{1}{1024} \begin{bmatrix} 236R + 677G + 112B \\ -103R - 346G + 450B + 131072 \\ 450R - 409G - 41B + 131072 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.084Y + 1.793(Cr - 128) \\ 1.084Y - 0.534(Cr - 128) - 0.213(Cb - 128) \\ 1.084Y + 2.115(Cb - 128) \end{bmatrix} = \\ = \frac{1}{1024} \begin{bmatrix} 1110Y + 1836Cr - 235012 \\ 1110Y - 547Cr - 218Cb + 97911 \\ 1110Y + 2166Cb - 277217 \end{bmatrix} \tag{16}$$

The floating point numbers are multiplied by 1024 and the appropriate divider is introduced at the front of the matrix. The number 1024 is selected for two reasons. First, the multiplication with any integer number greater or equal to 1000 results to another integer number, thus, avoiding the decimal points of the original transformation. Second, 1024 is a power of 2 (2^{10}=1024) and, therefore, the divider can be implemented with 10 right shifts of the final result. The implementation of equations (15) and (16) is depicted in Figure 7 and Figure 8, respectively.

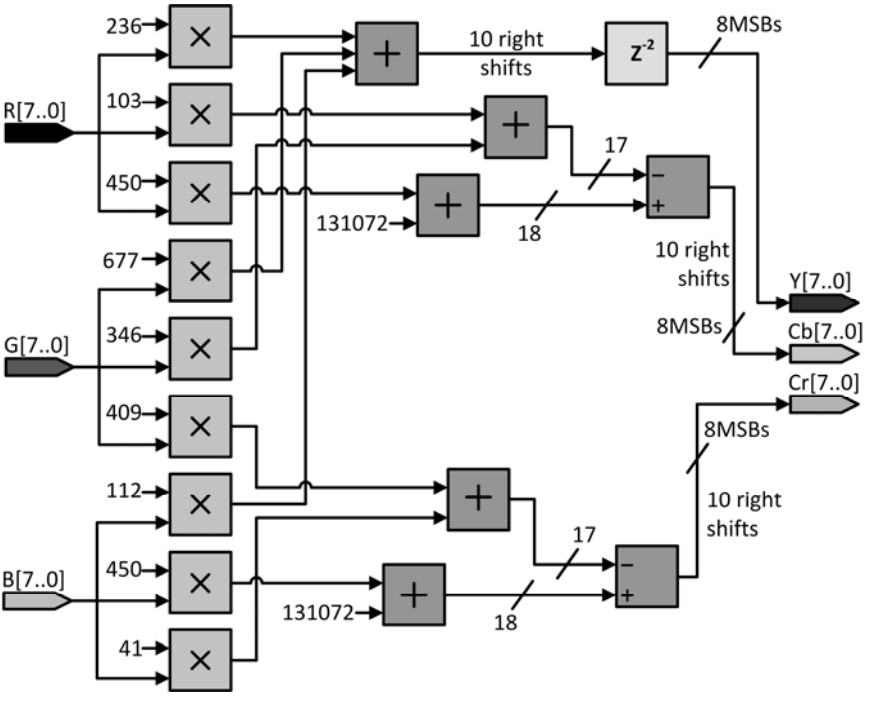


Fig. 7. Implementation of the RGB→YCbCr transformation

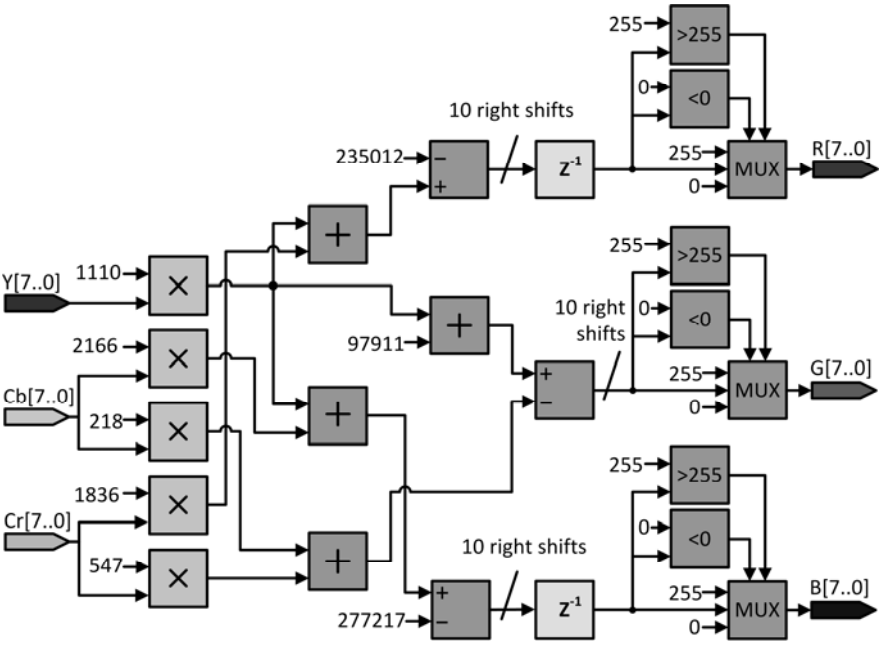


Fig. 8. Implementation of the YCbCr→RGB transformation

The above implementations employ many multiplications, which are considered to be computationally intensive operations. For this reason these multiplications are implemented with parallel shifts, as the following example indicates.

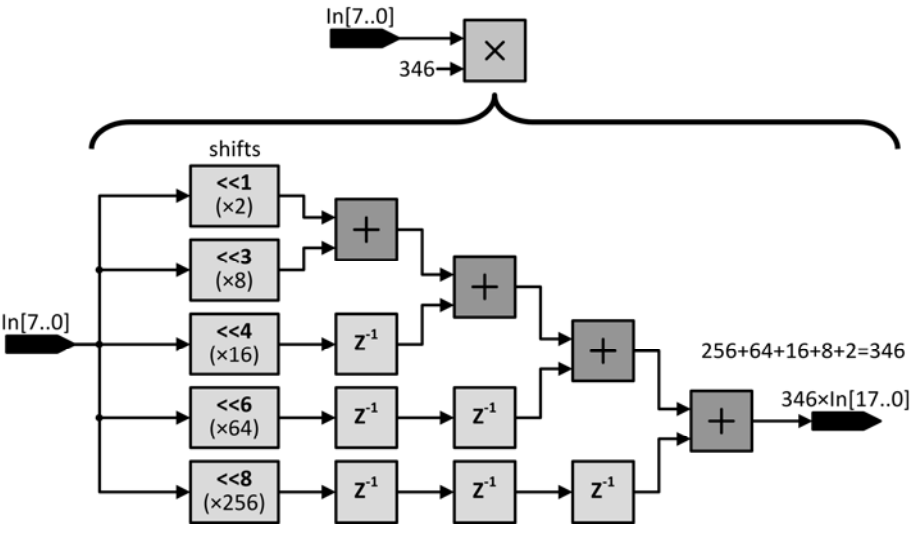


Fig. 9. Implementation of one of the multiplications of Figures 7 and 8

4.2 Calculation of StretchingLUT Modules

The stretching function of equation (1) comprises among others, a multiplication and a division. A straight-forward implementation of these operations would be inefficient, since they are considered expensive in terms of recourses. For this reason, equation (1) is implemented in an incremental way.

$$Stretching(x) = Stretching(x-1) + \frac{255}{Y_{\max} - Y_{\min}} \forall x \in [Y_{\min}, Y_{\max}] \tag{17}$$

The idea behind this alternative approach is that the stretching transformation divides the luminance channel into equal increments whose size is determined by the incremental factor $255/(Y_{max}–Y_{min})$. Consequently, every stretched value differs from the previous and from the next by the incremental factor. This means that a LUT could store all the possible incremental factors and feed the appropriate to an accumulator, which would calculate the stretched luminance values. This is depicted in Figure 10a.

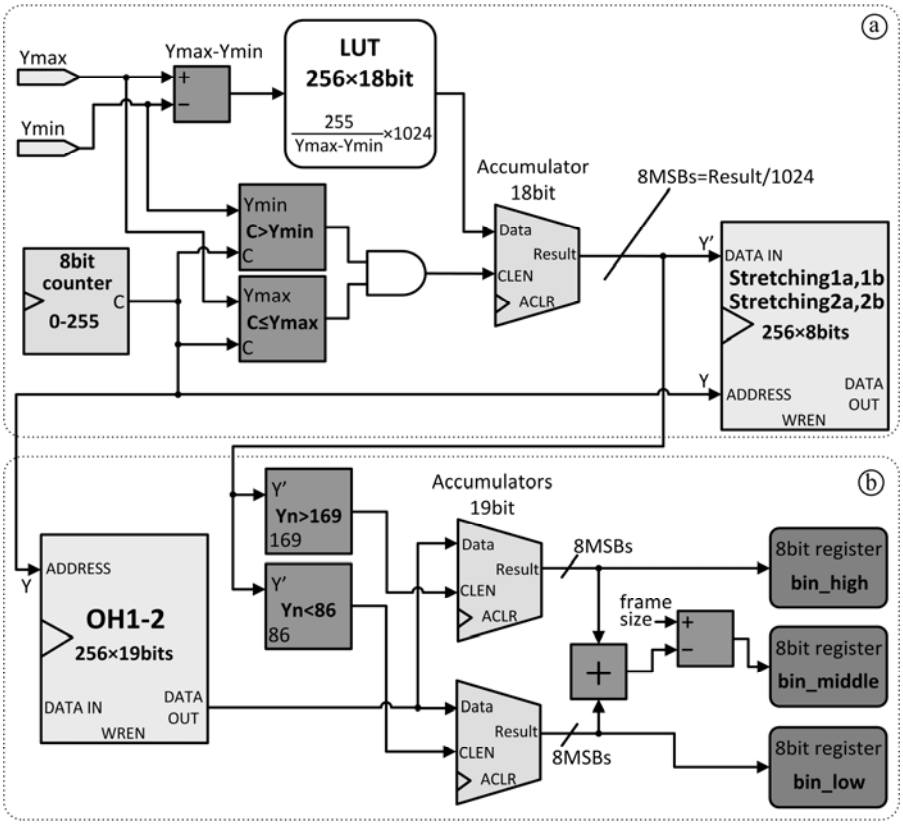


Fig. 10. a. Calculation of the StretchingLUTs. b. Calculation of the 3-bin histogram.

Similarly to the previous subsection, in order to avoid floating point arithmetic and maintain the accuracy of the calculations, the LUT stores the incremental factors multiplied by 1024. The final result is calculated by 10 right shifts (division by 1024). This is depicted in the following equations.

$$accumulator(x) = accumulator(x-1) + \frac{255 \times 1024}{Y_{\max} - Y_{\min}} \forall x \in [Y_{\min}, Y_{\max}] \quad \text{with } accumulator(Y_{\min}) = 0 \tag{18}$$

$$StretchingLUTs(x) = \frac{accumulator(x)}{1024} \tag{19}$$

4.3 Global Parameter Calculation

The global parameters, which are necessary for the main enhancement function, depend upon a 3-bin histogram of the stretched luminance component, as equations (8)-(13) indicate. Figure 10b depicts their implementation. The stretched luminance value Y' is driven into two comparators, while the original luminance value Y is driven into the

address bus of the appropriate original histogram (OH1 for odd frames or OH2 for even frames). The data output of the original histogram is sent to two accumulators. Depending on the output of the two comparators, the corresponding accumulator is activated and sums the number of pixels having the current Y' luminance value. When all the luminance values of the interval [Y_{min}, Y_{max}] have been processed, the three registers, that are depicted in Figure 10b, will store the number of pixels of the stretched luminance component that belong to the high bin (light tones), the middle bin (mid-tones) and the low bin (dark tones). Despite the fact that the accumulators are 19-bit wide, only the 8 most significant bits are registered, since high precision is not vital for global image statistics.

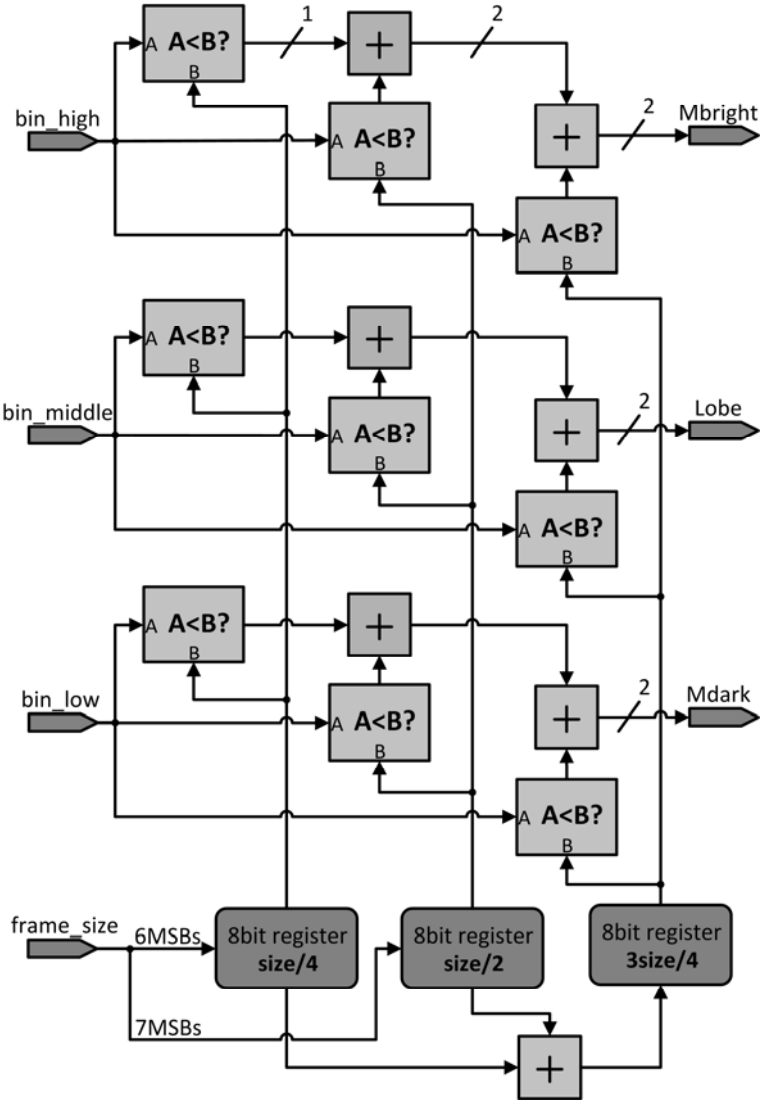


Fig. 11. Implementation of the global parameters

Figure 11 depicts the proposed implementation for the calculation of the global parameters. The 3-bin histogram is correlated with the desired values of the global parameters. These are 2-bit parameters which can have four different values. Their values are inversely proportional to the percentage of image pixels that a particular bin has. If for example *bin_low* occupies more than ¾ of the image size, the value of M_{dark} will be 00. On the contrary, if *bin_low* occupies less than ¼ of the image size, M_{dark} will be 11. The same associations hold for *bin_middle* with the *Lobe* parameter and *bin_high* with M_{bright}.

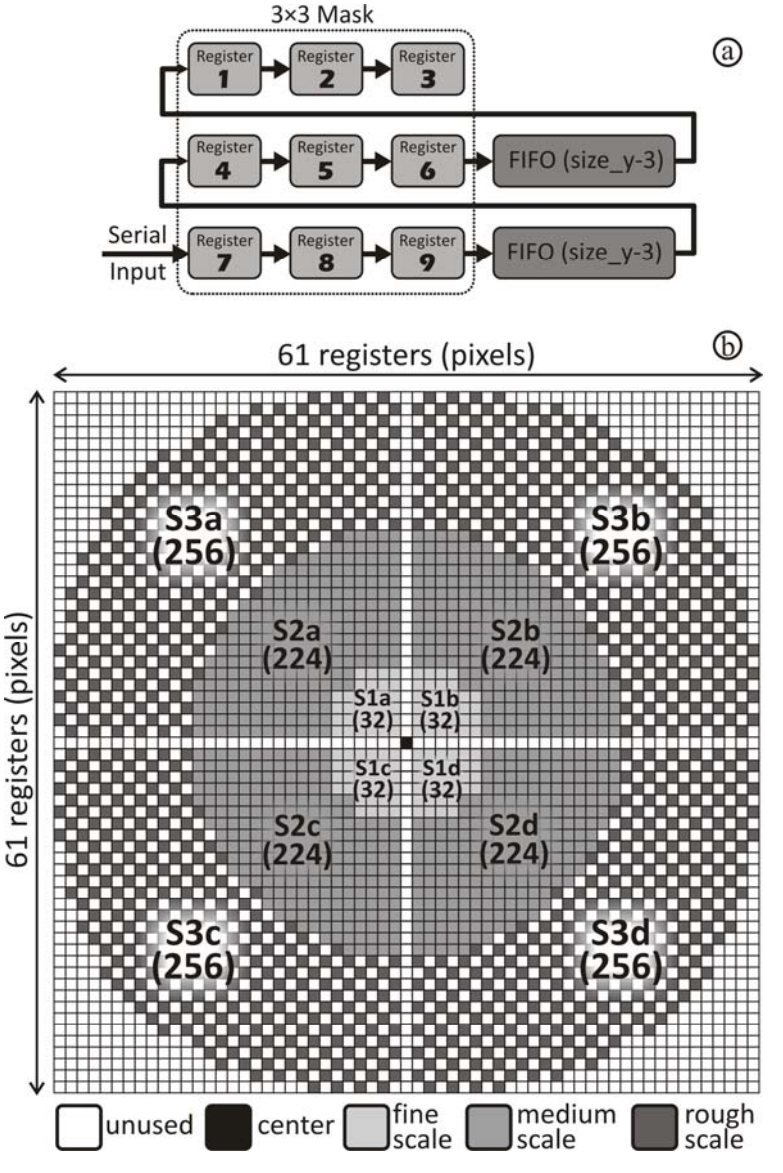


Fig. 12. a. A 3×3 serpentine architecture. b. The proposed serpentine architecture.

4.4 Surround Calculation

The average surrounding luminance of every pixel is a key local parameter of the main enhancement function. Its calculation introduces a considerable challenge: the surround calculation is principally a parallel operation, since many pixel values have to be averaged instantly, while the pixel values are fed to the system sequentially from the camera. For this reason, a serpentine memory architecture [8] is employed, allowing parallel access to many pixel values. Figure 12a depicts a 3×3 serpentine architecture.

The original enhancement algorithm requires large surround sizes of three different spatial scales. In order to implement this, a wider serpentine architecture is employed. The whole layout of the serpentine memory consists of 61×61, 24-bit registers and (60 × *py*)-61 wide FIFOs, (where *py* is the width of the picture). Five different types of registers are located in the proposed mask. At every clock cycle the central register contains the original luminance value of the pixel to be enhanced. The registers that appear white in Figure12b are not used in the computation, while the three different concentric areas of registers S_1, S_2, S_3 represent the three scales of surrounding neighborhoods. The number of registers in each spatial scale has been carefully selected in order to be a power of 2. This bypasses the expensive divisions, which are required for averaging, by using right shifts in the final result.

$$S_1 = S_1 a + S_1 b + S_1 c + S_1 d = 128\, pixels \tag{20}$$

$$S_2 = S_1 + S_2 a + S_2 b + S_2 c + S_2 d = 1024\, pixels \tag{21}$$

$$S_3 = S_2 + S_3 a + S_3 b + S_3 c + S_3 d = 2048\, pixels \tag{22}$$

$$Surround_1 = \frac{S_1}{128} \quad 7 \text{ right shifts} \tag{23}$$

$$Surround_2 = \frac{S_2}{1024} \quad 10 \text{ right shifts} \tag{24}$$

$$Surround_3 = \frac{S_3}{2048} \quad 11 \text{ right shifts} \tag{25}$$

The third scale (S_3) is different than the other two. Some of the registers participating in the computation are symmetrically scattered, in order to minimize the number of needed registers and cover a wider area of pixels. In order to sum at the same time the registers of the serpentine, large parallel adders are employed as Figure 13 depicts.

The final multiscale surround value is obtained by averaging the three surround values as follows:

$$S = \frac{2 \times S_1 + S_2 + S_3}{4} \tag{26}$$

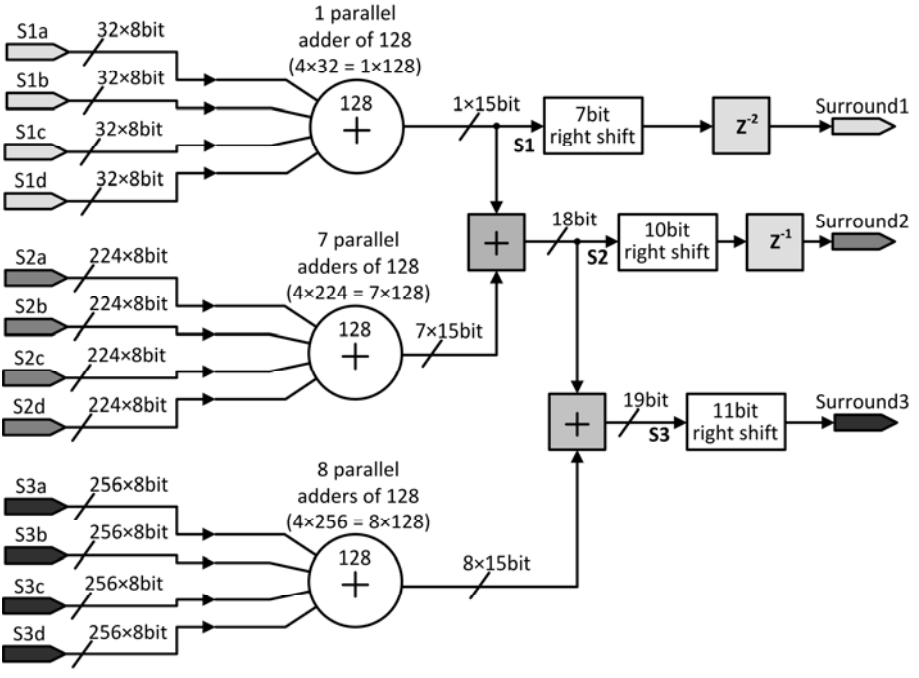


Fig. 13. Parallel summation of the serpentine registers

For every pixel (i,j) its luminance value Y_{ij} and its multiscale surround value S_{ij} are fed to the *StretchingLUT* modules, in order to get the stretched values. This is depicted in Figure 14.

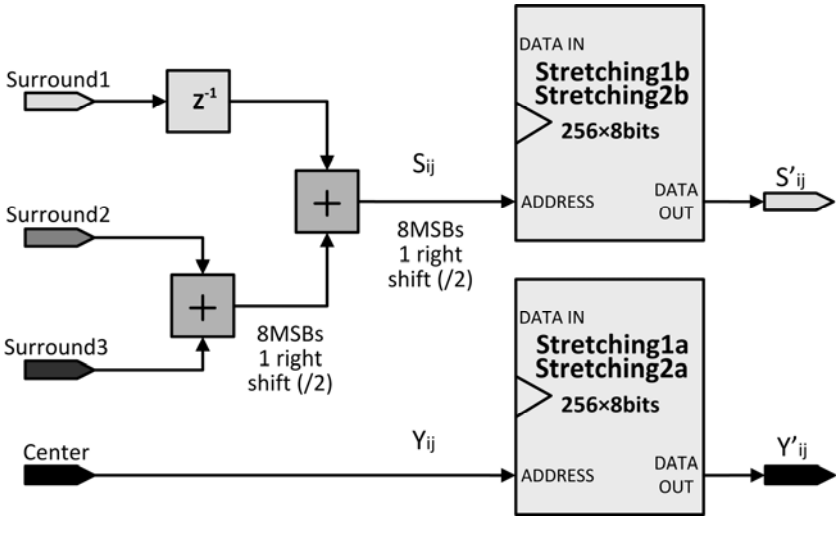


Fig. 14. Calculation of the local parameters

4.5 Enhancement Function

The enhancement function, as described by the equations (2)-(5), comprises several divisions and multiplications, which are expensive in terms of resources. In order to bypass the use of dividers and multiplicators, the enhancement function is stored into a LUT. This *FinalLUT* module, shown in Figure 6, is a ROM which is addressed with the four parameters of the main enhancement function and outputs at every clock cycle the equations' result. Figure 15 depicts the addressing of the *FinalLUT* module.

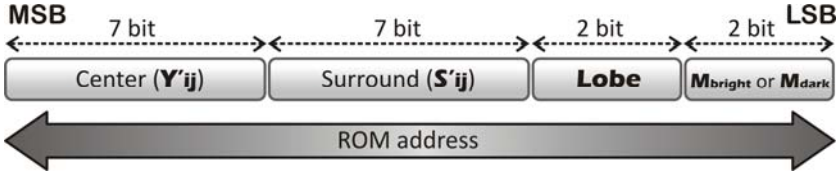


Fig. 15. Addressing of the *FinalLUT*

A reduction of the precision of the local parameters (7bits instead of 8bits) is introduced, in order to maintain the size of the *FinalLUT* within the ranges of current FPGA technology. The size of the ROM memory used is $2^{18}\times8$ bits = 256KB. As it will be shown later, this precision reduction does not affect the image quality of the final output of the system. Apart from resource efficiency, the use of the LUT allows rapid future improvements to the system. This can be done by simply changing the data of the LUT, instead of redesigning the whole system.

4.6 Alternative Implementation

As the frame sizes increase, the FPGA's memory resources become inadequate, and the FIFO memory that was used to create the second scan must be relocated outside the FPGA, as an external memory. An external memory is a costly component to include in a hardware implementation. Furthermore, it increases considerably the complexity of the system.

The only reason for the existence of the FIFO memory is the requirement to find the minimum and maximum luminance values of every frame. In a video sequence however, adjacent frames usually present small differences, since not much can change in 1/25 of a second. In fact, the global statistics of the frames remain practically unchanged. Taking this into consideration, the first stage of the hardware implementation described in the previous sections can be omitted. This can be done by using the global parameters M_{dark}, M_{bright} and Lobe and the *StretchingLUTs* of the previous frame. The pipelining structure of this alternative architecture is depicted in Figure 16.

As it is shown in Figure 16 only one scan for every frame is needed. Frame k enters the FPGA and is enhanced using the *StretchingLUT* and the global parameters (M_{dark}, M_{bright} and *Lobe*) of frame k-2. At the same time, its global statistics (histogram, Y_{max} and Y_{min}) and the parameters of frame k-1 (*StretchingLUT*, M_{dark}, M_{bright} and *Lobe*) are computed, in order to be used for the enhancement of the next frame. This implementation presents the highest performance in terms of execution time and frame size.

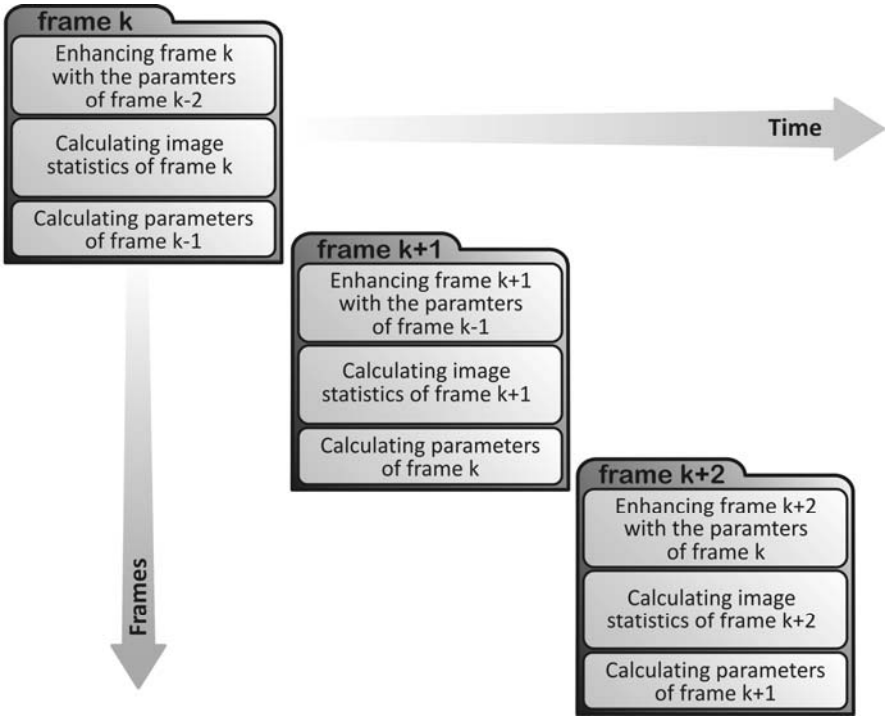


Fig. 16. The pipelining procedure of the alternative implementation

5 Hardware and Software Comparison

The accuracy reduction of the local parameters inevitably induces errors, compared to the software implementation of the algorithm. Figure 17 depicts the results of an error

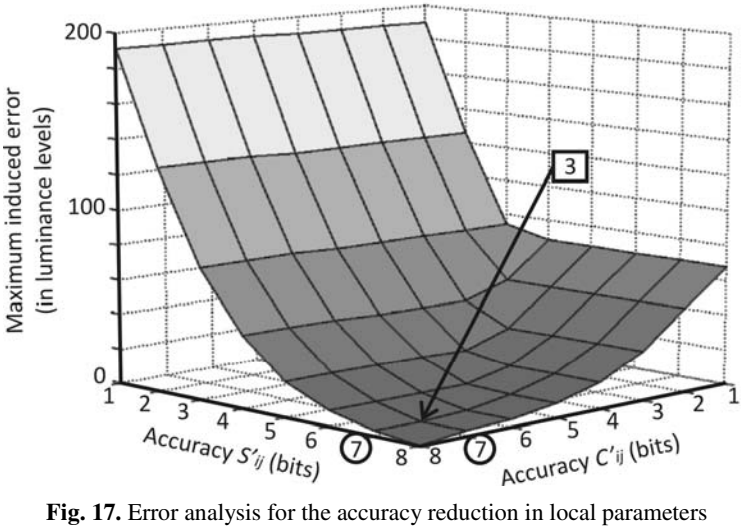


Fig. 17. Error analysis for the accuracy reduction in local parameters

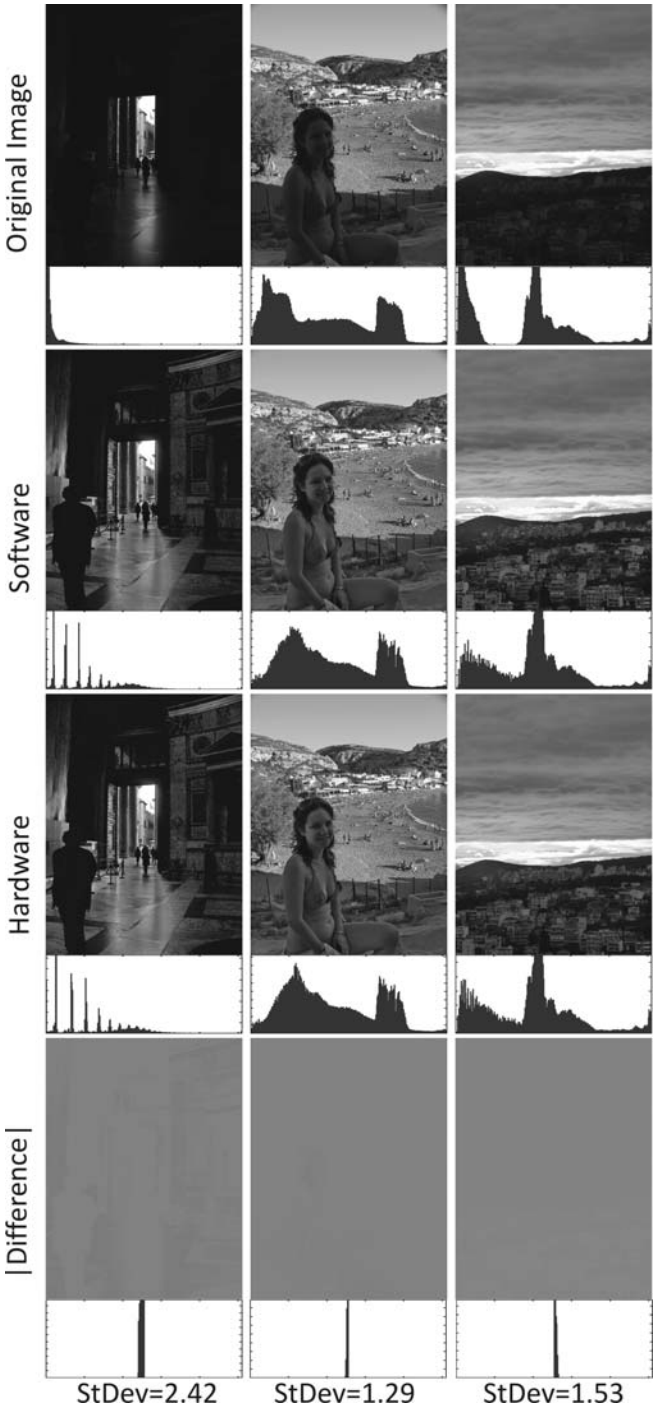


Fig. 18. Comparison between hardware and software

analysis for the accuracy reduction of the local parameters. This analysis shows that the maximum induced error can only be of three gray levels. A human observer is not capable of discriminating such errors, especially in video sequences. This conclusion is also confirmed in Figure 18, which depicts a visual comparison between the two results, as well as the standard deviation of their absolute difference.

6 Discussion and Conclusions

The proposed hardware architectures were synthesized in Altera's Stratix II GX: EP2SGX130GF1508C5 FPGA device. Table 1 depicts the simulation results from Altera Quartus II 5.1 CAD tool.

Table 1. Simulation results in Altera Quartus II 5.1 CAD tool

Altera Stratix II Simulation Results			
	Original Implementation		**Alternative Implementation**
	Color (24bit)	Grayscale (8bit)	Color (24bit)
Frame size	400×400 pixels	640×480 pixels	2.5 MPixels
Frame rate	25 fps	25 fps	25 fps
Total ALUTs	50,037/106,032 (47%)	44,034/106,032 (41%)	49,763/106,032 (47%)
Total registers	43,873	43,873	43,793
Total memory bits	5,492,736/ 6,747,840 (81%)	5,099,522/ 6,747,840 (75%)	2,609,151/ 6,747,840 (39%)
Total Combinational Functions	19,405	18,476	18,841
DSP Blocks	0	0	0
Maximum Frequency	66.66 MHz		

For both implementations, the maximum frequency of the system is 66.66MHz. This frequency is determined by the slowest module of the system, which are the parallel adders in the surround calculation. The original implementation allows the real-time rendering of color frames with size 400×400 pixels, or grayscale frames with size 640×480 pixels. The alternative implementation on the contrary, allows the real-time

processing of color frames with sizes up to 2.5MPixels. Both implementations outperform all the similar existing systems.

The above characteristics of the proposed implementations, allows the system to have many potential applications. Such applications are consumer electronics (e.g., digital cameras, mobile phones, video-call systems, and video surveillance systems), robotics (machine vision, assembly lines), driver's assistance (automotive), aerial/ satellite photography and medical imaging.

References

1. Battiato, S., Castorina, A., Mancuso, M.: High Dynamic Range Imaging for Digital Still Camera: an overview. Journal of Electronic Imaging 12, 459–469 (2003)
2. Jobson, D., Rahman, Z., Woodell, G.: A Multi-scale Retinex for Bridging the Gap between Color Images and the Human Observation of Scenes. IEEE Transactions Image Processing 6, 965–976 (1997)
3. Kimmel, R., Elad, M., Shaked, D., Keshet, R., Sobel, I.: A Variational Framework for Retinex. International Journal of Computer Vision 52, 7–23 (2003)
4. Hines, G., Rahman, Z., Jobson, D., Woodell, G.: DSP Implementation of the Retinex Image Enhancement Algorithm. In: Proceedings of the SPIE 5438: Visual Information Processing XIII, pp. 13–24 (2004)
5. Hines, G., Rahman, Z., Jobson, D., Woodell, G., Harrah, S.: Real-time Enhanced Vision System. In: Proceedings of the SPIE 5802: Enhanced and Synthetic Vision, pp. 127–134 (2005)
6. Seponara, S., Fanucci, L., Marsi, S., Ramponi, G.: Algorithmic and Architectural Design for Real-time and Power-efficient Retinex Image/Video Processing. Journal of Real-Time Image Processing 1, 267–283 (2007)
7. Vonikakis, V., Andreadis, I., Gasteratos, A.: Fast Centre-surround Contrast Modification. IET Image Processing 2(1), 19–34 (2008)
8. Benedetti, A.: Image Convolution on FPGAs: the Implementation of a Multi-FPGA FIFO structure. EUROMICRO 1, 123–130 (1998)

A Lifting-Based Discrete Wavelet Transform and Discrete Wavelet Packet Processor with Support for Higher Order Wavelet Filters

Andre Guntoro and Manfred Glesner

Department of Electrical Engineering and Information Technology,
Institute of Microelectronic Systems,
Technische Universität Darmstadt,
Karlstr. 15, 64283 Darmstadt, Germany
{guntoro,glesner}@mes.tu-darmstadt.de

Abstract. The major challenge in the wavelet transforms is that there exist different classes of wavelet filters for different kinds of applications. In this chapter, we propose a generalized lifting-based wavelet processor that can perform various forward and inverse Discrete Wavelet Transforms (DWTs) and Discrete Wavelet Packets (DWPs) that also supports higher order wavelet filters. Our architecture is based on Processing Elements (PEs) which can perform either prediction or update on a continuous data stream in every two clock cycles. We also consider the normalization step which takes place at the end of the forward DWT/DWP or at the beginning of the inverse DWT/DWP. Because different applications require different number of samples for the transforms, we propose a flexible memory size that can be implemented in the design. To cope with different wavelet filters, we feature a multi-context configuration to select among various forward and inverse DWTs/DWPs. For the 16-bit implementation, the estimated area of the proposed wavelet processor with 8 PEs configuration and 2×2×512 words memory in a 0.18-μm technology is 2.5 mm square and the estimated operating frequency is 319 MHz.

1 Introduction

For the last two decades the wavelet theory has been studied extensively [4, 7, 11, 17, 19] to answer the demand for better and more appropriate functions to represent signals than the ones offered by the Fourier analysis. Contrary to the Fourier analysis, which decomposes signals into sine and cosine functions, wavelets study each component of the signal on different resolutions and scales. In analogy, if we observe the signal with a large *window*, we will get a coarse feature of the signal, and if we observe the signal with a small *window*, we will extract the details of the signal.

One of the most attractive features that wavelet transforms provide is their capability to analyze the signals which contain sharp spikes and discontinuities. The better energy compacting support the wavelet transforms offer and also the

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 154–173, 2010.

localizing feature [5] of the signal in both time and frequency domains these transforms support have made wavelet outperforms the Fourier transform in signal processing and has made itself into the new standard of JPEG2000 [9,15].

Along with recent trends and research focuses in applying wavelets in image processing, the application of wavelets is essentially not only limited to this area. The benefits of wavelets have been studied by many scientists from different fields such as mathematics, physics, and electrical engineering. In the field of electrical engineering wavelets have been known with the name multi-rate signal processing. Due to numerous interchanging fields, wavelets have been used in many applications such as image compression, feature detection, seismic geology, human vision, etc.

Contrary to the Fourier transform, which uses one basis function (and its inverse) to transform between domains, there are different classes of wavelet kernels which can be applied on the signal depending on the application. Because different applications require different treatments, researchers have tried to cope with their own issues and implemented only a subset of wavelets which are suitable for their own needs such as ones that can be found in image compression [6,10,15,22] and speech processing [1,8,14,16]. The power of wavelet tools is then limited due to these approaches.

In this chapter we propose a novel architecture to compute forward and inverse transforms of numerous DWTs (Discrete Wavelet Transforms) and also DWPs (Discrete Wavelet Packets) based on their lifting scheme representations. Most lifting-based wavelet processors are dedicated to compute wavelet filters which are used only in JPEG2000 image compression where the wavelet coefficients can be represented as integers such as Andra in [2] which required two adders, one multiplier, and one shifter on each row and column processor to compute (5,3) and (9,7) filters with the prerequisite that prediction or update constants of the actual and the delayed samples are equal (i.e. $c(1 + z^{-1})$). Barua in [3] described the similar architecture for FPGAs that optimizes the internal memory usage. Dillen in [13] detailed the combined architecture of (5,3) and (9,7) filters for JPEG2000. Another example is from Martina, which encompassed multiple MAC structure with recursive architecture in [18].

Our new proposed architecture takes into account that each lifting step representation of an arbitrary wavelet filter may have two different update constants and the Laurent polynomial may have higher order factors (i.e. $c_1 z^{-p} + c_2 z^{-q}$), which are common in various classes of wavelet filters such as Symlet and Coiflet wavelet filters. Additionally, the proposed architecture also considers the normalization step which takes place at the end of the forward DWT/DWP or at the beginning of the inverse DWT/DWP for the applications that require to conserve the energy during the transform. In order to be flexible, the proposed architecture provides a multi-context configuration to choose between various forward and inverse DWTs/DWPs. Because wavelet transforms work with large number of samples, the proposed architecture can be configured to have an arbitrary memory size (i.e. the powers of two) to cope with the application demands.

The rest of the chapter is organized as follows. Section 2.1 describes the second generation of wavelets and the concepts regarding wavelet transforms and wavelet packets. The proposed architecture, including the processing element, the MAC-unit, the configuration and the context switch, the memory, the controller, are explained in Section 3. Section 4 discusses the performance of the proposed architecture and finally Section 5 concludes the contribution.

2 Backgrounds

2.1 Lifting Scheme

Contrary to the filter approach, which separates the signal into low and high frequency parts and performs the decimation on both signals afterwards, the second generation of wavelets reduces the computation by performing the decimation in advance. The second generation of wavelets, more popular under the name of lifting scheme, was introduced by Sweldens [21]. The basic principle of lifting scheme is to factorize the wavelet filter into alternating upper and lower triangular 2×2 matrix.

Let $H(z)$ and $G(z)$ be a pair of low-pass and high-pass wavelet filters:

$$H(z) = \sum_{n=k_l}^{k_h} h_n z^{-n} \tag{1}$$

$$G(z) = \sum_{n=k_l}^{k_h} g_n z^{-n} \tag{2}$$

where h_n and g_n are the corresponding filter coefficients. $N = |k_h - k_l| + 1$ is the filter length and the corresponding Laurent polynomial degree is given by $h = N - 1$. By splitting the filter coefficients into even and odd parts, the filters can be rewritten as:

$$H(z) = H_e(z^2) + z^{-1} H_o(z^2) \tag{3}$$

$$G(z) = G_e(z^2) + z^{-1} G_o(z^2) \tag{4}$$

and the corresponding polyphase representation is:

$$P(z) = \begin{bmatrix} H_e(z) & G_e(z) \\ H_o(z) & G_o(z) \end{bmatrix} \tag{5}$$

Daubechies and Sweldens in [12,21] have shown that the polyphase representation can always be factored into lifting steps by using the Euclidean algorithm to find the greatest common divisors. Thus the polyphase representation becomes:

$$P(z) = \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \prod_{i=n}^{1} \begin{bmatrix} 1 & a_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b_i(z) & 1 \end{bmatrix} \tag{6}$$

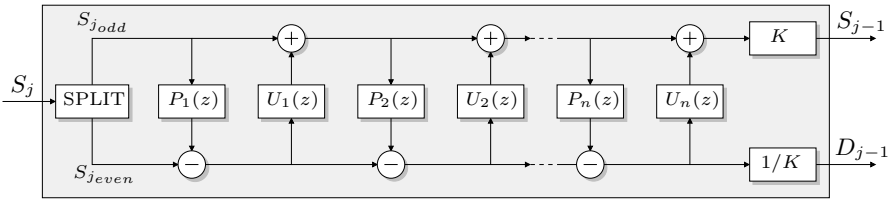


Fig. 1. Forward lifting steps

where $a_i(z)$ and $b_i(z)$ are the Laurent polynomials and K is the normalization factor.

Fig. 1 shows the arrangement of the lifting scheme representation. The Laurent polynomials $b_i(z)$ and $a_i(z)$ are expressed as predictor $P_i(z)$ and updater $U_i(z)$. The signal S_j is split into even and odd parts. Prediction and update steps occur alternately. The predictor $P_i(z)$ predicts the odd part from the even part. The difference between the odd part and the predicted part is computed and used by the updater $U_i(z)$ to update the even part. At the end, the low-pass and the high-pass signals are normalized with a factor of K and $1/K$ respectively.

By factoring the wavelet filters into lifting steps, it is expected that the computation performed on each stage (either it is a prediction or an update) will be much less complex. As an example, the famous Daub-4 wavelet filter with the low-pass filter response:

$$H(z) = \frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{3+\sqrt{3}}{4\sqrt{2}}z^{-1} + \frac{3-\sqrt{3}}{4\sqrt{2}}z^{-2} + \frac{1-\sqrt{3}}{4\sqrt{2}}z^{-3} \tag{7}$$

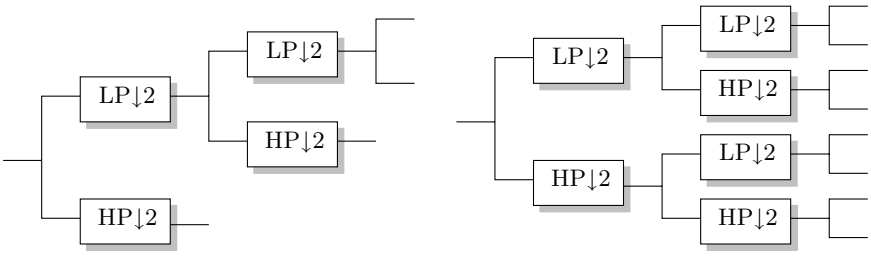
can be factored into lifting steps:

$$P(z) = \begin{bmatrix} \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}+1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & -z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} + \frac{2-\sqrt{3}}{4}z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{bmatrix} \tag{8}$$

Since the finding of the greatest common divisors is not necessarily unique, the result of the Laurent polynomials may also differ. The Daub-6 and the popular (5,3) and (9,7) wavelet filters can be factored into lifting steps with maximum degree of ± 1 [12] whereas Symlet-6 and Coiflet-2 (the lifting computations are not detailed here due to page limitation) may have two update/prediction terms and also $z^{\pm 5}$ factor on its Laurent polynomials.

2.2 Wavelet Transform and Wavelet Packet

Wavelet transform is a multi-resolution signal analysis. In the traditional wavelet transforms, only the low-pass signal is used on the next transformation level to generate a multi-resolution representation of the corresponding signal. In wavelet packets, both low-pass and high-pass signals are analyzed, resulting equally



(a) DWT (b) DWP

Fig. 2. Two different transformations

spaced frequency bands. Fig. 2 depicts both schemes. Note that the illustration uses wavelet transforms based on filter-approach instead of lifting-scheme in order to ease understanding the concept for both schemes. LP and HP correspond to low-pass and high-pass filter pair and ↓2 corresponds to down-sampling by two. It is obvious that DWT will require less computation time compared to DWP, because at each level, the number of samples is decreased by two. Also, the controller that controls the processor to perform DWTs and their inverses is straightforward, while the controller to perform DWPs and their inverses is more complicated due to the fact that the number of frequency bands that need to be processed increases two fold at each transform. As an example, performing four levels wavelet packet on a signal leads to 16 frequency bands whereas performing four levels wavelet transform generates 5 frequency bands.

Not only the challenges on the controller, the major issue in DWP is that the resulting HP signals are much smaller than the LP parts in normal circumstances. Thus performing multi-level DWP using integer arithmetics would make these HP signals go to zero, which lead to lower achievable SNR values, if it is not carefully performed.

3 Proposed Architecture

The lifting-based forward DWT/DWP splits the signal into even and odd parts at the first stage. The split signals are processed by an alternating series of predictors and updaters (on some wavelet filters, an updater may come before a predictor). On the final stage, the multiplication with the normalization factor takes place in order to conserve the energy. The inverse DWT/DWP performs exactly everything backwards. It starts with the multiplication with normalization factor, continues with a series of updaters and predictors, and finishes with the merging of the outputs.

