UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Brazilian Computer Society - SBC
Brazilian Microelectronics Society - SBMicro
Center for Advanced Microelectronics Technology - CEITEC
IEEE Circuits and Systems Society – Region 9 – Chapter RS
IEEE Student Branch UFRGS
IEEE-CAS Student Branch UFRGS

SIM 2008

23rd South Symposium on Microelectronics

Proceedings

General Chair

Marcelo Johann

Editors

Alessandro Girardi
Juan Pablo Martinez Brito

Published by

SBC

Bento Gonçalves, May 5-6, 2008

UFRGS - II - GME
Av. Bento Gonçalves, 9500 - Bloco IV
Caixa Postal 15064
CEP 91501-970 - Porto Alegre - RS - BRASIL
Phone: +55 51 3308 6155 Fax: +55 51 33087308
www.sbc.org.br/sim

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

South Symposium on Microelectronics (23. : 2008 : Porto Alegre, RS, Brazil).

Proceedings / XXIII South Symposium on Microelectronics,; editors Alessandro Girardi, Juan Pablo Marinez Brito. – Porto Alegre : SBC, 2008.
192 p. : il.

ISBN 978-85-7669-117-4

Conhecido também como SIM 2008.

1. Microeletrônica. I. Girardi, Alessandro. II. Brito, J. P. M. IV. SIM (23. : 2008 : Porto Alegre). V. Proceedings.

Printed in Porto Alegre, Brazil.

Cover: Ricardo Augusto da Luz Reis (UFRGS)
Logo by Ricardo Augusto da Luz Reis (UFRGS)

Foreword

Welcome to the 23rd edition of the South Symposium on Microelectronics. This symposium, originally called Microelectronics Internal Seminar (SIM), started in 1984 as an internal workshop of the Microelectronics Group (GME) at the Federal University of Rio Grande do Sul (UFRGS) in Porto Alegre. From the beginning, the main purpose of this seminar was to offer the students an opportunity for practicing scientific papers writing, presentation and discussion, as well as to keep a record of research works under development locally.

The event was renamed as South Symposium on Microelectronics in 2002 and transformed into a regional event, reflecting the growth and spreading of teaching and research activities on microelectronics in the region. The proceedings, which started at the fourth edition, have also improved over the years, receiving ISBN numbers, adopting English as the mandatory language, and incorporating a reviewing process that also involves students. Works submitted to this seminar may represent different levels of research activity, ranging from early undergraduate research assistant assignments to advanced PhD works in cooperation with companies and research labs abroad. The review process is therefore an important step to train young students and to provide feedback that helps to improve fresh new papers that may further be submitted to nationwide and international conferences.

This year SIM takes place at Bento Gonçalves togheter with the 10^{th} edition of the regional microelectronics school (EMICRO), a series of basic and advanced short courses provided by local professors and invited speakers.

These proceedings include 36 papers organized in 8 topics: CAD tools, analog and RF design, networks-on-chip, image and video processing, digital design and fault tolerance.

We would finally like to thank all individuals and organizations that helped to make this event possible. SIM 2008 was co-organized together with EMICRO UFRGS, promoted by the Brazilian Computer Society (SBC), the Brazilian Microelectronics Society (SBMicro) and IEEE CAS Region 9, receiving financial support from FINEP and Fapergs Brazilian agencies. Special thanks go to the authors and reviewers that spent precious time on the preparation of their works and helped to improve the quality of the event.

Bento Gonçalves, May 5, 2008
Alessandro Girardi
Juan Pablo Martinez Brito
Marcelo Johann

XXIII SIM - South Symposium on Microelectronics

Bento Gonçalves – RS – Brazil
May 5-6, 2008

General Chair

Marcelo de Oliveira Johann (UFRGS) – *johann@inf.ufrgs.br*

SIM Program Chairs

Alessandro Girardi (UNIPAMPA) – *alessandro.girardi@unipampa.edu.br*
Juan Pablo Martinez Brito (UFRGS) – *juan@inf.ufrgs.br*

EMICRO Program Chairs

Ricardo Augusto da Luz Reis (UFRGS) – *reis@inf.ufrgs.br*
Julio Leão da Silva Jr. (CEITEC) – *julio@ceitec.org.br*

List of Reviewers

Adriel Ziesemer Junior
André Mariano
André Aita
Braulio Mello
Bruno Zatt
Caio Alegretti
Carolina Neves
Cláudio Diniz
Cristiano Lopes
Cristina Meinhardt
Dalton Colombo
Eduardo Conrad Junior
Eduardo Flores
Eduardo Costa
Fábio Ramos
Felipe Marques
Fernando Paixão Cortes
Filipe Vieira
François Rivet
Guilherme de Freitas
Guilherme Corrêa
Júlio C. B. Mattos
José Augusto Nacif
Josue de Freitas
Juan Pablo Martinez Brito
Leandro Rosa
Leandro Pieper
Leomar Rosa Jr
Lisane Brisolara
Luciano Agostini
Luciano de Paula
Luiz Fernando Ferreira
Marcelo Porto
Matheus Braga
Márcio Kreutz
Murilo Pesati
Paulo Butzen
Rafael Cancian
Rafael Soares
Reginaldo Tavaves
Sandro Silva
Sandro Soares
Sidinei Ghissoni
Thaísa Silva
Thiago Assis

Table of Contents

ANALOG & RF DESIGN

NETWORKS-ON-CHIP

CAD TOOLS I

FAULT TOLERANCE

DIGITAL DESIGN

IMAGE & VIDEO PROCESSING I

IMAGE & VIDEO PROCESSING II

CAD TOOLS II

ANALOG & RF DESIGN

A Study on Components Sizing for CMOS Bandgap Voltage References

[1]Rafael Tambara Blumer, [1]Filipe Costa Beber Vieira, [1,2]Cesar Ramos Rodrigues
raftamb@mail.ufsm.br, filipecbv@mail.ufsm.br, cesar@ieee.org

[1]GMICRO - Microeletronics Group
[2] Electrical Engineering Department
Federal University of Santa Maria, UFSM
Santa Maria, Brazil

Abstract

The bandgap reference (BGR) is a block extensively used in analog circuits in implementation voltage references. This topology is widely used due its good performance and simplicity of operation. However, factors deriving from process variations limits the performance of the BGR. To circumvent this problem, a method for matching its components is proposed. This methodology trades with process parameters and power consumption to achieve a compromise between area and matching. The circuit was designed in XFAB-0.6 µm CMOS technology, has an output voltage of 1.268 V and consumes 40 µA. The area occupied by the BGR is 400x230 µm². The circuit is currently being fabricated.

1. Introduction

The bandgap reference (BGR) is one of most used architectures for voltage reference implementation. Main characteristic of references is stability. It is desirable that they can generate a fixed voltage, with immunity to power supply, temperature and variations in parameters during the fabrication process. For these reasons, the BGR are commonly found in ADCs, DACs, DRAMs, flash memories and variable gain amplifiers.

Many studies are being developed for designing high performance CMOS BGRs circuits. Although, components tolerance, and process variations severely degrades BGR performance [BRI 07]. To contribute on this issue, a simple design procedure is presented in this paper. In the proposed methodology transistors and resistors are sized aiming the best component matching for a given power consumption.

In the section 2, we describe the operational characteristics of a BGR circuit. The design procedure and theory about the matching components is developed in section 3. The section 4 shows the simulated results. The circuit layout and final conclusions are presented in the section 5 and 6 respectively.

2. Circuit Description

The schematic of BGR circuit studied in this paper is shown in fig. 1.

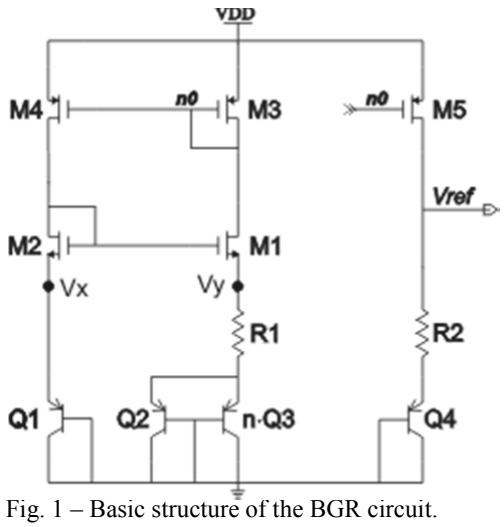


Fig. 1 – Basic structure of the BGR circuit.