As a predictor and an updater perform a similar computation, the hardware architecture for both functions is exactly the same. Taking this into account, we propose a novel wavelet processor which is based on M processing elements to cope with M lifting steps. Due to the nature of the lifting scheme, wavelet filters

that have longer lifting scheme representations can easily be broken down into smaller lifting steps that the processor can compute (i.e. M lifting steps each). Which means that the processor that implements M processing elements is not limited to perform the transform up to M lifting steps only.

The core behind our proposed architecture is the processing element (PE), which performs the prediction or the update. To maximize the performance, the PE utilizes the parallelism by using a pipeline mechanism to guarantee the outputs to be available in every clock cycle (actually every two clock cycles as detailed later). As the lifting scheme breaks a wavelet filter into smaller predictions and updates, the resulting predictor and updater can be limited to have a maximum Laurent polynomial degree of one. Nevertheless, the predictor or the updater of higher order wavelet filters may have the higher factors as well. Without loss of generality, we can formulate the predictor or the updater polynomial as:

$$l(z) = c_1 z^{-p} + c_2 z^{-q} \tag{9}$$

with c_1 and c_2 as the polynomial constants and $|p - q| \leq N$. This implies that on each stage (either as a predictor or an updater), the PE would perform two multiplications and two additions. As an example, the first predict and update steps of Daub-4 can be written as:

$$\begin{bmatrix} s' \\ d \end{bmatrix} = \begin{bmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ d \end{bmatrix} = \begin{bmatrix} s + d \cdot \sqrt{3} \\ d \end{bmatrix} \tag{10}$$

$$\begin{aligned} \begin{bmatrix} s' \\ d' \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} + \frac{2-\sqrt{3}}{4} z^{-1} & 1 \end{bmatrix} \begin{bmatrix} s' \\ d \end{bmatrix} \\ &= \begin{bmatrix} s' \\ d + s' \cdot \frac{-\sqrt{3}}{4} + s' \cdot \frac{2-\sqrt{3}}{4} z^{-1} \end{bmatrix} \end{aligned} \tag{11}$$

which perform one multiplication and one addition in order to solve s' (as shown at the top resulting term in Eq. 10) and two multiplications and two additions to solve d' (as shown at the bottom resulting term in Eq. 11).

3.1 Architecture of the Processing Element

Taking into account that multipliers are expensive in term of area and the PE receives two samples (s and d) at once, we have decided to lower the input rate by half. From the performance point of view, the processing rate of the PE will be equal to the processor speed and no longer twice as fast. This also implies that the bottleneck issues on the input and output ports with the memory will not occur. From the hardware implementation point of view, the PE requires only one multiplier and one adder. This optimization, as detailed later, is accomplished by multiplexing the operands of the multiplier inputs (the *multiplier* and the *multiplicand*) and by feeding the adder result back via the multiplexer.

Fig. 3 depicts the proposed PE. The PE has two selectors S1 and S2 to choose the prediction or the update samples that correspond to the factors p and q from the Laurent polynomial. Two constants which represent the filter coefficients are

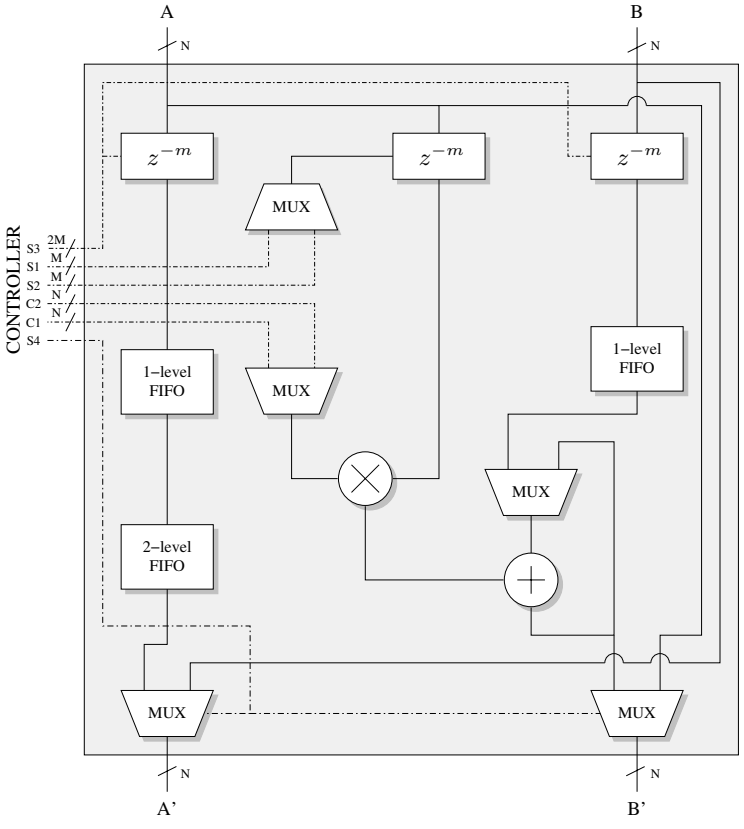


Fig. 3. Block diagram of the processing element

defined and configured by the controller. By delaying the actual samples, selector S3 controls the prediction or the update that requires future samples. Selector S4 is a bypass selector. Because lifting steps of the higher order wavelet filters may require distance prediction or update samples, the maximum depth of the unit delay z^{-m}, that determines the maximum delay level, can be freely chosen during the design.

Fig. 4 details the MAC (Multiply-and-Accumulate) unit which is implemented inside the PE. Both multiplier unit and adder unit require only one clock cycle to perform their function. C_1 and C_2 correspond to the Laurent polynomial constants, whereas M_1 and M_2 correspond to the outputs of the samples that are selected by S1 and S2. The multiplexer for M_1 and M_2 as a matter of fact does not exist and is drawn here only to illustrate the MAC process. A shifter is utilized as a replacement of the more expensive divider.

The PE is divided into 3 blocks. The first block organizes the input samples from both channels. The second block chooses the inputs of the multiplier and performs the multiplication. As mentioned earlier, the PE utilizes only one multiplier which is time-shared in order to perform two multiplications. The first

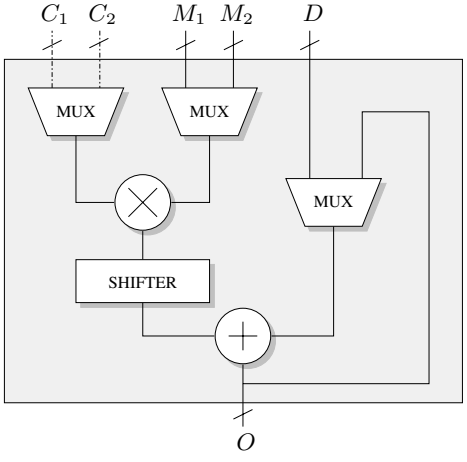


Fig. 4. Multiply-And-Accumulate unit

clock cycle performs the first multiplication (i.e. $C_1 \times M_1$) and the second cycle performs the second multiplication (i.e. $C_2 \times M_2$). The third block performs the summation between the reference sample and the prediction/update values. Similar technique is applied here in order to utilize only one adder. As shown in Fig. 4, the first addition cycle performs $D + 2^{-R}(C_1 \times M_1)$ and the second addition cycle adds-up the first one with $2^{-R}(C_2 \times M_2)$. Whilst the input data are integer, the shifter performs the division on the multiplication result with 2^R where R can be freely chosen. Two 1-level FIFOs (First In First Out) are implemented to deal with the multiplier delay and a 2-level FIFO is implemented to compensate the delay which is introduced by the adder.

3.2 Normalization

As the multiplication with the normalization factor can take place at the end of the transform in case of forward DWT/DWP or at the beginning of the transform in case of inverse DWT/DWP, two special processing elements to handle this function are required. Although the normalization step is different compared to the prediction or the update step in a manner that both inputs s and d are multiplied with constants K and $1/K$ respectively, we know for sure that two multiplications take place. To perform this normalization step, we extend the functionality of the PEs that are located on the top and on the bottom of the proposed wavelet processor instead of implementing a dedicated normalizer unit. Three additional multiplexers are needed to add the normalization factor unit into the PE. Fig. 5 shows the PE which is used on the top and on the bottom of the proposed architecture. By enabling S5 and setting S1 and S3 to zero, two inputs of the multiplexer before the multiplier correspond to the actual samples s and d (with the normalization factors $K = C_1$ and $1/K = C_2$). The first multiplication product passes through the multiplexer and the 1-level FIFO

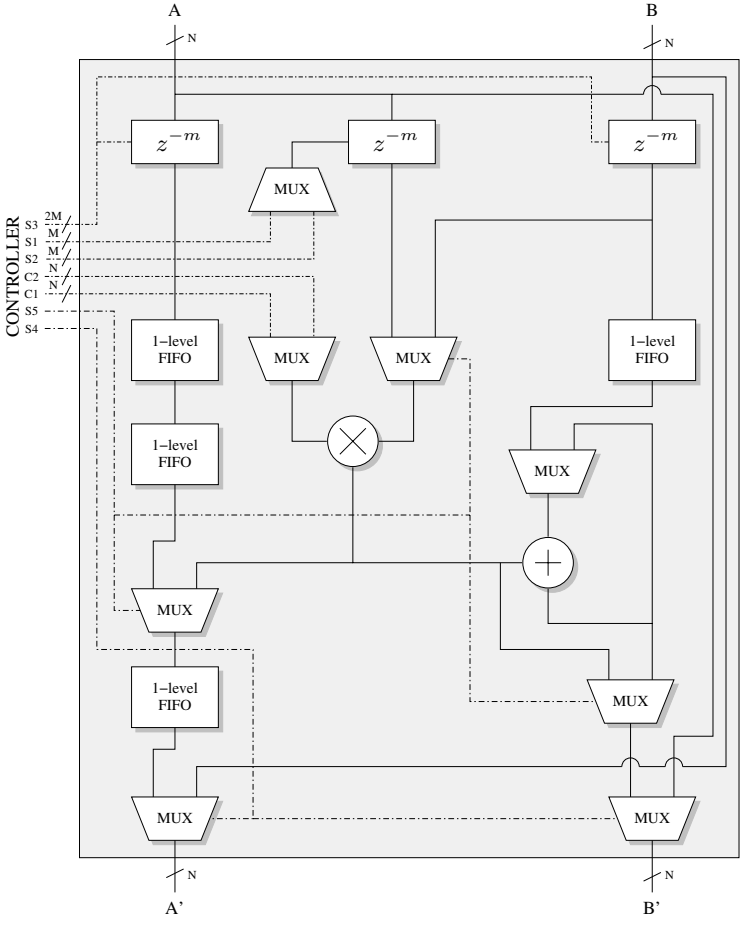


Fig. 5. Block diagram of the processing element which is located on the top and on the bottom of wavelet processor

resulting $s' = Ks$ (the left side) and the second multiplication product passes through the multiplexer resulting $d' = d/K$ (the right side). Whereas the first normalization (i.e. $s' = Ks$) takes place first, instead of adding a 1-level FIFO on the right output port, the 2-level FIFO is split into two 1-level FIFOs to make both outputs synchronized and to minimize the latency.

3.3 Context Switch

To cope with various lifting-based forward and inverse DWTs/DWPs, we have separated the configuration dependent parameters from the PE. Figs. 3 and 5 show how the inputs of the multiplexer selectors and the multiplier constants are separately drawn on the left side of the figures to emphasize the separation.

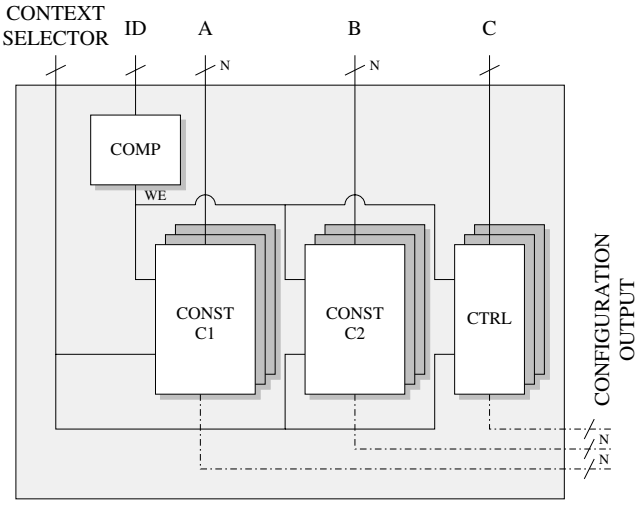


Fig. 6. Context Switch for the PE

In addition the PE is designed to be simple. Thus, no finite state machine is required to control the PE. To support different classes of wavelet filters that require different types of configurations, we have implemented a multi-context configuration on each PE as depicted in Fig. 6. Each PE is assigned with a row index as a unique ID for the configuration. Multiplier constants use the signal data paths to save the wiring cost whereas the multiplexers configuration requires additional controller path. Context switch is implemented as a memory module where the address is controlled by the context selector and the write enable signal is controlled by the output comparator.

The active configuration can easily be selected by using this context-based controller to cope with various wavelet filters. One benefit of having a multi-context configuration is that the proposed wavelet processor can be configured to perform the corresponding inverse DWTs/DWPs in a very simple manner. Additionally, the issues regarding the boundary condition can be relaxed by utilizing special wavelet filters on the signal boundaries which require less or no delayed/future samples (e.g. Haar wavelet) instead of exploiting the periodicity or the mirroring of the signal. Lastly, by using the context-based configuration, the DWTs/DWPs that exercise longer wavelet filters can simply be broken into smaller lifting steps. The configuration of each group of the lifting steps will be stored in the context memory and will be used to compute the transform.

3.4 Memory Controller

Taking into account that the predictions and the updates occur alternately, the outputs of a PE will be cross-linked with the input of the next PE. Due to the nature of lifting steps, the prediction and the update are computed *in-place*. It

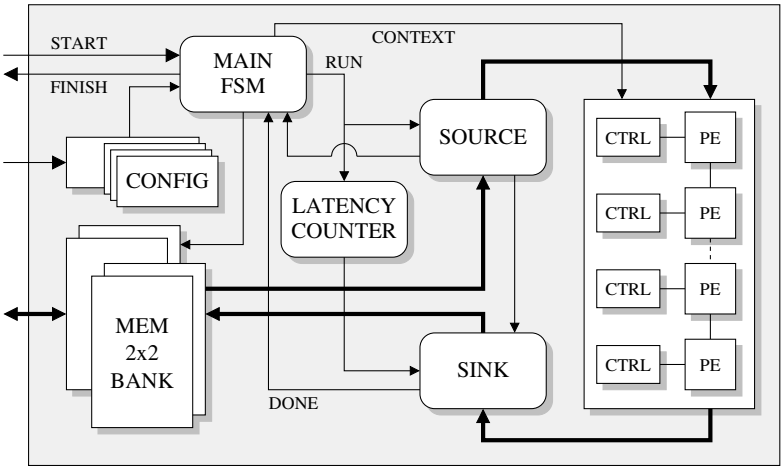


Fig. 7. The Proposed Wavelet Processor

means that it is not necessary to save the result or the temporary result into a different memory. One simple implementation of the proposed wavelet processor would consist of one PE. By configuring each context with the corresponding lifting step, the DWT/DWP and their inverses could be computed with this simple implementation. Although it is possible to use only one PE, a typical wavelet processor will have M chained PEs configuration to boost the performance and to minimize memory access.

Wavelet transform is a multi-resolutional signal processing tool. To achieve the required results, the signal needs to be transformed iteratively. In case of a DWT, only the low-pass part of the signal is taken into account as an input for the next transform. As a pair of low-pass and high-pass wavelet filter is used to compute the transform, the size of the signal decreases by two after each transformation level in this case. In contrary, a DWP uses both low-pass and high-pass parts of the signal in order to achieve equally spaced frequency bands after each transformation level. The total size of the signal on DWP remains the same and the amount of the processed data will slightly increase. It is due to the fact that low-pass and high-pass parts are treated independently during the computation and for each part of the signal, a signal extension, which will be detailed later on, is required to compute the transform on the boundary regions.

Fig. 7 depicts the block diagram of the processor along with the PEs and their configuration controller. The PEs that are located on the top and on the bottom of the wavelet processor have an extra capability to perform the normalization.

Main FSM

The main finite state machine controls the wavelet processor. When the transform is initiated, the FSM reads the necessary configurations, such as the transformation level, forward/inverse mode, transform/packet mode, used contexts,

etc. from the *config* block. This configuration, as detailed later, is divided into two categories. The first category is related to the functionalities of the processor and the second one is related to the lifting configuration.

The main FSM prepares the source and the sink addresses where the data will be read and stored, and also the length of the data needed to be processed. We exploit the periodicity extension to cope with the boundaries issue in order to compute the transform on those regions. This implies that source address does not always start on the top of the page. Address masking techniques are applied here to localize the page. The FSM takes care of the possibility of having a longer wavelet transform that has to be split into several lifting steps on the target PEs. The FSM allows multi-level forward/inverse DWT and DWP to take place by means of iteration process.

Config

The config block contains the configuration of the wavelet transform. Two different configuration categories are managed by this block. The functionality part manages:

- Selecting the type of the transform that will be performed: DWT or DWP.
- Selecting the transform mode: forward transform or inverse transform.
- The amount of memory that will be involved during the transform. Note that the processor can perform the transform on an arbitrary size of the sample. For an example, the value 0 indicates that the transform will be performed on the whole memory. The value 1 will make the transform processes the half of the memory and so on.
- Number of levels the transform will compute. This is effective to perform multi-level transform on a 1D signal. In contrary, for a 2D or higher dimension, the number of levels should always be set to 1.

The lifting part stores the configuration of the contexts used during the transform. It holds an important key to support wavelet transforms that use longer wavelet filters. If the number of lifting steps of the wavelet filters used for the transforms are larger than the available PEs, these lifting steps have to be split into several smaller steps that can be fit into the available PEs. The configuration of each lifting itself is stored on the context configuration of the PEs. This block stores only the corresponding context IDs that will be used. Thus, by selecting the right ID one after another, the wavelet transform with longer lifting steps can be performed. Basically, it tells us which context should be used for the corresponding lifting step.

Beside storing the context IDs, it also holds the read and write offset addresses to start the transform and also the latency value for each lifting. It is important to note that in order to compute the wavelet transform, except for Haar wavelet filter, past and future samples are required. This becomes an issue when the transform on the signal boundary is performed. To cope with this boundary issue, the periodicity extension is used to locate these samples. These offsets hold the information of the corresponding starting sample for this periodicity extension.

Memory

The memory is organized as 2×2 banks. This configuration describes that the processor has two main banks (which are called bank 0 and bank 1) and each main bank consists of one primary bank and one shadow bank. With this technique, while the processor performs the transform on one bank (either bank 0 or bank 1), the next data can be placed on the other bank. Thus, it improves the overall performance by minimizing the delay caused by the data preparation.

The memory write and read accesses are exclusive, which means that writing to the memory will write to the primary bank and reading from the memory will read from its shadow. This state is switchable automatically, controlled by the *FSM*. When the transform takes place, the *FSM* grants the memory access of the selected bank to the *source* and *sink* blocks. Writing to or reading from this bank is forbidden and it will generate an error (as an indication of a busy signal). Nevertheless, the external interface can still read from and write to the memory of the other non-selected bank. Thus, the previous resulting transform, which is stored in this non-selected bank, can be read, and also the external interface can prepare the new data for the next transform.

Source and Sink

These blocks generate and automatically increment the read and write addresses. The *source* reads data from the memory and transfers it to the PEs. The *sink* reads data from the PEs and writes it to the memory. A special case is considered when performing transformations that are longer than the available PEs. During the in-between transformation, in case of forward transform, the *sink* will write the data (which corresponds to the intermediate results) to the memory in adjacent manner (resulting L-H-L-H-...). During the final transformation, the *sink* writes the LP and the HP signals into two different pages (resulting L-L-...-H-H-...). The similar handling is also performed by the *source* when performing the inverse transform.

To access the correct page, two address masks are used. The first mask is responsible for the data indexing, and the second mask is responsible for the page indexing.

Latency Counter

This block delays the *run* signal from the *main FSM* to initiate the sink process. The delay amount is different for every lifting steps and it is defined in the *config* block.

Details of the Memory Access

Fig. 8 illustrates the N-level and multiple lifting steps DWT. White and grey represent the primary and the shadow banks and diagonal pattern represents the in-between transformation. During the setup, the data is prepared and stored in one bank (this bank is write-only and its shadow is read-only). When the

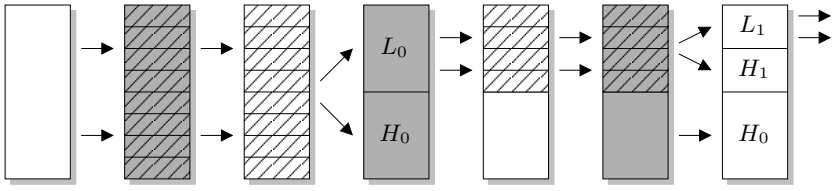


Fig. 8. Forward DWT Process

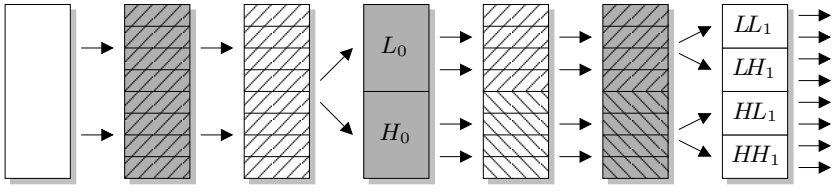


Fig. 9. Forward DWP Process

transformation is initiated, this state is reversed, and the source and the sink control the address lines. For each lifting steps, the source reads the written data, and the sink writes the in-between transformation result to the shadow bank. This state is reversed again every time one lifting step is finished, which makes the shadow bank as the primary bank and vice versa. During the last lifting step, the sink stores the LP and the HP results into two different pages. This whole process is performed N times with each iteration decreases the data by half. At each finishing level, a memory copy to transfer the previous HP result to the shadow bank is performed when necessary, e.g. when the lifting steps are odd.

For the DWPs, the HP signal is also transformed, as depicted in Fig. 9. Instead of executing/finalizing the transformation on each signal (LP, and then HP) on each level, the in-between transformations are performed on both signals. With this technique, the banks are not switched during the in-between transformation for both LP and HP signals. Thus, the *FSM* can trigger the *source* to initiate the next data transfer for the next band/page (e.g. HP) without waiting the *sink* to finish from the previous transform. This solution decreases the data preparation time that is caused by exploiting the periodicity extension and the PEs latency. No copy transfer is performed on the DWPs/IDWPs.

4 Results and Performances

Our wavelet processor is written in VHDL and is based on the modular and parametric approach to make the design adaptable. In this paper, we provide the synthesis results of our wavelet processor that contains 8 PEs to process forward/inverse DWTs/DWPs with 8-level unit delays to support higher order

Table 1. Estimated area and frequency of proposed wavelet processor with 8 PEs and 2×2 memory banks

Data Width	Est. Area (in mm^2)	% Area for Logic	Est. Frequency (in MHz)
16-bit	2.501	30.60 %	319.49
20-bit	3.120	31.38 %	298.51
24-bit	3.780	32.63 %	285.71
28-bit	4.376	32.57 %	262.47
32-bit	5.134	34.63 %	241.55

Table 2. Comparison with other Lifting-Based Architectures

Arch.	Speed	Area	Filter	Transform	Data Width	Mem. Size
Andra [2]	200 MHz (0.18-μm)	2.8 mm^2	(5,3) (9,7)	DWT IDWT	16-bit	128
Dillen [13]	110 MHz (FPGA)	–	(5,3) (9,7)	DWT IDWT	16-bit	256
Seo [20]	150 MHz (0.35-μm)	5.6 mm^2	(5,3) (9,7)	DWT IDWT	12-bit	512
Wang [23]	100 MHz (0.18-μm)	1.1 mm^2	Daub-4	DWP	18-bit	8
Ours	242 MHz (0.18-μm)	5.1 mm^2	Arbitary	DWT,IDWT DWP,IDWP	32-bit* Configurable	512* Configurable

wavelet filters and 16 available contexts to configure 16 different transforms. The design is synthesized using 0.18-μm technology. Because wavelet transforms deal with large numbers of samples, 2×2×512 words memory is integrated into the processor for this implementation. Note that the wavelet processor is also designed to be flexible in respect with the number of the samples the processor can handle. In other word, the processor can be synthesized with an arbitrary size of the memory, as long as it follows an integer power of two rule. The size of the memory corresponds to the maximum number of samples the wavelet transforms can be performed by the processor.

The estimated area and frequency of various data width implementations are reported in Table 1. For the 16-bit configuration, the proposed wavelet processor consumes 2.5 mm^2 chip area and has a maximum operating speed of 319 MHz. As a comparison, architecture from Andra with 16-bit data width in [2] can only compute (5,3) and (9,7) filters and required 2.8 mm^2 with 200 MHz operating frequency. The details of the comparisons with the other architectures are summarized in Table 2. Note that our proposed architecture has flexible data width and memory size.

Table 3. Lifting coefficients of Daub-6, Symlet-6, and Coiflet-2 wavelet filters

Type	Daub-6	Symlet-6	Coiflet-2
Updater	2.425 z^0	-0.227 z^0	-2.530 z^0
Predictor	0.079 z^{-1} -0.352 z^0	-1.267 z^{-1} 0.216 z^0	-0.240 z^{-1} 0.342 z^0
Updater	-2.895 z^1 0.561 z^2	0.505 z^1 -4.255 z^2	3.163 z^1 15.268 z^2
Predictor	-0.020 z^{-2}	0.045 z^{-3} 0.233 z^{-2}	0.006 z^{-3} -0.065 z^{-2}
Updater		-18.389 z^3 6.624 z^4	-63.951 z^3 13.591 z^4
Predictor		0.144 z^{-5} -0.057 z^{-4}	0.001 z^{-5} 0.002 z^{-4}
Updater		-5.512 z^5	-3.793 z^5
Normalizer	0.432 2.315	-0.599 -1.671	0.108 9.288

Table 4. SNR values of different data width implementations (in dB) for 4-level forward and inverse DWT

	Daub-6				
Source	16-bit	20-bit	24-bit	28-bit	32-bit
Sinusoid	42.90	67.04	89.38	115.00	138.52
Sawtooth	40.93	65.19	88.34	113.31	137.03
Step	44.98	67.07	87.95	114.19	138.88
Random	40.17	64.92	88.62	113.06	136.87
	Symlet-6				
Sinusoid	37.04	61.95	88.40	111.85	134.88
Sawtooth	35.75	60.22	85.84	108.89	133.17
Step	34.97	64.94	91.83	112.53	140.07
Random	36.52	61.18	85.93	109.37	133.51
	Coiflet-2				
Sinusoid	31.35	55.13	78.56	101.70	124.05
Sawtooth	29.80	52.85	76.83	100.13	123.19
Step	31.86	56.75	79.89	101.45	123.60
Random	29.01	52.83	77.53	101.93	125.27

In order to realize the fixed-point multiplication between the samples and the coefficients, we utilized an integer multiplier and a shifter to reduce the hardware cost. As the compensation, this implementation leads to errors caused by the rounding of the wavelet coefficients and the cropping of the multiplication results. To measure the level of correctness of our design, we perform DWTs/DWPs and their corresponding inverse transforms on some predefined signals. Four different 8-bit full-swing signals, which are used as references, are forward and inverse transformed using Daub-4, Symlet-6, and Coiflet-2 wavelet filters with no integer coefficients. The random signal has a uniform distribution.

Table 5. SNR values of different data width implementations (in dB) for 4-level forward and inverse DWP

	Daub-6				
Source	16-bit	20-bit	24-bit	28-bit	32-bit
Sinusoid	39.66	63.92	87.49	111.65	136.02
Sawtooth	37.45	62.05	85.26	109.80	134.00
Step	41.11	63.85	86.79	112.82	137.83
Random	37.19	61.75	84.11	109.34	133.41
	Symlet-6				
Sinusoid	35.41	60.03	85.22	108.95	131.86
Sawtooth	33.79	58.24	82.98	106.96	130.10
Step	34.25	62.31	87.14	108.17	134.37
Random	33.35	58.40	82.67	106.87	129.71
	Coiflet-2				
Sinusoid	29.26	53.07	76.74	100.13	123.01
Sawtooth	27.09	51.00	74.44	98.75	121.57
Step	29.49	53.75	76.65	98.84	122.46
Random	26.75	51.33	74.64	98.49	120.88

The lifting step coefficients of these wavelet filters are summarized in Table 3. These coefficients are shortened to save space. Because the coefficients have to be represented as integers, depending on the data width, they will be magnified with some factor, and the result will be rounded and used as lifting coefficients. ModelSim is used to compare and verify the results. The SNR is computed using:

$$SNR_{(dB)} = 20 \times log_{10}\left(\frac{\sum |signal|}{\sum |signal - result|}\right) \tag{12}$$

where *signal* corresponds the input vector and *result* corresponds the output of the forward and inverse transforms.

Because wavelet transform is a multi-resolution signal processing tool, we perform four-level DWTs and DWPs to give a better overview of the performance of our wavelet processor. The SNR values of the different data width implementations for 4-level DWTs and DWPs are reported in Table 4 and Table 5 respectively. Depending on the data widths, SNR values vary between 29 dB and 140 dB in case of DWTs and between 27 dB and 138 dB in case of DWPs, which are sufficient for most applications. DWPs achieve slightly lower SNR values due to the fact that the high-pass signals after each transformation level get smaller and tend toward zero. Thus information losses are affected at these bands. The 16-bit implementation achieves lower SNR values due to the fact that the lifting coefficients have a large dynamic range that is between 0.001 and 64. The same

reason applies for Coiflet-2 wavelet filter. The improvement of the SNR values can be achieved by increasing the data width.

The proposed wavelet processor can accept input data stream and perform the computation in every two clock cycles made possible by the pipeline structure and the resource sharing. The total latency on each PE is 4 clock cycles. One clock cycle is consumed by the input registers, 1+1 by the multiplier (two multiplications are performed), and 1+0 by the adder (two summations are performed where one cycle is "stolen" from the multiplier). Additional sample latency (2 clock cycles per future sample) will add-up to the total latency on the PEs which require this feature. The PE that is configured to perform the normalization step has latency of 3 clock cycles.

For the wavelet processor with M PEs, the total time needed to compute L-stage forward/inverse DWT is:

$$T_{DWT} = L(T_s + T_d) + 2S(1 - 0.5^L) + S(1 - 0.5^{L-1}) \tag{13}$$

where S is the signal length, T_s is the setup delay and $T_d = \sum_1^{m=M} T_{PE_m}$ is the circuit delay with T_{PE_m} as the PE latency delay of the m-th PE. The second term is the contribution of the actual transform whereas the last term is the result of the memory copy process.

In case of a L-stage forward/inverse DWP, the total time is formulated as:

$$T_{DWP} = L(T_s + T_d) + LS \tag{14}$$

The second term is the contribution of the low-pass and high-pass parts which have to be processed as well. No memory copy process takes place on performing forward/inverse DWP.

5 Conclusions

The facts are that wavelets have a very wide spectrum and there exists different classes of wavelet filters that can be used depending on the application. We have proposed a novel architecture that is able to compute various wavelet transforms and their inverses based on their lifting scheme representations. Because of diversities in application's need, we have designed the wavelet processor that can perform not only DWTs, but also DWPs.

The proposed wavelet processor is based on M chained PEs to compute the prediction/update of the lifting steps, and it can be configured easily to support higher order lifting polynomials, as the result of the factorization of the higher order wavelet filters. To cope with different wavelet filters, the developed wavelet processor includes a multi-context configuration so that users can easily switch between transforms (including their inverses). The wavelet processor is full-customized to manage different application demands which require different accuracy. Additionally, the architecture takes into account the energy conservation property of the wavelet transform by providing the normalization step that occurs at the end of the forward DWT/DWP or at the beginning

of the inverse DWT/DWP. Due to its locality property, wavelet transform has a straightforward implementation in hardware. Considering also that wavelet transforms work with arbitrary number of samples, we deliver this freedom into our wavelet processor. Using 0.18-μm technology, the estimated area of the proposed wavelet processor with 16-bit configuration and 2×2×512 words memory is 2.5 mm^2 and the estimated operating speed is 319 MHz.

References

1. Agbinya, J.: Discrete wavelet transform techniques in speech processing. In: Proc. of the IEEE TENCON. Digital Signal Processing Applications, TENCON 1996, vol. 2, pp. 514–519 (1996)
2. Andra, K., Chakrabarti, C., Acharya, T.: A VLSI architecture for liftingbased forward and inverse wavelet transform 50(4), 966–977 (2002)
3. Barua, S., Carletta, J., Kotteri, K., Bell, A.: An Efficient Architecture for Lifting-based Two-Dimensional Discrete Wavelet Transforms. In: Proc. of the Great Lakes Symposium on VLSI, GLSVLSI 2004 (2004)
4. Blinowska, K.J., Durka, P.J.: Introduction to wavelet analysis. Br. J. Audiol. 31(6), 449–459 (1997)
5. Bultheel, A.: Wavelets with applications in signal and image processing (2003)
6. Calderbank, R., Daubechies, I., Sweldens, W., Yeo, B.-L.: Lossless image compression using integer to integer wavelet transforms. In: Proc. of the Intl. Conference on Image Processing, ICIP 1997, vol. 1, pp. 596–599. IEEE Press, Los Alamitos (1997)
7. Calderbank, R., Daubechies, I., Sweldens, W., Yeo, B.-L.: Wavelet transforms that map integers to integers. Appl. Comput. Harmon. Anal. 5(3), 332–369 (1998)
8. Carnero, B., Drygajlo, A.: Perceptual speech coding and enhancement using frame-synchronized fast wavelet packet transform algorithms 47, 1622–1634 (1999)
9. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview 46(4), 1103–1127 (2000)
10. Dang, P., Chau, P.: Reduce complexity hardware implementation of discrete wavelet transform for JPEG 2000 standard. In: Proc. of the IEEE Intl. Conference on Multimedia and Expo., ICME 2002, August 26-29, vol. 1, pp. 321–324 (2002)
11. Daubechies, I.: The wavelet transform, time-frequency localization and signal analysis. IEEE Trans. on Information Theory 36, 961–1005 (1990)
12. Daubechies, I., Sweldens, W.: Factoring Wavelet Transforms into Lifting Steps. J. Fourier Anal. Appl. 4(3), 245–267 (1998)
13. Dillen, G., Georis, B., Legat, J., Cantineau, O.: Combined line-based architecture for the 5-3 and 9-7 wavelet transform of JPEG 2000 13(9), 944–950 (September 2003)
14. Ferens, K., Kinsner, W.: Adaptive wavelet subband coding for music compression. In: Kinsner, W. (ed.) Proc. of the Data Compression Conference, DCC 1995 (1995)
15. Grgic, S., Kers, K., Grgic, M.: Image compression using wavelets. In: Kers, K. (ed.) Proc. of the IEEE Intl. Symposium on Industrial Electronics, ISIE 1999, vol. 1 (1999)
16. Kaisheng, Y., Zhigang, C.: A wavelet filter optimization algorithm for speech recognition. In: Intl. Conference on Communication Technology Proc., ICCT 1998, October 22-24, vol. 2, p. 5 (1998)

17. Mallat, S. (ed.): A Wavelet Tour of Signal Processing. Academic Press, Incorporated, London (1998)
18. Martina, M., Masera, G., Piccinini, G., Zamboni, M.: A VLSI architecture for IWT (Integer wavelet Transform). In: Masera, G. (ed.) Proc. of the 43rd IEEE Midwest Symposium on Circuits and Systems, vol. 3 (2000)
19. Meyer, Y.: Wavelets and Operators. Press Syndicate of the University of Cambridge (1992)
20. Seo, Y.-H., Kim, D.-W.: A New VLSI Architecture of Lifting-Based DWT. In: Bertels, K., Cardoso, J.M.P., Vassiliadis, S. (eds.) ARC 2006. LNCS, vol. 3985, pp. 146–151. Springer, Heidelberg (2006)
21. Sweldens, W.: The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions. Wavelet Applications in Signal and Image Processing 3, 68–79 (1995)
22. Usevitch, B.: A tutorial on modern lossy wavelet image compression: foundations of JPEG2000 18(5), 22–35 (2001)
23. Wang, C., Gan, W.: Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Packet Transform 54(5), 422–426 (2007)

On the Comparison of Different Number Systems in the Implementation of Complex FIR Filters

Gian Carlo Cardarilli[1], Alberto Nannarelli[2], and Marco Re[1]

[1] Department of Electronics, University of Rome Tor Vergata, Rome, Italy
{g.cardarilli,marco.re}@uniroma2.it
[2] DTU Informatics, Technical University of Denmark, Kongens Lyngby, Denmark
an@imm.dtu.dk

1 Introduction

In modern electronic systems, complex arithmetic computation plays an important role in the implementation of different Digital Signal Processing (DSP) and scientific computation algorithms [1], [2]. Most of the interest in complex signal processing is related to the implementation of wireless communication systems based on new concepts and architectures [3]. A very interesting tutorial paper on complex signal processing and its applications has been presented recently [4]. In [4], the importance of the use of complex signal processing in wireless communications systems has been shown. Regarding communication systems, one of the most critical computation to be implemented in hardware is complex FIR filtering. In fact, FIR filters are generally characterized by a high order (number of taps) to obtain sharp transition bands that, in case of high speed real time computation, require many resources and have high power dissipation. In particular, for complex FIR filters, the hardware complexity is mostly determined by the number of complex multipliers (i.e. each complex multiplication is actually implemented with four scalar multiplications). Different solutions have been proposed to lower the hardware complexity of the complex multiplication either at algorithmic level (Golub Rule) [5], or by using different number systems such as the Quadratic Residue Number System (QRNS) [6], [2] and the Quater-Imaginary Number System (QINS) [7].

The aim of this work is to compare in terms of performance, area and power dissipation, the implementations of complex FIR filters based on the traditional Two's Complement System (TCS), the QRNS and the QINS (or radix-2j) implemented in the Redundant Complex Number Systems (RCNS) [8].

Previous work was done on both the QRNS ([6], [9]) and on the radix-2j and the RCNS ([10], [11], [12]). In this work, we compare for a specific application, the complex FIR filter, the performance and the tradeoffs of TCS, QRNS and RCNS. The results of the implementations show that the complex filter implemented in QRNS has the lowest power dissipation and the smallest area with respect to filters implemented in TCS and RCNS.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 174–190, 2010.

The work is organized as follows: in Section 2 a background on the QRNS and the radix-2j number systems is given; the FIR filter architectures for the three number systems are described in Section 3; the synthesis results and the comparisons are discussed in Section 4. Finally, the conclusions are drawn in Section 5.

2 The Quadratic Residue Number System

A Residue Number System (RNS) is defined by a set of P relatively prime integers $\{m_1, m_2, \ldots, m_P\}$ which identify the RNS base. Its dynamic range is given by the product $M = m_1 \cdot m_2 \cdot \ldots \cdot m_P$.

Any integer $X \in \{0, 1, 2, \ldots M-1\}$ has a unique RNS representation given by:

$$X \overset{RNS}{\rightarrow} (\langle X \rangle_{m_1}, \langle X \rangle_{m_2}, \ldots, \langle X \rangle_{m_P})$$

where $\langle X \rangle_{m_i}$ denotes the operation $X \ mod \ m_i$ [13]. Operations on different m_i (moduli) are done in parallel

$$Z = X \ op \ Y \quad \overset{RNS}{\rightarrow} \quad \begin{cases} Z_{m_1} = \langle X_{m_1} \ op \ Y_{m_1} \rangle_{m_1} \\ Z_{m_2} = \langle X_{m_2} \ op \ Y_{m_2} \rangle_{m_2} \\ \ldots \quad \ldots \quad \ldots \\ Z_{m_P} = \langle X_{m_P} \ op \ Y_{m_P} \rangle_{m_P} \end{cases} \tag{1}$$

As a consequence, operations on large wordlengths can be split into several modular operations executed in parallel and with reduced wordlength [13].

The conversion of the RNS representation of Z can be accomplished by the Chinese Remainder Theorem (CRT):

$$Z = \left\langle \sum_{i=0}^{P} \overline{m_i} \cdot \langle \overline{m_i}^{-1} \rangle_{m_i} \cdot Z_{m_i} \right\rangle_M \qquad \text{with } \overline{m_i} = \frac{M}{m_i} \tag{2}$$

and $\overline{m_i}^{-1}$ obtained by $\langle \overline{m_i} \cdot \overline{m_i}^{-1} \rangle_{m_i} = 1$.

To better explain the CRT, we show an example in which we convert the RNS representation $\{3,\ 6,\ 5\}$, with RNS base $\{\ 5,\ 7,\ 8\ \}$, to integer. The dynamic range of the RNS base $\{\ 5,\ 7,\ 8\ \}$ is $M = 280$. We start by computing the values $\overline{m_i} = \frac{M}{m_i}$

$$\overline{m_1} = \frac{280}{5} = 56 \qquad \overline{m_2} = \frac{280}{7} = 40 \qquad \overline{m_3} = \frac{280}{8} = 35$$

To compute $\overline{m_i}^{\ -1}$, we have to find a number x such that

$$\langle \overline{m_i} \cdot x \rangle_{m_i} = 1 \tag{3}$$

For this reason, x is called the multiplicative inverse of $\overline{m_i}$ and indicated as $\overline{m_i}^{\ -1}$. By computer iterations, we find

$$\overline{m_1}^{\ -1} = 1 \qquad \overline{m_2}^{\ -1} = 3 \qquad \overline{m_3}^{\ -1} = 3$$

Finally, applying (2) to the set of residues $\{3, 6, 5\}$ we get

$$\left\langle \sum_{i=1}^{3} \overline{m_i} \cdot \langle \overline{m_i}^{-1} \rangle_{m_i} \cdot Z_i \right\rangle_{280} = \langle\ 56 \cdot 1 \cdot 3 + 40 \cdot 3 \cdot 6 + 35 \cdot 3 \cdot 5\ \rangle_{280} = \langle\ 1413\ \rangle_{280} = 13$$

We can easily verify that

$$\langle 13 \rangle_5 = 3, \quad \langle 13 \rangle_7 = 6, \quad \langle 13 \rangle_8 = 5$$

In the complex case, we can transform the imaginary term into an integer if the equation $q^2 + 1 = 0$ has two distinct roots q_1 and q_2 in the ring of integers modulo M (Z_M). A complex number $x_R + jx_I = (x_R, x_I) \in Z_M \times Z_M$, with q root of $q^2 + 1 = 0$ in Z_M, has a unique Quadratic Residue Number System representation given by

$$\begin{aligned} (x_R, x_I) \overset{QRNS}{\rightarrow} & (X_i, \hat{X}_i) \quad i = 1, 2, \ldots, P \\ & X_i = \langle x_R + q \cdot x_I \rangle_{m_i} \\ & \hat{X}_i = \langle x_R - q \cdot x_I \rangle_{m_i} \end{aligned} \tag{4}$$

The inverse QRNS transformation is given by

$$\begin{aligned} (X_i, \hat{X}_i) \overset{RNS}{\rightarrow} & (X_{Ri}, X_{Ii}) \quad i = 1, 2, \ldots, P \\ & X_{Ri} = \langle 2^{-1}(X_i + \hat{X}_i) \rangle_{m_i} \\ & X_{Ii} = \langle 2^{-1} \cdot q^{-1}(X_i - \hat{X}_i) \rangle_{m_i} \end{aligned} \tag{5}$$

where 2^{-1} and q^{-1} are the multiplicative inverses of 2 and q, respectively, modulo m_i:

$$\langle 2 \cdot 2^{-1} \rangle_{m_i} = 1 \qquad \text{and} \qquad \langle q \cdot q^{-1} \rangle_{m_i} = 1\ .$$

Then, by applying the CRT we get

$$\begin{aligned} (X_{R1}, X_{R2}, \ldots, X_{RP}) &\overset{CRT}{\rightarrow} x_R \\ (X_{I1}, X_{I2}, \ldots, X_{IP}) &\overset{CRT}{\rightarrow} x_I \end{aligned} \tag{6}$$

Moreover, it can be proved that for all the prime integers which satisfy

$$p = 4k + 1 \qquad\qquad k \in N$$

the equation $q^2 + 1 = 0$ has two distinct roots q_1 and q_2.

As a consequence, the product of two complex numbers $x_R + jx_I$ and $y_R + jy_I$ is in QRNS

$$(x_R + jx_I)(y_R + jy_I) \overset{QRNS}{\rightarrow} (\langle X_i Y_i \rangle_{m_i}, \langle \hat{X}_i \hat{Y}_i \rangle_{m_i}) \tag{7}$$

and it is realized by using two integer multiplications instead of four.

We illustrate an example of QRNS multiplication in the ring modulo 13. The complex multiplication to perform is

$$(x_R + jx_I)(y_R + jy_I) = (3 + j)(2 + j2) = 4 + j8$$

For $m = 13$ the root is $q = q_1 = 5 \leftrightarrow \langle 5 \cdot 5 \rangle_{13} = -1$. The conversion to QRNS according to (4) gives

$$\begin{array}{ll} X = \langle 3 + 5 \cdot 1 \rangle_{13} = 8 & Y = \langle 2 + 5 \cdot 2 \rangle_{13} = 12 \\ \hat{X} = \langle 3 - 5 \cdot 1 \rangle_{13} = 11 & \hat{Y} = \langle 2 - 5 \cdot 2 \rangle_{13} = 5 \end{array}$$

The two QRNS multiplications (modulus 13) are:

$$X \cdot Y = \langle 8 \cdot 12 \rangle_{13} = 5 \qquad \hat{X} \cdot \hat{Y} = \langle 11 \cdot 5 \rangle_{13} = 3$$

And finally, the conversion QRNS to integer according to (5) gives

$$\begin{array}{lll} z_R = \langle 7(5 + 3) \rangle_{13} = 4 & \text{being} & 2^{-1} = 7 \\ z_I = \langle 7 \cdot 8(5 - 3) \rangle_{13} = 8 & \text{and} & q^{-1} = 8 \end{array}$$

3 The Radix-2j Number System

It is well known that an integer x can be represented by a digit-vector

$$X = (x_{n-1}, \ldots, x_1, x_0)_r$$

such that

$$x = \sum_{i=0}^{n-1} x_i \cdot r^i$$

where r is the radix of the representation. By choosing $r = 2j$, we obtain a *Quater-Imaginary* Number System (QINS) [7]. Complex numbers can be represented in QINS by vectors with the non-redundant digit set $\{0, 1, 2, 3\}$. Therefore, a complex number $a + jb$ is represented in QINS as:

$$\begin{aligned} a + jb = {} & x_{n-1}(2j)^{n-1} + x_{n-2}(2j)^{n-2} + \ldots + \\ & + x_3(-8j) + x_2(-4) + x_1(2j) + x_0(1) \\ = {} & (x_{n-1}, \ldots, x_1, x_0)_{2j} \end{aligned} \tag{8}$$

The above expression, shows that the real part is represented by the digits of even weight, while the imaginary one by the digits of odd weight. Furthermore, the sign is embedded in the representation. The imaginary number j cannot be represented by (8). To represent j, we need the power -1, which corresponds to $-\frac{1}{2}j$, that in the conventional number systems (e.g. binary) is only needed to represent fractional numbers. Table 1 shows how the real and imaginary

Table 1. Representation of real and imaginary integers in QINS

Real		Imaginary	
-8	00200.0	-8j	01000.0
-7	00201.0	-7j	01010.2
-6	00202.0	-6j	01010.0
-5	00203.0	-5j	01020.2
-4	00100.0	-4j	01020.0
-3	00101.0	-3j	01030.2
-2	00102.0	-2j	01030.0
-1	00103.0	-1j	00000.2
0	00000.0	0j	00000.0
1	00001.0	1j	00010.2
2	00002.0	2j	00010.0
3	00003.0	3j	00020.2
4	10300.0	4j	00020.0
5	10301.0	5j	00030.2
6	10302.0	6j	00030.0
7	10303.0	7j	103000.2
8	10200.0	8j	103000.0

numbers, in the range $[-8, 8]$ and $[-8j, 8j]$ respectively, are represented in QINS. Every complex number $x_R + jx_I$ can be obtained by overlapping the real and imaginary parts. For example, according to Table 1, $4 - 5j$ is represented by the digit vector 11320.2.

From Table 1 we can notice that for a given number of digits the representation is not symmetric with respect to the zero. For example, in the two's complement binary system with 8 digits we can represent the dynamic range $\{-128, 127\}$. In the QINS, for the real part, with 3 digits (equivalent to 64 different values) the dynamic range representable is $\{-12, 51\}$.

3.1 Addition

The addition of two QINS numbers can be performed by changing the *carry rule* according to (8). First, because the even weight digits represent the real part and the odd weight the imaginary one, the carry is propagated by skipping a digit. Second, because two adjacent even (or odd) weight digits have opposite sign, the carry propagated acts as a borrow. For example, if a positive weight digit generates a carry, this positive value will decrement the next digit with negative weight, and vice-versa. In addition, the propagation of borrows can generate negative digits (e.g. -1). Therefore, because of the quaternary representation of the QINS, the negative digits are converted into positive (modulo operation) and an always positive carry propagated. Summarizing the addition algorithm is implemented as:

$$x_i, y_i, s_i \in \{0, 1, 2, 3\}$$
$$c_i \in \{\overline{1}, 0, 1\}$$

$$s_i = (x_i + y_i + c_i) \bmod 4$$

$$c_{i+2} = \begin{cases} \overline{1} \text{ if } (x_i + y_i + c_i) \geq 4 \\ 1 \text{ if } (x_i + y_i + c_i) < 0 \\ 0 \; otherwise \end{cases}$$

For example, if we wnat to add $x_R = 1$ and $y_R = 3$ in QINS we get:

$$\begin{array}{ll} X\text{: } 0\,0\,0\,0\,3.\,0 + \\ Y\text{: } 0\,0\,0\,0\,1.\,0 + \\ c\text{: } \;1\,0\,\overline{1}\,0\,0.\,0 = \\ S\text{: } \overline{1\,0\,3\,0\,0.\,0} & \rightarrow \; s_R = 4 \end{array}$$

3.2 The Redundant Complex Number Systems

The implementation of the basic arithmetic operators in radix-$2j$ can take advantage of the Signed-Digit (SD) representation [14], which allows carry free addition. The combination of radix-$2j$ and SD representation, resulted in the Redundant Complex Number Systems (RCNS), which is described in [8], [10], [11], [12] and [15].

We now briefly recall the characteristics of the RCNS. The RCNS is a redundant positional number system based on the radix rj where its digits can assume the $2\alpha + 1$ values: $A_\alpha = \{\overline{\alpha}, \cdots, \overline{1}, 0, 1, \cdots, \alpha\}$ where $\overline{\alpha} = -\alpha$.

In the case of the radix $2j$, two possible RCNSs [10] are:

1. RCNS 2j, 2 with digit set $A_2 = \{\overline{2}, \overline{1}, 0, 1, 2\}$
2. RCNS 2j, 3 with digit set $A_3 = \{\overline{3}, \overline{2}, \overline{1}, 0, 1, 2, 3\}$

In this work, RCNS 2j, 2 is used to recode the multiplier, and RCNS 2j, 3 is used for the signed-digit additions, as illustrated next.

4 FIR Filter Architecture

A complex FIR filter of order N is expressed by

$$\underline{y}(n) = \sum_{k=0}^{N-1} \underline{a}_k \underline{x}(n-k) \tag{9}$$

where $\underline{x}$, $\underline{y}$ and $\underline{a}_k$ denote complex numbers. We consider the implementation of a FIR filter in transposed form because its structure is more regular with respect to the filter order N and it does not require a tree of adders. The filter

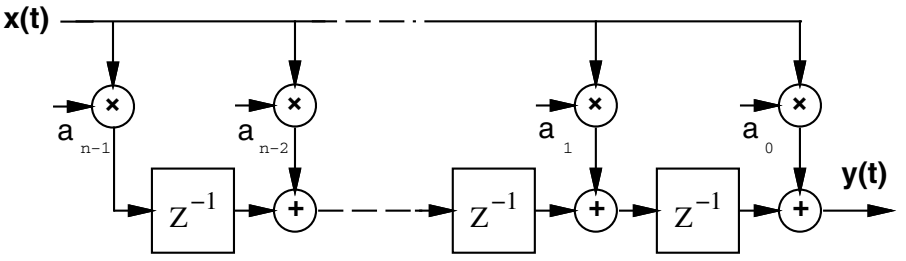


Fig. 1. Structure of FIR filter in transposed form

in transposed form (Fig. 1) can be regarded as the sequence of groups, often referred as *taps*, composed of:

- a complex multiplier;
- a complex adder implemented with one adder for the real part, and one for the imaginary part;
- a register to store the real and imaginary parts.

We perform our design space exploration for programmable N-tap complex FIR filters with input and coefficients size of 10 bits for both the real part and imaginary parts. The 20 bit dynamic range of the filter guarantees error free operations[1].

4.1 TCS FIR Filter

A single tap of the The programmable N-tap TCS complex FIR filter is realized as sketched in Fig. 2. It is composed of two branches: the real branch (top part of Fig. 2) and the imaginary branch (bottom part of Fig. 2). The real and imaginary products are both realized with two Booth multipliers each, and the resulting partial products are accumulated in a Wallace's tree structure which produces a carry-save (CS) representation of the product at each side of the filter. We chose to keep the product in carry-save (CS) format to speed-up the operation, and delayed the assimilation of the CS representation to the last stage of the filter. In both branches (real and imaginary) of each tap we need to add the CS representation of the product to the value stored in the register (previous tap). Again, to avoid the propagation of the carry, we can store the CS representation. For this reason, we need to implement the addition with an array of 4:2 carry-save adders (CSA), as shown in Fig. 2.

We convert the CS representation of y_{Re} and y_{Im} with two carry-propagate adders at the filter output.

[1] These wordlengths are derived from the specification of an actual digital filter for satellite TV broadcasting.

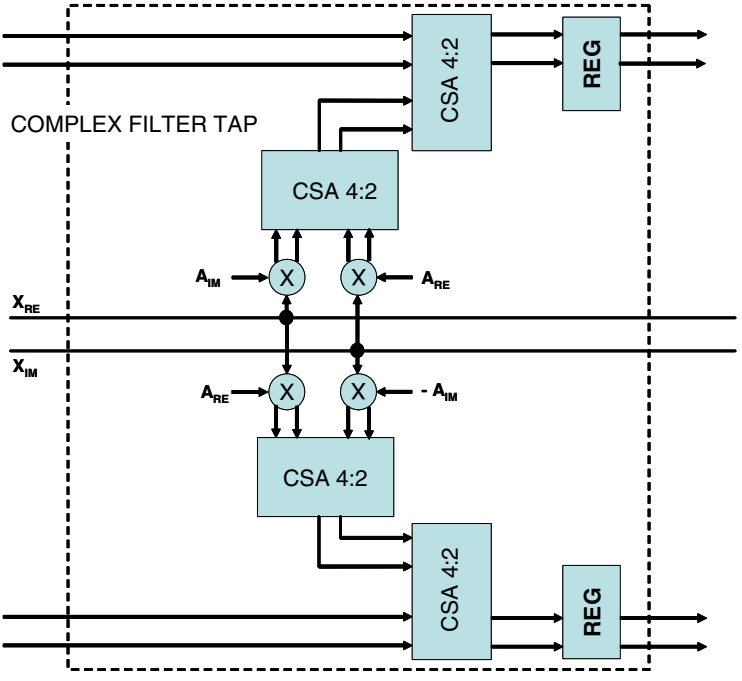


Fig. 2. Structure of tap in TCS complex FIR filter

4.2 QRNS FIR Filter

The architecture of the QRNS filter, is a direct consequence of (1), (7) and (9), and it can be realized by two RNS filters in parallel as shown in Fig. 3. Each RNS filter is then decomposed into P filters working in parallel, where P is the number of moduli used in the RNS representation. In addition, the RNS filter requires both binary to QRNS and QRNS to binary converters.

In order to have a dynamic range of 20 bits, as required by the specifications, we chose the following set of moduli:

$$m_i = \{5, 13, 17, 29, 41\}$$

such that

$$log_2(5 \cdot 13 \cdot 17 \cdot 29 \cdot 41) > 20.$$

For each path mod m_i, we have to build a FIR filter with a structure similar to that of Fig. 1. Therefore, we need to implement modular multiplication and addition.

Implementation of Modular Addition

The modular addition

$$\langle a_1 + a_2 \rangle_m$$

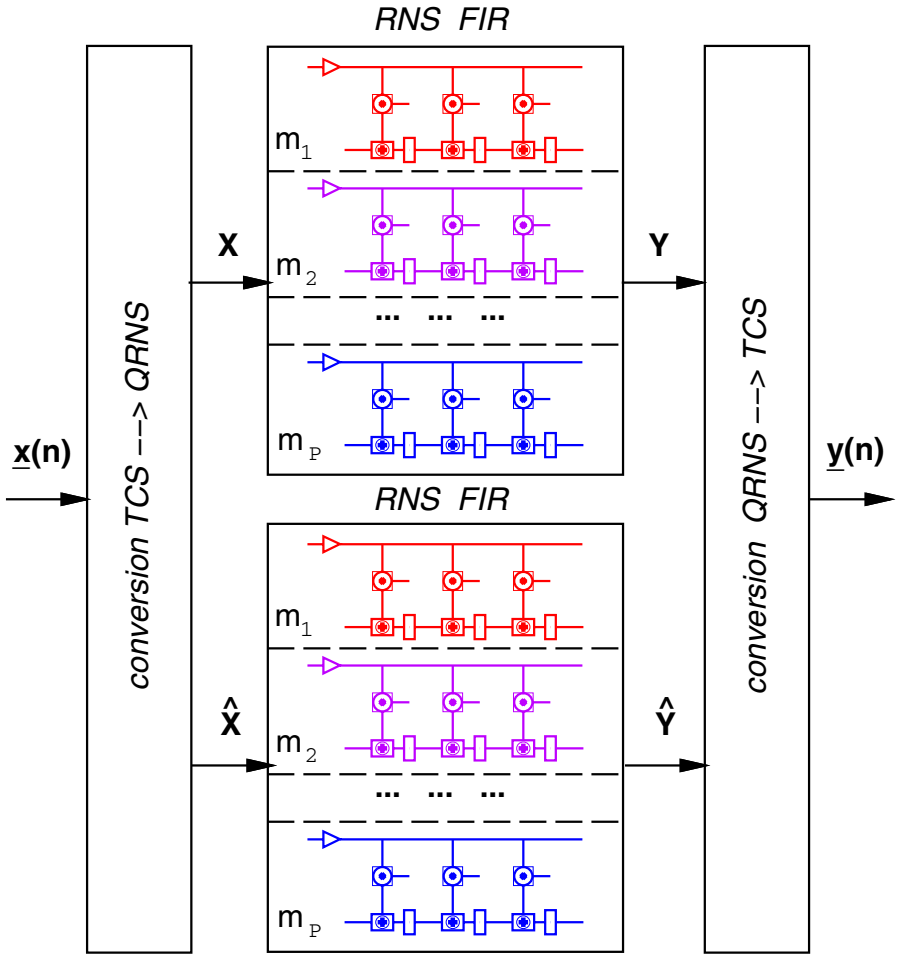


Fig. 3. QRNS FIR Filter architecture

can be implemented by two additions. If the result of $a_1 + a_2$ exceeds the modulo (it is larger than $m-1$), we have to subtract the modulo m. In order to speed-up the operation we can execute in parallel the two operations:

$$(a_1 + a_2) \quad \text{and} \quad (a_1 + a_2 - m).$$

If the sign of the three-term addition is negative, it means than the sum $(a_1 + a_2) < m$ and the modular sum is $a_1 + a_2$, otherwise the modular addition is the result of the three-term addition. The above algorithm can be implemented with two binary adders as shown in Fig. 4.

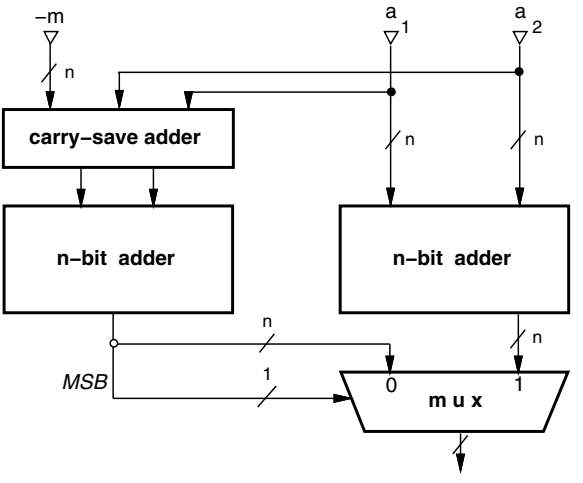


Fig. 4. Architecture of the modular adder

Table 2. Example of isomorphic transformation for $m = 5$ ($q = 2$)

n	w	$\langle q^w \rangle_m = n$
0	N/A	
1	0	$\langle 2^0 \rangle_5 = 1$
2	1	$\langle 2^1 \rangle_5 = 2$
3	3	$\langle 2^3 \rangle_5 = 3$
4	2	$\langle 2^2 \rangle_5 = 4$

Implementation of Modular Multiplication by Isomorphism

Because of the complexity of modular multiplication, it is convenient to implement the product of residues by the isomorphism technique [16] . By using isomorphisms, the product of the two residues is transformed into the sum of their indices which are obtained by an isomorphic transformation. According to [16], if m is prime there exists a primitive radix q such that its powers modulo m cover the set $[1, m-1]$:

$$n = \langle q^w \rangle_m \qquad \text{with } n \in [1, m-1] \text{ and } w \in [0, m-2].$$

An example of isomorphic transformation is shown in Table 2 for $m = 5$. In this case, the primitive radix is $q = 2$.

Both transformations $n \rightarrow w$ and $w \rightarrow n$ can be implemented with $m - 1$ entries look-up tables, if the moduli are not too large (less than 8-bit wide). Therefore, the product of a_1 and a_2 modulo m can be obtained as:

$$\langle a_1 \cdot a_2 \rangle_m = \langle q^w \rangle_m$$

where

$$w = \langle w_1 + w_2 \rangle_{m-1} \qquad \text{with } a_1 = \langle q^{w_1} \rangle_m \text{ and } a_2 = \langle q^{w_2} \rangle_m$$

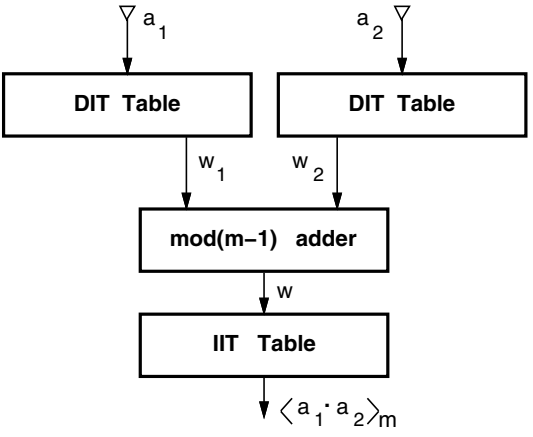


Fig. 5. Structure of isomorphic multiplication

In order to implement the modular multiplication the following operations are performed:

1) Two Direct Isomorphic Transformations (DIT) to obtain w_1 and w_2;
2) One modulo $m-1$ addition $\langle w_1 + w_2 \rangle_{m-1}$;
3) One Inverse Isomorphic Transformations (IIT) to obtain the product.

The architecture of the isomorphic multiplier is shown in Fig. 5. Special attention has to be paid when one of the two operands is zero. In this case there exists no isomorphic correspondence and the modular adder has to be bypassed.

For example, for the modular multiplication $\langle 3 \cdot 4 \rangle_5 = 2$ using the isomorphic transformation of Table 2, we have

$$
\begin{aligned}
&1)\ 3 = \langle 2^3 \rangle_5 \overset{DIT}{\rightarrow} w_1 = 3 \\
&\quad\ 4 = \langle 2^2 \rangle_5 \overset{DIT}{\rightarrow} w_2 = 2 \\
&2)\ \langle 2 + 3 \rangle_4 = 1 \\
&3)\ 1 \overset{IIT}{\rightarrow} \langle 2^1 \rangle_5 = 2
\end{aligned}
$$

Implementation of FIR Filter Modulo m

By using the isomorphism technique, the product of the two residues is transformed into the sum of their indices which are obtained by an isomorphic transformation. As a result, in each tap, the modular multiplication is reduced to a modular addition followed by an access to table (inverse isomorphism). The two input DIT tables of Fig. 5 do not need to be replicated in every tap. By observing that in computing the product $A_k X(n-k)$ the term X is common to all taps and it can be converted once in the input conversion unit, and that the term A_k can be stored directly as the index of the isomorphism. Therefore, the structure of each modular tap can be simplified as shown in Fig. 6.

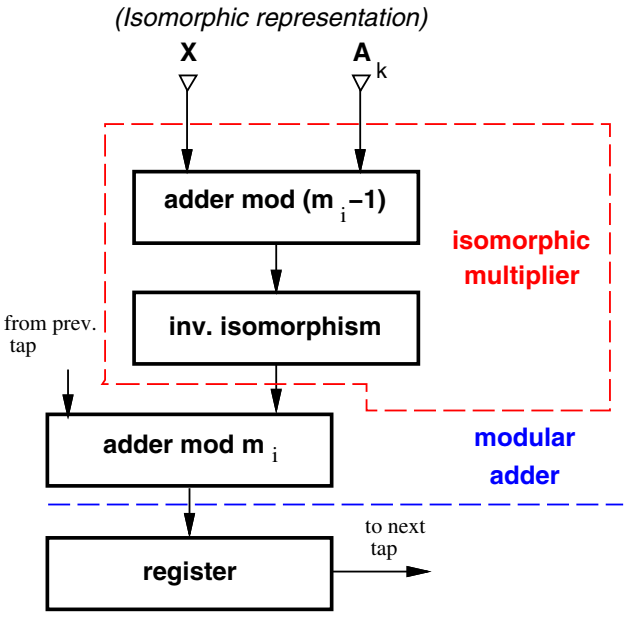


Fig. 6. Structure of RNS tap for filter in transposed form

4.3 Radix-2j Filter (RCNS)

Because of the radix-2j representation, the filter tap is simply implemented with a multiplier and an adder. We implement the multiplier as described in [10]. The complex $\underline{x}$ and $\underline{a}_k$ are converted in non-redundant QINS and then $\underline{a}_k$ is recoded into RCNS 2j, 2. The partial products (PPs) are then accumulated by a tree of arrays of signed-digit full-adders (SDFA) which operates in RCNS 2j, 3.

In RCNS 2j, 3, the complex number

$$\underline{X} = (X_{n-1}, \ldots, X_i, \ldots, X_1, X_0, X_{-1})$$

has digits in the set $X_i = \{\overline{3}, \overline{2}, \overline{1}, 0, 1, 2, 3\}$, which encoded in binary as

$$X_i = 2x_i^1 + x_i^0 \qquad \text{with } x_i^1, x_i^0 \in \{\overline{1}, 0, 1\} \tag{10}$$

Both x_i^1 and x_i^0 are then encoded with two bits each as shown in Table 3. Therefore, the resulting binary encoding of X_i is illustrated in Table 4. Four bits are necessary to represent each RCNS 2j, 3 digit. With the encoding of Table 4 the SDFA of Fig. 7 can be derived.

By arranging the SDFAs in a tree the 10 PPs are reduced to 2 as shown in Fig. 8. An extra array of SDFAs adds the product $\underline{x} \cdot \underline{a}_k$ to the partial sum coming from the previous tap. As for the TCS case, we keep the carry-save representation of the digits until the last stage of the filter where we perform the conversion from RCNS 2j, 3 to radix-2 (binary) integers. Due to the CS representation of digits we need to store 8N bits in the tap's registers.

Table 3. Binary encoding of x_i^1 and x_i^0

x_i^1	x_i^{P1}	x_i^{M1}
1	0	1
0	0	0
0	1	1
1	1	0

x_i^0	x_i^{P0}	x_i^{M0}
1	0	1
0	0	0
0	1	1
1	1	0

Table 4. Binary encoding of X_i

X_i	x_i^1	x_i^0	x_i^{P1}	x_i^{M1}	x_i^{P0}	x_i^{M0}
$\overline{3}$	$\overline{1}$	$\overline{1}$	0	1	0	1
2	1	0	0	1	0	0
			0	1	1	1
$\overline{1}$	0	$\overline{1}$	0	0	0	1
			1	1	0	1
	$\overline{1}$	1	0	1	1	0
0	0	0	0	0	0	0
			1	1	0	0
			0	0	1	1
			1	1	1	1
1	0	1	0	0	1	0
			1	1	1	0
	1	$\overline{1}$	1	0	0	1
2	1	0	1	0	0	0
			1	0	1	1
3	1	1	1	0	1	0

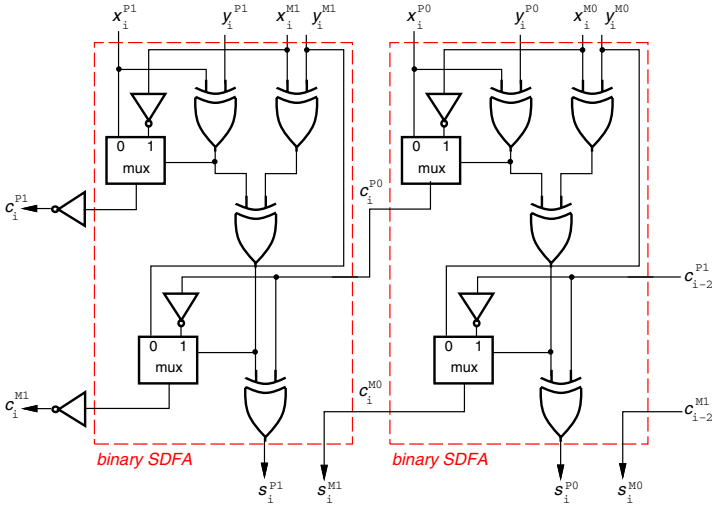


Fig. 7. Implementation of SD full-adder (SDFA)

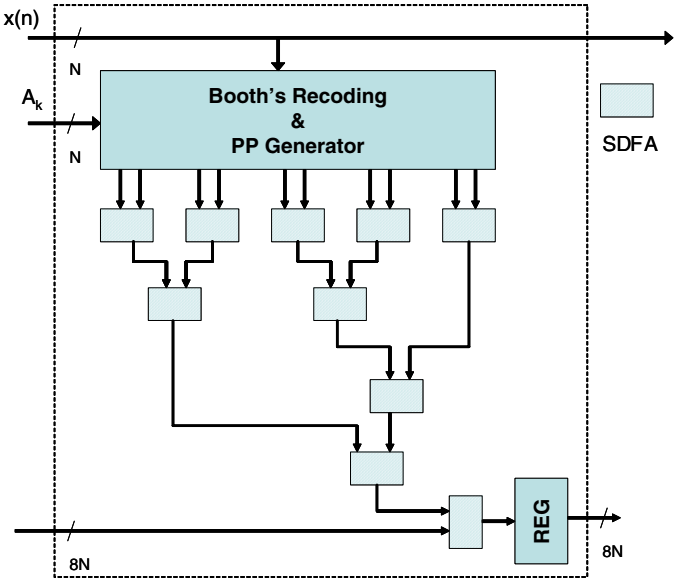


Fig. 8. Structure of RCNS tap

5 Filters Implementation

The filters are implemented in the 90 nm STM library of standard cells [17] and they have been synthesized by Synopsys Design Compiler. All the filters can be clocked at $f_{max} = 300\ MHz$. By interpolating the results obtained by synthesis on filters of different order (number of taps), we obtain the trends shown in Fig. 9 for the area and Fig. 10 for the power. The values of area and power dissipation for the single tap (Fig. 2, Fig. 6 and Fig. 8) determine the slopes of the curves in the figures. The conversions from the TCS to the other number systems (and vice versa) are a constant contribution that does not depend on the number of taps, but only on the dynamic range of the filters. Table 5 reports the data for tap and conversion contribution for the three number systems.

The results show that complex filters implemented in QRNS consume significantly less power than the corresponding ones in TCS and RCNS. The expression for the power dissipated dynamically [18] in a system composed of n cells is

$$P_{dyn} = V_{DD}^2 f \cdot \sum_{i=1}^{n} C_{Li} a_i \tag{11}$$

where

V_{DD} is the power supply voltage;
f is the clock frequency;
C_{Li} is the load connected to the i-th cell (both active load and interconnections);

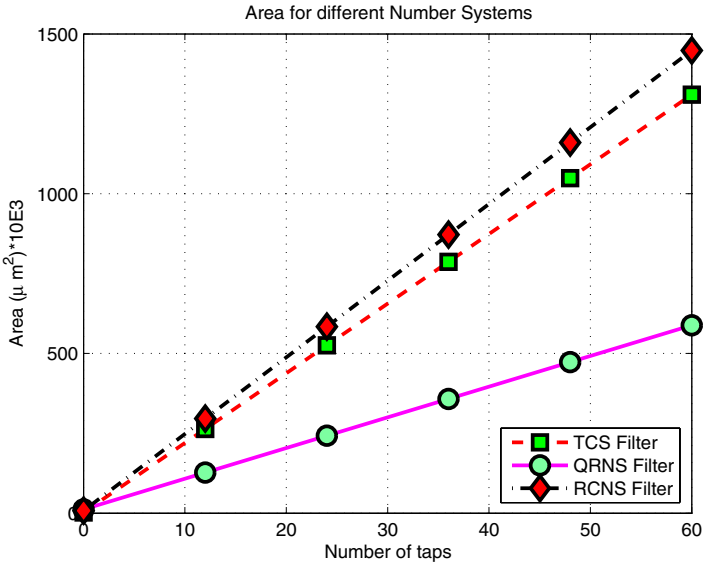


Fig. 9. Trends in area for increasing N

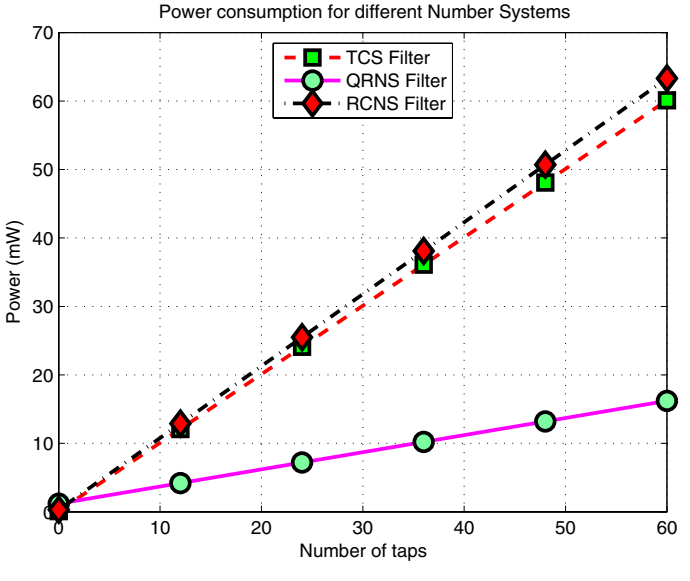


Fig. 10. Trends in power dissipation (at 100 MHz) for increasing N

a_i is the activity factor of the i-th cell, which is the measure of how many transitions occur at its output. The activity factor is normally related to the clock $a_i \in [0, 1]$.

Table 5. Values of area and power dissipation

	Area		P at 100 MHz	
	tap	conv.	tap	conv.
TCS	21.8K	2.0K	1.00	0.10
QRNS	9.6K	12.0K	0.25	1.20
RCNS	23.9K	8.0K	1.05	0.30
	$[\mu m^2]$		$[mW]$	

The lower power dissipation in the QRNS filter is due to the combination of two factors:

1. As clearly shown in Fig. 9, the smaller area results in a global reduced capacitance $\sum_{i=1}^{n} C_{Li}$ (including shorter interconnections).
2. The work in [19] showed that the number of transitions, i.e. the switching activity, for vectors of the same number of bits k, in RNS is lower than in TCS

$$\left(\sum_{i=1}^{k} a_i\right)_{RNS} < \left(\sum_{i=1}^{k} a_i\right)_{TCS}$$

Therefore, the switched capacitance $\sum_{i=1}^{n} C_{Li} a_i$, and by (11) the power consumption, in QRNS is smaller than in TCS and RCNS.

6 Conclusions

In this work, the use of different number representations for the implementation of complex FIR filters has been investigated.

Complex multipliers determine the performance, area and power dissipation of complex filters. Previously in [10], complex multipliers in TCS and RCNS were evaluated, while in [9], complex filters in QRNS and TCS were compared. Here we extended the comparison to complex filters implemented in TCS, QRNS and RCNS.

The experimental results on complex filters with 20 bit dynamic range show that for the TCS and the RCNS the area and power dissipation are similar and confirms the findings of [10]. As for the QRNS, the results presented here, confirm those of [9], based on the implementation of TCS and QRNS complex filters in a 0.35 μm technology.

To summarize, this work shows that for complex high order FIR filters implementations based on QRNS offer significant advantages in area and power dissipation without any performance degradation.

References

1. Oppenheim, A.V., Shafer, R.V.: Digital Signal Processing. Prentice Hall, Englewood Cliffs (1995)
2. Mitra, S.K., Kaiser, K.: Handbook for Digital Signal Processing. Wiley-Interscience, Hoboken (1993)

3. Brodersen, R.W., Chen, M.S.-W.: Digital Complex Signal Processing Techniques for Impulse Radio. In: Proc. of IEEE GLOBECOM 2006 Global Telecommunications Conference, November 2006, pp. 1–5 (2006)
4. Martin, K.W.: Complex signal processing is not complex. IEEE Transactions on Circuits and Systems I, 51, 1823–1836 (2004)
5. Moharir, P.S.: Extending the scope of Golub's method beyond complex multiplication to binary converters. IEEE Transactions on Computers C-34(5), 484–487 (1985)
6. Sodestrand, M., Jenkins, W., Jullien, G.A., Taylor, F.J.: Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. IEEE Press, New York (1986)
7. Knuth, D.E.: The Art of Computer Programming 2: Seminumerical Algorithms, 3rd edn. Addison-Wesley Publishing Company, Reading (1998)
8. Aoki, T., Amada, H., Higuchi, T.: Real/Complex Reconconfigurable Arithmetic using Redundant Complex Number Systems. In: Proc. of 13th IEEE Symposium on Computer Arithmetic, July 1997, pp. 200–207 (1997)
9. D'Amora, A., Nannarelli, A., Re, M., Cardarilli, G.C.: Reducing Power Dissipation in Complex Digital Filters by using the Quadratic Residue Number System. In: Proc. of 34th Asilomar Conference on Signals, Systems, and Computers, November 2000, pp. 879–883 (2000)
10. Aoki, T., Hosci, K., Higuchi, T.: Reduntant Complex Arithmetic and its Application to Complex Multiplier Design. In: Proc. of 29th IEEE International Symposium on Multiple-Valued Logic, May 1999, pp. 200–207 (1999)
11. Ohi, Y., Aoki, T., Higuchi, T.: Redundant Complex Number Systems. In: Proc. of 25th IEEE International Symposium on Multiple-Valued Logic, May 1995, pp. 14–19 (1995)
12. Aoki, T., Ohi, Y., Higuchi, T.: Redundant Complex Number Arithmetic for High-Speed Signal Processing. In: VLSI Signal Processing VIII (1995 IEEE Workshop on VLSI Signal Processing), October 1995, pp. 523–532 (1995)
13. Szabo, N., Tanaka, R.: Residue Arithmetic and its Applications in Computer Technology. McGraw-Hill, New York (1967)
14. Avizienis, A.: Signed-Digit Number Representations for Fast Parallel Arithmetic. IRE Trans. Electronic Computers EC-10, 389–400 (1961)
15. Nielsen, A.M., Muller, J.-M.: Borrow-Save Adders for Real and Complex Number Systems. In: Proc. 2nd Conf. on Real Numbers and Computers (April 1996)
16. Vinogradov, I.: An Introduction to the Theory of Numbers. Pergamon Press, New York (1955)
17. STMicroelectronics, 90nm CMOS090 Design Platform, http://www.st.com/stonline/prodpres/dedicate/soc/asic/90plat.htm
18. Weste, N.H.E., Eshraghian, K.: Principles of CMOS VLSI Design, 2nd edn. Addison-Wesley Publishing Company, Reading (1993)
19. Stouraitis, T., Paliouras, V.: Considering the alternatives in low-power design. IEEE Circuits and Devices Magazine 17, 22–29 (2001)

Time Efficient Dual-Field Unit for Cryptography-Related Processing

Alessandro Cilardo and Nicola Mazzocca

Università degli Studi di Napoli Federico II
Dipartimento di Informatica e Sistemistica
via Claudio 21, 80125 Naples, Italy
acilardo@unina.it

Abstract. Computational demanding public key cryptographic algorithms, such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve (EC) cryptosystems, are critically dependent on modular multiplication for their performance. Modular multiplication used in cryptography may be performed in two different algebraic structures, namely $GF(N)$ and $GF(2^n)$, which normally require distinct hardware solutions for speeding up performance. For both fields, Montgomery multiplication is the most widely adopted solution, as it enables efficient hardware implementations, provided that a slightly modified definition of modular multiplication is adopted. In this paper we propose a novel unified architecture for parallel Montgomery multiplication supporting both $GF(N)$ and $GF(2^n)$ finite field operations, which are critical for RSA ad ECC public key cryptosystems. The hardware scheme interleaves multiplication and modulo reduction. Furthermore, it relies on a modified Booth recoding scheme for the multiplicand and a radix-4 scheme for the modulus, enabling reduced time delays even for moderately large operand widths. In addition, we present a pipelined architecture based on the parallel blocks previously introduced, enabling very low clock counts and high throughput levels for long operands used in cryptographic applications. Experimental results, based on $0.18\mu m$ CMOS technology, prove the effectiveness of the proposed techniques, and outperform the best results previously presented in the technical literature.

1 Introduction

The increasing centrality of networking and Internet applications are stimulating an ever-growing demand for high-performance implementations of cryptographic algorithms and protocols. Two widely adopted public-key cryptosystems, in particular, are the Rivest-Shamir-Adleman (RSA) [11] and the Elliptic Curve (EC) [1] cryptosystems. While various standardization bodies recommend prime fields $GF(N)$ or binary extension fields $GF(2^n)$ for elliptic curve cryptosystems, RSA cryptography is essentially based on integer modular arithmetic, similar in its implementation to $GF(N)$ operations. Both types of finite fields have in common that the multiplication of elements implies a reduction operation, either modulo a prime N or modulo an irreducible binary polynomial $N(x)$ of degree n.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 191–210, 2010.

The so-called Montgomery algorithm [9] has proved to be the most effective implementation technique for modular multiplication [2, 17]. It is in fact based on a slightly different definition of the modular product, which enables particularly efficient implementations.

Originally introduced for integer numbers (and thus for $GF(N)$ arithmetic), Montgomery multiplication has been effectively extended to binary fields $GF(2^n)$ [8]. As a consequence, during the last years several works have addressed the problem of implementing *unified* arithmetic blocks, suitable for computing operations in both fields using the same underlying hardware [4, 6, 12, 13, 14, 18].

In this paper, we propose a novel unified architecture for parallel Montgomery multiplication supporting both $GF(N)$ and $GF(2^n)$ operations. The hardware unit interleaves multiplication and modulo reduction in a parallel scheme. Furthermore, it relies on a modified Booth recoding technique for the multiplicand and a radix-4 scheme for the modulus, enabling reduced time delays for moderately large operand widths. We also present a pipelined architecture based on the parallel component previously introduced, enabling very low clock counts and high throughput levels for long operands used in cryptographic applications. Experimental results, based on $0.18\mu m$ CMOS technology, prove the effectiveness of the proposed techniques, and outperform the best results previously presented in the technical literature.

The paper is structured as follows. Section 2 provides a brief introduction to the properties of Montgomery multiplication algorithm. Section 3 presents the state-of-the-art of architectures suitable for unified integer/$GF(N)$ and $GF(2^n)$ arithmetic. Section 4 describes the proposed parallel arithmetic unit supporting unified Montgomery multiplication. Section 5 presents a high-throughput pipelined core based on the previously introduced parallel multiplier. Section 6 presents our results and compares them to the state-of-the-art. Section 7 concludes the paper with some final remarks.

2 Modular Multiplication Algorithm

A slight variant of standard modular multiplication, Montgomery multiplication performs the following operation:

$$A \cdot B \cdot R^{-1} \bmod N$$

where $R = 2^n$ is a power of two and n is equal to, or slightly larger than the number of bits in the modulus N, ensuring $R > N$. The value R^{-1} is the inverse of R modulo N, i.e. a number such that $R^{-1}R \bmod N = 1$. In order for such a number to exist, it suffices that $\gcd(N, R) = 1$. Since in both Elliptic Curve cryptography based on prime fields and in RSA cryptography N is always an odd number, this condition is always satisfied when R is a power of two. Montgomery multiplication can be performed with the following algorithm.

Algorithm 1. *Montgomery Modular Multiplication*

Input:

N, R and $\tilde{N}$ such that $R \cdot R^{-1} - N \cdot \tilde{N} = 1$,
$A, B < N$

Output:

$P \equiv A \cdot B \cdot R^{-1} \bmod N, \quad P < N$

Algorithm:

1. $Q = AB \cdot \tilde{N} \bmod R$
2. $P = \frac{AB+Q \cdot N}{R}$
3. if $P > N$ then $P = P - N$

The above algorithm returns a quantity P which is congruent with $AB \cdot R^{-1}$ modulo N (step 2), and is less than N (at step 2, $P = \frac{AB+Q \cdot N}{R} < \frac{N \cdot N+Q \cdot N}{R} < \left[\frac{N}{R} + \frac{Q}{R}\right] \cdot N < 2N$). The multiple $Q \cdot N$ of the modulus is defined at step 1 in such a way as to make the quantity $AB + Q \cdot N$ divisible by R [9].

An interesting property enabled by Montgomery multiplication is the possibility to work on N-*residues* of numbers, defined as $\overline{A} = A \cdot R \bmod N$. It can be easily seen that the Montgomery product of two numbers in N-residue form is still in N-residue form: $\overline{A} \cdot \overline{B} \cdot R^{-1} \bmod N = AR \cdot BR \cdot R^{-1} \bmod N = (AB) \cdot R \bmod N = \overline{AB}$. This also holds true for modular addition: $(\overline{A} + \overline{B}) \bmod N = \overline{A+B}$. All operations used in RSA and EC cryptography can be reduced to a composition of modular multiplications and additions, and can thus always handle operands in Montgomery form.

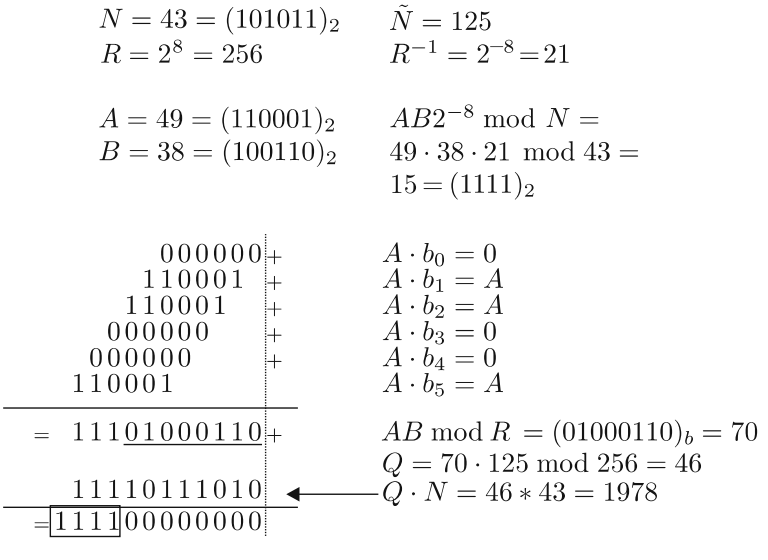


Fig. 1. An example of Montgomery multiplication execution

Notice that Algorithm 1 requires a magnitude comparison (Step 3) in order to ensure the result is actually less than the modulus N. However, when many consecutive multiplications are to be performed, we can allow intermediate results to be in the range $[0, 2N[$ with a proper choice for R. In fact, if we choose $R > 4N$, it can be easily seen that the reduction algorithm accepts multiplicands $A, B < 2N$, i.e. not necessarily less than N: $P = \frac{AB+Q\cdot N}{R} < \frac{2N\cdot 2N+Q\cdot N}{R} < \left[\frac{4N}{R} + \frac{Q}{R}\right] \cdot N < 2N$, so the algorithm preserves the invariant that inputs and output are less than $2N$. By avoiding magnitude comparison, the above version of Montgomery algorithm greatly improves performance, so we will refer to this version of the algorithm in the following. Figure 1 provides an example of execution of the Montgomery algorithm variant exploiting the above property.

The central operation of Montgomery algorithm, i.e. the computation of the product $A \cdot B$ and the multiple of the modulus $Q \cdot N$, can be implemented in a very efficient way, as it is suitable for deeply pipelined and systolic implementations [2, 10, 16, 17]. For scalable implementations, a natural choice is to partition operands into words, and process them separately. Precisely, we will refer in this paper to the so-called *finely integrated operand scanning* (FIOS) method [7], reported below.

Algorithm 2. *FIOS method for w-bit words*

Input:

$A = \sum_{i=0}^{m-1} A_i(2^w)^i$, $B = \sum_{i=0}^{m-1} B_i(2^w)^i$,
$N = \sum_{i=0}^{m-1} N_i(2^w)^i$, $\tilde{N} = \sum_{i=0}^{m-1} \tilde{N}_i(2^w)^i$,
with $A_i, B_i, N_i, \tilde{N}_i < 2^w$ and
$0 \le A, B < 2N, \quad m \cdot w \ge 2 + \lceil \log_2 N \rceil$ (i.e. $2^{m\cdot w} > 4N$)

Output:

$P \equiv A \cdot B \cdot 2^{-n} \bmod N$, with $n = m \cdot w$

Algorithm:

1. $P = 0$
2. for $j = 0$ to $m - 1$
3. $\quad C = 0$
4. $\quad Q_j = (P_0 + B_j A_0)\tilde{N}_0 \bmod 2^w$
5. $\quad$ for $i = 0$ to $m - 1$
6. $\quad\quad S := P_i + B_j A_i + Q_j N_i + C$
7. $\quad\quad$ if $(i \neq 0)$ then $P_{i-1} := S \bmod 2^w$
8. $\quad\quad C := S/2^w$
9. $\quad P_{m-1} := C$

The w-bit words of operands A, B, and N are processed in two nested loops. During the execution of the algorithm, temporary variables S and C can be stored in a $2w + 1$ bit and $w + 1$ bit register, respectively, while variable P

needs a full precision register since it is shared among consecutive "rows" (i.e., m iterations of the inner loop with constant j).

Authors in [8] extended Montgomery multiplication to binary fields $GF(2^n)$, by adopting polynomial representation and replacing the factor $R^{-1} = 2^{-n}$ with x^{-n}. With polynomial representation, $GF(2^n)$ field elements can be handled as binary polynomials and multiplication can be performed modulo an irreducible polynomial $N(x)$. Addition of $GF(2^n)$ elements is performed as a bitwise XOR of their components, while multiplication/division by powers of x are performed by left/right-shifting an element's components. As a result, the structure and the basic operations of Montgomery algorithm in $GF(2^n)$ turn out to be very similar to the integer/$GF(N)$ case. Essentially, the control-flow of the algorithm (including the above FIOS variant) remains unchanged, shift operations are also identical, while integer addition is replaced by a bitwise XOR. The $GF(2^n)$ counterpart of Algorithm 2 is presented, for example, in [13].

3 State-of-the-Art in Unified Field Arithmetic

Since the structure of Montgomery variants for $GF(N)$ and $GF(2^n)$ are similar, several authors have proposed *unified* hardware solutions for computing both operations with the same processing unit. To enable this approach, Savaş et al. proposed in [14] a basic building block able to perform a one-digit addition in both $GF(N)$ and $GF(2^n)$ fields. The basic component is the *Dual Field Adder*, i.e. an ordinary full adder whose carry input can be disabled, so that the sum output is simply the XOR of the two input bits (i.e., their $GF(2)$ sum). Figure 2 shows a possible implementation of such a component. Based on a similar idea, Großschädl [4] proposed a bit-serial unified multiplier processing the multiplicand in full precision. Montgomery modular reduction is computed by interleaving the addition of partial products and the modulus. A hardware

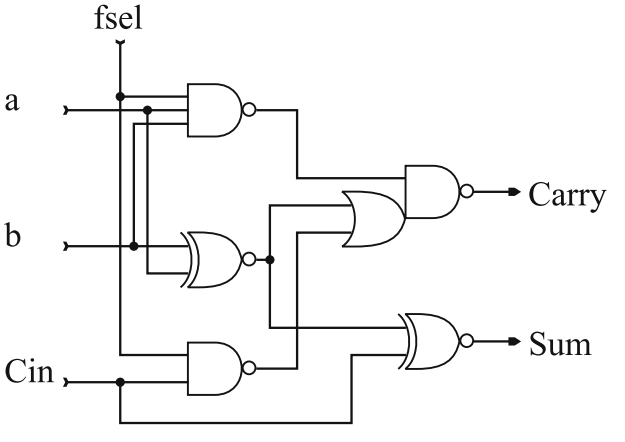


Fig. 2. An implementation of the Dual Field Adder [5, 6]

solution for dual-field arithmetic is also presented by Wolkerstorfer in [18]. The author introduces a low power design enabling short critical paths and high clock frequencies by using carry save adders. In [6], the authors present the design of a low-power multiply/accumulate (MAC) unit for efficient arithmetic in finite fields. The unit combines integer and polynomial arithmetic into a single functional unit supporting both $GF(N)$ and $GF(2^n)$ fields. The emphasis is mostly put on power consumption, as the authors show that a properly designed unified multiplier may consume significantly less power if used in polynomial mode compared to integer mode.

The fastest solution for unified field multiplication was proposed by Satoh and Takano [13]. They present a scalable elliptic curve cryptographic processor supporting both $GF(N)$ and $GF(2^n)$ finite fields. The core of the processor is a parallel dual-field multiplier, based on a Wallace tree scheme. The delay for a multiplication is logarithmic in the input-size, although it is different for the two types of fields. In fact, a sub-portion of the Wallace tree is used for obtaining a $GF(2^n)$ product, while the whole structure, including a fast carry propagation adder, is required for $GF(N)$ operations. The authors evaluate different parallelisms, developing the multiplier for word sizes of 8, 16, 32, or 64 bits, depending on the desired trade-off between area requirements and performance. One advantage of their approach is that it does not require any special full adder, such as the dual-field adder, unlike works in [4, 6, 14] and others. This makes it possible to optimize the partial product addition network. Furthermore, at a higher level, the performance of point multiplication over an elliptic curve is improved by converting on-the-fly the integer multiplicand in a redundant form.

Finally, a recent solution proposes a fast modular arithmetic-logic unit [12] that is scalable in the digit size and the field size. The datapath is based on chains of carry save adders to speed up arithmetic operations over large integers in $GF(N)$. This enables efficient execution of modular multiplication and addition/subtraction. The unit is prototyped in FPGA technology achieving interesting throughput levels, although inferior to the ASIC-based work presented in [13].

4 Parallel Montgomery Multiplier

In this section, we propose a novel unified architecture for parallel Montgomery multiplication supporting both $GF(N)$ and $GF(2^n)$ operations. Unlike previously proposed parallel multipliers, such as the solution in [6, 13], the hardware unit merges multiplication and Montgomery reduction, allowing a word-level modular multiplication to be performed is a single cycle. The proposed multiplier relies on a modified Booth recoding scheme for integer multiplication, and a radix-4 scheme for $GF(2^n)$ multiplication and Montgomery reduction. As a result, the number of partial products to be added in the parallel unit can be approximately halved, resulting in both reduced area and improved speed.