This topology follows the basic principles of operation from BGRs: supply-independent biasing and temperature-independent reference signal generation [RAZ 01].

Supply-independent biasing aims stability of the signal reference with respect to V_{DD} oscillations. This feature requires that the circuit must bias itself. Auto biasing is obtained with the feedback action in current mirrors M_1-M_3 and M_2-M_4.

The generation of temperature-independent reference is another condition for BGR operation. Assuming that M_1-M_2 e M_3-M_4 are identical pairs, we note that $V_X=V_Y$ when $I_{D1}=I_{D2}$. Considering these conditions, and all transistors operating in strong inversion, the voltage on the R_1 can be expressed as (1).

$$V_{R1} = V_T \cdot \ln(n) \tag{1}$$

Consequently, the current of M_1 and M_2 transistors are given by:

$$I_{D1} = I_{D2} = \frac{V_T \cdot \ln(n)}{R_1}. \tag{2}$$

From the current mirror M_3-M_5, an output voltage V_{ref} is obtained as follows:

$$V_{ref} = I_{D5} \cdot R_2 + V_{BE4} = \left[\frac{(W/L)_5}{(W/L)_3} \cdot \frac{V_T \cdot \ln(n)}{R_1} \right] \cdot R_2 + V_{BE4}. \tag{3}$$

Deriving (3) as a function of temperature and considering $R_1=R_2$, we obtain:

$$\frac{\partial V_{ref}}{\partial T} = \frac{(W/L)_5}{(W/L)_3} \cdot \frac{K}{q} \cdot \ln(n) + \frac{\partial V_{BE4}}{\partial T} = 0 \Rightarrow K_{ref} = \frac{(W/L)_5}{(W/L)_3} = \frac{\dfrac{-\partial V_{BE4}}{\partial T}}{\dfrac{K}{q} \cdot \ln(n)}, \tag{4}$$

where K is the Boltzmann's constant, q is the electronic charge, $\partial V_{BE4} / \partial T = -1.826 mV / °C$ and n ratio between currents from Q_3 and Q_2 transistors. Equation (4) establishes the current mirror (M_3-M_5) gain.

In fig. 3 (a), the full BGR schematic is shown, including the start-up circuit. This circuit drives the BGR out of the degenerates bias point when the supply is turn on. The blocks diagram of fig. 3 (b) show operation stages of the start-up circuit.