The basic full-precision algorithm for a radix-4 digit-serial interleaved Montgomery multiplication is given below (see for example [15]). For the sake of

clarity, we refer to the integer/$GF(N)$ version of the algorithm. As explained in Section 2, the extension to binary fields $GF(2^n)$ is straightforward, provided that a dual-field data path is available.

Algorithm 3. *Radix-4 Montgomery Modular Multiplication*

Input:

$2 < N < 4^k$,
$\tilde{N}$ such that $4^{k+1} \cdot 4^{-(k+1)} - N \cdot \tilde{N} = 1$,
$A = \sum_{i=0}^{k} A_i 4^i < 2N, \quad B = \sum_{i=0}^{k} B_i 4^i < 2N, \quad$ with $A_i, B_i < 4$

Output:

$P \equiv A \cdot B \cdot 4^{-(k+1)} \bmod N, \quad P < 2N$

Algorithm:

1. $P = 0$
2. for $i = 0$ to k
3. $\quad Q_i = (P_0 + B_i \cdot A_0) \cdot \tilde{N}_0 \bmod 4$
4. $\quad P = (P + B_i \cdot A + Q_i \cdot N)/4$

It can be easily proved that, by using $k+1$ iterations (i.e., by computing $A \cdot B \cdot 4^{-(k+1)} \bmod N$, $A, B < 2N$) the final value of P is still less than $2N$. In fact, we have $P = \frac{A \cdot B + Q \cdot N}{4^{k+1}} < \left[\frac{4N}{4^{k+1}} + \frac{Q}{4^{k+1}}\right] \cdot N < 2N$, where $Q = \sum_{i=0}^{k} Q_i 4^i$. Notice that Q_i only depends on the two least significant bits of $(P_0 + B_i \cdot A_0)$ and N, so it can be computed by a simple circuit or a look-up table. Its value is defined in such a way as to make the least significant digit of $(P + B_i A + Q_i N)_4$ zero at each iteration. Figure 3 gives an example of radix-4 Montgomery multiplication execution.

In the following, we will call $AA^{(i)}$ and $NN^{(i)}$ a partial product $B_i \cdot A$ and a multiple of the modulus $Q_i \cdot N$, respectively. In the case of radix-4, B_i and Q_i are 2-bit numbers. Thus, the value sets of $AA^{(i)}$ and $NN^{(i)}$ are as follows:

$$AA^{(i)} \in \{0, A, 2A, 3A\}, \quad NN^{(i)} \in \{0, N, 2N, 3N\}$$

requiring two extra adders to compute $3A$ and $3N$ on the fly. In the case of $GF(2^n)$ operations, using polynomial representation, $B_i(x)$ and $Q_i(x)$ are polynomial of degree less than 2, so the value sets of $AA^{(i)}(x)$ and $NN^{(i)}(x)$ are as follows:

$$AA^{(i)}(x) \in \{0,\ A(x),\ xA(x),\ xA(x) + A(x)\}$$
$$NN^{(i)}(x) \in \{0,\ N(x),\ xN(x),\ xN(x) + N(x)\}$$

In standard multipliers, Booth recoding scheme is normally used in order to avoid the expensive calculation of the multiple $3A$ in the $AA^{(i)}$ value set. The recoding scheme takes the bits of the multiplier $(b_{2i+1}, b_{2i}, b_{2i-1})$ as input and generates a recoded $AA^{(i)}$ according to Table 1, where b_{-1} is defined to be 0. As a consequence, Booth recoding scheme transforms the value set of $AA^{(i)}$ into $\{-2A, -A, 0, +A, +2A\}$. All elements in the set are calculated with simple operations such as bit inversion and/or bit shift. For $GF(2^n)$ operations, elements are

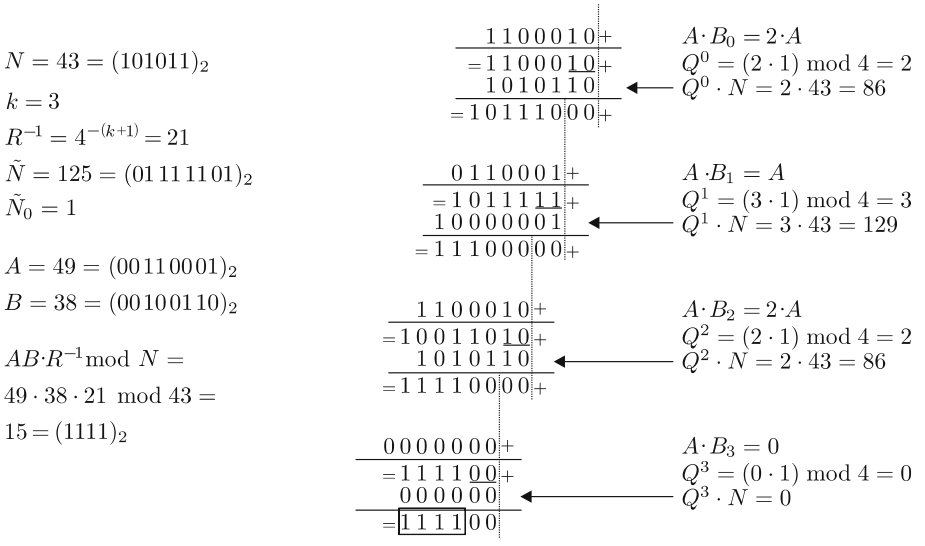


Fig. 3. An example of radix-4 Montgomery multiplication execution

Table 1. Partial product generation for integers and binary polynomials

Field Select	Three input bits			Recoded digit	Recoded partial product	Control signals		
$fsel$	b_{2i+1}	b_{2i}	b_{2i-1}	B_i	$AA^{(i)}$	inv	trp	shl
1	0	0	0	0	0	–	0	0
1	0	0	1	1	$+A$	0	1	0
1	0	1	0	1	$+A$	0	1	0
1	0	1	1	$+2$	$+2A$	0	0	1
1	1	0	0	-2	$-2A$	1	0	1
1	1	0	1	-1	$-A$	1	1	0
1	1	1	0	-1	$-A$	1	1	0
1	1	1	1	0	0	–	0	0
0	0	0	0	0	0	–	0	0
0	0	0	1	0	0	–	0	0
0	0	1	0	1	$A(x)$	0	1	0
0	0	1	1	1	$A(x)$	0	1	0
0	1	0	0	x	$xA(x)$	0	0	1
0	1	0	1	x	$xA(x)$	0	0	1
0	1	1	0	$x+1$	$xA(x)+A(x)$	0	1	1
0	1	1	1	$x+1$	$xA(x)+A(x)$	0	1	1

handled as binary polynomials. In this case, a pure radix-4 polynomial multiplication is adopted. In other words, multiples $AA^{(i)}(x)$, calculated as in Table 1, only depend on radix-4 digits (b_{2i+1}, b_{2i}).

For the proposed parallel Montgomery multiplier, in addition to summing partial products $AA^{(i)}$, we also need to sum modulus multiples $NN^{(i)}$ (or $NN^{(i)}(x)$ for $GF(2^n)$ multiplication). In [15] authors adopt a method named *Montgomery recoding scheme* to change the possible values of $NN^{(i)}$ so that they can all be obtained by simple shifts and inversions, similar to Booth recoding. Let (sp_1, sp_0) be the 2 bits in the least significant digit (LSD) of the partial product to be reduced $SP = P + AA$ and (n_1, n_0) be the 2 bits in the LSD of the modulus N. According to the input condition that N has to be odd, n_0 is always 1. Then, Montgomery recoding scheme takes (sp_1, sp_0, n_1) as input and generates a recoded $NN^{(i)}$ value according to Table 2, where Q_i represents the recoded quotient digit for an $NN^{(i)}$ multiple at the i-th iteration. Montgomery recoding scheme transforms the value set of NN into $\{-N, 0, +N, +2N\}$.

In polynomial mode the addition becomes a bitwise XOR. For this reason, we need to sum a different value of $NN^{(i)}(x)$ in order to reduce the least significant digits $(sp_1, sp_0)_2$ of $SP(x) = P(x) + AA(x)$. Notice that, in order to perform modular multiplication in $GF(2^n)$ with the same recoding scheme, we use an additional control signal, $fsel$ (field select), which allows us to switch between integer-mode ($fsel = 1$) and polynomial mode ($fsel = 0$). In Table 2 we show the unified Montgomery recoding scheme, including polynomial mode for $GF(2^n)$.

Due to the two recoding schemes, it is easy to calculate all the elements in the value sets of $AA^{(i)}$ and $NN^{(i)}$. Notice that, for integer multiplication, this technique changes the range of the Montgomery algorithm output, which may now be negative.

Table 2. Montgomery moduli generation for integers and binary polynomials

Field Select	**Three input bits**			**Recoded quotient**	**Recoded modulus**	**Control signals**		
$fsel$	sp_1	sp_0	n_1	Q_i	$NN^{(i)}$	inv	trp	shl
1	0	0	0	0	0	–	0	0
1	0	0	1	0	0	–	0	0
1	0	1	0	-1	$-N$	1	1	0
1	0	1	1	$+1$	$+N$	0	1	0
1	1	0	0	$+2$	$+2N$	0	0	1
1	1	0	1	$+2$	$+2N$	0	0	1
1	1	1	0	$+1$	$+N$	0	1	0
1	1	1	1	-1	$-N$	1	1	0
0	0	0	0	0	0	–	0	0
0	0	0	1	0	0	–	0	0
0	0	1	0	1	$N(x)$	0	1	0
0	0	1	1	$x+1$	$xN(x) + N(x)$	0	1	1
0	1	0	0	x	$xN(x)$	0	0	1
0	1	0	1	x	$xN(x)$	0	0	1
0	1	1	0	$x+1$	$xN(x) + N(x)$	0	1	1
0	1	1	1	1	$N(x)$	0	1	0

The core of the proposed parallel Montgomery multiplier is made of a sequence of *Partial Product Generators* (PPGs) and *Montgomery Modulues Generators* (MMGs), wired as in Figure 4. Their outputs are summed together, making up an unrolled implementation of the loop in Algorithm 3.

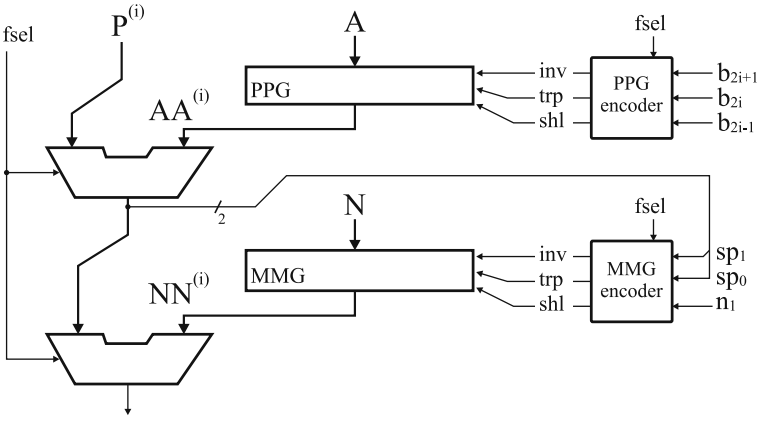


Fig. 4. The basic row in the proposed radix-4 parallel Montgomery multiplier

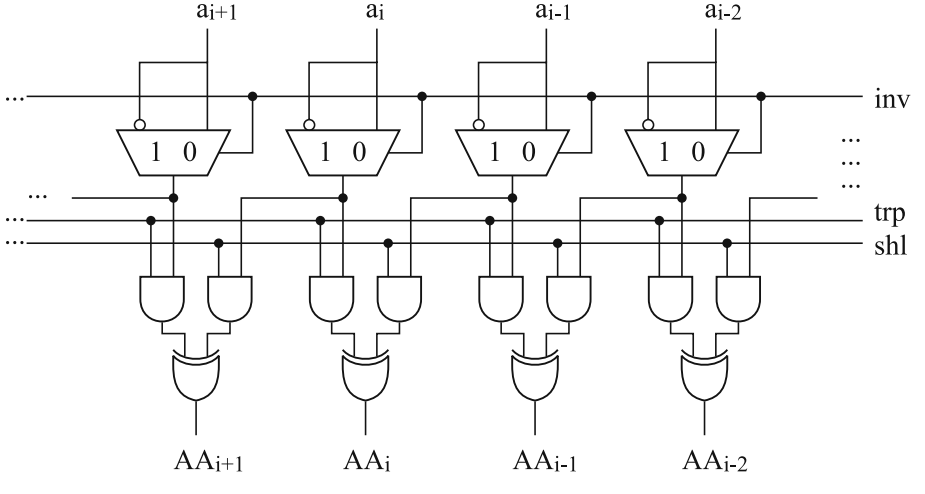


Fig. 5. The internal structure of a Partial Product Generator (PPG) [6]. A similar circuit is used for the Montgomery Modulus Generator (MMG).

The structures of PPGs and MMGs are identical, and are similar to that described in [6]. The corresponding circuit is depicted in Figure 5. PPGs and MMGs are controlled by an encoder via the three signals *inv* (invert), *trp* (transport), and *shl* (shift left), which represent the recoded digit B_i and the recoded

quotients Q_i, respectively. Precisely, when $inv = 1$, the corresponding modulus is negative, i.e. $NN^{(i)} = -N$. Control signal $trp = 1$ means $NN^{(i)} = N$ (no left-shift). On the other hand, when $shl = 1$, a 1-bit left-shift has to be performed, i.e. $NN^{(i)} = 2N$. Finally, $NN^{(i)} = 0$ is generated by $trp = shl = 0$. Notice that in $GF(2^n)$ mode, i.e. when $fsel = 0$, the input value $inv = 0$, $trp = 1$, $shl = 1$ generates the multiple $xN(x) + N(x)$ needed for radix-4 Montgomery reduction. Similar considerations hold true for the Partial Product Generator used to calculate the values of $AA^{(i)}$.

Selection signals inv, trp, and shl depend on the multiplier digit bits $b_{2i+1}, b_{2i}, b_{2i-1}$, in the case of PPG, and the two least significant bits (sp_1, sp_0) of SP and n_1, in the case of MMG, according to the equations below, derived from Table 2. For PPGs, selection signals can be written as follows:

$$\begin{aligned} inv &= fsel \cdot b_{2i+1} \\ trp &= \overline{fsel} \cdot b_{2i} + b_{2i} \cdot \overline{b_{2i-1}} + fsel \cdot \overline{b_{2i}} \cdot b_{2i-1} \\ shl &= \overline{fsel} \cdot b_{2i+1} + b_{2i+1}\overline{b_{2i}} \cdot \overline{b_{2i-1}} + fsel \cdot \overline{b_{2i+1}} \cdot b_{2i} \cdot b_{2i-1} \end{aligned} \tag{1}$$

For MMGs, selection signals can be written as follows:

$$\begin{aligned} inv &= fsel \cdot \overline{sp_1} \cdot \overline{n_1} + \overline{fsel} \cdot sp_1 \cdot sp_0 \cdot n_1 \\ trp &= sp_0 \\ shl &= sp_1 \cdot \overline{sp_0} + \overline{fsel} \cdot sp_1 \cdot \overline{n_1} + \overline{fsel} \cdot \overline{sp_1} \cdot sp_0 \cdot n_1 \end{aligned} \tag{2}$$

A parallel $(w \times w)$-bit multiplier for signed/unsigned modular multiplication contains $\lfloor w/2 \rfloor + 1$ PPGs and $\lfloor w/2 \rfloor + 1$ MMGs and the same number of PPG/MMG encoder circuits generating selection signals inv, trp, and shl.

Partial products $AA^{(i)}$ and moduli $NN^{(i)}$ are $w + 2$ bits long as they are represented in two's complement form. Besides a bitwise complement of their binary representation, negative multiples need a 1 to be added at the least significant position of the partial product. Let $ca^{(i)}$, $cn^{(i)}$ denote such bits. We will thus have $ca^{(i)} = 1$ and $cn^{(i)} = 1$ when the partial products $AA^{(i)}$ and the Montgomery moduli $NN^{(i)}$ are negative, respectively.

Notice that the parallel multiplier handles internal operands in carry-save form to reduce the architectural critical path. Special care must be put, in this case, for summing negative numbers. In principle, we would need to sign extend possibly negative partial products $AA^{(i)}$ and moduli $NN^{(i)}$ to full $2w$-bit length, causing a large waste of full-adders in each row of the multiplier. By recoding the addends, however, we can have only positive-weight bits to be added in the multiplier, provided that a suitable constant K is added along with them as the last row in the multiplier array [3]. Let $P = (-2^n)p_n + \sum_{i=0}^{n-1} 2^i p_i$ be a two's complement number. Recoding works as follows:

$$P = (-2^n)p_n + \sum_{i=0}^{n-1} 2^i p_i = -2^n + \left[2^n \overline{p_n} + \sum_{i=0}^{n-1} 2^i p_i \right]$$

where all number's components have a positive weight, while the only negative term is constant. If we have many partial products P to be summed together,

we can thus recode them as shown above, sum their positive components p_i (including $\overline{p_n}$) by adopting a usual array multiplier, separate their constant terms -2^n and accumulate them in a single full-length constant K to be added as the last row.

Some further optimizations can be applied to reduce the architectural critical path of the design. Let (S, C) denote a carry-save pair. In a non-optimized Montgomery multiplier with modified Booth recoding, the sum of the partial products and the Montgomery moduli in the carry-save stages (CSAs) proceeds as follows:

$$\begin{aligned}
&\ldots \\
&\left(Stmp^{(i)}, Ctmp^{(i)}\right) = AA^{(i)} + S^{(i)} + C^{(i)} \\
&\left(S^{(i+1)}, C^{(i+1)}\right) = N^{(i)} + Stmp^{(i)} + Ctmp^{(i)} \\
&\left(Stmp^{(i+1)}, Ctmp^{(i+1)}\right) = AA^{(i+1)} + S^{(i+1)} + C^{(i+1)} \\
&\ldots
\end{aligned}$$

$(Stmp^{(i)}, Ctmp^{(i)})$ is given by the sum of the i-th recoded partial product $AA^{(i)}$ and the previous $AA^{(j)}$, $0 \leq j < i$ with the recoded moduli $NN^{(j)}$, $0 \leq j < i$. Recoding of partial products and moduli, however, also implies the sum of the sign bits ca and cn. In principle, this would require the use of two additional CSA stages. Indeed, since ca and cn are in the right-most positions of partial products and moduli, we can juxtapose them with other partial products and moduli down in the multiplier array, since these are left-shifted and so leave free slots on the right. For the sake of clarity, Figure 6 gives a practical example of this organization, for the case $w = 6$. The generic stage within the proposed multiplier scheme performs the following operation:

$$\begin{aligned}
&\ldots \\
&\left(Stmp^{(i)}, Ctmp^{(i)}\right) = S^{(i-1)} + C^{(i-1)} + AA^{(i+1)} + ca^{(i)} \\
&\left(S^{(i)}, C^{(i)}\right) = Stmp^{(i)} + Ctmp^{(i)} + NN^{(i)} + cn^{(i)} \\
&\ldots
\end{aligned}$$

Overall, we need:

- $\lfloor w/2 \rfloor + 1$ CSA stages to compute $Stmp^{(i)}$, $Ctmp^{(i)}$
- $\lfloor w/2 \rfloor + 1$ CSA stages to compute $S^{(i)}$, $C^{(i)}$

The main optimizations adopted consist in (see Figure 6):

- reorganizing the sum of the LSB $ca^{(i)}$ and $cn^{(i)}$ of the output carry vector in order to avoid additional CSA stages. Notice that, although interchangeable for the accumulation of partial products and moduli, bits $ca^{(i)}$ are needed for the determination of the next modulus $NN^{(i+1)}$ to be summed. The MMG selection circuit must take this into account, and read also the bit $ca^{(i)}$ to anticipate the evaluation of $NN^{(i+1)}$
- postponing the sum of the least significant bits $\{s_1^{(i)}, s_0^{(i)}, c_0^{(i)}\}$ of $S^{(i)}$ and $C^{(i)}$ respectively, to save area and CSA stages. Similar to the previous optimization, these operations imply a complication of the MMG selection network,

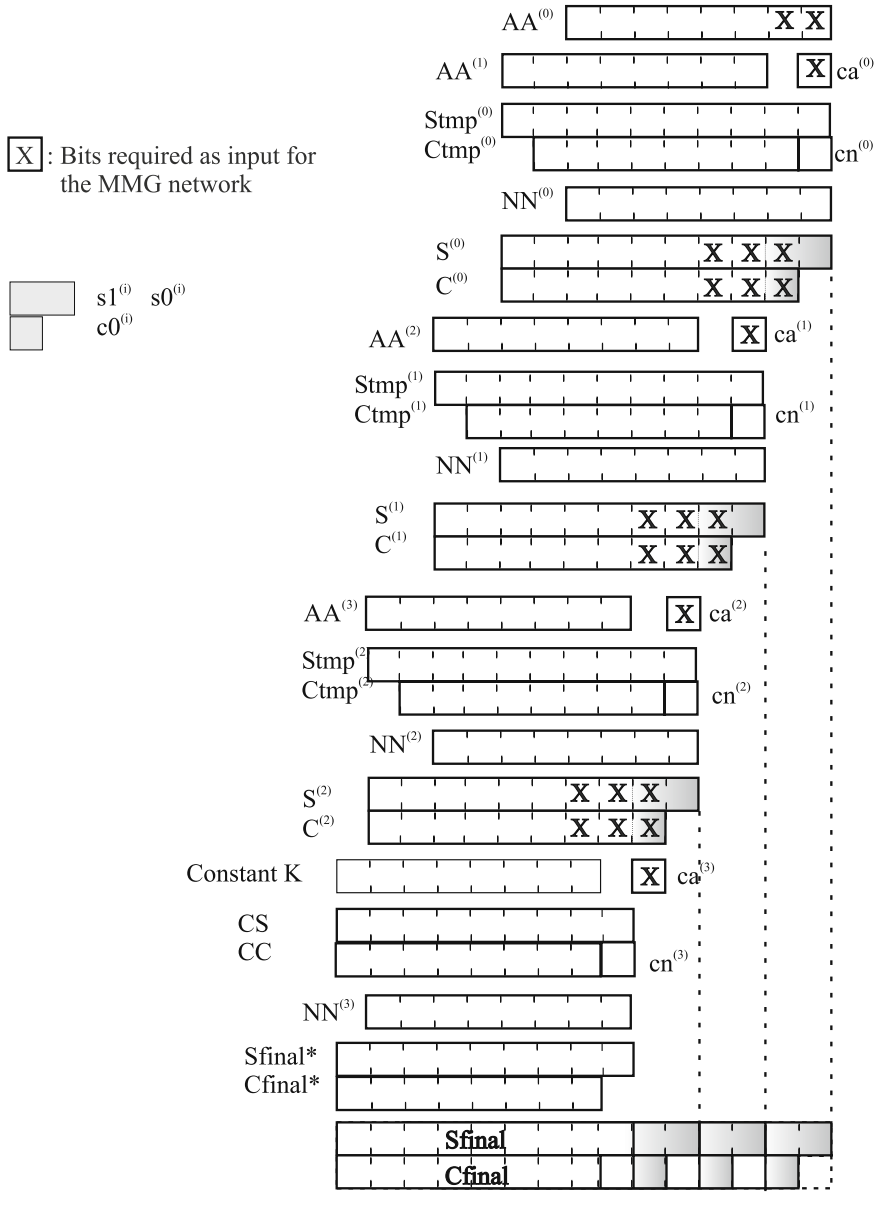


Fig. 6. Addition of Partial Products and Montgomery Moduli with Booth Recoding in an optimized scheme for $w = 6$

which needs more inputs to infer the values of bits sp_1, sp_0, handled here in redundant, carry-save form

– reversing the order of the sum of $AA^{(i)}$, $NN^{(i)}$, in order to improve the critical path. This operation does not alter the computation of $NN^{(i)}$, due

to the encoding network previously described, which tests the bits needed for the computation of the modulus before the addition of the $AA^{(i+1)}$ vector.

After the final stage, we need a *Dual-Field Carry-Look-Ahead* adder (not shown in Figure 6) that converts the Carry/Sum pair back to non-redundant form. The structure of the Dual-Field Carry Look-Ahead is depicted in Figure 7. The essential idea is to disable carry generation throughout the adder structure in $GF(2^n)$ mode, i.e. when $fsel = 0$. In this case, all internal carry signals C_i are zero, independent of propagate conditions P_i. As a result, output bits S_i coincide with propagate signals $P_i = a_i \oplus b_i$, i.e. a $GF(2)$ sum. The fundamental advantage of this solution is that it enables the reuse of highly-optimized fast carry look-ahead circuits which are normally available for a given target technology.

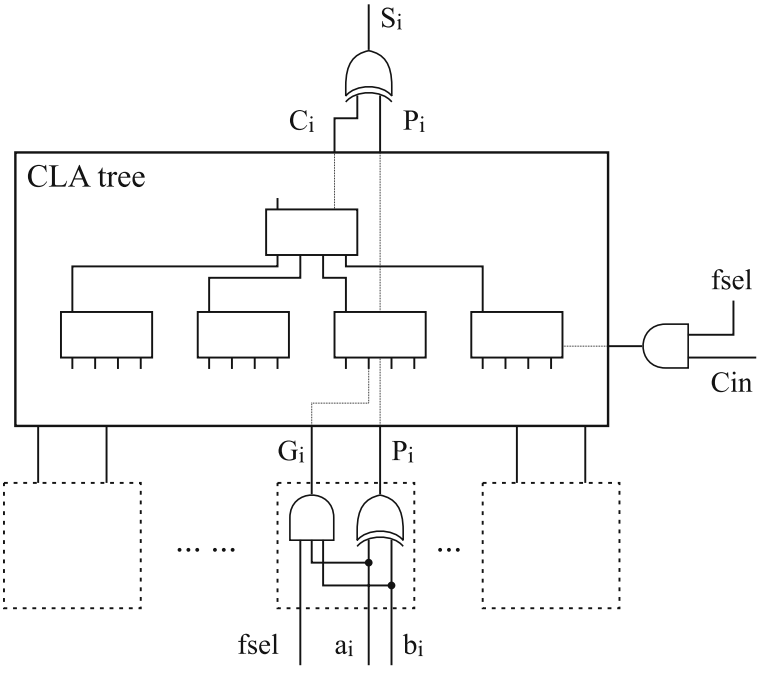


Fig. 7. Dual-Field Carry Look-Ahead adder

5 Pipelined Montgomery Multiplier

Previous works (e.g. Satoh and Takano's 64-bit multiplier [13]) suggest that it is normally convenient to adopt a large parallelism for achieving higher throughput levels. Our parallel architecture has a relatively complex selection network and a linear critical path, which results in large time delays as the word size increases. In order to achieve high throughput levels and propose a scalable scheme, we present in this section a pipelined architecture, using the parallel

unit as the basic building block. The architecture can process single words of w bits. By partitioning long operands into w-bit words, a full-length Montgomery multiplication can be carried out based on the FIOS variant of the Montgomery algorithm (see Algorithm 2).

We implemented the unit for a bit length w of 64 bits. Figure 8 shows the internal structure of a 64x64-bit unit composed of eight pipelined modules. The "smaller" multipliers on the right are in fact four instances of the parallel unit presented in the previous section: in other words, they can generate the recoded multiples (i.e. Q_i recoded as the signals shl, trp, inv) of the modulus N and the multiplicand A for the whole row, in addition to adding them. The four "larger" multipliers on the left side of Figure 8, on the other hand, only need to sum the multiples of N and A, as determined by right-multipliers. Since left-multipliers are much simpler in their structure and have consequently a shorter delay, they are designed so that they process longer data. Furthemore, right-multipliers also need an additional input signal, called $first\_word$, which can enable/disable the generation of multiples of the modulus Q_i. This is necessary to process intermediate words during a row scanning of the FIOS algorithm (steps 5-8 in Algorithm 2), where we need to process new w-bit words in the pipelined unit reusing a previously generated value of Q_i.

As we use two's complement representation in the carry-save form, it is desirable to keep intermediate sums in carry-save form and convert the final result back to binary form only at the end of the pipelined structure. We thus need to transfer carry-save numbers between subsequent multiplier modules having different output/input sizes. This required the use of a suitable technique [19] to sign-extend the carry-save pair and properly propagate sign information.

Figure 9 describes how the pipelined unit is used to process multi-word operands, showing how the portions of the operands are scheduled in the pipeline. Numbers in parentheses indicate which of the eight blocks in the unit works on which portion of operands A, N, and B at which clock cycle (starting from cycle 1 for the top right-most multiplier). The unit has a latency of eight cycles, introducing a stall at the end of each row only if the number of words m is less than 8. This makes the unit particularly suitable for high-performance multiplication on large multi-word operands, when many words on the same row are to be processed consecutively. The throughput of the architecture is one multiplication word per clock cycle in this case.

Right modules in Figure 8 have a 16×16 bit size, while left modules have a 48×16 bit size. The architecture is designed so that the single blocks, especially the smaller right-multipliers, can be optimized to minimize the clock period. Notice that, with a slight modification to the scheme of Figure 8, the first and the last row (possibly connected to an external bus) may be designed with a smaller height than the multipliers in the second and third row, so as to balance the delay of each stage in the pipeline. The carry-save stages are followed by a Dual-Field Carry-Look-Ahead adder, not shown in Figure 8, converting results back to the non-redundant form.

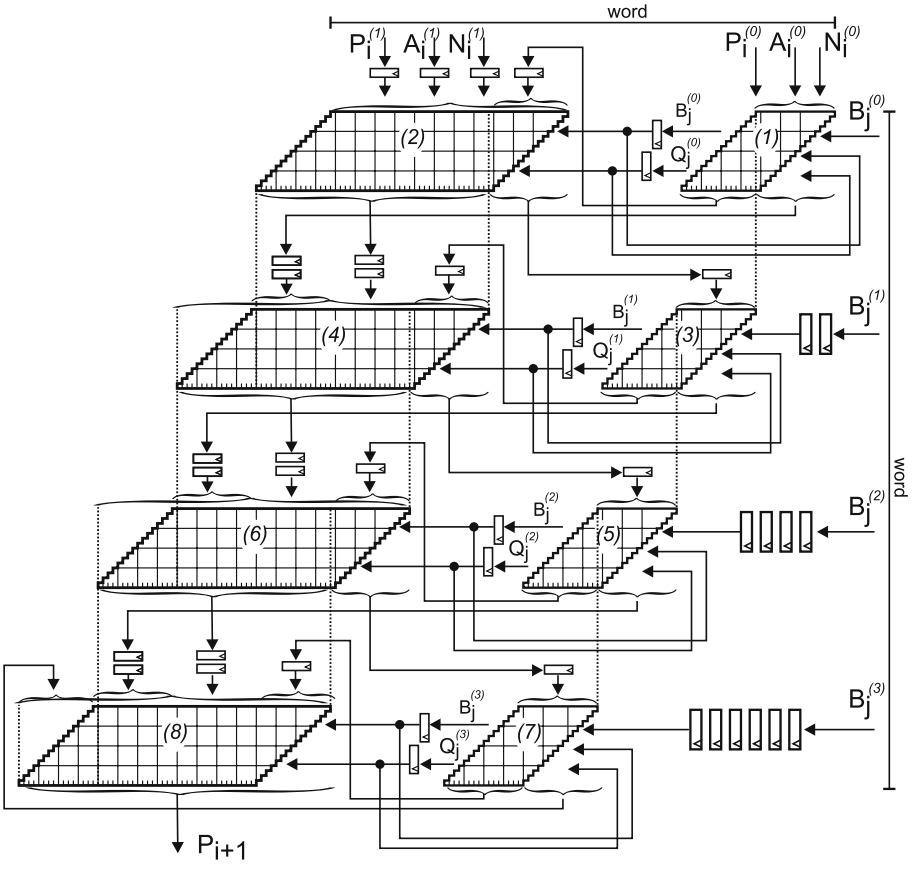


Fig. 8. Pipelined architecture of the arithmetic core. Superscript numbers in parentheses indicate the different portions into which a single w-bit word is partitioned inside the pipelined unit.

The overall architecture of the dual-field multiplication unit is shown in Figure 10. From the scheme in Figure 9 it is clear that at the beginning of each row we need to drive in the unit three different words, namely A_0, N_0, and B_j, while the words of the intermediate result P are stored internally in a dedicated memory. This is the only case when we need three concurrent accesses to the external memory. To overcome this problem and limit the number of external buses, we observe that it is convenient to store the first word of the modulus N, N_0, in an internal register. This trick only requires w additional flip-flops and some selection logic, independent of the full size of the operands and the modulus. N_0 is stored before starting a multiplication (or a sequence of multiplications sharing the same modulus). As a consequence, at the beginning of each row in the multiplication pipeline we only need A_0 and B_j, while for the subsequent words we need A_i and N_i (B_j is constant through the row), which are driven into the multiplication unit through the same pair of buses.

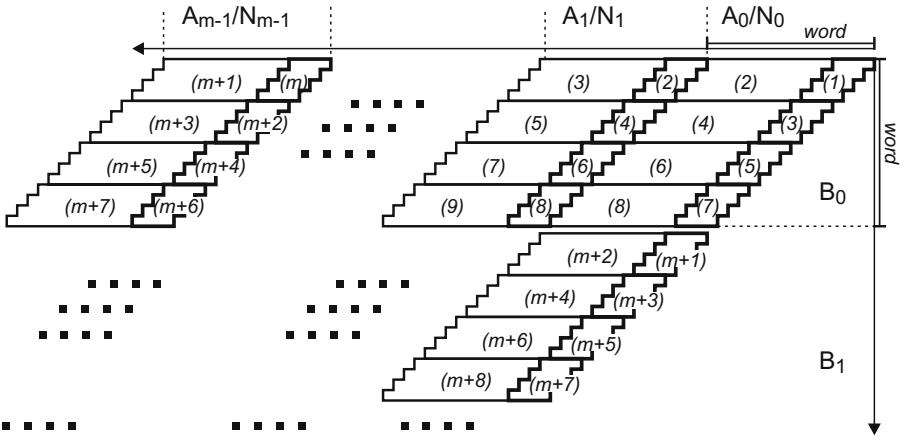


Fig. 9. Scheduling for a multi-word Montgomery multiplication. A_i, N_i, and B_j are w-bit words. Word sub-portions enter the pipelined w-bit unit according to the schedule indicated in parentheses.

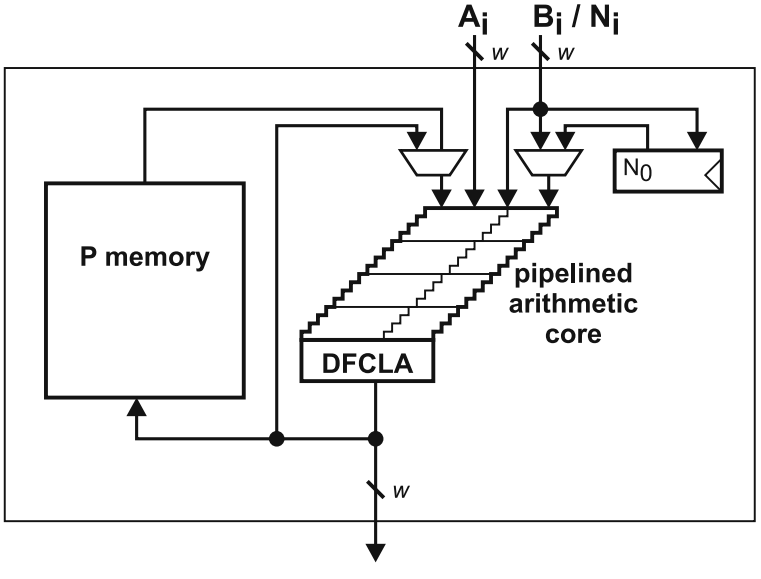


Fig. 10. Overall architecture of the dual-field multiplication unit

6 Experimental Results and Comparisons

The pipelined multiplier core of Figure 8 was described in VHDL and then synthesized for a CM0S 0.18μm standard cell library technology by using Cadence Build Gates synthesis tool. Post-synthesis area requirements are estimated to be $1316k\mu m^2$, while the minimum clock period is $12.2ns$.

Although there are different related works presenting unified Montgomery multiplication (see Section 3), we only compare our results with the multiplier introduced in [13], since it achieves the highest throughput among the various works available in the literature. Both their work and ours are synthesized as a CMOS ASIC, but the design in [13] relies on a $0.13\mu m$ technology, more advanced than the $0.18\mu m$ target used in our design. When implemented in the same technology, our solution is thus likely to enable even better improvements than emphasized in the following discussion. The table below reports some results referred to integer (i.e. $GF(N)$) modular multiplication for different operand lengths, choosing the field sizes indicated by NIST standards for elliptic curve cryptography. Performance improvements are especially evident in terms of clock counts.

	Satoh and Takano [13]		**This work**	
	ASIC $0.13\mu m$ clock period: 7.26 ns		ASIC $0.18\mu m$ clock period: 12.2 ns	
$GF(N)$ field size	clock count	throughput [Mbit/s]	clock count	throughput [Mbit/s]
192	45	587.7	27	582.9
224	66	467.5	36	510.0
256	66	534.3	36	582.9
284	91	429.9	45	517.3
521	231	310.7	90	474.5

Authors in [13] emphasize that a higher frequency could be used if the unified multiplier were used only in $GF(2^n)$ mode, since the output of their unit is connected, in this case, to a subportion of the Wallace tree in the multiplier. If a dual clock frequency were allowed, $GF(2^n)$ operations would be worse in our case, while remaining superior for the more critical integer/$GF(N)$ arithmetic. In the case a dual frequency implementation is not possible, on the other hand, our multiplier has better performance also for $GF(2^m)$, and comparisons with the multiplier in [13] appear similar to those given in the above table for the integer/$GF(N)$ case.

7 Conclusions

The approach presented in this paper, based on dual-field parallel Montgomery multiplication, proves to be a promising choice, especially for the reduction in clock count. As a future work, we plan to study new techniques to further reduce the delay of the parallel Montgomery unit, described in Section 4, thereby improving the clock period and the throughput achievable by the pipelined unit.

Acknowledgements

This work was partially supported by Regione Campania and Ditron S.R.L. in the framework of “Progetto Metadistretto del settore ICT - Misura 3.17:

Sistema di comunicazione per l'integrazione delle informazioni nella distribuzione commerciale dei punti vendita".

References

1. Blake, I.F., Seroussi, G., Smart, N.P.: Elliptic Curves in Cryptography. Cambridge University Press, Cambridge (1999)
2. Blum, T., Paar, C.: High-Radix Montgomery Modular Exponentiation on Reconfigurable Hardware. IEEE Transactions on Computers 50, 759–764 (2001)
3. Burgess, N.: Removal Of Sign-Extension Circuitry From Booth's Algorithms Multiplier-Accumulators. Electronics Letters 26, 1413–1415 (1990)
4. Großschädl, J.: A bit-serial unified multiplier architecture for finite fields $GF(p)$ and $GF(2^n)$. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, p. 202. Springer, Heidelberg (2001)
5. Großschädl, J., Kamendje, G.A.: Instruction set extension for fast elliptic curve cryptography over binary finite fields $GF(2^m)$. In: Proceedings of the 14th IEEE Int. Conference on Application-specific Systems, Architectures and Processors (ASAP 2003), pp. 455–468. IEEE Computer Society Press, Los Alamitos (2003)
6. Großschädl, J., Kamendje, G.A.: Low Power Design of a Functional Unit for Arithmetic in Finite Fields $GF(p)$ and $GF(2^m)$. In: Chae, K.-J., Yung, M. (eds.) WISA 2003. LNCS, vol. 2908, pp. 227–243. Springer, Heidelberg (2004)
7. Koç, Ç.K., Acar, T., Kaliski, B.S.: Analyzing and Comparing Montgomery Multiplication Algorithms. IEEE Micro 16, 26–33 (1996)
8. Koç, Ç.K., Acar, T.: Montgomery Multiplication $GF(2^n)$. Designs, Codes and Cryptography 14, 57–69 (1998)
9. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44, 519–521 (1985)
10. Örs, S.B., Batina, L., Preneel, B., Vandewalle, J.: Hardware Implementation of a Montgomery Modular Multiplier in a Systolic Array. In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2003), p. 184b (2003)
11. Rivest, R.L., Shamir, A., Adleman, L.: A Method for obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21, 120–126 (1978)
12. Sakiyama, K., Preneel, B., Verbauwhede, I.: A Fast Dual-Field Modular Arithmetic Logic Unit and its Hardware Implementation. In: Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2006), pp. 787–790 (2006)
13. Satoh, A., Takano, K.: A Scalable Dual-Field Elliptic Curve Cryptographic Processor. IEEE Transanctions on Computers 52, 449–460 (2003)
14. Savaş, E., Tenca, A.F., Koç, Ç.K.: A Scalable and Unified Multiplier Architecture for Finite Fields $GF(p)$ and $GF(2^m)$. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 281–296. Springer, Heidelberg (2000)
15. Son, H.K., Oh, S.G.: Design and Implementation of Scalable Low-Power Montgomery Multiplier. In: Proceedings of the IEEE International Conference on Computer Design (ICCD 2004), pp. 524–531 (2004)
16. Tsai, W.C., Shung, C.B., Wang, S.J.: Two systolic architectures for modular multiplication. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 8, 103–107 (2000)

17. Walter, C.D.: Systolic Modular Multiplication. IEEE Transactions on Computers 42, 376–378 (1993)
18. Wolkerstorfer, J.: Dual-field arithmetic unit for $GF(p)$ and $GF(2^m)$. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 500–514. Springer, Heidelberg (2003)
19. Tenca, A.F., Tawalbeh, L.A.: Carry-Save Representation is Shift-Unsafe: The Problem and Its Solution. IEEE Transanctions on Computers 55, 630–635 (2006)

A Temperature-Aware Placement and Routing Algorithm Targeting 3D FPGAs

Kostas Siozios and Dimitrios Soudris

National Technical University of Athens (NTUA),
School of Electrical & Computer Engineering,
9 Heroon Polytechneiou, Zographou Campus, 157 80 Athens, Greece
{ksiop, dsoudris}@microlab.ntua.gr

Abstract. In current reconfigurable architectures, the interconnect structures increasingly contribute to the delay and power consumption budget. The demand for increased clock frequencies and logic availability (smaller area foot print) makes the problem even more important, leading among others to rapid elevation in power density. Three-dimensional (3D) architectures are able to alleviate this problem by accommodating a number of functional layers, each of which might be fabricated in different technology. Since power consumption is a critical challenge for implementing applications onto reconfigurable hardware, a novel temperature-aware placement and routing (P&R) algorithm targeting 3D FPGAs, is introduced. The proposed algorithm achieves to redistribute the switched capacitance over identical hardware resources in a rather "balanced" profile, reducing among others the number of *hotspot* regions, the maximal values of power sources at *hotspots*, as well as the percentage of device area that consumes high power. For evaluation purposes, the proposed approach is realized as a new CAD tool, named 3DPRO (3D-Placement-and-Routing-Optimization), which is part of the complete framework, named 3D MEANDER. Comparing to alternative solutions, the proposed one reduces the percentage of silicon area that operates under high power by 63%, while it leads to energy savings (about 9%), with an almost negligible penalty in application's delay ranging from 1% up to 5%.

1 Introduction

For decades, semiconductor manufacturers have been shrinking transistor size in ICs to achieve the yearly increases in speed and performance described by Moore's Law, which exists only because the RC delay was negligible in comparison with signal propagation delay [1]. For submicron technology, however, the RC delay becomes a dominant factor. This has generated many discussions concerning the end of device scaling as we know it, and has hastened the search for solutions beyond the perceived limits of current 2D devices.

One emerging solution to this problem is the 3D integration, which replaces a large number of long interconnects needed in 2D structures with shorter ones. Such architectures mitigate many of the limitations that the 2D devices exhibit. Among others, they provide: (*i*) higher logic density in the same foot print area, (*ii*) shorter

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 211–231, 2010.

interconnections among the logic blocks, (*iii*) reduced signal propagation delay, (*iv*) greater versatility and resource utilization, and (*v*) lower power consumption.

One of the most critical challenges for efficient application implementation in 3D FPGAs is the power management, and hence the thermal problem, which has already been studied for 2D architectures [6, 7, 17]. This problem is exacerbated in the 3D devices for two reasons: (*i*) the vertically stacked layers cause a rapid increase of power density [9], and (*ii*) the thermal conductivity of the dielectric layers inserted between device layers for insulation is very low compared to silicon and metal.

Moreover, an obvious consequence of this trend is the increased power consumption per area unit. In recent years, power density in 2D FPGAs has doubled every three years [1], and this rate is expected to increase as feature sizes, frequencies and technologies scale faster than operating voltages. As the power density will continue increasing in future technologies (according to "*A-power*" law), the power consumption is regarded as a limiting factor to the increasing scales of integration predicted by Moore's law [1].

Thermal management of Field-Programmable Gate Array (FPGA) devices is more critical compared to ASIC solutions, as they dissipate more power, while their operating temperatures usually exceed the critical one. Also, the leakage current increases exponentially with temperature, causing a positive feedback loop between leakage power and temperature.

Eliminating and managing power consumption for reconfigurable architecture requires appropriate algorithm support. Realizing applications on 2D FPGAs is a well studied problem; however, there are only a few solutions regarding 3D architectures [8, 13, 14, 16].

In [13] a P&R approach for 3D ICs is presented, having as criterion to minimize the total wire-length, the applications delay, and the on-chip temperature. Even though the framework supports reconfigurable architectures, however, the thermal feature is available sorely for ASIC designs.

A similar approach is shown in [14], where the P&R algorithm optimizes the energy consumption and the thermal profile of a 3D standard-cell device under the supplied timing constraint. The employed algorithm focuses on the energy consumption of interconnect-related components. Unfortunately, the software implementation is not publically available, in order to evaluate this approach against to our proposed solution.

In [16] a thermal-driven 3D floor-planning algorithm that provides a trade-off between runtime and quality is presented. The algorithm tries to reduce the total wire-length, as well as the maximum on-chip temperature, compared to a non thermal-driven approach.

In [8] a P&R algorithm and its software implementation targeting to explore alternative interconnection schemes for 3D FPGAs are introduced. The employed cost functions pay effort to minimize the application delay, the power/energy consumption, as well as the total wire length, ignoring about their distribution. This tool is part from an open-source CAD framework, named *3D MEANDER*, for mapping applications onto 3D FPGAs.

All these approaches realize digital applications on 3D devices having as goal to minimize the total power/energy consumption of the design, ignoring about the spatial distribution of its sources. Moreover, none of them is aware during the P&R

procedure about spatial distribution of parameters that affect the power/energy consumption (*i.e.*, switched capacitance). This results both to increased power/energy consumption, as well as to significant variations of on-chip temperature values across the 3D device. Among others, this non-uniformity in power consumption leads to increased cooling costs, as the IC packaging has to be designed for the worst case scenario.

The rest paper is organized as follows. In Section 2, we formulate the temperature-aware P&R problem, while the employed methodology in order to derive the temperature model is described in Section 3. The algorithmic steps of the proposed temperature-aware P&R algorithms are introduced in Section 4. Section 5 evaluates the efficiency of applying such a temperature-aware approach against to other implementations for application mapping, while conclusions are summarized in Section 6.

2 Problem Formulation

Power consumption of FPGAs is generally grouped into three categories: (*i*) dynamic power, (*ii*) static power, and (*iii*) interface (I/O) power. These components are governed by the process technology and traditionally maintain constant percentages of the device's total power. The dynamic part of power consumption (formulated in the Equation (1)), occurring due to signal transition as the load capacitance is charged (or discharged), still dominates the total power consumption. In this equation, f represents the clock frequency of the signal, V_{dd} is the supply voltage, while Cap_i and $Activity_i$ are the capacitance and switching activity, respectively, of element i.

$$P_{switching} = 0.5 \cdot f \cdot V_{dd}^2 \cdot \sum_{i=1}^{Nets} \{Cap_i \cdot Activity_i\} \quad (1)$$

When a lower bound on the supply voltage is set by external constraints (as often happens in real-world designs), or when the performance degradation due to lowering of the supply voltage is intolerable, then the only means of reducing power consumption is by lowering the effective capacitance and/or the switching activity (*i.e.*, switched capacitance). Throughout this paper, we discuss an algorithm for managing the spatial distribution of this product ($Cap_i \cdot Activity_i$) over the 3-D FPGA device, leading to a more "*uniform*" temperature profile.