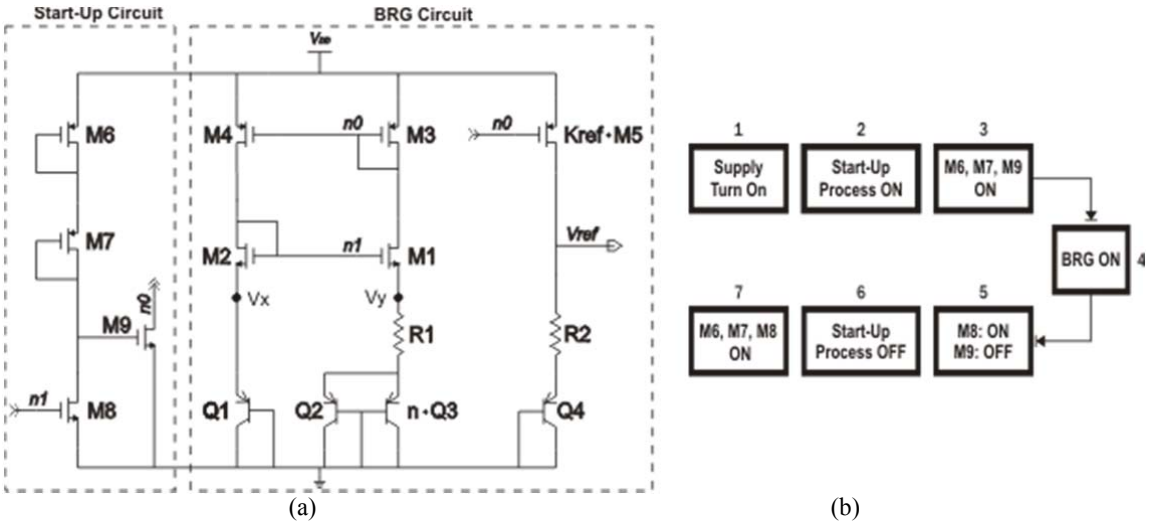


Fig. 3 – BGR circuit: a) Complete schematic; b) Operation stages of the start-up circuit.

The M_6-M_7 transistors must have long channels to avoid an excessive current consumption after the start-up process.

3. Circuit Design

This section presents a different way for sizing the BGR components in fig. 1. The mathematical routine uses the current consumption specification as starting point.

I. *Resistor design*

One of the most important characteristic of an electric circuit is the current consumption I_S. After the start-up circuit is disabled, the total consumption must be equal to the summation of currents from M_1, M_2, and M_5:

$$I_S = I_{D1} + I_{D2} + I_{D5}. \tag{5}$$

From fig.3, we can see that currents I_{D1}, I_{D2} and I_{D5} are related by aspect ratios from the transistors of the current mirrors, thus

$$I_{D2} = I_{D1}, \text{ and} \tag{6}$$

$$I_{D5} = K_{REF} \cdot I_{D1}. \tag{7}$$

Substituting (1), (6) and (7) in equation (5) and considering R_1=R_2, we can determine the circuit resistance values as a function of I_S:

$$R_1 = R_2 = \frac{V_T \cdot \ln(n) \cdot (2 + K_{ref})}{I_S} \tag{8}$$

The standard deviation between resistances of two identical rectangular devices depends on their areas [PEL 89]. This relation can be expressed as:

$$\sigma R\left(\frac{\Delta R}{R}\right) = \frac{A_R}{\sqrt{AREA_{resistor}}}. \tag{9}$$

Equation (9) is provided by foundry to model mismatching between resistors. The constant A_R is a process parameter and depends on the resistor type chosen in the project. For XFAB-0.6 μm CMOS technology, this data can be extracted from graph in fig. 4.

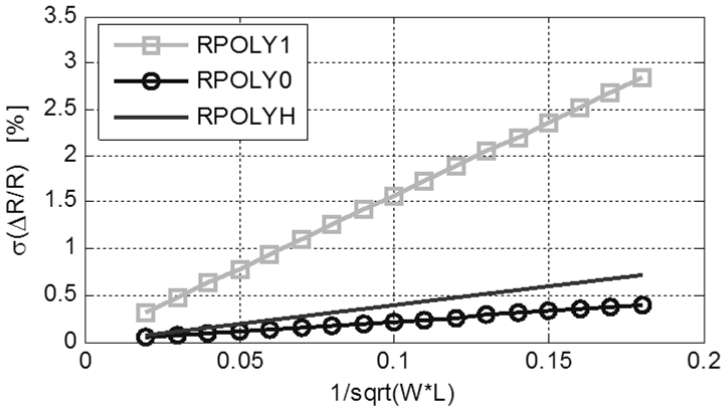


Fig. 4 – Poly resistor matching XC06.

Thus, using information from fig. 4, one can choose resistor type and correspondent sheet resistance ($R_\square$). The required resistance for BGR circuit can be defined from equations (10) trough (12):

$$R = R_\square \cdot N_\square \tag{10}$$

$$W_{resistor} = \sqrt{\frac{AREA_{resistor}}{N_\square}} \tag{11}$$

$$L_{resistor} = W_{resistor} \cdot N_\square \tag{12}$$

where $N_{\square}$ define a square numbers used for reach the desire resistance.