Definition: Application Hypergraph

We consider as application hypergraph a directed hypergraph $AppG(L, N)$, where each vertex $l_i \in L$ represents a logic functionality of the target application, while the directed hyperedge $n_{i,j} \in N$ encodes the communication between logic functionalities l_i and l_j. The weight associated to hyperedge $n_{i,j}$, denoted as $communication\_weight_{i,j}$, represents the communication load/bandwidth from vertex l_i to l_j.

Definition: Platform Graph
We consider as platform graph a directed graph $PlatG(C,W)$ where each vertex $c_i \in C$ represents an element of the target architecture (*e.g.*, logic block, processor, memory, etc.), while the directed edge $w_{i,j} \in W$ denotes a communication path between hardware elements c_i and c_j. The weight of the edge $w_{i,j}$, denoted as $interconnection\_weight_{i,j}$, encodes the fabricated interconnection hardware resources among these logic blocks.

3D Temperature-Aware Placement and Routing Problem
Given the architecture graph $PlatG(C,W)$ consisted by a set of V ($V = i \times j \times k$) slices (S), where $S = \{S_1(x_1, y_1, z_1), \cdots, S_v(x_i, y_j, z_k)\}$ find a placement ($Place: L \rightarrow C$) and a routing ($Route: N \rightarrow W$) of the application hypergraph on the available hardware resources (platform graph), in order each logic function (L) and the appropriate communication (N) to occupy uniquely a logic resource (C) and the available routing fabric (W), respectively. The derived P&R solution is accepted if the following conditions are satisfied:

(i) $S_i \cap S_j = 0$ for all the $i,j \in PlatG$.
(ii) The interconnection of each layer is accomplished with the minimum routing resources.
(iii) Distribute uniformly the switched capacitance over the 3D device.
(iv) Meeting timing/power/area constraints of the application.
(v) Employ an acceptable number of vertical links (in terms of the selected 3D bonding technology).

The proposed temperature-aware P&R solution was evaluated against to existing (*i.e.*, non temperature-aware) P&R algorithms with the usage of the 20 biggest MCNC benchmarks [15]. During this experimental setup, the P&R algorithms were applied to identical (*i.e.*, with same amount of logic resources and interconnection fabric) 3D FPGAs. The results show significant reduction (about 63%) on area percentage that operates under high temperatures, while we also achieve energy savings about 9%.

3 Methodology for Deriving the Temperature Model

The proposed methodology for deriving the employed temperature model that calibrates our P&R algorithms, is depicted in Figure 1. For this reason, a representative number of benchmarks from [15] were implemented (with the same P&R algorithms) onto 3D FPGAs. Then we visualize the variation of switched capacitance over each layer, in order to determine the number, as well as the spatial distribution of *hotspot* regions. As a *hotspot* we refer to the device region where the temperature is higher than 70% of the maximum temperature of the 3D FPGA. This step was presented in a previous work regarding the 2D architectures [7]. Then, the application is P&R on the target 3D architecture with the derived temperature-aware algorithm. This step is described in more detail in upcoming sections.

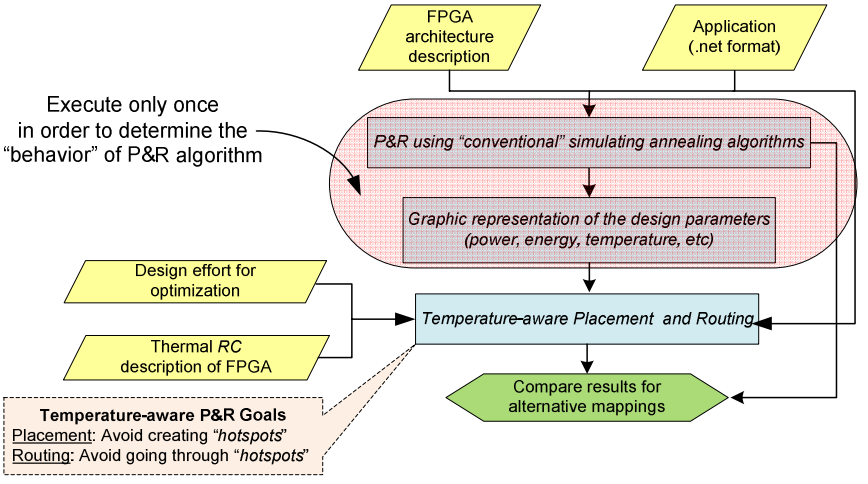


Fig. 1. Proposed methodology for temperature-aware P&R

Our interest lies to control the temperature sources across the 3D device. In order to model this, each of the functional layers is divided into a grid with dimensions $X \times Y$, where every point (x, y, z) is assumed to be small enough in order its temperature to be constant. In general, the steady-state temperature of each location across the FPGA is a function of the total power consumption regarding all the on-chip heat sources. Equation (2) gives the parameters of the thermal RC circuit [7] that models the temperature across the device.

$$T_{xyz} = \sum_{x=1}^{X} \left\{ \sum_{y=1}^{Y} \left\{ \sum_{z=1}^{Z} \{R_{xyz} \times P_{xyz}\} \right\} \right\} \tag{2}$$

In this equation, the value of R_{xyz} refers to the transfer thermal resistance, while the P_{xyz} represents the power consumption of the slice placed at spatial location (x, y, z). The on-chip temperature for this point is represented as T_{xyz}.

4 Proposed P&R Algorithm Targeting 3D FPGA Devices

Fundamentally, the problem of 3D P&R is related to topological arrangements of the application's functionality to slices (*i.e.*, logic blocks) of the 3D FPGA, while satisfying the design timing, power and area constraints. The proposed temperature-aware P&R algorithm pays effort to minimize the on-chip temperature gradient, obtaining an even uniformly spatial distribution of switched capacitance, in respect to the application's timing constraints. This approach can be though as a power management strategy. The result is an application mapping with fewer *hotspot* regions, as compared to a conventional (*i.e.*, timing-aware) approach.

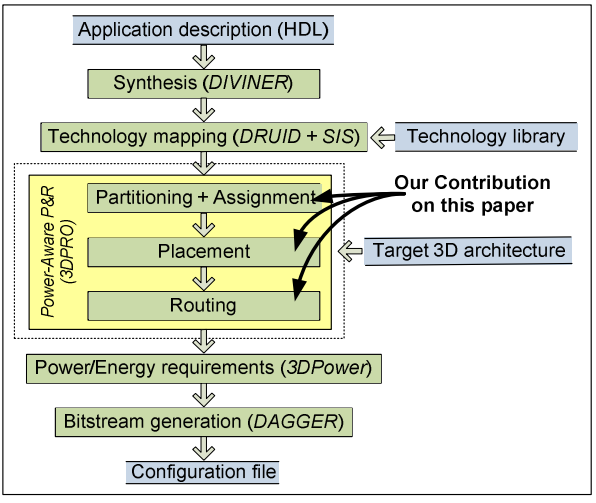


Fig. 2. The 3D MEANDER Framework

Figure 2 shows the tool flow, named *3D MEANDER*, for realizing applications on 3D FPGAs. This flow adopts some existing CAD tools from the 2D toolset [5], which do not need to be aware of the 3D FPGA topology (*i.e.*, technology platform independent). To the best of our knowledge, this toolset is the first complete framework in academia for mapping applications on 3D reconfigurable devices starting from hardware description language up to configuration file generation.

The proposed temperature-aware algorithm (shown in Algorithm 1) is implemented within the *3DPRO* tool [8]. The calculation of application's delay is based on Elmore model [12]. Regarding the algorithmic complexity of this tool, it is similar to existing solutions [10, 13]. Regarding the calculation of temperature variations across the 3D FPGA, we employ models introduced in [2], appropriately extended in order to be aware about the third dimension. These models were integrated in the *3DPower* tool.

```
Function TEMPERATURE-AWARE P&R FOR 3D FPGAs()
     Input netlist: technology mapped application
     Input target architecture: target 3D FPGA device
     Partition(netlist, target architecture);
     Place(netlist, target architecture);
     Route(netlist, target architecture);
     Calculate statistics(netlist);
End Function
```

Algorithm 1. The proposed temperature-aware P&R

Since our proposed P&R approach is temperature-aware, it poses new challenges to application implementation on 3D devices. Detail description of each algorithmic step (*i.e.*, partitioning, placement, routing) will be given in the upcoming sections.

4.1 Application Partitioning and Layer Assignment

The first step of the proposed temperature-aware P&R algorithm deals with the application partitioning into Z balanced sections. This number (Z) is equal to the 3D device layers. The employed partitioning algorithm (shown in Algorithm 2) is based on [11], as it tries to minimize the interlayer communication, however its cost function was appropriately extended to spread as much as possible the spatial distribution of application's switched capacitance across the 3D device, without affecting the total power/energy consumption, the application's delay or the area requirements.

This procedure is done by recursive bi-partitioning of the application hypergraph $AppG$, such that to minimize the value of the employed cost function (depicted in Equation (3)). Since the net length is tightly firmed to its resistance and capacitance values, we can manage the power consumption sources by weighting each net according to its switched capacitance.

```
Function PARTITION(netlist, number of layers)
     while accept (partition=True) do
          Subgraphs ← split(netlist, number of layers);
          C ← calculate(connections among subgraphs);
          If (C > crititcal) then
               try to repartition the netlist;
          else
               accept partitions ← True
          end if
     end while
end Function
```

Algorithm 2. The proposed temperature-aware partitioning

We associate the switched capacitance criticality of a logic element as weight to the corresponding vertex in the hypergraph, while the timing criticality is shown as weight to the corresponding hyperedge. These weights encourage the partitioning algorithm to split the application in a way that balances both of these factors. The criticalities of the graph (*i.e.*, weights of vertexes and hyperedges) are updated at each partitioning level, while the partitioning process stops when both the switched capacitance distribution and the timing constraints are met.

Next, the algorithm assigns the derived application segments on the device layers by taking into consideration a number of design parameters. More specifically, the algorithm tries not to assign segments that consume high power close to each other, or on the middle of the 3D stack, as it is more difficult to dissipate heat. This task is accomplished in conjunction to the effort for minimizing the interlayer communication or other design constraints (*i.e.*, delay, power/energy consumption, etc).

Additionally, since our algorithm can be used for architecture-level exploration, rather than providing only an output, we calculate the Pareto-based space of alternative application partitions. These solutions balance the area occupied by active hardware resources, the number of interlayer connections and the variation of power

consumption (*i.e.*, power sources) among layers. In order to quantify each of the derived application partitioning, we employ cost function shown in Equation (3).

$$cost_{function} = (k \times power_{variation}) + (1 - k) \times \{(p \times interlayer_{connections}) + ((1 - p) \times area_{balance})\} \quad (3)$$

where $power_{variation}$ denotes the variation of power sources over the 3D FPGA, the $interlayer_{connections}$ is equal to the total amount of hyperedge-cut, while the $area_{balance}$ corresponds to the variation of area occupied by active hardware resources among the device layers. The employed factors k and p provide higher flexibility to the cost function, as they can be used to tune the algorithm for further optimizing the partitioning result. Finally, we have to mention that both the cost function, as well as the criticalities of the hypergraph (*i.e.*, weights of vertexes and edges), are updated after each iteration, while the partitioning stops when both the distribution of switched capacitance and the timing constraints are met.

4.2 Application Placement

After the partitioning step, the placement algorithm assigns the application's logic functionalities (L) to available hardware modules (C). As the majority of applications realized onto FPGAs utilize only a subset of the available hardware resources, this non-uniformity leads to high variation of power consumption across the device [8]. This problem gets even worst in 3D devices due to high power/temperature variation among layers.

The proposed temperature-aware placement algorithm tries to place the logic functionalities (L) in a way that minimizes the maximal switched capacitance values (referred as *hotspots*), as well as to distribute it across the whole 3D FPGA. As the switching activity depends on the functionality implemented inside the logic blocks, while the capacitance is proportional to the interconnection length and the number of hardware modules that form each network, the proposed algorithm pays effort to handle in an efficient way their product (*i.e.*, switched capacitance).

More specifically, by placing on adjacent spatial locations logic functionalities connected through nets with high switching activity, these nets probably will be shorter (exhibit smaller capacitance), leading to reduced power consumption. Unfortunately, it is not always possible to place close all these blocks, as this might lead to increased application delay (*i.e.*, delay of the slowest path). Also, the placement of functionalities with high bandwidth requirements should be assigned onto the same functional layer, since there is plethora of routing resources, as compared to the reduced connectivity of vertical connectivity.

The proposed placement approach (shown in Algorithm 3) is based on simulated annealing. During the placement pairs of logic blocks are selected and swapped randomly, until either the resulted placement is good enough, or the maximum number of iterations is reached. The efficiency of a placement is characterized by calculating its cost function, shown in Equation (4), where:

$$\Delta Cost = \alpha \times \frac{\Delta Temperature_{cost}}{Previous\ Temperature_{cost}} + (1-\alpha) \times \left[\beta \times \frac{\Delta Wire_{cost}}{Previous\ Wire_{cost}} + (1-\beta) \times \frac{\Delta Time_{cost}}{Previous\ Time_{cost}}\right] \quad (4)$$

where

$$Temperature_{cost} = T\left[\sum_{i=1}^{Total_{Nets}} \left\{Activity(i) \times \left[q(i) \times \left[\frac{bb_x(i)}{C^{\varepsilon}_{av,x}(i)} + \frac{bb_y(i)}{C^{\varepsilon}_{av,y}(i)} + r \times \frac{bb_z(i)}{C^{\lambda}_{av,z}}\right]\right]\right\}\right]$$

$$Wire_{cost} = \sum_{i=1}^{Total\_Nets} \left\{q(i) \times \left[\frac{bb_x(i)}{C^{\varepsilon}_{av,x}(i)} + \frac{bb_y(i)}{C^{\varepsilon}_{av,y}(i)} + \frac{bb_z(i)}{C^{\lambda}_{av,z}}\right]\right\}$$

$$Time_{cost} = \sum_{\forall i,j \in application} \{Delay(i,j) \times criticality(i)^{const}\}$$

In this cost function, factors α of cost function balance the effort for reducing either the total wire length or the delay. However, in both cases, the algorithm tries to reduce the switching activity. The $bb_x(i)$, $bb_y(i)$ and $bb_z(i)$ parameters denote the dimensions of the 3D bounding box for network i, while the $q(i)$ is a scaling factor of the bounding box, used to make more accurate estimations about the wire-length for nets with more than 3 terminals [10]. The $delay(i,j)$ denotes the delay between a source-sink path of a network, the factor *const* is a constant, while the $criticality(i)$ gives the importance, in terms of how close to the critical path, is the network i. Finally, the $activity(i)$ represents the switching activity value for the network i. In order to calculate this parameter, the transition density for all the hardware elements of network i has to be summarized.

The $C_{av,x}(i)$, $C_{av,y}(i)$ and $C_{av,z}(i)$ parameters represent the average width of routing tracks across the x, y and z direction, respectively, for the bounding box of network i, while they are used in order to be taken into consideration the available routing resources during the placement. Their values depend solely on the fabricated interconnection resources, while they are constant during the placement. The values of ε and λ control the relative cost of employing narrower and wider routing channels. More specifically, when their values are 0, then the cost function results to the conventional bounding box approach. Otherwise, as higher the values of these parameters are, then more and more tracks from narrowest routing channels have increased cost value, compared to the wider channels. We employ a different relative cost (λ) for the vertical interconnections, as the placement algorithm has to pay effort to not waste this kind of connections. Finally, by using an additional factor, denoted as r, we discourage the placer to put functions that exchange data in different layers.

```
Function PLACE(netlist, target architecture)
P ← Make an initial Placement();
T ← Initial Temperature;
R_limit ← Initial R_limit;
While (Exit_Criterion() not TRUE)  // outer loop
    {
    While (loop_criterion() not TRUE)  //inner loop
        {
        P_new ← Random swap placements(P, P_limit);
        ΔCost ← Cost(P_new) - Cost(P);
        r ← random value(0,1);
        if (r<e^(-ΔCT)) P ← P_new;  // accept movement
        }
    R_limit ← Update(R_limit);
    T ← Update(Temperature)
    }
End Function
```

Algorithm 3. The proposed temperature-aware placement algorithm

Even though it is true that such an approach can reach arbitrary close to the global minimum, if the cooling schedule is slow enough, it suffers from long run times for large circuits. In contrast to most of the existing approaches that start from a random initial placement, our solution employs a more "*sophisticated*" assignment of logic blocks, leading to shorter runtimes. This is achieved by taking into consideration during the initial placement apart from the timing and the wire-length constraints, the minimization of switched capacitance variation. Such info is available from the partitioning step (shown in previous section).

4.3 Application Routing

By defining the placement on the 3D FPGA, the routing algorithm forms the appropriate connections among the utilized logic blocks (C) through the available interconnection fabric (W). As the vertical interconnections are limited, compared to horizontal tracks, the routing algorithm sets their weight to a higher value, in order to discourage the unnecessary bends between horizontal and vertical wires. Also, this penalty forces the router not to connect logic blocks placed on one layer by using interconnection fabric from different layers.

The proposed routing algorithm (shown in Algorithm 4) is based on Pathfinder negotiated congestion [4]. Initially, a number of networks are allowed to share the same routing fabric, which is gradually prohibited, until to the final routing where every network employs dedicated routing fabric. Such an approach finds the narrowest horizontal and vertical channels for which the application is fully routable.

```
Function ROUTE(netlist, target architecture)
horizontal ← initial horizontal channel width;
vertical ← initial vertical channel width;
while (routing ← optimal connection of logic blocks) do
route netlist();
   if (succeed routing) then
      optimal routing ← find narrowest channels;
      do
         T ← meet timing constraints();
         if(T ← True) distribute switched
capacitance();
      while(T ← True)
   else increase horizontal/vertical channel widths;
end while
end Function
```

Algorithm 4. The proposed temperature-aware routing

By discouraging routing algorithm to form connections that cross *hotspot* regions, it is possible to spread the switched capacitance over the 3D device, while it also achieve the timing and total power/energy constraints. However, this is not always feasible, as it might increase the application's delay or its power/energy consumption. The efficiency of a derived application routing, is quantified with the cost function. The mathematical expression regarding this function is shown in Equation (5).

$$\begin{aligned} \Delta Cost(n) = Criticality(i) \times Delay(i,j) + \big(1 - Criticality(i)\big) \\ \times [\gamma \times switching(i) \times capacitance(n) + (1 - switching(i)) \\ \times b(n) \times h(n) \times p(n)] \end{aligned} \tag{5}$$

In this expression, the factor γ defines the importance of temperature control during the routing procedure. The parameters $b(n)$, $h(n)$ and $p(n)$ represent the base cost, the historical congestion cost and the present congestion cost for the hardware element n, respectively. In order to come to acceptable solutions the value of $p(n)$ increases with the execution time, in order to avoid the overuse of routing resources. The factor $capacitance(n)$ corresponds to the normalized capacitance of resource n, while the $switching(i)$ refers to the importance of the switching activity for the network i. Equation (6) gives the mathematic expression of this parameter.

$$switching(i) = \min\left\{Max\_switching, \frac{switching(i)}{maximum\_switching}\right\} \tag{6}$$

Here the $Max\_switching$ corresponds to the maximum allowed value of switching activity regarding the network i, while the ratio $\frac{switching(i)}{maximum\_switching}$ gives the normalized switching activity over all the application's interconnection networks. As the value of the $Max\_switching$ parameter closes to 1 (*e.g.*, 0.99), then more and more interconnection networks with high switching activity will be taken in consideration during the routing congestion.

Comparing the proposed cost function with existing from literature [8, 10, 13], it has identical timing parameter. So, whenever a routing connection is timing critical, the routing algorithm pays effort to reduce the delay of the network. However, the second part of the cost function tries to handle the spatial distribution of switched capacitance. Whenever a connection exhibits high switching activity, the proposed algorithm tries to form the required connections through paths that exhibit reduced capacitance (in order to eliminate the temperature values). On the other hand, when the networks do not exhibit increased switching activity, the routing algorithm reduces the routing congestion and increases the application's operation frequency.

5 Experimental Results

We implement the proposed temperature-aware P&R algorithm in C++, as part to an existing open-source tool for 3D FPGAs, named 3DPRO [8]. The experimental results were retrieved using the 3DPRO tool, without and with the power-aware P&R feature. This section provides comparisons among the proposed temperature-aware P&R algorithm and the alternative solutions found in relevant literature, considering the 20 biggest MCNC benchmarks. The average complexity of the employed benchmark circuits, as listed in Table 1, is about 3,410 4-input LUTs, while each of the layers contains an array of 56×56 slices. In terms of the target 3-D devices, the average percentage of utilized logic resources is almost 97%.

The target 3D PFGA platforms (were each of the benchmarks is mapped) is inspired by the one proposed in [8]. Such a 3D device is constructed by stacking a number of identical 2D FPGAs on individual functional layers, providing appropriate communication among them by interlayer vias. These connections are realized inside vertically adjacent 3D Switch Boxes (SBs). The employed 3D architecture has a vias distribution with smaller fabrication costs compared to conventional 3D FPGAs, without any degradation in application performance, or increment of total power/energy consumption. The features of this architecture are summarized as follows:

- It consists of four functional layers ($Z = 4$).
- The percentage of vertical interconnections (i.e. vias) per functional layer is 30% (as derived in [8]).
- The spatial location (x, y) of each vertical interconnection per layer remains invariant.
- The vertical interconnection fabric was modeled based on the approach shown in [3].
- There are 4 bit connections between layers for each 3D SB.
- The hardware resources (both logic and interconnection) among layers are identical.
- Each application is P&R onto the smallest 3D FPGA.
- The employed 3D devices for the alternative mappings have identical hardware resources.

Table 1. Complexity of the employed benchmark applications, and utilization of the logic resources of the target 3-D FPGA

Benchmark	# of 4-LUTs	2D FPGA		3D FPGA	
		FPGA array	% utilized logic resources	FPGA array	% utilized logic resources
alu4	1544	40×40	96.50%	20×20×4	96.50%
apex2	1920	44×44	99.17%	22×22×4	99.17%
apex4	1290	36×36	99.54%	18×18×4	99.54%
bigkey	2391	49×49	99.58%	25×25×4	95.64%
clma	8879	95×95	98.38%	48×48×4	96.34%
des	2092	46×46	98.87%	23×23×4	98.87%
diffeq	1974	45×45	97.48%	23×23×4	93.29%
dsip	2020	45×45	99.75%	23×23×4	95.46%
elliptic	4969	71×71	98.57%	36×36×4	95.85%
ex1010	4618	68×68	99.87%	34×34×4	99.87%
ex5p	1135	34×34	98.18%	17×17×4	98.18%
frisc	4561	68×68	98.64%	34×34×4	98.64%
misex3	1425	38×38	98.68%	19×19×4	98.68%
pdc	4631	69×69	97.27%	35×35×4	94.51%
s298	1948	45×45	96.20%	23×23×4	92.06%
s38417	7694	88×88	99.35%	44×44×4	99.35%
s38584	7884	89×89	99.53%	45×45×4	97.33%
seq	1826	43×43	98.76%	22×22×4	94.32%
spla	3752	62×62	97.61%	31×31×4	97.61%
Tseng	1605	41×41	95.48%	21×21×4	90.99%
Average:	**3407.9**	**56×56**	**98.37%**	**29×29×4**	**96.61%**

Figure 3 depicts the variation of switched capacitance over the layers of the 3D FPGA, with a timing-aware mapping. The employed application, named *alu4*, is one of the 20 biggest MCNC benchmarks, consisted of 1522 4-input LUTs. These logic modules are assigned to four equal sized layers, each of which occupies an array of 20×20 slices. The picture of Figure 3 is a very useful instrument to architecture designers, in order to specify the spatial distribution of *hotspot* regions over the device.

From this figure we conclude that the switched capacitance vary a lot, even for hardware resources assigned to adjacent spatial locations onto the same layer. Also, it is possible to locate regions on the layers with excessive high values of switched capacitance. In order to understand the thermal characteristics and prevent circuit failure, it is important to detect such *hotspots* regions. By specifying their spatial distribution, the designer can concentrate his/her efforts to control the switched capacitance on certain regions only, but not on the whole device, reducing among others the design/fabrication cost.

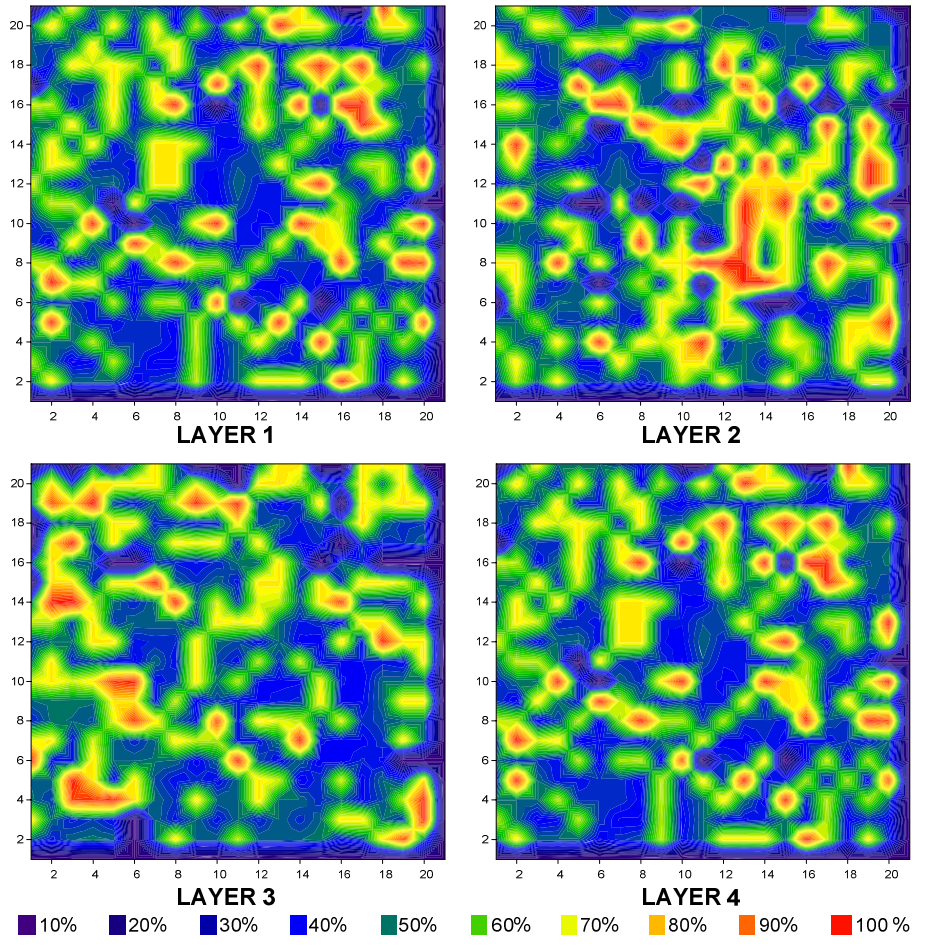


Fig. 3. The variation of switched capacitance for *alu4* benchmark with a conventional P&R algorithm

The proposed temperature-aware P&R algorithm can assist to provide a solution to this problem, as it is aware about the distribution of switched capacitance across the 3D FPGA. Figure 4 plots the corresponding variation of switched capacitance regarding the same application and 3D device, for the proposed algorithm. In contrast to the conventional approach (shown in Figure 3), the proposed one exhibits more balanced variation of switched capacitance, and hence for power consumption and for on-chip temperature.

Additionally, the maximal values of switched capacitance are lower, leading to cheaper and more reliable devices. One more conclusion might be derived from these two graphs. More specifically, the proposed approach distributes more uniformly the switched capacitance for the layers placed on the middle of the 3D stack. This feature is critical for the thermal efficiency of the target 3D architecture, as it is more difficult to dissipate heat from these layers.

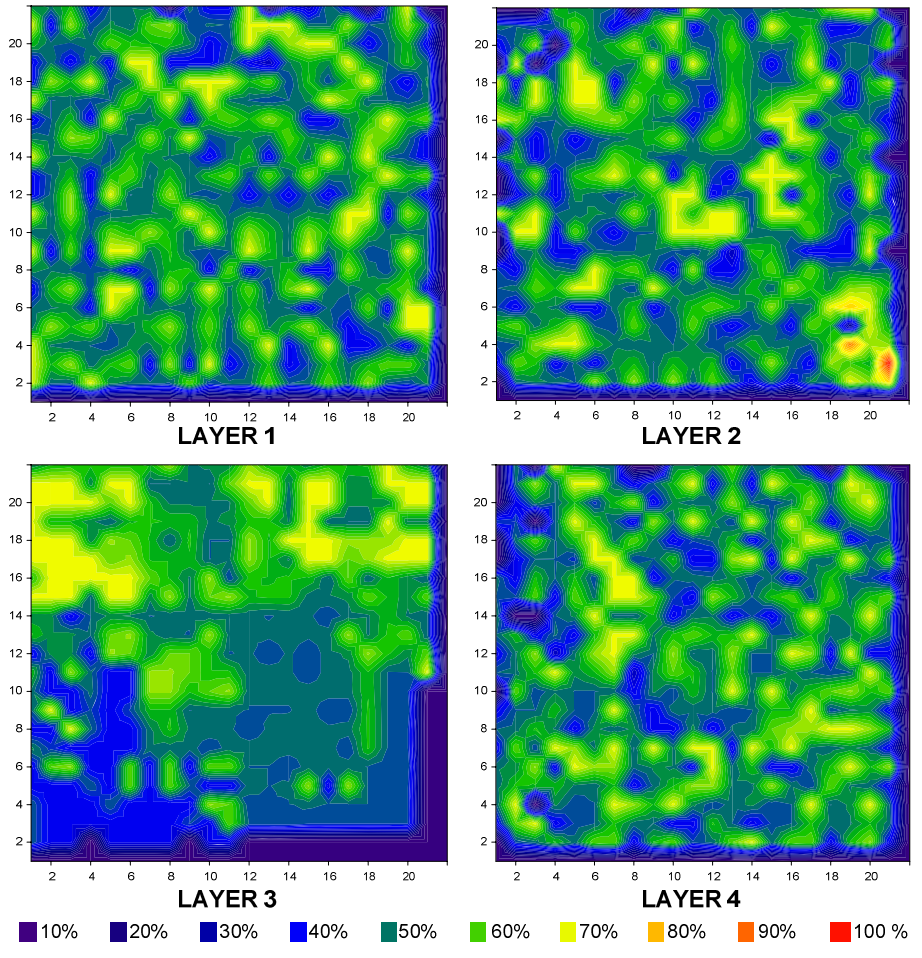


Fig. 4. The variation of switched capacitance for *alu4* benchmark with the proposed temperature-aware P&R

For shake of completeness we employ the proposed temperature-aware P&R algorithm for two setups. Both of them were realized by appropriately tuning the parameters of the cost functions. More specifically, the first of them affects an approach where the importance of application's delay is thought to be similar to the temperature distribution, while in the second experimental setup, the employed cost functions are tuned to achieve even more uniform temperature distribution on the 3D FPGA.

Figure 5 compares the average (over the 20 biggest MCNC benchmarks) area percentage of the 3D FPGA that operates under high power sources for the two flavors of the proposed approach (non-aggressive and aggressive) against to conventional P&R. As we may conclude, the proposed solution achieves to reduce the percentage of area that operates under high power values (*i.e.*, belonging to *hotspot*

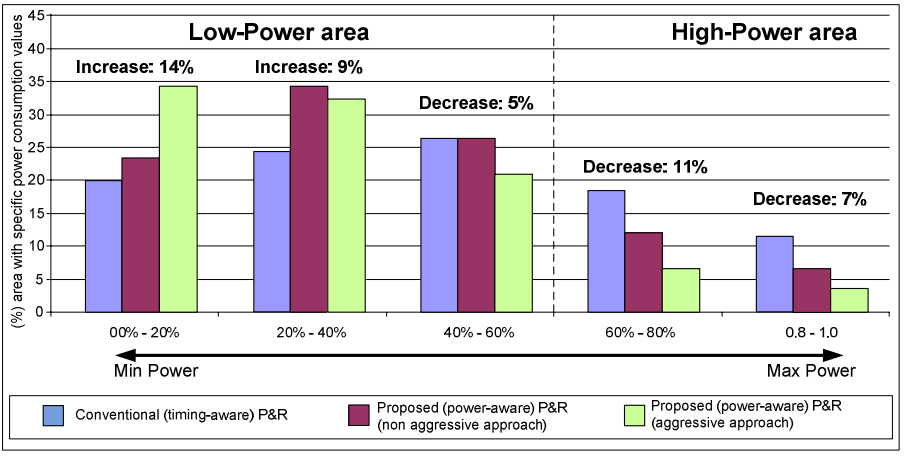


Fig. 5. Variation of area percentage that operates under specific power consumption for alternative P&R algorithms

Table 2. Comparison in terms of total wire-length ($\times 10^3$m) for alternative P&R algorithms

Benchmark	2D FPGA		3D FPGA		
	Timing-aware P&R	Temperature-aware P&R [18]	Timing-aware P&R	Proposed (Temperature-aware) P&R	
				non- Aggressive	Aggressive
alu4	50.23	54.46	37.09	35.65	37.81
apex2	67.67	65.63	53.16	56.06	58.99
apex4	62.39	65.69	37.92	38.16	37.76
bigkey	60.01	68.15	43.68	48.57	32.71
clma	301.42	309.4	280.5	294.6	430.44
des	57.46	62.09	43.82	45.44	52.7
diffeq	46.77	54.48	31.08	36.31	34.88
dsip	45.81	49.56	33.69	35.46	29.9
elliptic	114.74	121.38	94.91	92.15	102.24
ex1010	41.34	46.55	31.47	33.78	36.9
ex5p	163.43	169.93	146.42	167	129.65
Frisk	108.35	114.56	91.26	99.83	174.05
misex3	50.56	56.91	37.88	38.26	39.31
pdc	174.33	183.36	160.37	173.32	238.78
s298	59.68	65.44	42.3	44.65	55.85
s38417	170.63	182.39	155.97	169.52	172.01
s38584	147.55	160.01	136.39	152.62	129.14
seq	63.46	68.55	48.74	54.49	52.8
spla	139.45	148.62	125.03	113.47	148.19
tseng	35.45	41.09	21.07	23.85	25.89
Average:	**98.04**	**104.41**	**82.64**	**87.66**	**101.00**

Table 3. Comparison in terms of delay (10^{-9}sec) for alternative P&R algorithms

Benchmark	2D FPGA		3D FPGA		
	Timing-aware P&R	Temperature-aware P&R [18]	Timing-aware P&R	Proposed (Temperature-aware) P&R	
				non-Aggressive	Aggressive
alu4	9.77	8.69	7.32	7.51	7.71
apex2	9.37	9.65	7.87	8.04	8.12
apex4	8.86	9.1	6.78	6.96	7.29
bigkey	6.03	9.71	5.14	4.96	5.22
clma	10.1	9.23	8.84	9.95	10.47
des	7.76	11.1	6.78	6.65	7.34
diffeq	6.15	6.32	7.71	7.67	7.77
dsip	7.99	10.9	5.14	4.99	5.13
elliptic	10.9	11.7	7.34	7.63	7.69
ex1010	18.3	18.3	7.87	8.29	8.52
ex5p	9.26	7.17	5.95	5.96	6.28
Frisk	16.1	13.4	6.51	6.62	6.87
misex3	11.7	8.34	7.32	7.68	7.79
pdc	20.4	18.7	8.41	8.37	8.52
s298	13.4	13.4	11.69	11.54	11.69
s38417	9.85	9.79	10.95	11.35	11.43
s38584	7.45	7.85	10.47	10.12	10.64
seq	9.52	8.17	7.32	7.41	7.72
spla	15.6	18	7.87	7.62	8.14
tseng	5.55	5.53	7.61	7.48	7.88
Average:	**10.70**	**10.75**	**7.74**	**7.84**	**8.11**

regions), while it spreads these power sources on the rest device in a more uniformly manner. This is especially critical for designing reliable and cheaper devices, as there is no need for expensive packaging solutions. Moreover, by transferring power consumption from *hotspot* regions to the rest architecture, we increase the device reliability and reduce its fabrication cost.

Apart from distributing uniformly the on-chip temperature; the proposed P&R algorithm pays effort not to increase either the application's delay or its total power/energy consumption. The upcoming Tables summarize the evaluation results of applying the proposed strategy to the 20 biggest MCNC benchmarks. The target 3D FPGA device was described in the beginning of this section, while the array dimensions for each benchmark is derived from Table 1.

As we have already mentioned, the interconnection network contributes to the temperature of target 3D architectures. Table 2 compares the total wire-length for application mapping onto 2D and 3D FPGAs. Based on the results, the proposed temperature-aware approach leads to increased wire-length (between 6% and 22%), as compared to a timing-aware P&R. However, both of temperature-aware flavors

Table 4. Comparison in terms of energy dissipation (10^{-9} Joule) for alternative P&R algorithms

Benchmark	2D FPGA		3D FPGA		
	Timing-aware P&R	Temperature-aware P&R [18]	Timing-aware P&R	Proposed (Temperature-aware) P&R	
				non- Aggressive	Aggressive
alu4	5.83	5.82	4.45	4.21	4.38
apex2	6.75	6.81	6.39	5.28	4.85
apex4	4.17	4.23	3.68	3.26	2.96
bigkey	7.95	8.27	7.10	8.49	7.76
clma	75.6	79.9	32.49	33.44	33.69
des	10.5	11.4	10.14	9.76	9.22
diffeq	3.51	3.5	9.28	8.35	8.27
dsip	7.82	8.01	5.58	5.77	5.10
elliptic	12.4	12.8	9.71	9.87	10.17
ex1010	16.2	16.3	9.91	8.31	8.68
ex5p	4.32	4.1	3.19	3.10	2.97
Frisk	12.1	11.2	19.01	17.02	15.70
misex3	5.55	5.25	4.12	3.79	4.01
pdc	22.3	21.4	14.02	15.83	13.70
s298	6.88	6.95	7.45	7.49	6.83
s38417	22.9	23.1	34.48	29.10	26.99
s38584	43.2	35.6	23.95	24.08	21.28
seq	6.32	6.08	5.72	4.70	5.02
spla	13.1	13.9	9.58	9.04	8.55
tseng	3.18	3.2	15.05	16.09	14.30
Average:	**15.13**	**14.98**	**11.76**	**11.35**	**10.72**

results to smaller wire-lengths than the implementation targeting 2D architectures [8] about 19%. As we will prove later, the increased values of this parameter cannot outperform the advantages of realizing applications with the proposed power-aware P&R algorithm.

Table 3 gives the delay for each of the 20 biggest MCNC benchmark with the usage of alternative P&R algorithms for 2D and 3D FPGAs. Based on the results, the proposed temperature-aware P&R algorithm increases slightly the application's delay, ranging from 1% up to 5%, while the performance improvement compared to solution targeting 2D FPGAs [7] is up to 27%. The almost negligible performance degradation (due to the extra constraints for forming connections) is acceptable, as it does not lead to significant variation of the application's functionality.

Table 4 gives the energy requirements for each of the 20 biggest MCNC benchmarks. As we may conclude from the results, the proposed temperature-aware P&R algorithms (non-aggressive and aggressive) lead to energy savings, compared to conventional (*i.e.*, timing-aware) P&R that range between 4% and 9%, respectively. Additionally, the energy savings compared to corresponding solution from literature [7] is up to 29%.

Table 5. Comparison in terms of area percentage that operates under high temperature values (*i.e.*, hotspot regions) for alternative P&R algorithms

Benchmark	2D FPGA		3D FPGA		
	Timing-aware P&R	Temperature-aware P&R [18]	Timing-aware P&R	Proposed (Temperature-aware) P&R	
				non- Aggressive	Aggressive
alu4	34%	17%	26%	14%	9%
apex2	41%	23%	30%	18%	9%
apex4	42%	29%	31%	23%	14%
bigkey	34%	19%	24%	15%	8%
clma	25%	12%	18%	10%	6%
des	39%	26%	30%	21%	9%
diffeq	37%	25%	28%	20%	9%
dsip	39%	27%	30%	22%	11%
elliptic	42%	30%	30%	26%	15%
ex1010	26%	15%	19%	12%	5%
ex5p	27%	17%	20%	14%	8%
Frisk	33%	24%	25%	19%	9%
misex3	31%	23%	23%	18%	11%
pdc	31%	26%	23%	17%	10%
s298	31%	24%	23%	14%	9%
s38417	27%	13%	20%	10%	7%
s38584	45%	30%	34%	25%	12%
seq	40%	31%	30%	24%	13%
spla	32%	23%	23%	18%	10%
tseng	33%	23%	25%	19%	8%
Average:	**34%**	**23%**	**26%**	**18%**	**9.6%**

Finally, we study the percentage of device area that operates under high power. This part of device area is mentioned as *hotspot* region, while its reduction is the main goal of the developed research. The two flavors of the proposed power-aware P&R algorithm achieve to reduce this percentage, compared to conventional (*i.e.*, timing-aware) P&R for the same 3D FPGA ranging from 30% up to 63%. Moreover, the reduction of area coverage for *hotspot* regions compared to existing approaches for 2D FPGAs [7] is about 58%.

The results presented in this section prove that the proposed temperature-aware P&R achieves to reduce the percentage of silicon area that operates under high power/temperatures (*hotspot* regions), which is the main goal of our research, without impact on other critical design parameters, even though there is an increase in total wire-length. This occurs due to the better application partitioning, partition to layer assignment, placement and routing.

The gains of employing the proposed temperature-aware P&R approach targeting 3D FPGAs can be summarized as follows: (*i*) it spreads the power/temperature sources across the 3D FPGA in a way that it is more easy to dissipate heat, (*ii*) it

reduces the peak values of power/temperature sources leading to cheaper fabrication cost for cooling, (*iii*) it reduces the total energy consumption increasing among other the battery life and the system's reliability, and (*iv*) it reduces significantly the percentage of silicon area that operates under high power/temperature consumption values (i.e. *hotspot* regions), which can be thought as a power/temperature management approach.

6 Conclusions

A novel temperature-aware P&R algorithm targeting to 3D FPGAs, as well as its software implementation at 3DPRO tool, was presented. This approach could also be used as a power management strategy, since it achieves to re-distribute the power budget over identical hardware resources in a way that the produced heat is easily to be dissipated. More specifically, the proposed P&R algorithm reduces about 63%, in average, the percentage of device area that operates under high temperature by appropriately controlling the switched capacitance. In addition to that, we achieve energy savings about 9% (in average), with an almost negligible penalty (ranging from 1% up to 5%) in application's delay.

Acknowledgment

This paper is part of the 03ED593 research project, implemented within the framework of the "Reinforcement Program of Human Research Manpower" (PENED) and co-financed by National and Community Funds (75% from E.U.-European Social Fund and 25% from the Greek Ministry of Development-General Secretariat of Research and Technology).

References

[1] International Technology Roadmap for Semiconductors, `http://www.intel.com/technology/silicon/itroadmap.htm`
[2] Poon, K., Yan, A., Wilton, S.: A Flexible Power Model for FPGAs. In: Glesner, M., Zipf, P., Renovell, M. (eds.) FPL 2002. LNCS, vol. 2438, pp. 312–321. Springer, Heidelberg (2002)
[3] Gupta, S., Hilbert, M., Hong, S., Patti, R.: Techniques for producing 3D ICs with High-Density Interconnect. In: VLSI Multi-Level Interconnection Conference (2004)
[4] McMurchie, L., Ebeling, C.: Pathfinder: A Negotiation-Based Performance-Driven Router for FPGAs. In: ACM/SIGDA Int. Sym. on Field-Programmable Gate Arrays, pp. 111–117 (1995)
[5] `http://proteas.microlab.ntua.gr`
[6] Telikepalli, A.: Designing for Power Budgets and Effective Thermal Management. Xcell Journal (56) (2006)
[7] Siozios, K., Soudris, D.: A Novel Methodology for Temperature-Aware Placement and Routing of FPGAs. In: IEEE Computer Society Annual Symposium on VLSI, pp. 55–60 (2007)

[8] Siozios, K., et al.: Exploring Alternative 3D FPGA Architectures: Design Methodology and CAD Tool Support. In: 17th Int. Conference on Field-Programmable Logic and Applications, pp. 652–655 (2007)
[9] Chiang, T.Y., et al.: Thermal Analysis of Heterogeneous 3D ICs with Various Integration Scenarios. In: International Electron Devices Meeting (IEDM), pp. 31.2.1–31.2.4 (2001)
[10] Betz, V., et al.: Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, Dordrecht (1999)
[11] Selvakkumaran, N., Karypis, G.: Multiobjective Hypergraph-Partitioning Algorithms for Cut and Maximum Subdomain-Degree Minimization. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 25(3), 504–517 (2006)
[12] Okamoto, T., Cong, J.: Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization. In: Int. Conference on Computer Aided Design, pp. 44–49 (1996)
[13] Ababei, C., et al.: Placement and Routing in 3D Integrated Circuits. IEEE Design & Test of Computers 22(6), 520–531 (2005)
[14] Das, S., et al.: Timing, Energy, and Thermal Performance of Three Dimensional Integrated Circuits. In: 14th ACM Great Lakes symposium on VLSI, pp. 338–343 (2004)
[15] Yang, S.: Logic Synthesis and Optimization Benchmarks, Version 3, Microelectronics Centre of North Carolina (1991)
[16] Cong, J., Wei, J., Zhang, Y.: A Thermal-Driven Floorplanning Algorithm for 3D ICs. In: International Conference on Computer Aided Design, pp. 306–313 (2004)
[17] Lesea, et al.: Powering Xilinx FPGAs (2002)

A Reconfigurable Network-on-Chip Architecture for Optimal Multi-Processor SoC Communication

Vincenzo Rana[1], David Atienza[2,3], Marco Domenico Santambrogio[1], Donatella Sciuto[1], and Giovanni De Micheli[4]

[1] Dipartimento di Elettronica e Informazione (DEI) - Politecnico di Milano, Via Ponzio 34/5, 20133 - Milano, Italy
{rana, santambr, sciuto}@elet.polimi.it
[2] Embedded Systems Laboratory (ESL) - Ecole Polytechnique Fédérale de Lausanne (EPFL), ESL-IEL-STI-EPFL, Station 11, 1015-Lausanne, Switzerland
david.atienza@epfl.ch
[3] Depto. de Arquitectura de Computadores y Automática (DACYA) - Universidad Complutense de Madrid (UCM), Avda. Complutense S/N, 28040-Madrid, Spain
[4] Integrated Systems Laboratory (LSI) - EPFL, LSI-ISIM-IC-EPFL, Station 14, 1015-Lausanne, Switzerland
giovanni.demicheli@epfl.ch

Abstract. Network-on-Chip (NoC) has emerged as a very promising paradigm for designing scalable communication architecture for Systems-on-Chips (SoCs). However, NoCs designed to fulfill the bandwidth requirements between the cores of an SoC for a certain set of running applications may be highly sub-optimal for another set of applications. In this context, methods that can lead to versatility enhancements of initial NoC designs to changing working conditions, imposed by variable sets of executed real-life applications at each moment in time, are very important for designing competitive NoCs in industrial SoCs.

In this work, we present a run-time reconfigurable NoC framework based on the partial dynamic reconfiguration capabilities of Field-Programmable Gate Arrays (FPGAs). This new NoC framework can dynamically create/delete express lines between SoC components (implementing dynamically circuit-switching channels) and perform run-time NoC topology and routing-table reconfigurations to handle interconnection congestion, with a very limited performance overhead. Moreover, we show in our experimental results that the addition of these dynamic reconfiguration capabilities into basic NoCs using our framework only implies a very limited area overhead (around 10% on average) with respect to the initial NoC designs; thus, it can bring great benefits when compared to traditional non-reconfigurable NoC design approaches for worst-case bandwidth requirements in SoCs with many possible sets of running applications.

Keywords: Networks on Chips, Systems on Chips, Topology Reconfiguration, Express Lines, Dynamic Reconfiguration, FPGA.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 232–250, 2010.

1 Introduction and Problem Description

Latest applications ported to embedded systems (e.g., scalable video rendering, communication protocols) demand a large computation power, while must respect other critical embedded design constraints, such as, short time-to-market, low energy consumption or reduced implementation size.

Thus, embedded systems are complex *Systems-on-Chip (SoCs)* that consist of a large number of components, such as, processing elements, storage devices and even reconfigurable devices, such as *Field-Programmable Gate Arrays (FPGAs)*, to enhance the flexibility of final SoCs to be used in different environments [5, 15]. Nevertheless, one of the most critical areas of MPSoC design is the definition of the suitable interconnect subsystem for all these SoC components, due to architectural and physical scalability concerns [3]. In fact, traditional shared bus interconnects are relatively easy to design, but do not scale well for latest and forthcoming SoC consumer platforms.

In order to cope with the large communication demands of such SoCs, the use of modular and scalable *Networks-on-Chips (NoCs)* has been proposed [3]. Then, designing custom-tailored NoC interconnects that satisfy the performance and design constraints of the SoC for all the different combinations of possible executed applications is a key goal to achieve optimal commercial products [2, 13]. However, as general-purpose processor cores are used to run software tasks of different applications in SoCs, the communication between the cores cannot be precharacterized and fully optimized, since the application processes can be mapped differently to the cores, typically with the support of the compiler. Thus, to provide predictable performance of the NoC, the bandwidth capacity of the different links must be sufficient to support the peak rate of traffic on the links of the possible different mappings of the tasks onto the final SoC. Otherwise, the network might experience traffic congestion and the latency for the traffic streams and, hence, the interconnect performance will become unacceptable, which needs to be avoided to provide appropriate consumer devices. As a result, NoCs designs that guarantee worst-case bandwidth conditions of SoC operation with multiple concurrent application often leads to over-sized topologies and links on regular operation of the SoC. In this context, the development of new methods and frameworks that increase the run-time versatility of initial static NoC designs to adapt to different working conditions, originated by the diversity of sets of applications at each moment, is an important research area in the NoC domain.

In this paper we introduce a novel run-time reconfigurable NoC framework, which exploits the partial dynamic reconfiguration capabilities of *FPGAs* to adapt at run-time the implemented NoC interconnect to the specific working requirements of the final SoC at each moment in time. In particular, the proposed NoC framework is able to reduce the latency of interconnecting the included SoC components by dynamically establishing or deleting a number of dedicated point-to-point connections between them (or express lines in the NoC literature [3]), which is particularly suited for video and audio streaming. Thus, circuit-switching communication can be dynamically configured in the SoC. In

addition, our framework enables a fast dynamic reconfiguration of routing tables (few cycles) and overall NoC topologies (few milliseconds), which provides new promising means to overcome congestion and consequently provide more reliable and high-performance NoC designs. Furthermore, our experimental results show that the addition of the dynamic reconfiguration capabilities in basic NoCs using our framework only involves limited area overheads (around 10% on average) with respect to initial NoC designs without reconfiguration capabilities. Hence, the proposed reconfigurable NoC framework is viable to be considered in commercial designs of SoCs.

It is possible to fully exploit the reconfigurable NoC proposed in this work in order to establish *dedicated* (among 2 or more switches of the network) and *long* (circuit-switching) communication channels among the cores of the reconfigurable system. This can be really useful, for instance, in the case of audio and video streaming that have to be dynamically carried out at run-time, without the possibility to obtain detailed information at design time. In this case, in fact, the proposed approach enables to dynamically reconfigure the communication infrastructure accordingly to the need that arise at run-time, in order to meet both latency and throughput requirements. This is possible since the reduction of the number of hops between two switches directly decreases the latency between them and the introduction of a new express line in the topology directly increases the overall throughput of the NoC.

The rest of the paper is organized as follows. In Section 2 we overview previous work in the field on reconfigurable NoCs. Then, in Section 3 we introduce our reconfigurable NoC architecture, spanning from the included basic NoC architecture to the additional components to enable the NoC Next, in Section 4 we discuss the major reconfiguration capabilities and methods to implement them in the proposed adaptive NoC framework. Later, in Section 5 we present the area, performance and latency evaluation of the reconfiguration capabilities of our framework in a real implementation on a large commercial FPGA implementing a multi-processor SoC. Finally, in Section 6 we draw the main conclusions of this work.

2 State of the Art

In recent years, several works focused their attention into the definition of a reconfigurable communication infrastructure for reconfigurable Systems-on-chip. For instance, in [11] the authors present a methodology for developing dynamic network reconfiguration processes, but they define a reconfiguration just as the change from one routing function to another while the network is up and running. For this reason they present a theoretical work based on the limiting assumption (not valid in the approach presented in this paper) that the network topology can be considered fixed (like in [7]).

The work presented in [6] describes an integrated modeling, simulation and implementation tool for reconfigurable NoCs. The work is based on the optimization of a single given application and no details are given about the reconfigurable

architecture of the NoC and about reconfiguration mechanisms. In addition to this, flow control is not supported and the proposed NoCs are quite expensive in terms of area usage (2733 slices, around 30% of the total slices of a Xilinx Virtex II Pro XC2VP20 device), for a 2x2 torus running at 85 MHz, while the proposed approach, with a lower area usage, allows the creation of a 3x3 mesh running at 170 MHz.

In [1], a dynamically reconfigurable NoC architecture is presented. This NoC can be dynamically configured with respect to routing, switching and data packet size, but all the required resources have to be allocated at design time, since at run-time it is only possible to dynamically change a limited number of parameters. A similar approach can be found in [10], where the proposed NoC can be configured at run-time, but only with respect to memories content, resources addressing and control parameters, while topology, buffers size and port connections have to be determined at design time.

In [9], a scalable dynamic NoC for dynamically reconfigurable FPGAs (CuNoC) is presented. The main idea behind CuNoC approach is to fill the whole reconfigurable devices with very small communication units called CUs, that can establish a communication channel between two different cores. The main drawback of this approach is, in addition to the huge power consumption, the high latency for each communication. In fact, the number of hops required for a communication is very high on average, as each packet has to pass through a high number of CUs (each one having a latency of 2 clock cycles), since it is not possible neither to define a custom topology nor to configure express lines between CUs. Furthermore, if an obstacle is present between two cores that need to communicate, it is necessary to go around it, which increases the number of hops of each packet.

In [14] the authors present CoNoChi, that is an adaptable NoC for dynamically reconfigurable hardware design. The reconfigurable device is divided in a matrix, and each cell of this matrix can hold either a computational module or a communication element (a switch or point-to-point interconnect). Since it is not possible to pass through a computational module, each communication channel has to go around all the computational elements placed on the reconfigurable device; thus, express lines cannot be configured at run-time. In addition to this, the area requirement for a single switch is very high, as it varies from 463 to 493 slices (around 5% of a XC2VP20 device). Then, the working frequency is quite low (it ranges from 66 to 73 MHz) and the actual latency of each switch is 5 clock cycles, which can be a significant penalty for interconnection mechanisms nowadays.

3 The Proposed Reconfigurable Architecture

3.1 Reconfiguration Support

In order to configure an FPGA with the desired functionality, we need to use one or more bitstreams. A bitstream is a binary file in which configuration information for a particular Xilinx device is stored, that is where all the data to

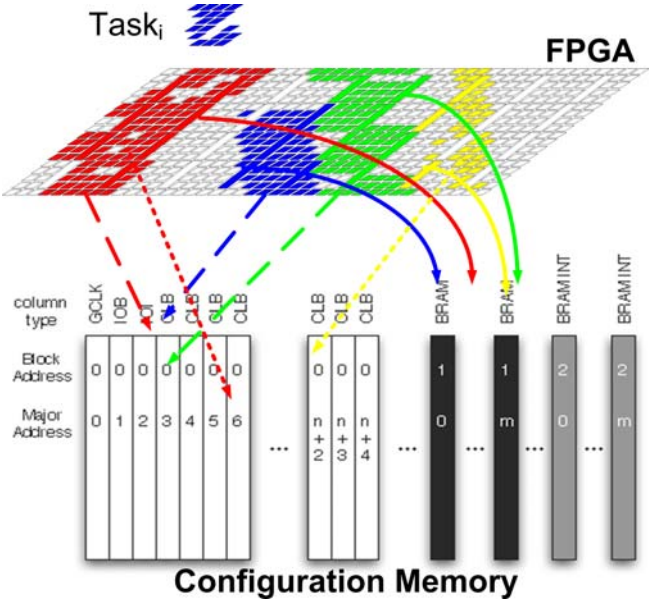


Fig. 1. Configuration memory setup

be copied on to the configuration SRAM cells, the configuration memory, are stored, along with the proper commands for controlling the chip functionalities. Therefore Virtex devices, such as Virtex II Pro and Virtex 4, are configured by loading application specific data into their configuration memory, as shown in Figure 1. On the Virtex FPGAs the configuration memory is segmented into frames. Virtex devices are partially reconfigurable and a frame is the smallest unit of reconfiguration. According to the device, this element can span the entire length of the FPGA, such as in the Virtex II Pro context, or just part of it, as in Virtex 4 devices. The number of frames and the bits per frame are specific for each device family. The number of frames is proportional to CLB width. Bitstreams can be either partial or full. A full bitstream configures the whole configuration memory and is used for static design or at the beginning of the execution of a dynamic reconfiguration system, to define the initial state of SRAM cells. Partial bitstreams configure only a portion of the device and are one of the end products of any partial reconfiguration flow.

FPGAs provide different means for configuration, under the form of different interfaces to the configuration logic on the chip. There are several modes and interfaces to configure a specific FPGA family, among them the the IEEE 1149.1 *Joint Test Action Group (JTAG)* download cable (the one used in this work), the SelectMAP interface, for daisy-chaining the configuration process of multiple FPGAs, configuration loading from PROMs or compact flash cards, microcontroller-based configuration, an *Internal Configuration Access Port (ICAP)* and so on, depending on the specific family. The ICAP provides an interface which can be

used by internal logic to reconfigure and read back the configuration memory. In every FPGA a configuration logic is built on the chip, with the purpose of implementing the different interfaces for exchanging configuration data and to interpret the bitstream to configure the device. A set of configuration registers defines the state of this configuration logic at a given moment in time. Configuration registers are the memory where the bitstream file has direct access. Actual configuration data is first written by the bitstream into these registers and then copied by the configuration logic on the configuration SRAMs.

3.2 Architecture Description

As previously hinted, the communication infrastructure of the proposed architecture is based on the NoC paradigm. Furthermore, in order to exploit a *2-layered* approach, in which the *computational layer* is completely decoupled from the *communication layer*, the proposed reconfigurable architecture mainly consists of two different parts: a *static part* and a *reconfigurable part*.

The **static part** consists of all the computational elements and the network interfaces. On the one hand, computational elements can be further divided into two categories. The first one consists of *masters*, that are the active components of the system, such as microprocessors (either a soft-core, as a MicroBlaze, or a hard-core as a PowerPC), that can initialize new transactions on the network (deployed in the communication layer); these components are connected to the communication infrastructure through *NI initiators* (see Figure 2). The second one consists of *slaves*, such as memories, that represent the components that act in a passive mode, by receiving and answering transaction coming from active elements; these components are connected to the communication infrastructure through *NI targets* (see Figure 2).

The **static part** consists of all the computational elements and the network interfaces. Computational elements can be further divided into *masters* (that are the active components of the system, such as microprocessors, that can initialize new transactions on the network and that are connected to the communication infrastructure through *NI initiators*, as shown in Figure 2) and *slaves* (such as memories, that represent the components that act in a passive mode, by receiving and answering transaction coming from active elements, and that are connected to the communication infrastructure through *NI targets*, as shown in Figure 2).

The **reconfigurable part** is composed by all the reconfigurable elements, used to adapt at run-time the structure of the system implemented on the FPGA. These elements can be either computational components or elements used to update the communication infrastructure. Network interfaces toward the communication infrastructure can implement bridges between On-chip Peripheral Bus (OPB), Processor Local Bus (PLB) or Open Core Protocol (OCP) and the network protocol, as shown in Figure 2. The only part of network interfaces (both initiator and target network interfaces) that has to be modified at run-time are routing tables, that are used to dynamically change the routing of packets on the network. Thus, all the network interfaces have been placed into the static part of the system and routing tables have been deployed on BRAM blocks. In this way it is possible to

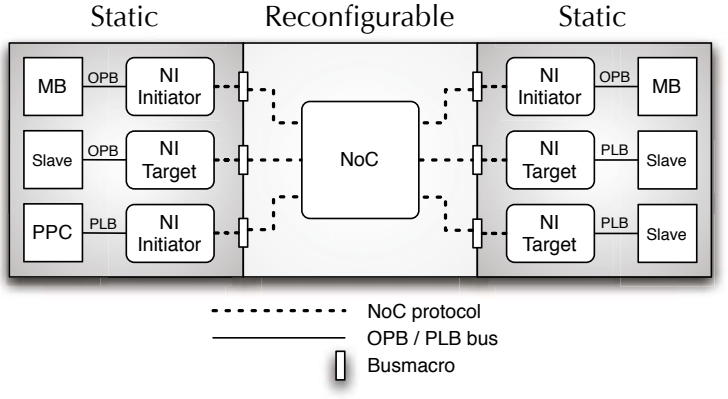


Fig. 2. Interfaces between static and reconfigurable parts

dynamically modify routing tables by changing the content of BRAM blocks at run-time, as described in Section 4.1.

This architectural solution enables connecting the static parts to the reconfigurable ones by using network interfaces that are considerably thinner of the ones used within the static part of the system. Regarding this static part, the used interconnect can be either OPB and PLB buses, or on an ad-hoc point-to-point communication infrastructure, as shown in Figure 2.

4 Reconfiguration Features

Each reconfigurable part of the system can be dynamically reconfigured at run-time to modify either a part or the whole underlying communication infrastructure. This reconfiguration can be done by the reconfiguration controller (see Figure 3), which is a master component present on the static part, trough partial reconfiguration operations.

The reconfiguration controller is connected both to the external dynamic memory (DDR) interface and to the ICAP interface through the OPB bus. The DDR memory is used to store partial bitstreams that can be used to reconfigure at run-time the reconfigurable device. In order to perform a reconfiguration process, the reconfiguration controller has to read the desired bitstream from the memory and to pass it to the ICAP interface, connected to the ICAP component, that will take care of the physical reconfiguration process. The reconfiguration controller is aware of both the current configuration of the reconfigurable NoC (routing tables, topology and express lines) and the current communication requirements, such as the cores that have to communicate and the required bandwidth and latency. In this way, the controller is able to adapt the underlying communication infrastructure in order to satisfy communication requirements, even when they vary at run-time.

As previously hinted, the proposed reconfigurable NoC can be dynamically adapted to the current operating scenario by modifying network interfaces

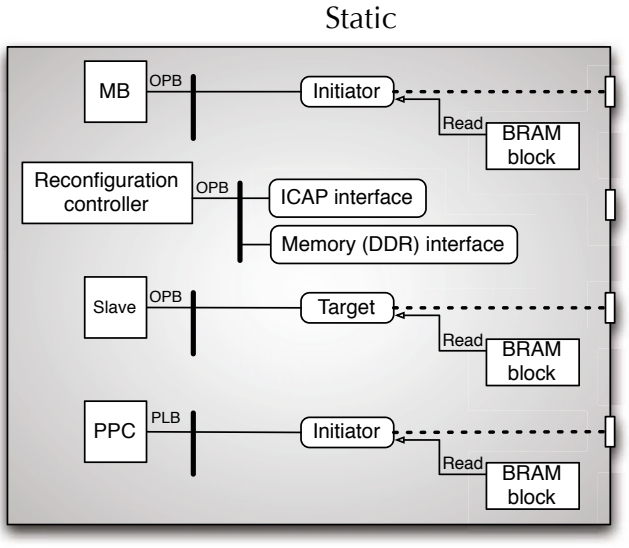


Fig. 3. A static part with a reconfiguration controller

routing tables at run-time, as described in Section 4.1. Furthermore, a dynamic change into the proposed reconfigurable NoC can also involve either the connection among switches (by inserting or removing express lines) or the whole topology, as described in Section 4.2.

4.1 Path Reconfiguration

Storing routing tables in BRAM blocks allows to dynamically change them at run-time in two different ways. The first solution is to write the new routing table with a simple *write operation* on the selected BRAM block. This write operation can be performed by the reconfiguration controller, that has to manage both the physical reconfiguration and the modification of BRAMs content, since routing tables have to be always consistent with respect to the current topology of the network. Using the reconfiguration controller for writing on BRAM blocks makes it necessary to directly connect it to each BRAM block, increasing the complexity and the area usage of the reconfigurable system.

A second solution consists of performing a partial dynamic reconfiguration of BRAM blocks, as described in [12]. This reconfiguration has to be performed by the reconfiguration controller, but in this case there is no need to directly connect it to each BRAM block, since these elements are updated by the controller using the configuration memory; thus, no area overhead is introduced. Performing this kind of reconfiguration enables dynamically changing BRAM blocks content (routing tables), in order to change the functionality of the network interfaces at run-time, while leaving unaltered all the logic implementing the functionality of the system; this allows a complete decoupling between routing tables and the logic that implements both the static and the reconfigurable components. The

main drawback of this solution is the increment of the time overhead of the network reconfigurations, as stated in Section 5.

4.2 Express Lines and Topology Reconfiguration

In order to exploit express lines reconfiguration, it is necessary to define a reconfigurable architecture that consists of several reconfigurable parts, in which it is possible to deploy the switches of the NoC. This can be done by means of the Early Access Partial Reconfiguration design flow [8] defined by Xilinx. This flow allows to implement a reconfigurable architecture containing an arbitrary set of reconfigurable regions (which shape is a rectangle spanning the whole height of the reconfigurable device, for FPGA of Virtex, Virtex II and Virtex II Pro families, or an arbitrary rectangle for FPGA of Virtex IV and Virtex V families). Both the static architecture and each reconfigurable module, which need to be placed in a single reconfigurable region, can be configured on the target device by using a specific bitstream, namely, a complete bitstream for the static part and a partial bitstreams for the reconfigurable modules. All the bitstreams generated by this flow are the ones used by the previously described reconfiguration controller to change the current configuration of the system; in other words, the reconfiguration controller is able to select a partial bitstream to be configured on the device in order to change the underlying communication infrastructure. In particular, if an express line has to be placed between to switches that belong to the same reconfigurable region, the reconfiguration controller has to configure a new version of the reconfigurable region in which the two switches are directly connected (through a new connection). A similar procedure can be applied to completely change the topology of the NoC. In this case, a deeper modification of the selected reconfigurable part is needed, in order to make it possible to change the number and the kind of the switches of the same reconfigurable part (and thus of the whole NoC).

The number of express lines that can be established between two reconfigurable regions has to be decided at design-time, since each bus-macro (which enables to establish a single reliable communication channel among different regions) has to be placed during the place and route phase of the architecture. Furthermore, the maximum number of express lines, which is always in the order of tens for FPGA of Virtex II, Virtex II Pro, Virtex IV and Virtex V families, is limited by the amount of available resources along the edge among reconfigurable and static regions; hence, it strictly depends both on the target reconfigurable device and on the shape of each reconfigurable or static region.

5 Experimental Results

This section presents a set of experimental results that validate the performance of the proposed reconfigurable architecture. These results have been achieved by implementing the proposed reconfigurable architectures on a Xilinx Virtex II Pro (XC2VP20) device. However, the same approach can be easily adapted to another device, even in a different family, such as Virtex IV and Virtex V.

5.1 Routing Tables Reconfiguration Analysis

Regarding **routing tables reconfiguration**, it can be performed in a few clock cycles if it is performed with a simple write operation. In particular, if routing tables reconfiguration is performed directly by the reconfiguration controller, the latency of the reconfiguration is only 2 clock cycles (0.02 μs at 100 MHz). On the other hand, by performing a partial dynamic configuration of BRAMs, even if both the area and the complexity overheads are not increased, the latency of the reconfiguration is considerably higher (2.24 ms at 100 MHz). Table 1 summarizes all the experimental results related to dynamic routing table reconfiguration.

Table 1. Routing tables reconfiguration experimental results

Reconfiguration model	Timing overhead (Clock cycles)	Timing overhead (ms) @ 100 MHz	Area overhead	Complexity increment
Write operation (reconfiguration controller)	2	0.00002	yes	yes
Partial dynamic reconfiguration of a single BRAM block	224242	2.24	no	no

5.2 Express Lines and Topology Reconfiguration Analysis

Even if **express lines reconfiguration** and **topology reconfiguration** can be used in order to achieve different modifications of the underlying network, from the timing overhead point of view, they are characterized by the same values, because the time required to reconfigure a reconfigurable region is exactly the same in both cases. Since the reconfiguration on Xilinx Virtex II Pro devices can only be performed with a 1D approach, the reconfiguration latency is directly related to the width of the reconfigurable region that has to be reconfigured. For instance the reconfiguration latency for a 4 slices width region, which can be filled with up to two switches, is around 21 ms, while a 20 slices width region, which can include up to ten switches, requires around 104 ms, as shown in Table 2.

In particular, regarding **express lines reconfiguration**, it can be exploited both to reduce the traffic on a part of the NoC and to decrease the latency between two switches. In order to better explain how it is possible to dynamically configure express lines on the proposed reconfigurable architecture, let us consider a simple 3x3 mesh network, similar to the one presented in Figure 4 (A). Without any express line, if the MicroBlaze 0 (MB 0) has to communicate with Slave 4, 3 hops (a path to a destination on a network can be considered as a series of hops, through switches) are necessary in order to go from Switch 0 (to which MB 0 is connected through an initiator network interface) to Switch 5 (to which Slave 4 is connected through a target network interface). To this end, each packet has to pass, for instance, through Switch 1 and Switch 2, in order to reach its final destination. In a similar way, the communication between PPC 0 and Slave 3 requires at least 2 hops (between Switch 6 and Switch 8), since each packet has also to pass through Switch 7.

Table 2. Express lines and topology reconfiguration results

Width of the reconfigurable slot (slices)	Reconfiguration latency (*ms*)	Bitstream size (Kb)
4	21	32
6	30	46
8	41	62
10	52	78
12	63	94
14	75	112
16	80	120
18	93	140
20	104	156

In the proposed reconfigurable architecture, it is possible to configure a direct connection between the port 3 of Switch 0 and the port 4 of Switch 5, and another one between the port 4 of Switch 6 and the port 3 of Switch 8. In this way, in addition to considerably reduce the congestion of Switches 1, 2 and 7, each communication between MB 0 and Slave 4 or PPC 0 and Slave 3 can be achieved with a single hop (from Switch 0 to Switch 4 and from Switch 6 and Switch 8), thus notably reducing the latency between these elements. The number of express lines that have to cross static parts has to be defined at design time (since the involved static parts have to be aware of them), while the number of express lines that lies within a single reconfigurable region only depends on the available resources of the selected region.

Since communication among the elements of the system can change at run-time in a non-predictable way, it is possible that the system reaches a status (for instance when the applications running on MB 0 and on PPC 0 change) in which MB 0 has to communicate with Slave 3 and PPC 0 has to communicate with Slave 4. With the configuration of Figure 4 (A), each master can reach the desired slave, by using both express lines, with 2 hops (from Switch 0 to Switch 8 and from Switch 6 to Switch 5). However, a problem that can arise is that these two paths share the link between the port 1 of Switch 5 and the port 2 of Switch 8, thus leading to a contention of the same resource. A possible solution is the partial dynamic reconfiguration of the reconfigurable region number 2 (Reconfigurable 2 in Figure 4), in order to achieve the configuration of the system shown in Figure 5, which can be achieved by adapting the routing tables according to the new configuration of the system, as described in Section 4.1). In this way, not only the congestion of the link between Switch 5 and Switch 8 is completely resolved, but also the latency of the two communication paths decreases to a single hop, *i.e.*, providing a circuit-based switching connection. Table 3 presents a comparison among the latency introduced by the NoC of Figure 4, the NoC of Figure 5 and a NoC in which express lines are not taken into account.

An important consideration is that, while the partial reconfiguration of the reconfigurable region 2 is performed, the communication among other parts of the system does not need to be interrupted, as long as it does not affect the

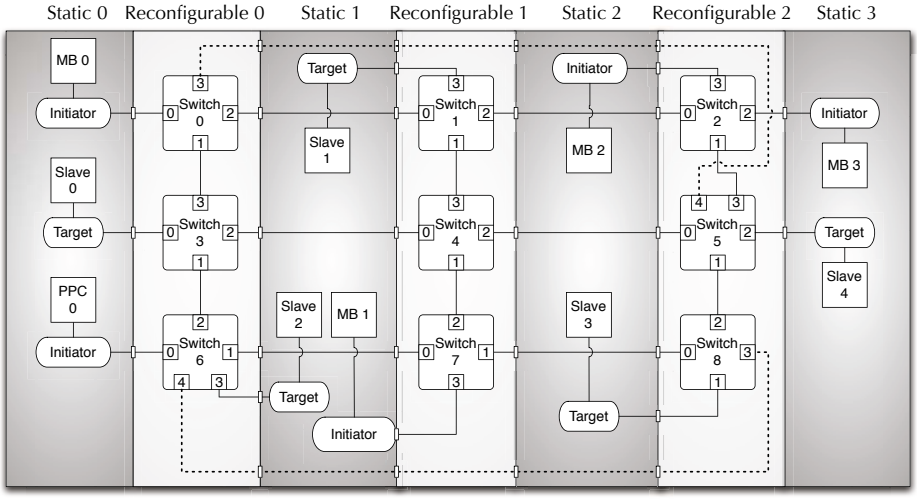


Fig. 4. Complete reconfigurable system schema, with an express line between Switch 0 and Switch 5 and another one between Switch 6 and Switch 8

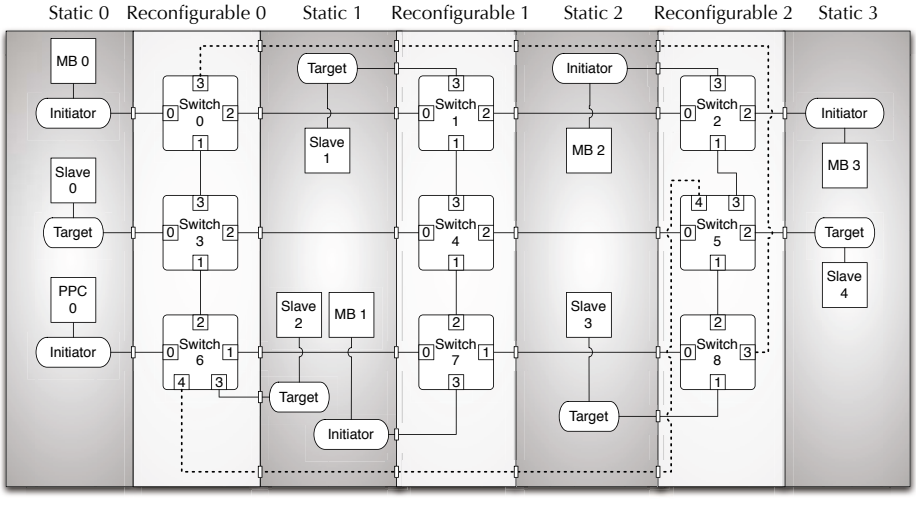


Fig. 5. Complete reconfigurable system schema, with an express line between Switch 0 and Switch 8 and another one between Switch 6 and Switch 5

region that is reconfigured. For instance, if MB 1 has to communicate with Slave 0 or Slave 1, this communication can take place even during the reconfiguration of the reconfigurable region 2.

The physical implementation of the previously presented architecture is shown in Figure 6, where A indicates the static part, while B, C and D represent the

Table 3. Latency introduced by the NoC

Source	Target	Figure 4 (number of hops)	Figure 5 (number of hops)	Mesh without express lines (number of hops)
MB 0	Slave 0	2	2	2
MB 0	Slave 1	2	2	2
MB 0	Slave 2	3	3	3
MB 0	Slave 3	3	2	5
MB 0	Slave 4	2	3	4
PPC 0	Slave 0	2	2	2
PPC 0	Slave 1	4	4	4
PPC 0	Slave 2	1	1	1
PPC 0	Slave 3	2	3	3
PPC 0	Slave 4	3	2	4

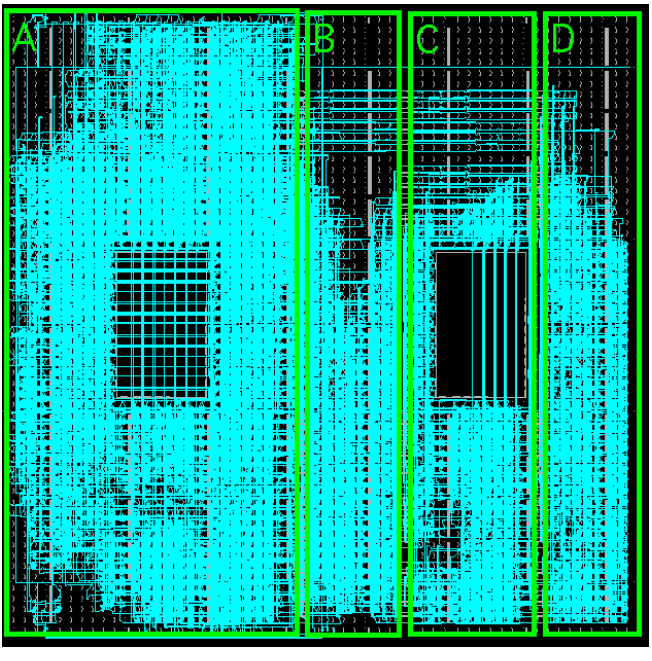


Fig. 6. Physical implementation of the reconfigurable 3x3 mesh

three reconfigurable regions (which width is, respectively, 16, 20, and 14 slices - the 18%, 22% and 16% of a XC2VP20 device). All the reconfigurable regions have been filled with three switches each one, in order to implement the previously presented 3x3 mesh.

Table 4 shows the experimental results regarding area usage and reconfiguration latency of the proposed architecture on a XC2VP20 device. The bus-macro overhead consists of 288 slices, while the complete 3x3 mesh requires 2237 slices.

Table 4. Area usage and reconfiguration latency results

	Area usage (slices)	**Area usage (%)**	**Reconfiguration latency (*ms*)**
Reconfigurable region B	800	8.6	80
Reconfigurable region C	637	6.9	104
Reconfigurable region D	800	8.6	75
Complete 3x3 mesh	2237	24.1	259
Bus-macro overhead	288	3.1	

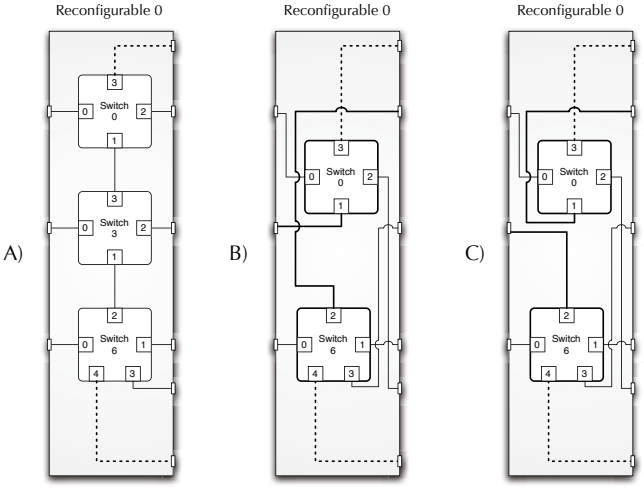


Fig. 7. Alteration of the original mesh topology through the reconfiguration of the original reconfigurable slot 0 (Reconfigurable 0) (A) with two different versions of the subnetwork (B and C)

Thus, the overhead introduced by the proposed approach represents the 10% (on average) of the initial NoC.

Furthermore, it is possible to configure at least two express lines in the implemented architecture, and since each express line of the presented design has a latency lower than 4 *ns*, it is possible to exploit each direct connection within a single clock cycle at 100 MHz (while the latency required by the connection passing through Switch 1, Switch 2 and Switch 5 is greater then 40 *ns*, *i.e.*, 4 clock cycles).

On the other hand, a **topology reconfiguration** can be exploited on the same architecture in order to adopt a specific NoC for each application that

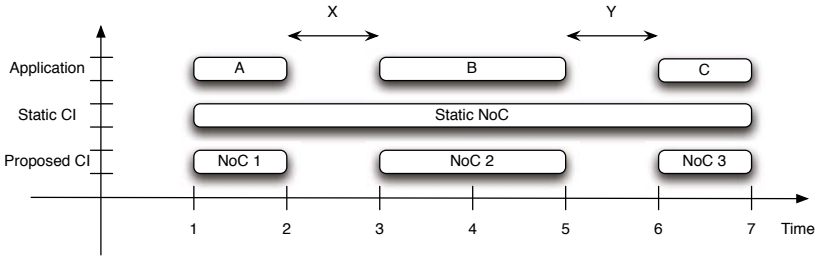


Fig. 8. Temporal evolution of a generic system

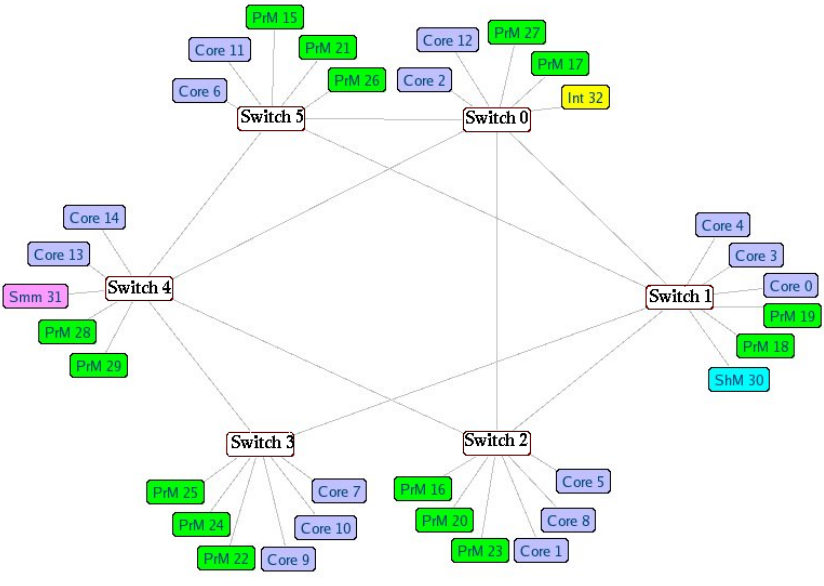


Fig. 9. Topology of the static NoC

has to be run on the system. In order to completely change the topology of the Network-On-Chip, a deeper modification of the selected reconfigurable part is needed, since both the number and the kind of the switches of the same reconfigurable part can be changed. As an example, let us consider that MB 0 has to communicate with both Slave 0 and Slave 2 with the lowest latency possible. In order to satisfy this strict requirement, it is necessary to change the original topology of the network, by altering the mesh (in particular, the original reconfigurable slot number 0 shown in Figure 7 A) as shown in Figure 7 B. In the reconfigurable module shown in Figure 7 B, in fact, MB 0, Slave 0 and Slave 2 are all connected to Switch 0, in order to make it possible for MB 0 to reach Slave 0 and Slave 2 without any hop.

Another case in which a reconfiguration of the topology can lead to meet communication requirements is, for instance, when both MB 0 has to communicate

Table 5. Specific NoCs experimental results

NoC	Number of switches	Average latency (clock cycles)	Average power consumption (mW)
Static NoC	6	5.96	278.021
NoC 1	4	3.9	211.789
NoC 2	4	4	204.308
NoC 3	4	4.07	216.519

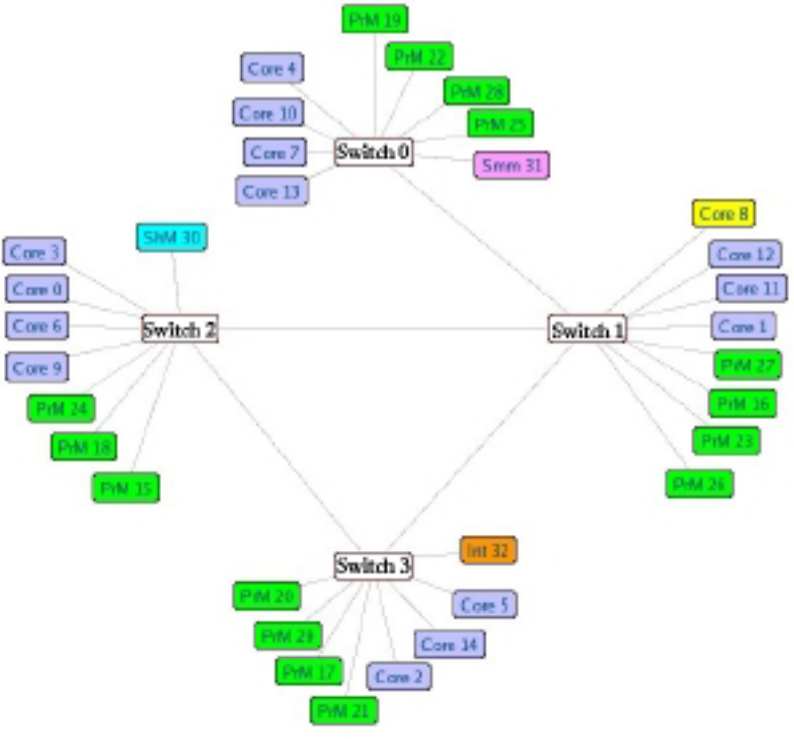


Fig. 10. Topology of the NoC 2

with Slave 2 and MB 1 has to communicate with Slave 0 with the lowest latency possible. In this case, the reconfigurable module shown in Figure 7 C can be configured in the reconfigurable slot number 0 (Reconfigurable 0), in order to connect both MB 0 and Slave 2 to Switch 0, and MB1 and Slave 0 to Switch 6. Thanks to this reconfiguration of the topology it is possible to establish both the required communication channels without any overhead in terms of hops between switches, since all the components that has to communicate between them have been connected to the same switch.

We have validated the proposed approach with three different versions of real-life SoC benchmarks, namely, a video processing application of 32 cores

Table 6. Area overhead, timing performance and features comparison among state of the art solutions and the proposed approach

Approach	CuNoC ([9])	CoNoChi ([14])	Proposed work
Switch size (slices)	from 72 to 491	from 363 to 493	from 86 to 267
Communication infrastructure size (slices)	All the available resources	NA	2727 for a 3x3 mesh
Frequency (MHz)	from 272 to 336	from 66 to 111	170
Single switch latency (clock cycles)	2	5	1
Single switch latency (*ns*)	from 6 to 7.4	from 45 to 76	5.9
Flow control	NA	NA	Supported
Path reconfiguration	Not supported	NA	Supported
Express lines reconfiguration	Not supported	Not supported	Supported
Topology reconfiguration	Not supported	Supported	Supported

(A), a Video Object Plane Decoder of 34 cores (B) and an image processing application of 23 cores (C). We refer the readers to [4] for the communication characteristics of these benchmarks. As shown in Figure 8, if these different applications have to deployed on the same system, it is possible to employ either a static network or three specific NoCs, each one designed ad-hoc for the particular application. The second choice can be adopted if the time interval that occurs between two consecutive applications is greater than the time overhead required by the reconfiguration process; thus, it is possible to transparently change the underlying NoC.

In order to test the application of our dynamically reconfigurable framework in this context, we have developed a static NoC and three specific ones for each of the three aforementioned SoC benchmarks application. As shown in Figure 9, the static NoC consists of 6 switches (1 switch of 8x8, 2 switches of 9x9, 2 switches 10x10 and 1 switch of 11x11), whileboth NoC 1 (for application A) and NoC 3 (for application C) consists of 4 switches (3 switches of 10x10 and 1 switch of 11x11) and NoC 2 (for application B) consists of 4 switches (1 switch of 10x9, 2 switches of 10x10 and 1 switch of 10x11), as shown in Figure 10. The static NoC option, as shown in Table 5, is characterized by a higher area usage, a higher average power consumption (evaluated as proposed in [2]) and a higher average latency, with respect to the three ad-hoc NoCs specifically designed for each application. Using the specific NoCs, it can be reported reductions of 34% in latency and 24% in power consumption. Finally, the overall latency for the reconfiguration of the NoC to be used at run-time is very limited, making it applicable in real-life scenarios where applications are switched dynamically by users.

As previously hinted, the reconfiguration latency of a reconfigurable region strictly depends on its size. For instance, the reconfiguration latency for a 4 slices width region (that can be filled with up to two switches) is around 21 ms, while a 20 slices width region (that can be filled with up to ten switches) requires around 104 ms.

Finally, Table 6 presents a comparison among state-of-the-art solutions and our approach, which shows the clear benefits of our approach regarding area overhead reduction, timing performance improvements and enhancements of the reconfiguration features.

6 Conclusions

NoCs have been proposed as a very promising scalable communication paradigm SoCs. However, methods that provide versatility enhancements of initial NoC designs to changing working conditions, imposed by variable sets of executed applications at run-time, are key to design competitive NoCs in industrial SoCs. In this work we have presented a novel NoC reconfigurable framework that can reconfigure the NoC topology at run-time, as well as enabling path reconfiguration and express lines creation/removal, while introducing an overhead on average of 10% of an initial static NoC design. Moreover, our experimental results have shown that in the proposed framework, on average, a reconfigurable switch only occupies 41% of the slices needed by a CoNoChi switch, the state-of-the-art reconfigurable NoC approach, whereas our reconfigurable NoC can run at almost double the frequency (170 MHz *vs.* 88.5 MHz) of CoNoChi. Finally, our approach introduces less than one tenth of the latency introduced by a CoNochi switch (respectively, 5.9 ms and 60.5 ms). Thus, it is a promising framework to be applied to commercial NoC-based SoC solutions.

Acknowledgments

This work was partially supported by the HiPEAC network of excellence (www.hipeac.net), the Swiss NSF Research Grant 20021-109450/1 and Spanish Government Research Grants TIN2005-5619, TIN2008-00508 and CSD00C-07-20811.

References

1. Ahmad, B., Erdogan, A.T., Khawam, S.: Architecture of a dynamically reconfigurable noc for adaptive reconfigurable mpsoc. In: First NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2006, June 15-18, pp. 405–411 (2006)
2. Angiolini, F., Meloni, P., Carta, S., Benini, L., Raffo, L.: Contrasting a NoC and a traditional interconnect fabric with layout awareness. In: Proceedings of Design, Automation and Test in Europe Conference (DATE 2006), Munich, Germany, pp. 124–129 (2006)

3. Benini, L., De Micheli, G. (eds.): Networks on chips: Technology and Tools. Morgan Kaufmann Publishers, San Francisco (2006)
4. Bertozzi, D., Jalabert, A., Murali, S., Tamhankar, R., Stergiou, S., Benini, L., De Micheli, G.: Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. IEEE Trans. Parallel Distrib. Syst. 16(2), 113–129 (2005)
5. Brebner, G., Levi, D.: Networking on chip with platform fpgas. In: Proceedings of the 2003 International Conference on Field-Programmable Technology (FPT), December 2003, pp. 13–20 (2003)
6. Ching, D., Schaumont, P., Verbauwhede, I.: Integrated modeling and generation of a reconfigurable network-on-chip. In: Proceedings of 18th International Conference on Parallel and Distributed Processing Symposium, April 26-30, p. 139 (2004)
7. Hansson, A., Goossens, K.: Trade-offs in the configuration of a network on chip for multiple use-cases. In: First International Symposium on Networks-on-Chip, NOCS 2007, May 7-9, pp. 233–242 (2007)
8. Xilinx Inc. Early Access Partial Reconfiguration Guide. Xilinx Inc. (2006)
9. Jovanovic, S., Tanougast, C., Weber, S., Bobda, C.: Cunoc: A scalable dynamic noc for dynamically reconfigurable fpgas. In: International Conference on Field Programmable Logic and Applications, FPL 2007, August 27-29, pp. 753–756 (2007)
10. Kumar, A., Hansson, A., Huisken, J., Corporaal, H.: An fpga design flow for reconfigurable network-based multi-processor systems on chip. In: Design, Automation and Test in Europe Conference and Exhibition, DATE 2007, April 16-20, pp. 1–6 (2007)
11. Lysne, O., Pinkston, T.M., Duato, J.: A methodology for developing dynamic network reconfiguration processes. In: ICPP, p. 77 (2003)
12. Montone, A., Rana, V., Santambrogio, M.D., Sciuto, D.: Harpe: a harvard-based processing element tailored for partial dynamic reconfigurable architectures. In: 22nd IEEE International Parallel and Distributed Processing Symposium - 15th Reconfigurable Architectures Workshop (April 2008)
13. Murali, S., Coenen, M., Radulescu, A., Goossens, K., De Micheli, G.: Mapping and configuration methods for multi-use-case networks on chips. In: Proceedings of the 2006 conference on Asia South Pacific design automation (ASP-DAC), pp. 146–151. ACM Press, New York (2006)
14. Pionteck, T., Koch, R., Albrecht, C.: Applying partial reconfiguration to networks-on-chips. In: International Conference on Field Programmable Logic and Applications, FPL 2006, August 28-30, pp. 1–6 (2006)
15. Vicentelli, A., Martin, G.: A vision for embedded systems: Platform-based design and software. IEEE Design and Test - Special Issue of Computers 18(6), 23–33 (2001)

Fast Instruction Memory Hierarchy Power Exploration for Embedded Systems

Nikolaos Kroupis[1] and Dimitrios Soudris[2]

[1] Department of Information and Telecommunication Technology,
Technological Institute of Larisa,
411 10 Larisa, Greece
nkroup@gmail.com
[2] School of Electrical & Computer Engineering, Department of Computer Science,
National Technical University of Athens,
9 Heroon Polytechneiou, Zographou Campus, 157 80 Athens, Greece
dsoudris@microlab.ntua.gr

Abstract. A typical instruction memory design exploration process using simulation tools for various cache parameters is a rather time-consuming process, even for low complexity applications. In order to design a power efficient memory hierarchy of an embedded system, a huge number of system simulations are needed for all the different instruction memory hierarchies, because many cache memory parameters should be explored. Exhaustive search of design space using simulation is too slow procedure and needs hundreds of simulations to find the optimal cache configuration. This chapter provides fast and accurate estimates of a multi-level instruction memory hierarchy. Using a detail methodology for estimating the number of instruction cache misses of the instruction cache levels and power models; we estimate within a reasonable time the power consumption among these hierarchies. In order to automate the estimation procedure, a novel software tool named FICA implements the proposed methodology, which automatically estimates the total energy in instruction memory hierarchy and reports the optimal one.

1 Introduction to Instruction Caches

Cache memories have become a major factor to bridge the bottleneck between the relatively slow access time to main memory and the faster clock rate of today's processors. The power consumed by the memory hierarchy of a micro-processor can contribute to as much as 50% of the total microprocessor system power [1].

A cache is a small but fast memory and it is placed closer to the CPU. A cache block is the amount of data transferred between the main memory and the cache from any memory operation. A cache can also be divided into sets where each set contains N (usually N is 1,2,4,8 etc.) cache blocks. Fig. 1 classifies a cache on the basis of its contents and organization. For a direct mapped cache, each set contains only one cache block. For an *n*-way set associative cache each set contains n cache blocks. Fig. 2 shows the implementation of a direct mapped cache.

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 251–270, 2010.