II. *Transistors design*

The threshold voltage and the current factor are the main sources of mismatching between MOS transistors. These random variations have a normal distribution with zero mean and standard deviation that depends on the area and the spacing between the components [PEL 89]. The current difference between two identical transistors is quantified according to data provided by foundry as follows:

$$\sigma\left(\frac{\Delta ID}{ID}\right) = \frac{AIDx}{\sqrt{W \cdot L}} \xrightarrow{with} x = (V_{gs} - V_{th}) \tag{13}$$

where *AIDx* is a process parameter that is strongly dependent from the overdrive gate voltage $(V_{gs}\text{-}V_{th})$ [XFA 05]. The foundry provides some *AIDx* values for specific bias conditions. The tab.1 shows the relation between the process parameter *AIDx* and $(V_{gs}\text{-}V_{th})$.

Tab.1 - Typical *AIDx* values for NMOS4 e PMOS4 devices.

Device	AID0.0	AID0.2	AID0.4	AID0.6	AID1.0	AID2.0	AID3.0
NMOS4	15.9	9.11	6.3	4.91	3.02	1.74	1.39
PMOS4	24.8	10.8	6.34	4.44	2.86	1.74	1.41

As Tab.1 presents only few operational conditions, equations were created to provide the intermediate solutions. These equations were created by curve fitting and can be expressed by equations (14)-(15). The results are compared to XFAB measures in fig. 5.

$$AID_{NMOS}(x) = -4,6 \cdot x^5 + 32,2 \cdot x^4 - 78,7 \cdot x^3 + 86,3 \cdot x^2 - 48 \cdot x + 15.8 \tag{14}$$

$$AID_{PMOS}(x) = -10,5 \cdot x^5 + 73,7 \cdot x^4 - 182,1 \cdot x^3 + 198,6 \cdot x^2 - 101,6 \cdot x + 24,7 \tag{15}$$

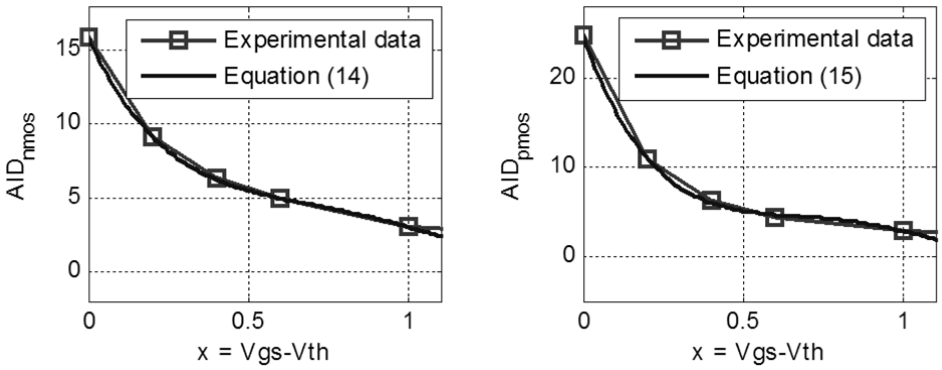


Fig. 5 – Process Parameter *AIDx*: a) AID_{nmos}; b) AID_{pmos}

The correct use of equation (13) requires a calibration of *AIDx* as a function of $(V_{gs}\text{-}V_{th})$. The V_{gs} value is approximated from simulated I_{DS} x V_{gs} curves, for NMOS and PMOS transistors with large dimensions (100µmx100µm), according to the current obtained from equation (2). Then, drawing square transistors and the current mismatching devices as $\sigma(\Delta ID/ID)$, *W-L* dimensions for NMOS (M_1,M_2) and PMOS (M_3,M_4) could be calculated for the specified mismatch as:

$$W_{NMOS} = L_{NMOS} = \frac{AIDx_{NMOS}}{\sigma\left(\Delta ID / ID\right)} \tag{16}$$

$$W_{PMOS} = L_{PMOS} = \frac{AIDx_{PMOS}}{\sigma\left(\Delta ID / ID\right)}. \tag{17}$$

Following the above routine, a BGR circuit with $\sigma(\Delta R/R) = \sigma(\Delta ID/ID)$ = 0.1%, current consumption of 40µA was projected for standard *RPOLY0* layer. Final dimensions obtained are shown in tab.2.

Tab.2 – Design Values

Component	Value
M1, M2	W = L = 79 μm
M3, M4, M5	W = L = 61 μm
M6, M7	W = 1 μm , L = 20 μm
M8, M9	W = 10 μm , L = 1 μm
Q1...Q4	25 μm^2 (*QPV5 Vertical BJT*)
R1, R2	W = 4 μm , L = 112 μm – 16.75 kΩ (*RPOLY0*)

4. Simulations Results

The BGR circuit from Fig. 3 (a) was simulated. The graph in fig. 6 depicts output voltage V_{ref} and the current I_S as a function of temperature. The circuit is supplied with 5 V.

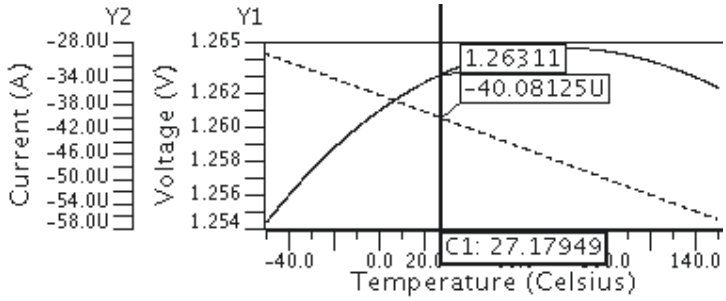


Fig. 6 – Reference voltage and consumed current as a function of temperature.

As expected, the output voltage and consumed current at 27 ºC are 1.263 V and 40.08 μA respectively. The total variation on the output voltage suffered was 11 mV on the analyzed band (-50 °C to 150 ºC), inferring a temperature coefficient of 40 ppm/ºC. The BGR has a PSRR of 44.6 dB under nomal operational condictions.

Another important result is shown in fig. 7. It shows output voltage variations in response to different supply voltages.

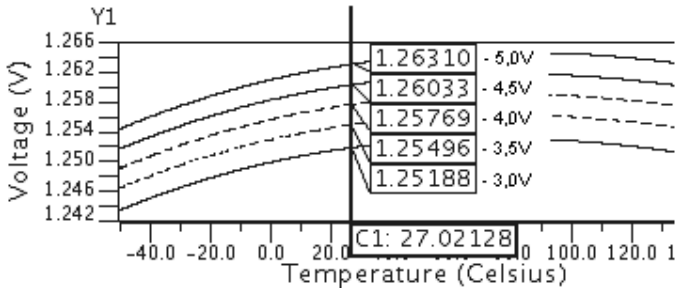


Fig. 7 – Reference voltage in function of supply variations.

We can notice the good stability of the circuit inside of 3-5 V band. Inside of this band was observed a maximum deviation of 12 mV in relation to the nominal conditions of operations.

The robustness of the project as analyzed with Monte Carlo simulation. In the Monte Carlo analysis, process and mismatch parameters are modified to verify the yield/limitation of the circuit when submitted to random variations associated to fabrication processes. The simulation was set up to 10000 rounds. The histogram of output voltage is shown in fig. 8.

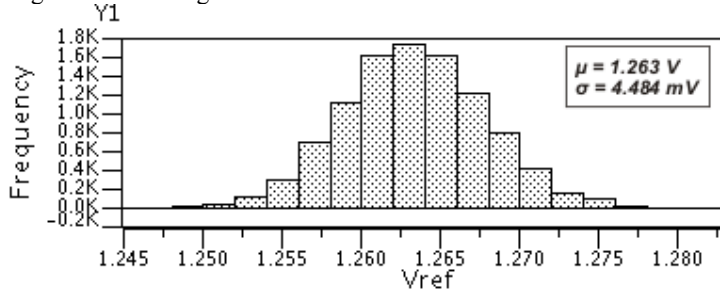
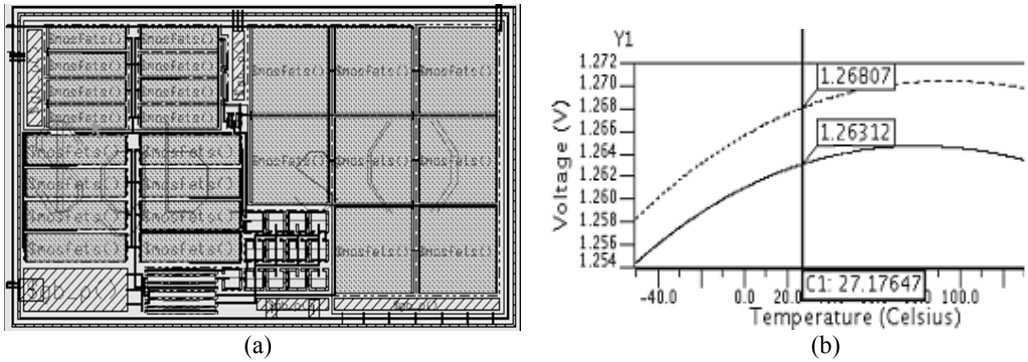