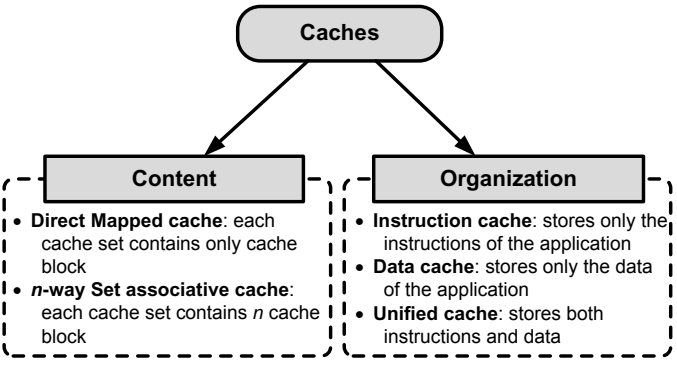


Fig. 1. Caches are classified on the basis of their content and organization

Each cache block includes a tag to show which memory location is present in this block, a data field holding the contents of that memory location, and a valid tag to show whether the contents of this cache block is valid or not. An address (referenced location) is divided into three sections. The index is used to select which cache block to check. The tag is compared against to the tag value in the line selected by the index. If the address tag matches the tag value in the block, that block contains the desired memory location. If the tag does not match the tag value in the block, then it is a cache miss. If the length of the data field is longer than the minimum addressable unit, then the least significant bits of the address are used as an offset to select the required value from the data field.

Nowadays the programmable systems usually contain one or two levels of caches, in order to reduce the main memory transfer delay and the power consumption. Tuning cache parameters to the needs of a particular application can save energy. Every application has different cache requirements that cannot be efficiently satisfied with one predetermined cache configuration. A single-level cache may have dozens of different cache configurations, and interdependent multi-level caches lead to thousands of different cache configurations. The simulation of cache memories is common practice to determine the best configuration of caches during the design of computer architectures. It has also been used to evaluate compiler optimizations with respect to cache performance. Exhaustively searching the design space is a too slow procedure, even if it would be fully automatic.

Unfortunately, the cache analysis of a program can increase significantly the program's execution time frequently by two orders of a magnitude. Thus, cache simulation has been limited to the analysis of programs with a small or moderate execution time and still requires considerable experimentation time before yielding results. In reality programs often run for a long time, but cache simulation simply becomes unfeasible with conventional methods. The huge time overhead of cache simulation is imposed by the necessity of tracking the execution order of instructions.

No certain cache configuration would be efficient for all applications, seeing as every application has different cache requirements. Thus, finding the best cache configuration for a particular application could save energy and time. But, to explore all possible cache configurations it is not so easy task it would not be a viable solution,

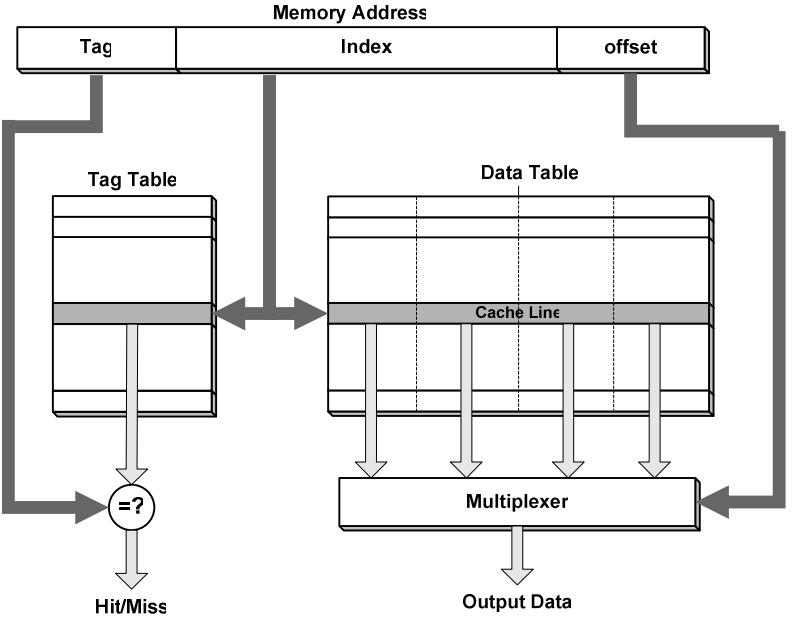


Fig. 2. Cache Memory Architecture

due to the prolonged of exploration time involved. For instance, if we consider only variations in the parameters of one level cache such as cache size, block (line) size and associativity, an designspace of dozens of configurations [2] will be explored ed in order to find out the optimal one for a given application. In the case of multilevel memory hierarchies, for instance hat include a second level of cache, where both levels have separated instruction and data caches, few hundreds of configurations have to be tested.

Each instruction memory hierarchy has different energy consumption. The energy consumed in the instructions memories is directly-related to the memory architectures parameters (e.g., cache size, block size, associativity) and to the number of accesses to every memory hierarchy level. In order to design an efficient embedded system the total instruction memory energy consumption should be shrank. Defining the instruction memory and the instruction cache parameters, the total energy consumed by the instruction memory hierarchy can be computed.

2 Simulation and Estimation Methods: Overview

Adjusting the parameters of an application's cache memory can save 60% of energy consumption, on average [2]. By tuning these parameters, the cache can be customized to a particular application. However, no single cache configuration would be effective for all applications. Thus, strategies to explore the cache parameters can be applied to customize the cache structure to a given application. The proposed techniques can be classified into two categories. The techniques of the first category are

targeting to reduce the exploration search space, while the second one to reduce the miss rate estimation time for simulation of a certain cache hierarchy.

In [3], an automated method for adjusting two-level cache memory hierarchy in order to reduce energy consumption in embedded applications was presented. The proposed method, two-level cache exploration heuristic considering TECH-CYCLES method, make a small search in the space of configurations of the two-level cache hierarchy, analyzing the impact of each parameter in terms of energy and number of cycles spent for a given application.

Zhang and Vahid [2] presented a cache architecture that can find the optimal set of cache configurations for a given application. Such architecture would be very useful in prototyping platforms, eliminating the need for time-consuming simulations to find optimal cache configurations. Gordon et. al [4] presented an automated method for tuning two-level caches to embedded applications for reducing energy consumption. The method is applicable to both a simulation-based exploration environment and a hardware based system prototyping environment. Platune was introduced by Givargis and Vahid in [5], which is used to explore automatically the large configuration space of such a SoC platform. The power estimation techniques for processors, caches, memories, buses, and peripherals combined with the design space exploration algorithm deployed by Platune, form a methodology for design of tuning frameworks for parameterized SOC platforms.

The previously-referred methods are based on the instruction set simulator, which provides cycle accurate estimations, but from the other hand it is a very slow procedure. Their main disadvantage is the huge needed time cost when we have to explore a large number of different instruction memory hierarchies.

New techniques have been proposed to reduce the simulation time, which were presented in [6], [7], [8] and [9]. In particular, a technique called inline tracing can be used to generate the trace of addresses with much less overhead than trapping or simulation. Measurement instructions are inserted in the program to record the addresses that are referenced during the execution. Borg, Kessler, and Wall [6] modified some programs at link time to write addresses to a trace buffer, and these addresses were analyzed by a separate higher priority process. The time required to generate the trace of addresses was reduced by reserving five of the general purpose registers to avoid memory references in the trace generation code.

Mueller and Whalley [7] provided a method for instruction cache analysis, which outperforms the conventional trace-driven methods. This method, named static cache simulation, analyzes a program for a given cache configuration and determines, prior to execution time, if an instruction reference will result in a cache hit or miss. The total number of cache hits and misses can be extracted from the frequency counters at program exit. In order to use this technique, the designer should make changes in the compiler of the processor, which are restricted most of the times, when we use commercial tools and compilers.

A simulation-based methodology, focused on an approximate model of the cache and the multi-tasking reactive software, that allows one to trade off-smoothly between accuracy and simulation speed, has been proposed by Lajolo et. al. [8]. The methodology reduces the simulation time, taking into account the intra-task conflicts and considering only a finite number of previous task executions.

Nohl et. al [9] presented a simulation-based technique, which meets the requirements for both, the high simulation speed and maximum flexibility. This simulation technique, called just-in-time cache compiled simulation technique, can be utilized for architecture design, as well as for end-user software development. This technique is integrated into the retargetable LISA processor design platform [10].

A brief description of the simulation/estimation methods and techniques were available in [11]. In this chapter, a novel methodology aiming to find the optimal instruction cache memory hierarchy of the system in terms of the power consumption. High estimation accuracy can be achieved within an affordable estimation time cost. The high-level estimation decisions are very useful for a fast exploration among several instruction cache configurations. The developed software tool based on the methodology explores many instruction cache configurations considering multi-level cache memory hierarchy. The basic concept of the methodology is the straightforward relationship for specific characteristics between the high-level application description code and its corresponding assembly code. The developed tool achieves speedup orders of magnitude in the miss rate and power consumption estimation and time cost comparing to existing methods, while the estimation accuracy is higher than 90%. The experimental results show the efficiency of the proposed methodology and the estimation tool in terms of accuracy and the exploration time for a system consisting by one or two levels of instruction cache.

3 Instruction Cache Miss Rate Estimation

The power consumption of the instruction memory hierarchy depends on the number of accesses to each memory level. The crucial point is to estimate the number of accesses to each cache level and to find the miss rate of each level as well as the total number of executed instructions. A miss rate estimation methodology based on the correlation between the high-level description code (e.g. C) of the application and its associated assembly code was proposed in [12]. In particular, the methodology is based on a set of analytical equations which calculate the number of cache misses of a loop proposed by Liveris *et.al*. [13]. Using the compiler of the chosen processor, the assembly code of the application can be derived. The crucial point of the methodology is that the number of conditional branches in both the C code and its assembly code is equal. Thus, executing the C code we can find the number of passes from every branch. The values correspond to the assembly code, and thus we can find how many times each assembly branch instruction is executed. Creating the Control Flow Graph (CFG) of the assembly code, the number of executions of all application's assembly instructions can be calculated. The miss rate estimation is accomplished by the assembly code processing procedure and the data extracted from the application execution. Thus, the estimation time depends on the code (assembly and C) processing time and the application execution time in a general-purpose processor. The total estimation time cost is much smaller than that obtained by the trace-driven simulation techniques.

A cache read miss from an instruction cache generally causes increased delay, because the processor has to wait (stall) until the instruction is fetched from main

memory. Cache misses can be classified into three categories the compulsory, the capacity and the conflict misses as following:

- *Compulsory misses* are those misses caused by the first reference to a datum. Cache size and associativity make no difference to the number of compulsory misses. Compulsory misses are sometimes referred to as cold misses.
- *Capacity misses* are those misses that occur regardless of associativity or block size, solely due to the finite size of the cache. The curve of capacity miss rate versus cache size gives some measure of the temporal locality of a particular reference stream.
- *Conflict misses* are those misses that could have been avoided, had the cache not evicted an entry earlier. Conflict misses can be further broken down into mapping misses, that are unavoidable given a particular amount of associativity, and replacement misses, which are due to the particular victim choice of the replacement policy.

In order to model the number of cache misses of a nested loop, analytical formulas have been proposed in [13]. Given the cache size (cache parameters), these analytical formulas can estimate the number of cache misses. The explanation of these formulas is presented in [13]. Here, we provide only the necessary information regarding with the high-level estimation formulas. Assuming a specific cache, in order to estimate the misses of an application, we split an application into a number of nested loops. For every loop the misses are estimated individually.

Depending on the loop size mapped to the cache size, the assumed loops are categorized in three different types: *Loop Type* 1, *Loop Type* 2 and *Loop Type* 3, each of which the capacity misses, in number of blocks, is shown in Fig. 3.

Given a nested loop with N iterations and a total size of instructions in assembly code, L_s, a cache memory with size, C_s, (in instructions), and a block size, B_s, (cache line length), the number of misses, N_misses, can be calculated by using the following formulas [13]:

Loop Type 1: if $L\_s \le C\_s$ then:

$$Num\ \_\ misses\ = \frac{L\_s}{B\_s} \tag{1}$$

Loop Type 2: if $C\_s < L\_s < 2 \times C\_s$ then:

$$Num\_misses = \frac{L\_s}{B\_s} + (N-1) \times 2 \times \frac{L\_s \bmod C\_s}{B\_s} \tag{2}$$

Loop Type 3: if $2 \times C\_s \le L\_s$ then:

$$Num\_misses = N \times \frac{L\_s}{B\_s} \tag{3}$$

The miss rate is given by the formula:

$$Miss\_rate = \frac{Num\_misses}{Num\_references} \tag{4}$$

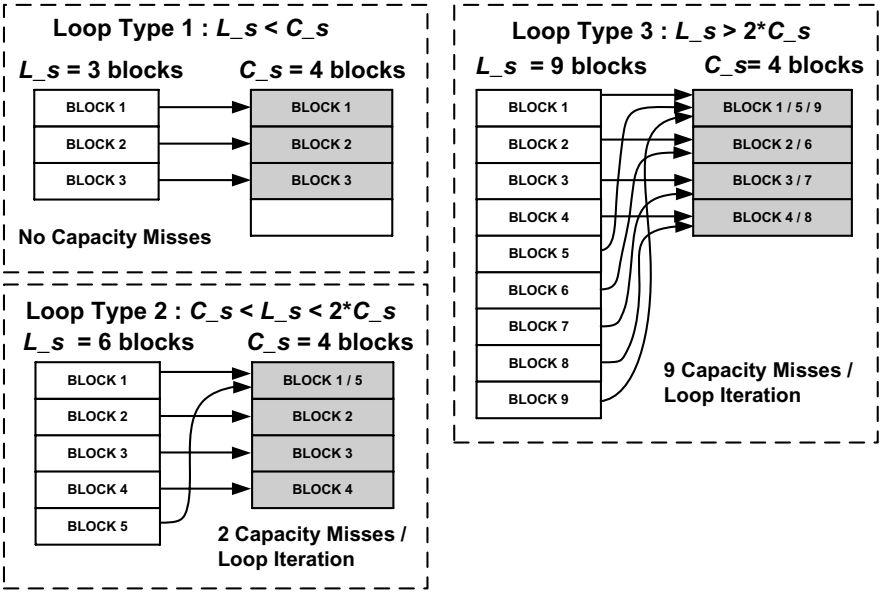


Fig. 3. Depending on the loop's size compared to cache size, the nested loops are classified into three categories

where *Num_references* is the number of memory references from the processor to memory with

$$Num\_references = \frac{L\_s}{B\_s} \times N \tag{5}$$

The proposed methodology consists of four stages illustrated in Fig. 4. The first stage aims at the calculation of the number of executions (passes) of all branches of the application C code. Thus, the number of executions of every leaf of the Control Flow Graph (CFG) is evaluated by the application execution. Determining the branches of the high-level application code, we can find the number of executions within these branches executing the code. This stage is a platform-independent process and thus, its results can be used in any programmable platform.

The second stage estimates the number of executions of each instruction and eventually, the total number of the executed instructions. Given the assembly code of the application, the second step creates the CFG of the application and associates the number of executions executed from the first stage. It consists of: (*i*) the determination of assembly code branches, (*ii*) the creation of CFG, (*iii*) the assignment of counter values to CFG nodes and (*iv*) the calculation of the execution cost of the rest CFG nodes.

The third stage of the methodology is platform-dependent and contains two steps: (*i*) the creation of all the unique execution paths of each loop and (*ii*) the computation of number of instructions and iterations associated with a unique path. Exploring all

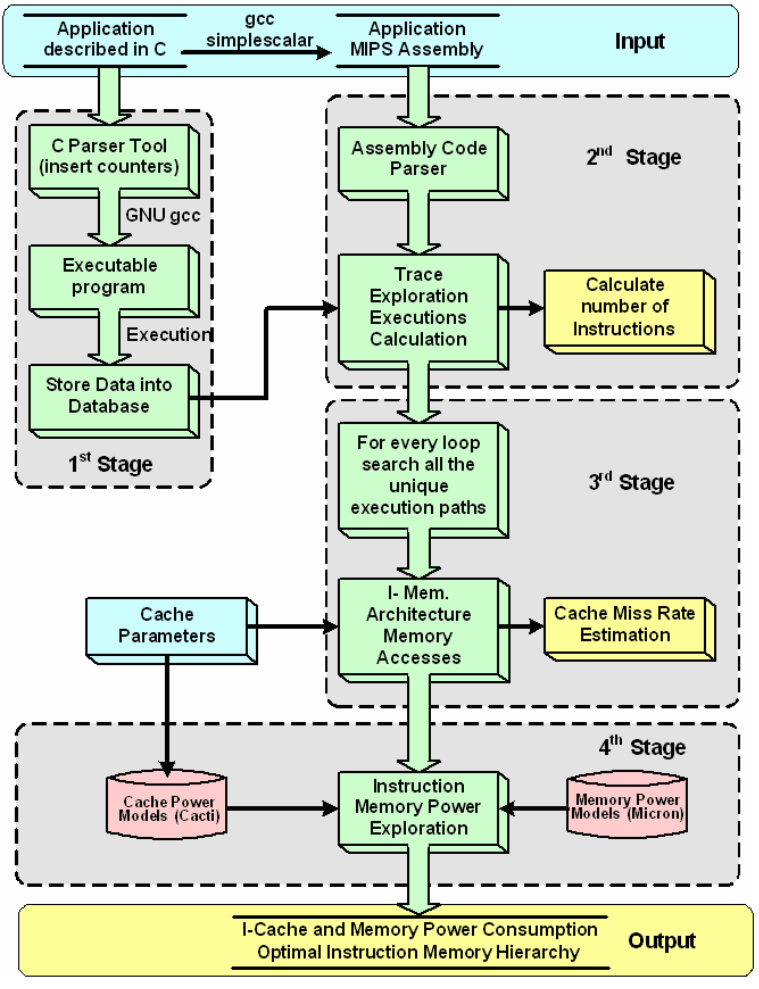


Fig. 4. The proposed methodology for estimating the miss rate and the power consumption of a multi-level instruction memory cache hierarchy

the paths of the CFG of an application, we determine the loops and the size (in numbers of instructions), as well as the number of executions of each loop. Furthermore, from the rest of the conditional branches (if / else), we create all the unique execution paths inside every loop, together with the number of executions of each unique path. Comparing the size in terms of number of instruction of every unique path with the instruction cache size the number of cache misses is estimated. The number of cache misses can be computed all together for variable cache sizes and architectures and multi-level memories, by a single run of FICA tool. This is one of the advantages of the methodology. The fourth stage contains the power model of the instruction cache and the memory, which is described in the next paragraph.

4 Instruction Memory Power Consumption

An architecture exploration among the variable instruction cache configurations is needed in order to find the optimal memory hierarchy in terms of power consumption. The power consumption of the memory depends on their characteristics (type, size etc.) and number of accesses. Power consumption models are estimates of the energy consumed by a cache per memory access [14] [15]. Cacti 4.0 power model [14] was developed to estimate the power consumption of on-chip cache memories. Given the basic cache parameters such as size, block size, associativity and design technology, it estimates the energy consumption per access (read and write) to this cache memory. The power model proposed by Micron [15] is targeting to the estimation of off-chip SDRAM and DDR memories based on memory parameters. Such, the most important parameter to estimate the power consumption on a memory is the number of accesses. Based on the cache miss ratio estimation of the third stage, analytical questions define the number of accesses of every cache memory. The last step of the fourth stage assigns the energy consumption per access to every cache level and computes the application's energy consumption of every level. Summing up the energy of each level, the total consumed energy of the instruction memory hierarchy is estimated.

Power models for the memory hierarchy are needed to find the optimal instruction memory hierarchy. Using the number of accesses (read/write) of each cache level, which are computed in the third stage and memory power models, the tool automatically estimates the total energy consumed in instruction hierarchy. The number of instruction cache levels varies from one to *n*, but typical embedded system contains usually one or two. The developed software tool with one run estimates all the cache miss rates and the energy consumed on them, architectures which contains from one to *n* and for all cache parameters and combination between the different caches. The tool reports the power consumption of every cache hierarchy and the designer can choose the optimal in terms of energy consumption.

5 The Estimation Methodology Using an Example

The proposed methodology is based on the correlation between the high-level description code (e.g. C) of the application and its corresponding assembly code. Using the compiler of the chosen processor, we can derive the assembly code of the application. Here, we are providing the proposed methodology using an example with a simple C code. The procedure of the four stages of the methodology of the sample C code are presented in detail in Fig. 5-6. The sample code contains one loop and a conditional branch into the loop, such there are two branches. The first stage detects (*Stage* 1*, Step* 1) the two branches and automatically inserts counters after every branch in C code (*Stage* 1*, Step* 2) and executing the C code (*Stage* 1*, Step* 3) we can find the number of executions of every branch (*Stage* 1*, Output*). The values of the counters provide the number of executions of every branch of the C code.

The second stage has as input the equivalent assembly code of the application, and parsing the assembly code (*Stage* 2*, Step* 1), the Control Flow Graph (CFG) of the application (*Stage* 2*, Step* 2) can be derived. Corresponding the values of the counters to the specific places in the CFG (*Stage* 2*, Step* 3), we can calculate how many times the

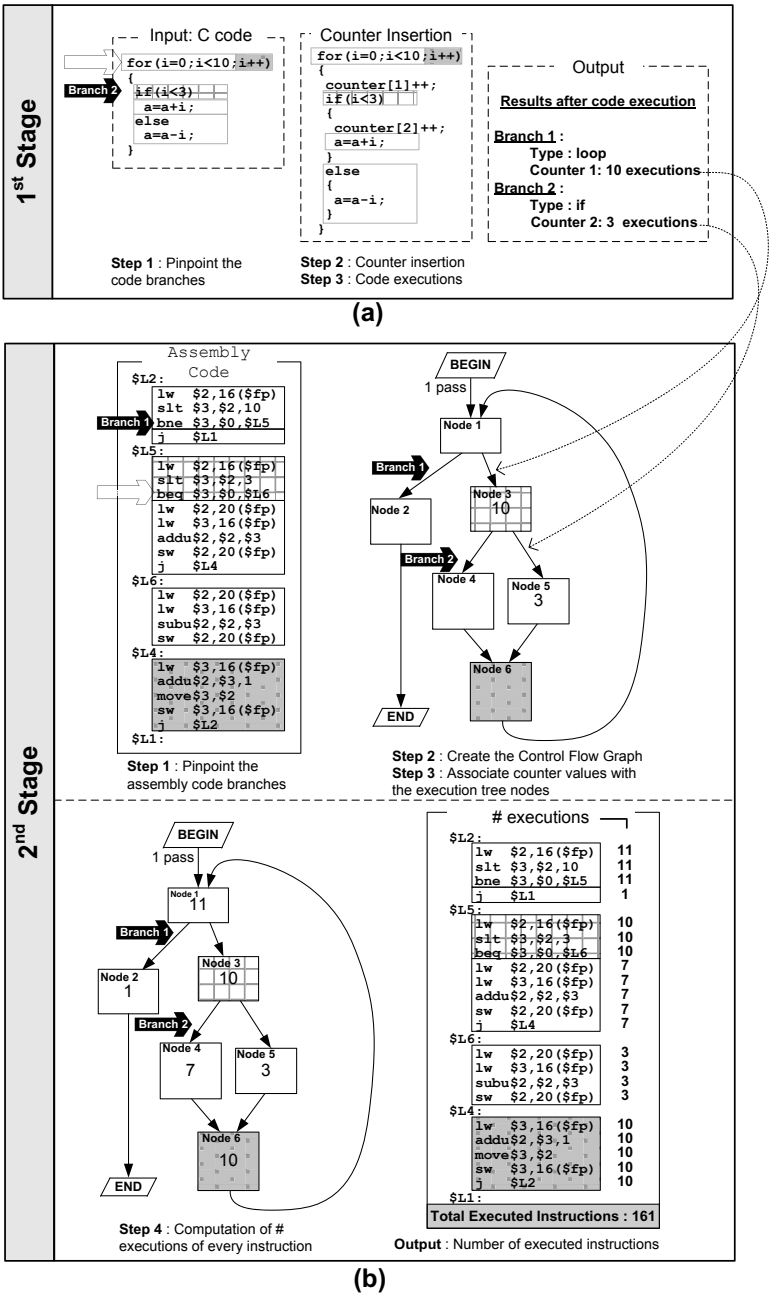
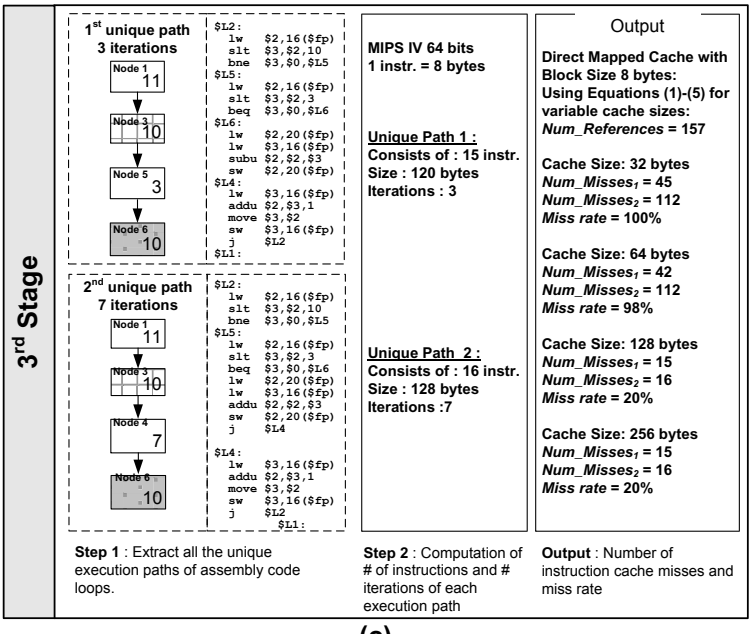
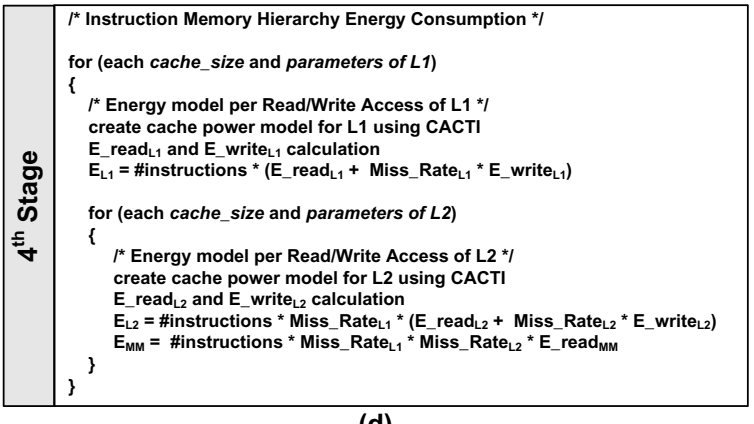


Fig. 5. First and second stage of the estimation methodology



(c)



(d)

Fig. 6. Third and forth stage of the estimation methodology

branches of the assembly code are executed. Using an iterative procedure we calculate the number of executions from all nodes of the CFG (*Stage* 2, *Step* 4). Thus, summarizing the number of executions of all application's assembly instructions, the total number of executed instructions can be calculated (*Stage* 2, *Output*).

The third stage the methodology explores the CFG of the application and can determine the loops and the size (in number of instructions) as well as the number of executions of each loop (*Stage* 3, *Step* 1). Taking into account the conditional

branches (*if / else*), we create all the unique execution paths inside every loop and eventually, we calculate the number of executions of every unique path (*Stage* 3*, Step* 2). Comparing the size of every unique path in terms of number of instruction with the instruction cache size, the number of cache misses is estimated. The number of cache misses is computed for variable cache memory sizes through a single run of the developed tool (*Stage* 3*, Output*), which is one of the main advantages of the proposed methodology.

The forth stage aims at the estimation of energy consumption of the instruction memory hierarchy. The energy consumed by the memory is dependent on the memory technology, the memory type and the number of accesses (reads/writes). Using the CACTI power model for the cache memories, we can create detail power parameters for various cache types. In the case of two levels of instruction cache, we need two loops to cover all the possible combinations between the two caches in terms of size and parameters. Thus, in the Fig. 6 (*Stage* 4) the outer loop explores all the possible L1 cache architectures, while the second one explores all the possible L2 cache architectures for every L1 architecture. Firstly, we create the power model for each cache architecture using the CACTI power model. Applying the cache parameter to CACTI for each cache architecture, we compute the *E_read* and *E_write* power parameters of the cache, for every read and write access, respectively. Secondly, multiplying the number of read and writes of every cache with the power values we compute the total energy consumption to each cache level. Finally, using the power model of memories we can also compute the energy consumed of the system's main memory.

6 Comparison Results

In order to evaluate the proposed estimation technique we compare the results, which are taken using the developed tool with the simulation-based measurements. We considered as implementation platform the 64-bit processor core MIPS IV, while the measurements were taken by Simplescalar tool [16], the accurate instruction set simulator of MIPS processor. Simplescalar includes instruction set simulator, fast-instruction simulator and cache simulator, and can simulate architectures with instruction, data and mixed instruction-data caches with one or two memory hierarchy layers. In order to evaluate the proposed methodology, a set of benchmarks from various signal processing applications, such as MPEG-4, JPEG, Filtering and H.263 are used. In particular, we use five Motion Estimation algorithms: (*i*) Full Search (FS) [17], (*ii*) Hierarchical Search (HS) [18], (*iii*) Three Step Logarithmic Step (3SLOG) [17], (*iv*) Parallel Hierarchical One Dimensional Search (PHODS) [17] and (*v*) Spiral Search (SS) [19]. It has been noted that their complexity ranged from 60 to 80% of the total complexity of video encoding (MPEG-4) [17]. Also, we have used the 1-D Wavelet transformation [20], the Cavity Detector [21] and the Self Organized Feature Map Color Quantization (CQ) [22]. We assumed L1 instruction cache memory size ranging from 64 bytes to 1024 bytes with block sizes 8 and direct-mapped cache architecture and L2 instruction cache with sizes varying between 128 bytes and 4 Kbytes. We performed both simulation and estimation computations in terms of the miss rate of instruction cache on L1 and L2. Moreover, we computed the actual time

cost for running the simulation and the estimation-based approaches as well as the average accuracy level of the proposed methodology.

Every cache level has its own local miss rate, which is the misses in this cache divided by the total number of memory accesses to this cache. Average miss rate is the misses in this cache divided by the total number of memory accesses generated by the processor. For example, in the case where there are two level of cache memories the average miss rate is given by the product of the two local miss rates of the two levels, i.e., (*MissRateL*1×*MissRateL*2). Average miss rate is what matters to overall performance, while local miss rate is a factor for evaluating the effectiveness of every cache level.

The accuracy of the proposed estimation technique is provided by the average estimation error. Table 1-3 presents the average percentage error of the proposed methodology compared to the simulation results taken using the Simplescalar tool, considering the abovementioned nine DSP applications. The last row of each table provides the average estimation error of miss rate of a two-level instruction cache memory hierarchy of each application. We choose to present the results of only two-level cache hierarch due to lack of space. Also, in order to reduce the results we present only the miss rate of L2 cache which its size is four times greater that L1, otherwise a lot tables and results must be presented. Depending on the application, the corresponding average values of estimation error ranges from 1% to 12%, while the total average estimation error of the proposed approach is less than 4% (i.e. 3.77%). The latter value implies that the proposed methodology exhibits high accuracy.

Table 1. Comparison between the estimation and the simulation results of the miss rate in L1 cache

Applications		L1 Cache Size (bytes)					Av. Error (%)
		64	128	256	512	1024	
FS	Simplescalar	100,0	100,0	99,8	99,2	76,8	1,10
	FICA	100,0	100,0	99,9	99,6	71,9	
HS	Simplescalar	99,9	97,3	92,6	66,4	2,8	2,55
	FICA	100,0	96,0	87,5	60,8	3,4	
PHODS	Simplescalar	100,0	100,0	99,6	96,7	31,7	2,08
	FICA	100,0	100,0	98,8	96,1	22,7	
3SLOG	Simplescalar	100,0	99,7	93,1	15,9	1,9	2,71
	FICA	100,0	99,4	96,9	7,4	0,9	
SS	Simplescalar	99,9	99,9	98,8	79,9	0,5	1,31
	FICA	100,0	99,2	98,4	75,0	0,0	
CAVITY	Simplescalar	100,0	100,0	94,3	61,4	16,9	6,43
	FICA	100,0	100,0	94,6	45,7	0,8	
CQ	Simplescalar	100,0	99,4	89,1	46,5	9,6	11,66
	FICA	100,0	98,7	84,2	3,5	0,0	
WAVELET	Simplescalar	98,7	89,9	50,3	1,3	1,1	2,30
	FICA	99,3	92,7	43,3	0,4	1,1	
FFT	Simplescalar	99,8	98,7	95,7	87,7	7,1	3,07
	FICA	100,0	100,0	96,1	75,3	6,0	

Table 2. Comparison between the estimation and the simulation results of the miss rate in L2 cache

Applications		L2 Cache Size (bytes)					Av.Error (%)
		256	512	1024	2048	4096	
FS	Simplescalar	99,8	99,2	77,0	0,1	0,0	1,16
	FICA	99,9	99,6	72,0	0,2	0,2	
HS	Simplescalar	92,7	68,2	3,0	2,3	53,3	10,28
	FICA	87,5	63,3	3,9	5,3	15,9	
PHODS	Simplescalar	99,6	96,8	31,8	0,8	0,7	2,81
	FICA	98,8	96,1	23,0	1,0	4,2	
3SLOG	Simplescalar	93,1	15,9	2,0	11,0	0,4	3,79
	FICA	96,9	7,4	1,0	7,2	2,2	
SS	Simplescalar	99,0	80,0	0,5	0,1	0,1	2,95
	FICA	98,4	75,6	0,0	0,0	9,4	
CAVITY	Simplescalar	94,3	61,4	17,9	0,5	0,5	6,82
	FICA	94,6	45,7	0,8	0,0	0,0	
CQ	Simplescalar	89,2	46,8	10,8	0,8	0,4	31,90
	FICA	84,2	3,5	0,0	0,0	100,0	
WAVELET	Simplescalar	50,9	1,5	2,1	1,5	1,4	5,22
	FICA	43,6	0,4	0,2	4,4	14,3	
FFT	Simplescalar	96,0	88,8	7,4	0,6	3,1	16,68
	FICA	96,1	75,3	9,8	7,9	63,1	

Table 3. Comparison between the estimation and the simulation results of the global miss rate in instruction cache memory hierarchy

Applications		L1 Cache Size : L2 = 4×L1 (bytes)					Av. Error (%)
		64	128	256	512	1024	
FS	Simplescalar	99,8	99,2	76,8	0,1	0,0	1,13
	FICA	99,9	99,6	71,9	0,2	0,1	
HS	Simplescalar	92,5	66,4	2,8	1,6	1,5	2,78
	FICA	87,5	60,8	3,4	3,2	0,5	
PHODS	Simplescalar	99,6	96,7	31,7	0,8	0,2	2,26
	FICA	98,8	96,1	22,7	1,0	1,0	
3SLOG	Simplescalar	93,1	15,9	1,9	1,7	0,0	2,90
	FICA	96,9	7,4	0,9	0,5	0,0	
SS	Simplescalar	98,8	79,9	0,5	0,1	0,0	1,16
	FICA	98,4	75,0	0,0	0,0	0,0	
CAVITY	Simplescalar	94,3	61,4	16,9	0,3	0,1	6,50
	FICA	94,6	45,7	0,8	0,0	0,0	
CQ	Simplescalar	89,1	46,5	9,6	0,3	0,0	11,61
	FICA	84,2	3,5	0,0	0,0	0,0	
WAVELE T	Simplescalar	50,3	1,3	1,1	0,0	0,0	1,82
	FICA	43,3	0,4	0,1	0,0	0,2	
FFT	Simplescalar	95,7	87,7	7,1	0,5	0,2	4,81
	FICA	96,1	75,3	9,5	6,0	3,8	

Based on the equations and using the number of accesses in each cache memory, the tool estimates the number of accesses of each cache level. We use CACTI 4.0 [14] power models for the caches design in technology 90nm and Micron [15] model for the off-chip system memory. The fourth stage of the methodology estimates the total

power consumption of the instruction memory hierarchy. The comparison results between the energy estimated by the FICA tool and using the conventional method (simulation and power models) are presented in Fig. (7)-(10). The simulation results from all these figures are produced by using a huge number of simulation with all the different cache hierarchies, while the estimated results only running once the tool. Fig. 7 presents the energy consumption results of the application FS for a system which contains only one level of cache. It can be easily deduced that the estimated results are similar to the simulation for all the cache sizes.

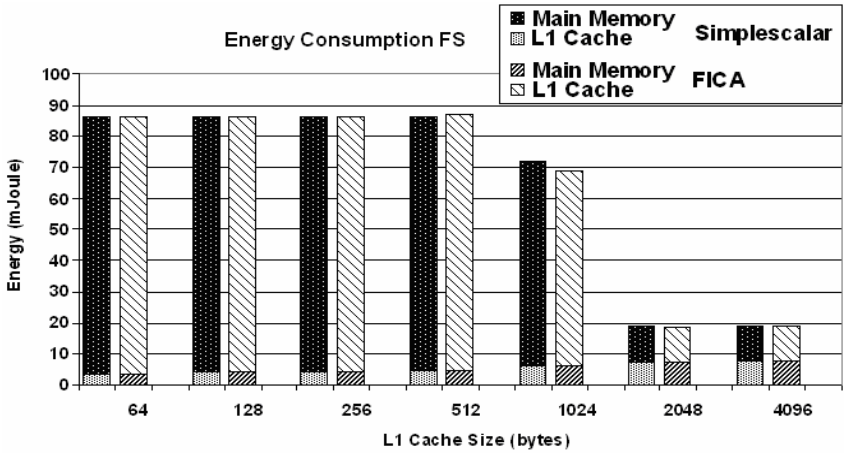


Fig. 7. Energy consumption comparison results between simulation and estimation for a system with L1 instruction cache of Full Search algorithm

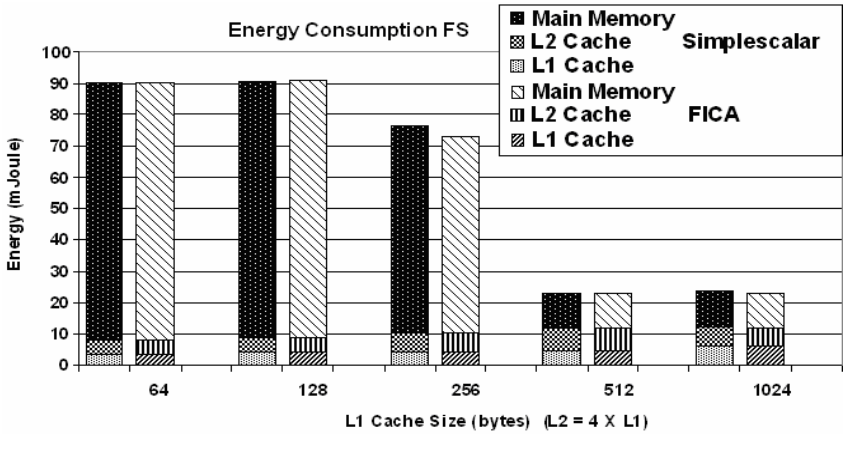


Fig. 8. Energy consumption comparison results between simulation and estimation for a system with L1 instruction cache of Full Search algorithm

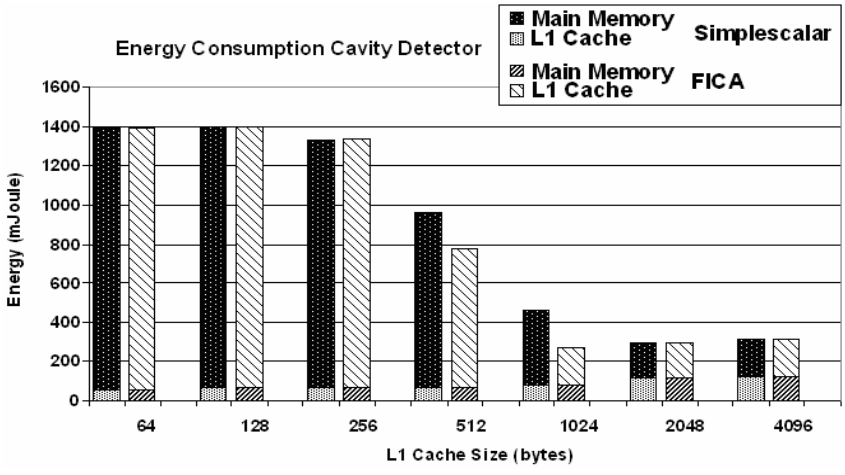


Fig. 9. Energy consumption comparison results between simulation and estimation for a system with L1 instruction cache of Cavity Detector

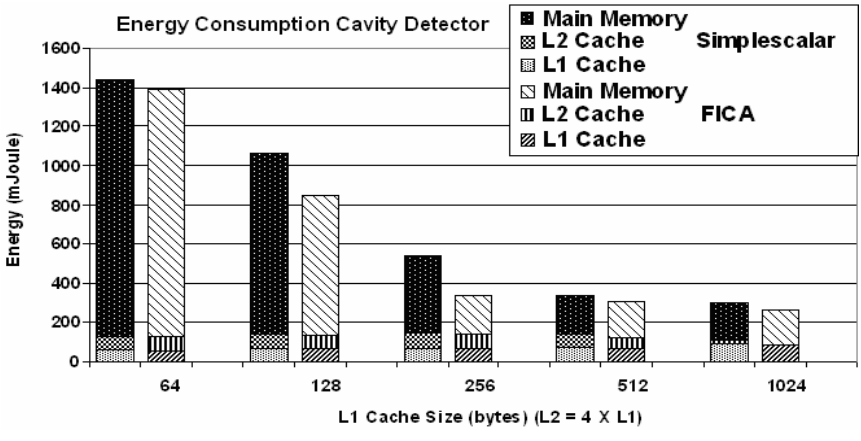


Fig. 10. Energy consumption comparison results between simulation and estimation for a system with L1 and L2 instruction caches of Cavity Detector

Apart from the accuracy of an estimation methodology (and tool), a second parameter very crucial for its efficiency is the required time cost to obtain the accurate estimates. Table 4 provides the required (average) time cost, in seconds, for performing the simulation and estimation procedure for all benchmarks. It is assumed an architecture with two levels of instruction cache and cache sizes for $L1$ from 64 bytes to 1024 bytes and $L2$ from 128 up to 4096 bytes there are 20 different combinations assuming that $L2>L1$. Using variable cache block sizes for $L1$ and $L2$ caches from 8 bytes to 32 bytes, there are totally 6 combinations assuming that $L1_{block\_size} \leq L2_{block\_size}$. In order to complete explorer the two-level instruction cache architecture 20×6=120 simulation procedures are needed for every application. The estimation and

Table 4. Speed up comparison results using our proposed methodology compared to the simulation time in a host machine Intel Pentium IV CPU 2GHz

Time (sec)	Applications							
	FS	**HS**	**3SLOG**	**PHODS**	**SS**	**Wavelet**	**Cavity**	**CQ**
Simulation	73,200	1,920	2,760	3,480	77,520	4,320	1,081,080	795,240
Estimation	4.8	27.3	7.2	9.45	7.2	105	15.3	27.45
Speed up	15,250	70	383	368	10,767	41	70,659	28,970

Table 5. Comparisons between existing and proposed methods

Method Technique	Processor type	Tool support	Accuracy	Time Code	Remarks
Processor Simulator [16]	MIPS	Simplescalar	100%	# exec instr. (GBytes)	-
Silva-Filho [3]	MIPS	-	100%	# exec instr. (GBytes)	Reduce the search space
Zhang [2]	MIPS	-	100%	# exec instr. (GBytes)	Reduce the search space
Gordon [4]	MIPS	-	100%	# exec instr. (GBytes)	Reduce the search space
Givargis [5]	MIPS	Platune	93%	# exec instr. (GBytes)	System Simulation
Borg [6]	MIPS	-	-	# exec instr. (GBytes)	Trace analysis
Lajolo [8]	Motorola 68332	POLIS	98%	# exec instr. (GBytes)	Task level estimation
Nohl [9]	LISA	JIT-CCS	100%	# exec instr. (GBytes)	Simulator modifications
FICA	MIPS	FICA	95%	Size of source code (Kbytes)	-

simulation computations were performed by a personal computer with Pentium IV, 2 GHz and 1 Gbyte RAM. It can be inferred that the proposed methodology offers a huge time speedup (orders of magnitude) compared with the simulation-based approach. Consequently, the new methodology/tool is suitable for performing estimations with a very high accuracy at the early design phases of an application.

The exploration time cost of the simulation-based approach is proportional to the size of the trace file of the application considered (order of GBs). In contrary, the corresponding time cost of the proposed methodology is (almost) proportional (linear) to the code size of the assembly code (order of KBs). From Table 4, it can be seen that the larger the number of loop iterations in C code (and of course in assembly code) is, the larger is the speedup factor of the new methodology. Regarding the proposed approach, we achieved time cost reduction between 40 to 70,000 times (i.e. up to four (4) orders of magnitude), depending on the application characteristics. Thus, accurate estimation within an affordable time cost allows a designer to perform design exploration of larger search space (i.e. exploration of additional design parameters).

In addition, the increasing complexity of modern applications, for instance image/video frame with higher resolution, will render the usage of simulation tools impractical. Therefore, the design of such complex systems, the high-level estimation tool will be the only viable and pragmatic solution.

Table 5 presents the comparison results between the methods referred in related work and the proposed one. The first column shows the list of methods, while the second one presents the used processor core on which each method has been developed. The third column provides the name of the software tool that supports the estimation/simulation of each method while the forth one shows the estimation accuracy. The most important comparison is presented in the fifth column; the time which needs each method to explore the instruction memory hierarchy is proportional to the number of executed instructions, while the proposed one is proportional to source code of the application. The time cost of the existing methods is increasing with the application computational complexity, while, the corresponding estimation time of the proposed method is dependent on the application's source code size.

7 Conclusions

A novel methodology for estimating the cache misses of multilevel instruction caches realized by an embedded programmable platform, was presented. The methodology was based on the straightforward relationship between the application high-level description code and its corresponding assembly code. Having as inputs both types of code, we extract specific features. Using the proposed methodology, we can perform estimation of application critical parameters during the early design phases, avoiding the time-consuming simulation-based approaches. The FICA tool is based on the proposed methodology and it is an accurate instruction cache miss rate estimator. The proposed methodology achieved estimations with smaller time cost than the simulation process, (i.e. orders of magnitude).

Acknowledgments

This work was partially supported by 03ED593 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds.

Also, it was partially sponsored by MOSART project (Mapping Optimization for Scalable multi-core ARchiTecture) funded by the EU (IST-215244), http://www.mosart-project.org.