Fig. 8 – Monte Carlo Simulation

5. Layout

The layout of the circuit is fundamental for a good matching between the components. Some techniques are recommended for layout design, such as: to use components with great dimensions, place the matched components as close as possible and use common-centroid layouts. The final layout is showed in fig. 9 (a). A comparison between simulated and extract results is performed in fig. 9 (b). The total area used by BGR is 400x230 μm^2.



(a) (b)

Fig. 9 – (a) Layout; (b) Comparison between the simulated and extracted results.

6. Conclusions

A bandgap reference circuit was designed and simulated in XFAB-0.6 μm CMOS technology. The circuit was especially designed for studying the compromise between matching and area in these topologies. So, a methodology proposed related process parameters and current consumption. As the final result, was obtained bandgap circuit with 1.268 V of output voltage and total consumption of 40 μA, as foreseen analytically. The area used is 400x230 μm². To validate the design efficiency the circuit will be prototyped.

7. References

[BRI 07] BRITO, J.P.M; Klimach, H.; Bampi, S. **A Design Methodology for Matching Improvement in Bandgap References.** Quality Eletronic Design, 2007. ISQED 2007. 8th IEEE International Symposium on, March, San Jose, USA.

[KIN 05] KINGET, P.R. **Device mismatch and tradeoffs in the design of analog circuits.** Solid-State Circuits, IEEE Journal of, vol.40, no.6pp. 1212-1224, June 2005.

[PEL 89] PELGROM, M.J.M.; Duinmaijer, A.C.J.; Welbers, A.P.G. **Matching properties of MOS transistors.** Solid-State Circuits, IEEE Journal of, vol.24, no.5pp. 1433-1439, Oct 1989.

[RAZ 01] RAZAVI, B. **Design of Analog CMOS Integrated Circuits.** New York: McGRAW-HILL, 2001.

[XFA 05] XFAB Semiconductor, **Process Specifications XC06 – 0.6μm Modular CMOS**, Document PS_06_03, 2005.

Device Characterization of an IBM 0.18μm Analog Test Chip

Giovano da Rosa Camaratta, Eduardo Conrad Jr, Luiz Fernando Ferreira, Fernando Paixão Cortes, Sergio Bampi
{grcamaratta, econradjr, lfferreira, fpcortes, bampi}@inf.ufrgs.br

Federal University of Rio Grande do Sul (UFRGS) - Informatics Institute
Postal Code: 15.064 – CEP: 91.501-970 – Porto Alegre – Brazil

Abstract

The design of an analog circuit depends on several factors such as good device modeling and technology characterization. In this context, in order to validate the design methodology and the electrical models of a target technology, electrical characterizations based on measurements are necessary. This paper presents the main modules and the characterization of a test chip developed with several RF blocks, oscillators and test devices. The chip was prototyped in IBM 0.18µm CMOS technology process through a MPW service. Preliminary measurements of the ring oscillator and test vehicles will also be presented.

1. Introduction

The development of ultra-scaled VLSI technologies, coupled with the demand for more signal processing integrated in a single chip, has set the trend for integrating analog circuits in digital CMOS technologies below 90 nm feature sizes.