References

[1] Segars, S.: Low power design techniques for micro-processors. In: International Solid State Circuit Conference (February 2001)

[2] Zhang, D., Vahid, F.: Cache configuration exploration on prototyping platforms. In: 14th IEEE International Workshop on Rapid System Prototyping, June 2003, pp. 164–170 (2003)

[3] Silva-Filho, A.G., et al.: Heuristic for Two-Level Cache Hierarchy Exploration Considering Energy Consumption and Performance. In: Vounckx, J., Azémard, N., Maurine, P. (eds.) PATMOS 2006. LNCS, vol. 4148, pp. 75–83. Springer, Heidelberg (2006)
[4] Gordon-Ross, A., Vahid, F., Dutt, N.: Automatic Tuning of Two-Level Caches to Embedded Applications. In: Design, Automation and Test in Europe, DATE, February 2004, pp. 208–213 (2004)
[5] Givargis, T., Vahid, F.: Platune: A Tuning framework for system-on-a-chip platforms. IEEE Trans. Computer-Aided Design 21, 1–11 (2002)
[6] Borg, A., Kessler, R., Wall, D.: Generation and analysis of very long address traces. In: International Symposium on Computer Architecture, May 1990, pp. 270–279 (1990)
[7] Mueller, F., Whalley, D.: Fast Instruction Cache Analysis via Static Cache Simulation. In: Proc. of 28th Annual Simulation Symposium, pp. 105–114 (1995)
[8] Lajolo, M., Lavagno, L., Sangiovanni-Vincentelli, A.: Fast instruction cache simulation strategies in a hardware/software co-design environment. In: Proc. of the Asian and South Pacific Design Automation Conference, ASP-DAC 1999 (January 1999)
[9] Nohl, A., Braun, G., Schliebusch, O., Leupers, R., Meyr, H.: A Universal Technique for Fast and Flexible Instruction-Set Architecture Simulation. In: Proc. of the 39th conference on Design automation, DAC 2002, New Orleans, Louisiana, USA, pp. 22–27 (2002)
[10] Hoffmann, A., Kogel, T., Nohl, A., Braun, G., Schliebusch, O., Wieferink, A., Meyr, H.: A Novel Methodology for the Design of Application Specific Instruction Set Processors (ASIP) Using a Machine Description Language. IEEE Transactions on Computer-Aided Design 20(11), 1338–1354 (2001)
[11] Balaji, R.: Fast Design Space Exploration of Instruction Caches. Msc Thesis, National University of Singapore (2003)
[12] Kroupis, N., Mamagkakis, S., Soudris, D.: An Estimation Methodology for Designing Instruction Cache Memory of Embedded Systems. In: ESTIMedia 2006, Fourth IEEE Workshop on Embedded Systems for Real Time Multimedia, Seoul, Korea, October 26-27 (2006)
[13] Liveris, N., Zervas, N., Soudris, D., Goutis, C.: A Code Transformation-Based Methodology for Improving I-Cache Performance of DSP Applications. In: Proc. of DATE, Paris, pp. 977–984 (2002)
[14] Tarjan, D., Shyamkumar, T., Jouppi, N.: CACTI 4.0 HPL Tech. Report HPL-2006-86 (June 2006)
[15] Calculating Memory System Power for DDR2, Technical Note, Micron Technology Inc. (2007)
[16] Austin, T., Larson, E., Ernst, D.: SimpleScalar: An Infrastructure for Computer System Modeling. Computer 35(2), 59–67 (2002)
[17] Kuhn, P.: Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation. Kluwer Academic Publisher, Boston (1999)
[18] Nam, K., et al.: A fast hierarchical motion vector estimation algorithm using mean pyramid. IEEE Transactions on Circuits and Systems for Video Technology 5(4), 344–351 (1995)
[19] Cheung, C.-K., Po, L.-M.: Normalized Partial Distortion Search Algorithm for Block Motion Estimation. Proc. IEEE Transaction on Circuits and Systems for Video Technology 10(3), 417–422 (2000)
[20] Lafruit, G., Nachtergaele, L., Vahnhoof, B., Catthoor, F.: The Local Wavelet Transform: A Memory-Efficient, High-Speed Architecture Optimized to a Region-Oriented Zero-Tree Coder. Integrated Computer-Aided Engineering 7(2), 89–103 (2000)

[21] Danckaert, K., Catthoor, F., De Man, H.: Platform independent data transfer and storage exploration illustrated on a parallel cavity detection algorithm. In: ACM Conference on Parallel and Distributed Processing Techniques and Applications III, pp. 1669–1675 (1999)

[22] Dekker, A.: Kohonen neural networks for optimal colour quantization. Network: Computation in Neural Systems 5, 351–367 (1994)

Timing Error Detection and Correction by Time Dilation

Andreas Floros, Yiorgos Tsiatouhas, and Xrysovalantis Kavousianos

Department of Computer Science
University of Ioannina

Abstract. Timing failures in high complexity - high frequency integrated circuits, which are mainly caused by test escapes and environmental as well as operating conditions, are a real concern in nanometer technologies. The Time Dilation design technique supports both on-line (concurrent) error detection/correction and off-line scan testing. It is based on a new scan Flip-Flop and provides multiple error detection and correction at the minimum penalty of one clock cycle delay at the normal circuit operation for each error correction. No extra memory elements are required, like in earlier design approaches in the open literature, reducing drastically the silicon area overhead, while the performance degradation is negligible since no extra circuitry is inserted in the critical paths of a design.

Keywords: On-Line Testing, Concurrent Testing, Timing Errors, Error Detection, Error Correction.

1 CMOS Nanotechnologies and Timing Errors

As modern CMOS technologies scale down in the nanometer era and the complexity of integrated circuits and systems increases, an ongoing difficulty to achieve adequate reliability levels and keep the cost of testing within acceptable bounds is reported [1-2]. The device size scaling, the operating frequency increase and the power supply reduction affect circuits' noise margins and reliability. The probability of transient faults generation increases and many times it is hard to achieve error rate specification levels.

Various mechanisms like crosstalk, power supply disturbance or ground bounce have been accused for timing error generation. The increased path delay deviations, due to process variations, and the manufacturing defects that affect circuit speed may also result in timing errors that are not easily detectable (in terms of test cost) in high frequency and/or high device count ICs. The already complex testing process can not sufficiently exercise the huge number of paths in modern circuit designs, and thus it can not effectively screen out all timing related defective ICs. Consequently, a considerable part of defective ICs may escape the fabrication tests. Additionally, and for the same reasons, timing verification turns to be a hard task escalating the probability of timing failures in a design. Furthermore, modern systems running at multiple frequency and voltage levels may suffer from an increased timing error rate due to numerous environmental and process related as well as data dependent variabilities that can affect circuit performance. Besides, *dynamic voltage scaling* (*DVS*) techniques for

C. Piguet, R. Reis, and D. Soudris (Eds.): VLSI-SoC 2008, IFIP AICT 313, pp. 271–285, 2010.

low power operation that reduces power supply voltage with marginal performance degradation have been proposed in the literature [3]. These exploit timing error detection and correction mechanisms to overcome increased timing error rates. In addition, transistor aging problems significantly impact the performance of nanometer circuits resulting in the appearance of timing errors during their normal lifetime [4-5]. Such an example is the *Negative Bias Temperature Instability* (*NBTI*) induced aging of PMOS transistors which degrades their threshold voltage over time increasing path delays. From the above, it is evident that concurrent on-line testing techniques for timing error detection and correction are becoming mandatory in order to achieve acceptable levels of error robustness and meet reliability requirements.

2 Timing Error Detection and Correction

Timing failures in a combinational logic circuit result in delayed responses at its outputs. As it is shown in Figure 1, in case of a delayed response arrival, after the triggering edge of the system clock *CLK*, the memory element will capture an erroneous value and a timing error is generated.

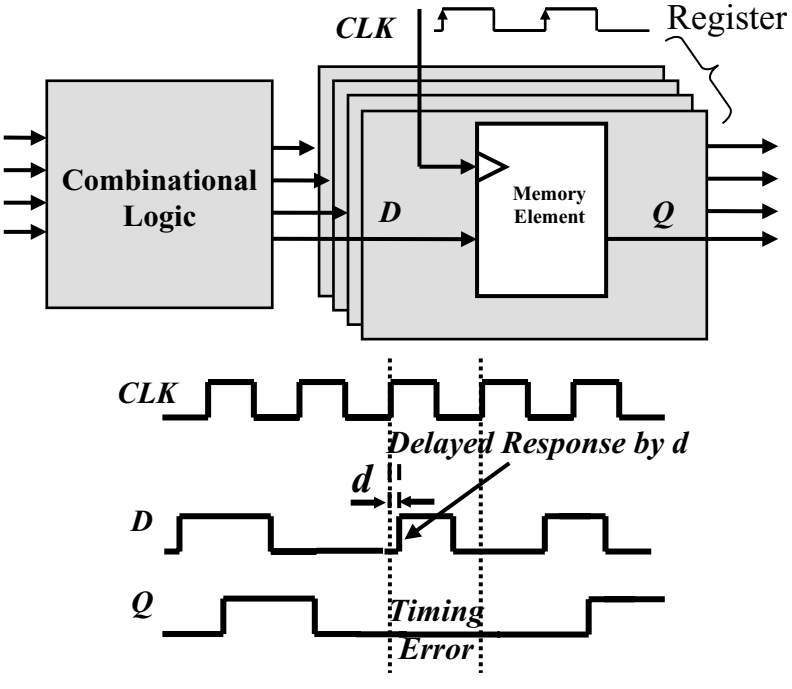


Fig. 1. Timing error generation

Various timing error detection techniques have been proposed in the open literature [6-11] that are based on the delayed response of timing faults to provide error tolerance using time redundancy techniques. A well known error detection scheme is based on the use of a comparator that is realized by a simple XOR gate [8-9, 11]. The

monitoring circuitry consists of an additional memory element plus a XOR gate for every memory element (main latch or Flip-Flop) in the design (see Figure 2a). The secondary memory element is clocked by a delayed version of the system clock that feeds the main memory element. This delay is equal to the maximum signal delay (d_{max}) that must be tolerated in order to achieve an acceptable level of timing error rate, plus the setup time of the used memory elements (t_{su}). Thus, the secondary memory element captures a delayed version of the data stored in the main memory element. In the presence of a timing error the stored data in the two memory elements differ, while the secondary memory element holds the correct delayed response of the combinational logic. The XOR gate "compares" the contents of the two memory elements and in case of discrepancy it raises its output to high indicating the error detection. The local error indication signals (*Error_L*) are collected by an OR gate (realized as an OR tree) to generate a global error indication signal (*Error_R*). This signal can be exploited to achieve error tolerance by performing a retry procedure after each error detection. During the retry operation the period of the system clock must be increased to provide the necessary time for correct response evaluation.

Alternatively, a cost efficient approach is to use only the XOR gate for error detection as it is shown in Figure 2b [8]. The XOR gate compares the data input and output signals of the main memory element for a predefined time period after the triggering

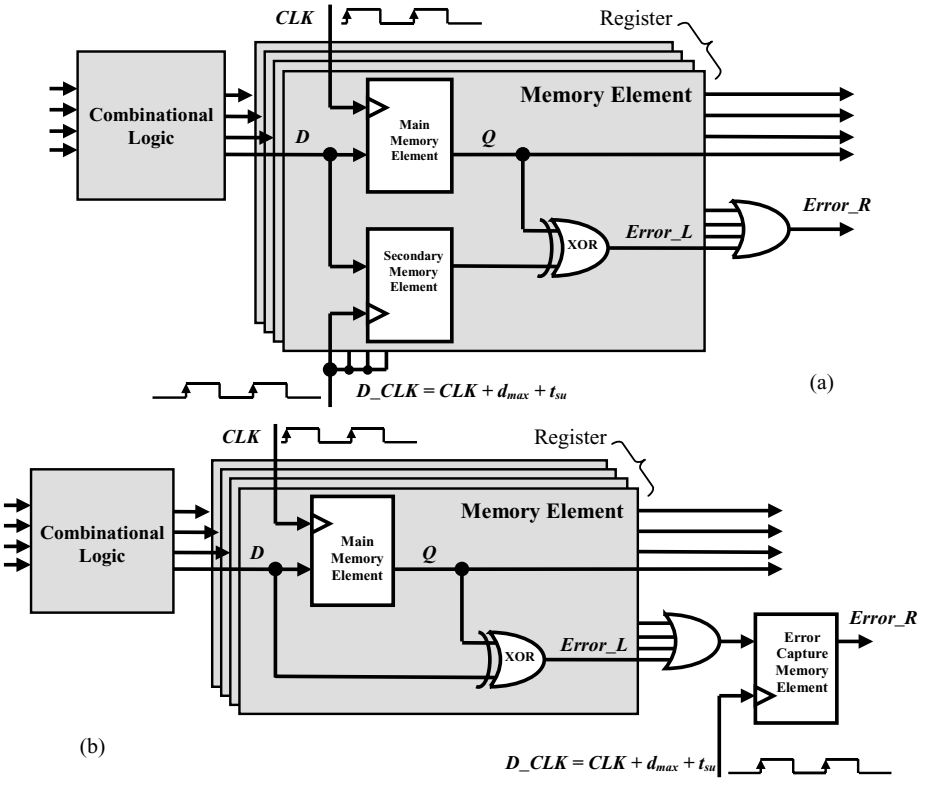


Fig. 2. Timing error detection: a) memory duplication and b) cost efficient design

edge of the system clock. This time period is also equal to the maximum signal delay that must be tolerated plus the setup time of the memory elements. In case of discrepancy between the two signals, the error indication signal raises at the XOR output. Possible very fast paths with propagation times close to or less than $d_{max}+t_{su}$ must be excluded from the timing error monitoring process, since they will induce false alarms. In general, fast paths with propagation times less than the system clock period minus ($d_{max}+t_{su}$), with a proper tolerance, can be also excluded.

2.1 The Razor Pipeline Architecture

A pipeline architecture (named *Razor*) with timing error detection and correction capabilities, targeting the substantial energy reduction of integrated circuits exploiting dynamic voltage scaling, has been presented in [3]. According to this architecture, the stage registers are constructed using the Razor Flip-Flops. Figure 3 illustrates a Razor Flip-Flop, which consists of the main system Flip-Flop plus an assistant shadow latch, a multiplexer (MUX) and a XOR gate. As discussed earlier, the shadow latch captures, with a proper delay with respect to the main Flip-Flop, the responses of the combinational logic. The XOR gate acts as a comparator and compares the outputs of the main Flip-Flop and the shadow latch.

In the error free case both the main Flip-Flop and the shadow latch will capture the same data. The comparison by the XOR gate provides a low local error indication signal (*Error_L*) and the pipeline continues to operate in the normal mode. In case of a delay in the evaluation of the logic stage S_j that exceeds circuit specifications, erroneous data are latched in the main Flip-Flop while the shadow latch will capture the correct (delayed) data, since it operates with a delayed clock. Consequently, the XOR output (*Error_L*) will rise to high indicating the detection of an error. The generation of a

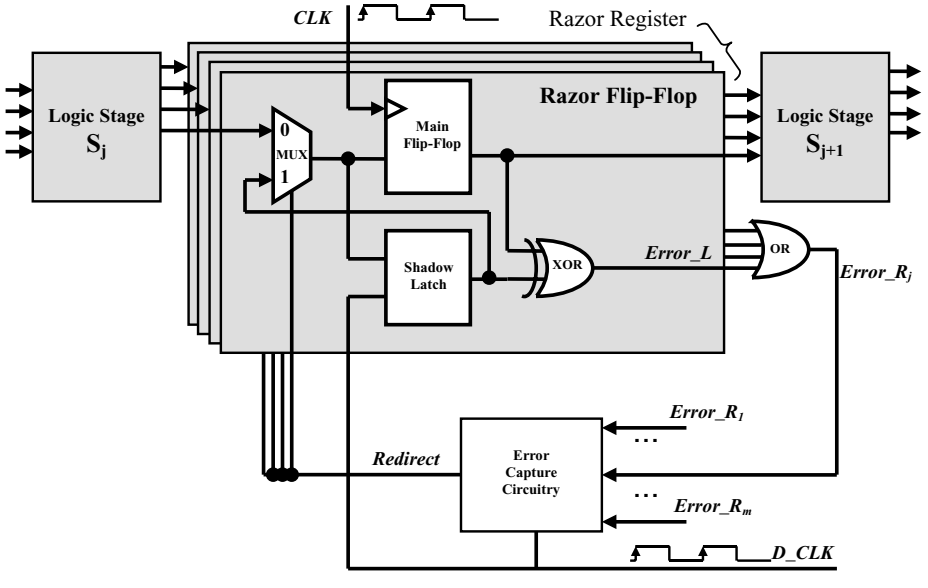


Fig. 3. The Razor timing error detection and correction design approach

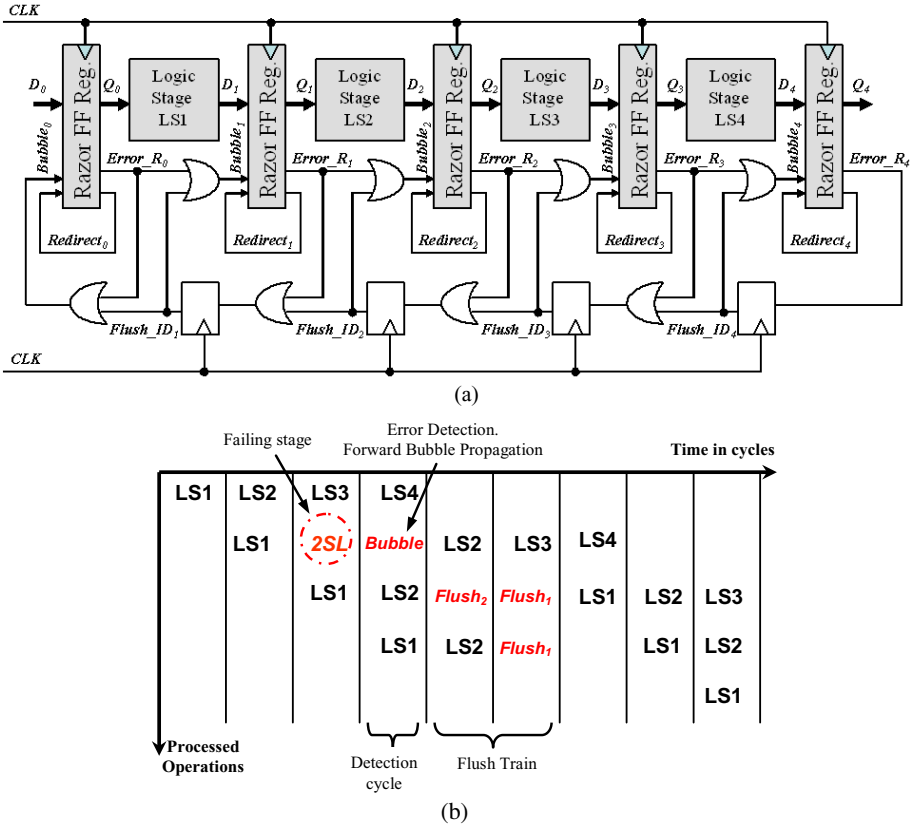


Fig. 4. Razor counterflow recovery: a) pipeline architecture and b) pipeline operation

timing error in a clock cycle (i+1) at a pipeline stage S_j implies that the data of stage S_{j+1} in the following cycle (i+2) are incorrect and must be flushed. This action is easy to be accomplished since the shadow latch contains the correct data without the need to re-compute them through the failing stage. The local error indication signal *Error_L* activates the register error indication signal *Error_R*$_j$ which is captured by the Error Capture Circuitry. This in turn sets the *Redirect* signal to high enabling the shadow latch to feed the main Flip-Flop with the correct data. These are injected into the pipeline in the next cycle (i+3) allowing stage S_{j+1} to compute the correct responses.

In the Razor architecture two approaches for pipeline error recovery have been adopted [3]. The first one is the clock gating technique where in case of an error detection the entire pipeline stalls by gating the next global clock edge for one cycle. This period is exploited by each stage to re-compute its result using the correct data of the shadow latches. The second approach used in Razor is the *counterflow* pipelining which is based on the namesake processor architecture [12]. This technique is illustrated in Figure 4 and is characterized by negligible timing constraints in the pipeline operation at the expense of few cycles, depending on the pipeline depth, for error recovery. When a register error indication signal is generated, there are two actions that follow. First, a *Bubble* signal is generated to nullify the computation in the following

stage. This signal indicates to all subsequent stages that the pipeline slot is empty. Second, a flush train is activated by asserting the ID of the stage generating the error indication signal. In the next cycle the correct data of the corresponding register shadow latches are injected into the pipeline allowing the errant instruction to continue its execution. In parallel, the flush train propagates the ID of the failing stage in the opposite direction to this of the instructions flow. At each stage that the flush train visits, the computation is nullified. When the first stage of the pipeline is reached the pipeline restarts its operation with the instruction that follows the failing one.

The Razor approach suffers from high silicon area cost since for every main Flip-Flop an extra latch, a multiplexer and a XOR gate are required. In addition an extra clock signal is used.

2.2 Scan Based Error Detection and Correction

Soft error detection and correction techniques for special purpose scan Flip-Flops in microprocessor circuits have been proposed in [1]. These techniques are suitable in designs where each system Flip-Flop consists of a pair of Flip-Flops (i.e. the main Flip-Flop and the scan Flip-Flop as it is shown in Figure 5) and can be also exploited to cover timing errors. The scan Flip-Flop is modified to operate as a shadow of the main Flip-Flop, latching the same data with a proper delay as discussed earlier (signals *CAPTURE* and *SCB* are delayed with respect to the system clock *CLK*). A XOR gate is used to compare the outputs of the Flip-Flop pair and detect possible errors in the system Flip-Flop. Three additional logic gates (a second XOR, an OR and an AND) are used in order to enable the trapping of any error indication signal (*Error_L*) in the pertinent scan Flip-Flop. This error indication is shifted out using the existing scan path in order to activate system recovery through re-execution. The main drawbacks of this technique are: a) the high silicon area cost due to Flip-Flop duplication and the insertion of extra logic gates, b) the performance degradation due to the complexity of the main Flip-Flop, c) the large number of control signals and d) although the global routing of error signals is reduced reusing existing scan facilities, there is a high penalty in error detection latency.

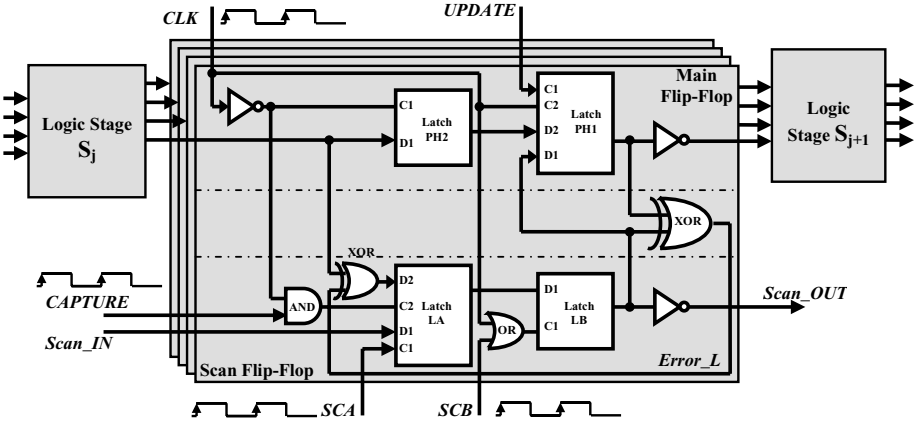


Fig. 5. Error trapping scan cell

3 The Time Dilation Scan Architecture

Recently, a low cost pipeline architecture has been proposed in [13] that is characterized by the ability to detect and correct timing errors. This architecture utilizes only a multiplexer and a XOR gate per system Flip-Flop reducing drastically the silicon area cost, while only a single clock cycle is required for error correction. This technique has been extended in [14] to scan designs forming the *Time Dilation* scan architecture.

Figure 6 illustrates the classical scan register configuration which is based on standard scan Flip-Flops. All scan Flip-Flops are connected together as one or more scan registers. The *Scan_IN* input of a scan Flip-Flop is driven by the Q output of the preceding scan Flip-flop in the shift register. When the *Scan_EN* signal is "high" the circuit is in the scan mode of operation, for testing purposes, and the scan Flip-Flops are driven by the *Scan_IN* inputs, else they are driven by the *D* inputs capturing the response data of the combinational logic.

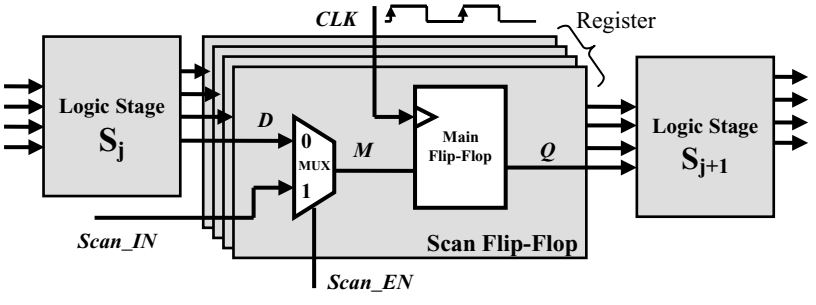


Fig. 6. The standard scan Flip-Flop design

3.1 The Time Dilation Scan Flip-Flop

The scan Flip-Flop used in the Time Dilation (TIMED) architecture is presented in Figure 7. The TIMED Flip-Flop provides the capability of error detection and correction by appending only a multiplexer (MUX-B) and a XOR gate in the structure of the standard (main) scan Flip-Flop. This hardware overhead is much lower than this of the next most attractive choice, the Razor topology, where except of the above two cells an additional shadow latch is required. Although we will present for convenience the application of the Time Dilation technique in pipeline architectures, it can be also applied in any sequential circuit design.

When the scan enable signal (*Scan_EN*) is "high" the TIMED Flip-Flop operates like a scan Flip-Flop to support the pertinent off-line testing activity. In the normal mode of operation (*Scan_EN*="low") the TIMED Flip-Flop behaves like an ordinary Flip-Flop enhanced with the ability to detect and correct timing errors. The XOR gate is used to directly compare the data at the *M* input and the *Q* output of the Main Flip-Flop for error detection, while the two multiplexers and the feedback path from the *M* line to the input of the additional MUX-B forms the required memory element (MUX-latch) that holds valid data for error correction.

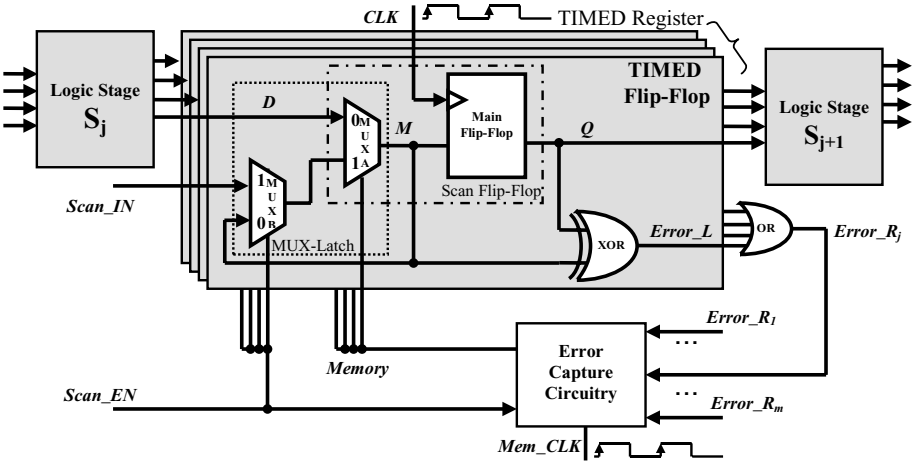


Fig. 7. The TIMED Flip-Flop and support circuitry

Briefly, the Time Dilation technique operates as follows. Suppose that a timing error is detected at the inputs of the combinational logic stage S_{j+1}, due to a delayed response of the previous stage S_j. Thus, the response of S_{j+1} will be erroneous and must be corrected. Then, the evaluation time of the circuit is extended by one clock cycle and S_{j+1} is fed with the delayed, but valid, response of S_j that has been captured in the MUX-latch, for error correction.

The MUX-latch is clocked by the *Memory* signal. In the error free case the *Memory* signal is exclusively controlled by the *Mem_CLK* signal, a delayed version of the clock signal *CLK* with a proper duty cycle. When the *Mem_CLK* signal is "high" the *Memory* signal is activated (turns also to "high") and the MUX-latch enters the memory state; else the MUX-latch is transparent. The time interval that the *Memory* signal is active must coincide with the time interval where new values arrive at the *D* inputs of the TIMED Flip-Flops, in all stage registers, due to an earlier evaluation of the pertinent logic stages according to the circuit specifications. Any signal transition at the *D* inputs of the TIMED Flip-Flops, earlier than the activation time of the *Memory* signal, is considered as violation of the timing specifications and must be detected. Obviously, the deactivation of the *Memory* signal (falling edge), and accordingly of the *Mem_CLK* signal, must occur before the triggering edge of the *CLK* signal and at a time distance at least equal to the delay time of the MUX-A plus the setup time of the Main Flip-Flop.

The XOR gate in the TIMED Flip-Flop detects timing errors and indicates them by setting signal *Error_L* to "high". An OR gate is used to collect the *Error_L* signals and to generate the register error indication signal $Error\_R_j$. Any register error indication signal is captured by a single Flip-Flop (Error Flip-Flop) triggered by the *Mem_CLK* signal which has been properly delayed. The final error indication signal, *Error*, is used to activate the error correction mechanism.

3.2 Timing Error Detection and Correction Using Time Dilation

In Figure 8 the operation of the TIMED Flip-Flop is presented. We study the normal mode of operation (not the scan mode) therefore the *Scan_EN* signal is considered always "low". In the i^{th} clock cycle the response of the logic stage S_j is within the timing specifications of the circuit. This means that it occurs during the high state of the *Memory* signal. Consequently, after the triggering edge of the clock *CLK* both the data input *M* and the output *Q* of the Main Flip-Flop will carry the same value until the falling edge of the *Memory* signal. Thus, the *Error_L* signal as well as the subsequent $Error\_R_j$ signal will be both zero at the time that the Error Flip-Flop is triggered. In that case, the pipeline's operation remains unaltered (*Error*="low"). In the next cycle (i+1) a timing fault occurs which induce a delayed response of stage S_j. Thus, a timing error is generated at the next triggering edge of the clock *CLK*. The data captured in the TIMED register between the S_j and S_{j+1} stages are erroneous and consequently the response of S_{j+1} stage at the (i+2) cycle will be also erroneous. Moreover, due to the fault, a transition occurs at the *D* input of a TIMED Flip-Flop, inside (i+2) cycle, after the triggering edge and before the activation of the *Memory* signal. Since the MUX-latch is transparent during this time interval, the transition passes to the *M* line. Now the value at the output of the MUX-latch (*M* line) differs from this at the output of the Main Flip-Flop (*Q* line). The first one is the correct response of S_j and the second the erroneous value captured on *Q*. So, the comparison by the XOR gate of the MUX-latch valid data with the erroneous data stored in the Main Flip-Flop sets the local error signal *Error_L* to "high" and generates a register error indication signal $Error\_R_j$ at the output of the register's OR gate. Next, the triggering edge of the *Mem_CLK* signal activates the *Memory* signal, setting the MUX-latches in the memory state, and after a proper delay captures the register error indication in the Error Flip-Flop, raising the *Error* signal to "high". This "high" value will extend the active duration of the *Memory* signal keeping all MUX-latches in the memory state. At this point the error has been detected. In addition, all the MUX-latches hold the correct (valid) responses of the S_j logic stage for the (i+1) clock cycle. The new responses of the S_j and S_{j+1} logic stages at the (i+2) cycle are blocked at the *D* inputs of the pertinent TIMED Flip-Flops and will be discarded since the response of S_{j+1} is erroneous. Entering the next cycle (i+3), the triggering edge of the clock *CLK* forces the valid data to move from the MUX-latches to the Main Flip-Flops in order to be available to the next pipeline stage S_{j+1}. Consequently, the error is corrected since the logic stage has correct data to perform, inside the (i+3) clock cycle, the failed evaluation of the (i+2) cycle. This is an one cycle penalty for correction. Next, the error indication signals *Error_L*, $Error\_R_j$ and *Error* turn successively to "low" and the *Memory* signal returns to its routine operation.

According to the above discussion, if a timing error occurs in a pipeline stage S_j during a particular clock cycle, then the data in the subsequent stage S_{j+1} are incorrect, during the next clock cycle, and must be flushed from the pipeline. However, the MUX-latches contain the correct data and thus the re-execution of the failed evaluation in the S_j stage is avoided. On the other hand, the S_{j+1} stage re-executes its evaluation using this time the correct input data with only one-cycle penalty in the pipeline operation.

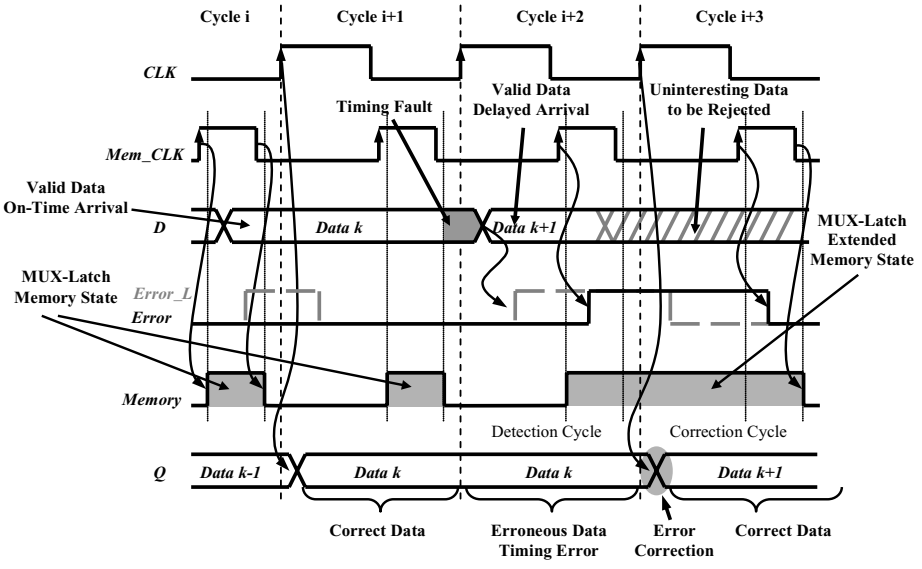


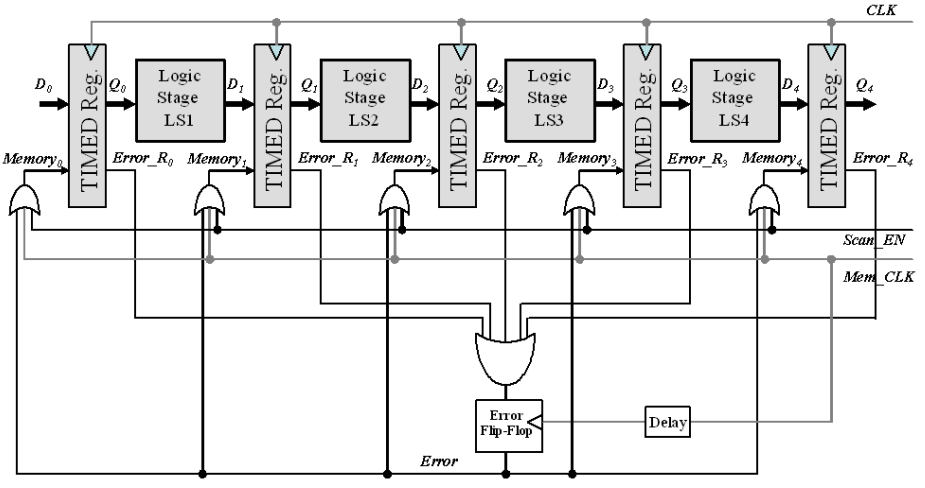
Fig. 8. TIMED Flip-Flop operation with a timing error in cycle i+2 and recovery in cycle i+3

A main characteristic and an advantage of the proposed topology is that no circuitry is inserted in the critical path from the *D* input to the *Q* output of the Flip-Flop or in the distribution path of the clock signal *CLK*. The additional MUX-B is inserted in the scan path which is not critical. A minor performance penalty is introduced by the small parasitic capacitances of the MUX-B and the XOR gate inputs that are driven by the *M* and *Q* signal lines. In addition, note that the silicon overhead of the OR gate at the output of a TIMED register is small (especially when a Domino design style is used), while the rest circuitry (the Error Capture Circuitry) is shared on the whole pipeline and thus its cost is insignificant. The area overhead related to the OR gates and the Error Capture Circuitry is also present in the Razor topology.

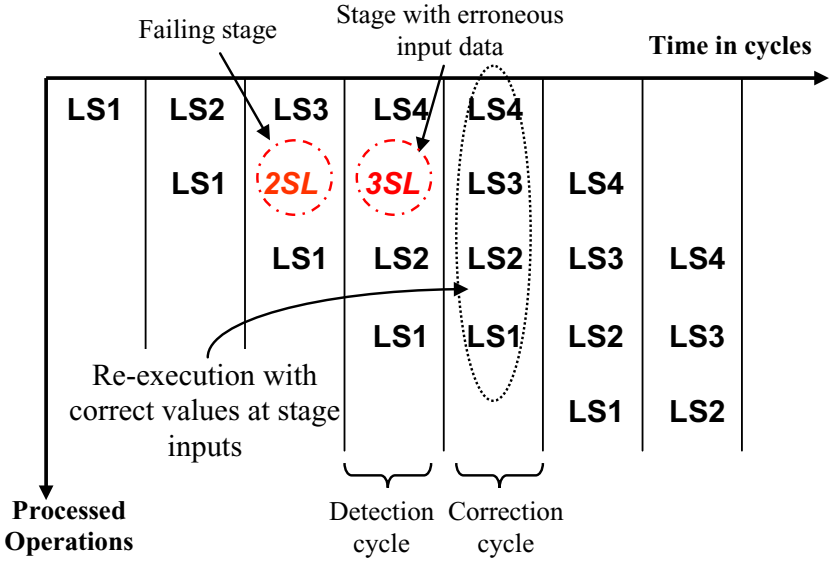
3.3 Pipeline Recovery

Every error detection is succeeded by a pipeline state recovery action. Figure 9 illustrates the pipeline recovery mechanism. The event of a timing error in a logic stage (lets say the LS2 stage) generates an error indication signal $Error\_R_2$ at the following TIMED register. This means that the response of the next stage LS3 at the subsequent clock cycle is incorrect (as indicated in Figure 9b) since its input data are not valid.

The error indication signal is latched by the Error Flip-Flop and the *Memory* signal remains "high" keeping all the MUX-latches of the TIMED Flip-Flops in all stage registers in the memory state. Thus, in the next clock cycle every stage is allowed to re-compute its response using the correct data stored in the MUX-latches. Actually, this seems to be like a "time dilation" in the duration of the failing clock cycle. Note here that there is no need for the failing stage LS2 to re-compute its response in the cycle where the failure occurred since the correct responses are already available in



(a)



(b)

Fig. 9. Time Dilation recovery: a) pipeline organization and b) pipeline operation

the following MUX-latches. The Time Dilation pipeline architecture can tolerate any number of errors in a clock cycle since all stages re-compute their responses with correct data at their inputs. In case that one or more stages fail in each clock cycle, the pipeline will continue to run at half of the normal speed.

Referring to the analysis of the Time Dilation architecture, there is no need to apply main clock gating to accomplish pipeline recovery, neither the Counterflow pipeline design technique [12] as in the Razor case. This is due to the fact that the pipeline performance is not affected by the recovery mechanism since there is not any prohibitive delay in the feedback path from the error indication signal generation to the activation of the memory state of the MUX-latches. The MUX-latches in the TIMED Flip-Flops are set to the memory state, by the *Memory* signal, independently of the generation or not of an error signal. Thus, at the time an error indication signal (*Error*="high") is captured in the Error Flip-Flop, the *Memory* signal is already active ("high") and the MUX-latches are in the memory state. This error indication signal simply extends the active state of the *Memory* signal for one clock period. Consequently, the following triggering edge of the clock *CLK* injects the correct data from the MUX-latches into the pipeline, allowing the "swerved" operation to continue. Later operations inside the pipeline are not flushed and continue to run after recovery. Hence, only a single cycle is required in the Time Dilation architecture for pipeline recovery as it is shown in Figure 9b.

Note that the delay of the *Mem_CLK* signal with respect to the system clock *CLK*, and consequently its duty cycle, must be properly selected to prevent data corruption in the MUX-latches due to possible existence of short paths in the combinational logic. To avoid this, a minimum path delay constraint is considered in the design. In order to meet this constraint in the presence of short paths, gates constructed of minimum size and high-threshold voltage transistors can be used and buffers may be added during logic synthesis (like in Razor [3]) to slow them down. The minimum path delay constraint is equal to the delay of the *Memory* signal with respect to the system clock *CLK*, plus the hold time of the MUX-latch. However, a trade-off arises. A large value for the minimum path delay constraint may increase the number of the required buffers in the design and consequently the silicon area penalty. On the other side, a small value for this delay constraint reduces the error tolerance due to the reduction of the maximum detectable signal delay.

4 Time Dilation Application

The Time Dilation architecture was applied in a 32-bit four stages pipeline datapath, that has been designed in a 90nm CMOS technology (V_{DD}=1V), with 870MHz clock frequency (1150ps period). The TIMED Flip-Flop has been designed in transistor level as a library standard-cell. Since the fastest response of the combinational logic is higher than 400ps, the delay of the *Mem_CLK* signal with respect to *CLK* is set to 300ps and its "on" time duration is equal to 550ps. The extra delay inserted to the *Mem_CLK* signal to drive the Error Flip-Flop is 250ps. Signal delays up to 350ps (30% of the clock cycle) from the triggering edge of the system clock *CLK* can be detected and corrected. The performance penalty introduced in the original scan design with the use of the TIMED Flip-Flop is less than $4^{o}/_{oo}$ and thus it is negligible.

In Figure 10 electrical simulations using SPECTRE are presented. A timing fault is injected at the first stage of the pipeline during the 4^{th} clock cycle. Consequently, the data captured at the *Q1_5* output of the corresponding TIMED Flip-Flop are erroneous and the same stands for the response of second stage at the 5^{th} cycle. Due to the

fault, a delayed response appears at the *D1_5* input of the TIMED Flip-Flop in the 5th cycle, after the triggering edge of *CLK*. This response is propagated to the *M1_5* (not shown) input of the main Flip-Flop since the MUX-latch is transparent (*Memory1*="low") during this time interval. Next, the *Memory1* signal is activated and the MUX-latch captures the correct data on *M1_5*. The XOR gate detects the difference between *M1_5* and *Q1_5* (due to the erroneous data on *Q1_5*) and sets signal *Error_R1* to "high". Consequently, the triggering edge of *Mem_CLK* also forces the global *Error* signal to "high". This extends the memory state of the MUX-latch holding the *Memory1* signal active ("high") within the 6th clock cycle. In this cycle the pipeline re-executes the stage responses with the correct data that are available in the MUX-latches. Thus, the error is corrected and the pipeline proceeds with its normal operation.

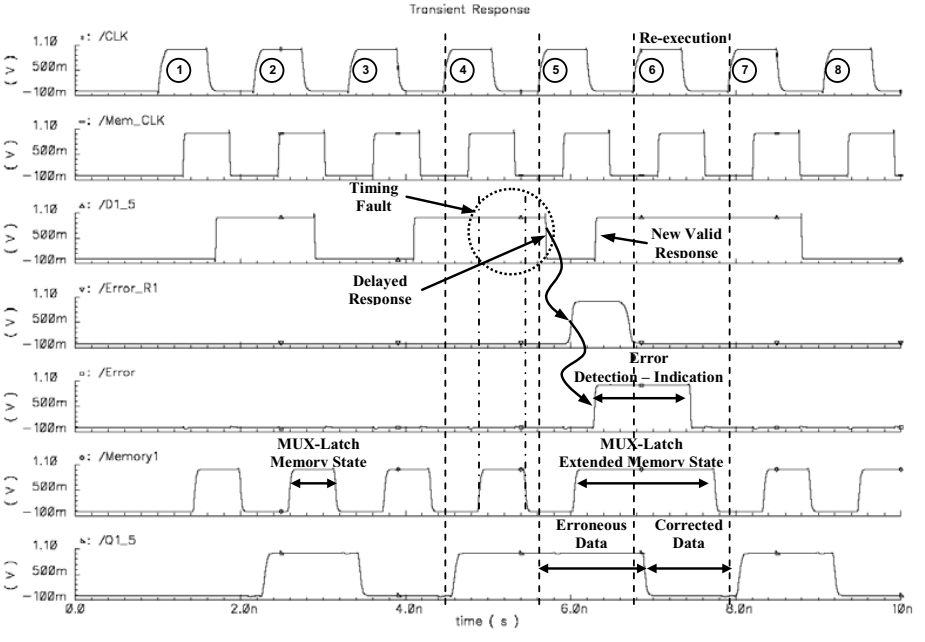


Fig. 10. Simulated waveforms from Time Dilation application in a 32-bit pipeline

5 Conclusions

Timing error detection and correction techniques are of great importance in today nanometer CMOS technologies. To cope with them, a new scan Flip-Flop design that provides timing error detection/correction capabilities and a pipeline architecture (under the name *Time Dilation*) which exploits this scan Flip-Flop for pipeline recovery after a timing error occurrence, have been proposed. This design approach is characterized by low silicon area requirements (about 24% reduction in Flip-Flop area with respect to *Razor* the most attractive alternative topology), negligible performance penalty and the minimum cost of only one clock cycle for pipeline recovery after each

error detection. Although the proposed technique is illustrated for pipeline architectures, it can be applied in general to any sequential circuit.

The Time Dilation technique can be utilized to provide aggressive power reductions in Dynamic Voltage Scaling (DVS) based circuits by tolerating timing errors in critical paths under worst case process and environmental variabilities or the presence of noise sources like di/dt noise in supply voltage and signal crosstalk. Moreover, Time Dilation offers the ability of using more relaxed design constraints or voltage and noise margins to ensure correct operation. Those constraints/margins are inserted to protect a design against uncertainty in circuit model parameters and worst case combination of variabilities. However, such a combination might be very rare or even impossible making this approach overly conservative from the performance point of view and demanding in design effort [3]. With technology scaling, process variations are increased and noise effects are getting more and more serious worsening the required constraints and margins in a design. Time Dilation accounts for both local and global process and temperature variations as well as noise sources that affect timing, eliminating the need to meet severe constraints and apply wide margins to ensure correct operation at a given (desired) performance.

References

[1] Mitra, S., Seifert, N., Zhang, M., Shi, Q., Kim, K.S.: Robust System Design with Built-In Soft-Error Resilience. IEEE Computer 38(2), 43–52 (2005)

[2] Mitra, S., Zhang, M., Waqas, S., Seifert, N., Gill, B., Kim, K.-S.: Combinational Logic Soft Error Correction. In: IEEE International Test Conference (2006)

[3] Austin, T., Blaauw, D., Mudge, T., Flautner, K.: Making Typical Silicon Matter with Razor. IEEE Computer 37(3), 57–65 (2004)

[4] Agarwal, M., Paul, B.C., Zhang, M., Mitra, S.: Circuit Failure Prediction and its Application to Transistor Aging. In: IEEE VLSI Test Symposium, pp. 277–284 (2007)

[5] Agarwal, M., Balakrishnan, V., Bhuyan, A., Kim, K., Paul, B.C., Wang, W., Yang, B., Cao, Y., Mitra, S.: Optimized Circuit Failure Prediction for Aging: Practicality and Promise. In: IEEE International Test Conference (2008)

[6] Nicolaidis, M., Zorian, Y.: On-Line Testing for VLSI – A Compendium of Approaches. Journal of Electronic Testing: Theory and Applications 12(1-2), 7–20 (1998)

[7] Metra, C., Degiampietro, R., Favalli, M., Ricco, B.: Concurrent Detection and Diagnosis Scheme for Transient, Delay and Crosstalk Faults. In: IEEE International On-Line Testing Workshop, pp. 66–70 (1999)

[8] Tsiatouhas, Y., Haniotakis, T.: A Zero Aliasing Built-In Self Test Technique for Delay Fault Testing. In: IEEE Symposium on Design for Testability of VLSI Systems, pp. 95–100 (1999)

[9] Anghel, L., Nicolaidis, M.: Cost Reduction and Evaluation of Temporary Faults Detecting Technique. In: Design Automation and Test in Europe Conference, pp. 591–598 (2000)

[10] Matakias, S., Tsiatouhas, Y., Arapoyanni, A., Haniotakis, T.: A Circuit for Concurrent Detection of Soft and Timing Errors in Digital CMOS ICs. Journal of Electronic Testing: Theory and Applications 20(5), 523–531 (2004)

[11] Nicolaidis, M.: Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies. In: IEEE VLSI Test Symposium, pp. 86–94 (1999)

[12] Sproull, R.F., Sutherland, I.E., Molnar, C.E.: The Counterflow Pipeline Processor Architecture. IEEE Design and Test of Computers 11(4), 48–59 (1994)
[13] Floros, A., Tsiatouhas, Y., Arapoyanni, A., Haniotakis, T.: A Pipeline Architecture Incorporating a Low-Cost Error Detection and Correction Mechanism. In: IEEE International Conference on Electronics, Circuits and Systems, pp. 692–695 (2006)
[14] Floros, A., Tsiatouhas, Y., Kavousianos, X.: The Time Dilation Scan Architecture for Timing Error Detection and Correction. In: IFIP/IEEE International Conference on Very Large Scale Integration, pp. 569–574 (2008)

Author Index