Several analog circuit blocks were designed using a full-custom design methodology based on the g_m/I_D characteristics, the same methodology that was used in previous works [COR 03]. In the first results, the technology characteristics and circuits performance were obtained with electrical simulations using the foundry-supplied typical BSIM3v3 model parameters for the target technology IBM 0.18µm CMOS technology.

In order to validate the design methodology and the electrical models, electrical characterizations based on measurements are necessary. Thus, a test chip was prototyped in IBM 0.18µm CMOS technology process and 10 not encapsulated chips and 5 encapsulated (64-pin QFN package) have been measured. The objective here is to use the electrical measurements to perform characterization to fine tune the basic design curves and validate the blocks performance.

2. Test Chip Structures

A test chip with the designed blocks and test vehicles in IBM 0.18µm CMOS technology was developed to characterize the design through electrical measurements. The chip microphotography is shown in **Fig. 1** and the area of each block is shown in Tab. 1. Each block that composes the test chip is briefly described as follows.

There are 6 RF analog building blocks implemented: a complete variable gain front-end architecture, an up-conversion mixer, a down-conversion mixer, a variable gain amplifier (VGA) [COR 08] and a voltage-controlled oscillator (VCO) using a full-custom design methodology based on the g_m/I_D characteristics [PAU 07] and a ring oscillator. All the design optimizes both speed and the power consumption, as well as reasonable sensitivity and gain are achieved.

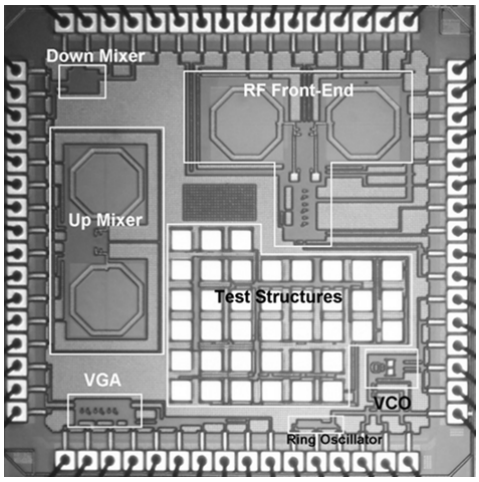


Fig. 1 – Chip die microphotography.

Tab. 1 – System chip area breakdown.

Block	Area (mm^2)
Up Mixer FI=1.4GHz	0.7100
Down Mixer 1.4GHz to 40MHz	0.0210
VGA (variable gain amplifier)	0.0350
Front-End	0.7700
VCO	0.0220
Ring Oscillator	0.0073
Test Structures	1.2480
Pad Ring	1.7600
Unused Space	2.2260
Total Area	6.8000

Several test structures was designed in order to allow the extraction of the transistors parameters applicable to all operation regions and all device sizes. The test structures inside the test chip include long (L = 10µm), short (L = 0.18µm), wide (W = 10µm) and narrow (W = 0.22µm) channel transistors. Other geometries are also presented, such as series-parallel associations of transistors (TAT's) of different sizes and shapes [GIR 03].

These structures will be used to characterize the behavior of the drain current, gate transconductance, output conductance, noise and intrinsic frequency in terms of geometry, association of unit transistors and layout strategies. The test structures will be measured through microprobes, so we used micro-pads to access the structures terminals (the PAD's, which were obtained from the IBM 0.18 µm technology library).

The chip microphotography, as shown in **Fig. 1**, has a total area (including 64 pads) of 6.8 mm². Most of the chip area is occupied by micro-pads, in order to access the device terminals for measurements. The area of each block is shown in Tab. 1.

Fig. 2a shows the micro-pads layout prototyped and Fig. 2b presents the micro-pads configuration and transistor location in order to access the structure terminals for testing and measurements. The test chip was fabricated containing a few equivalent composite transistors formed by different associations in 0.18µm CMOS technology composed by unit transistors of equal size. These associations and a reference transistor (a single transistor showing the same W/L relation as a functional reference) are used to achieve by comparative measuring the analog behavior [GIR 03]. Fig. 3a shows an electrical composition of the prototyped TAT associations and the Fig. 3b the PMOS and NMOS TAT layout example.

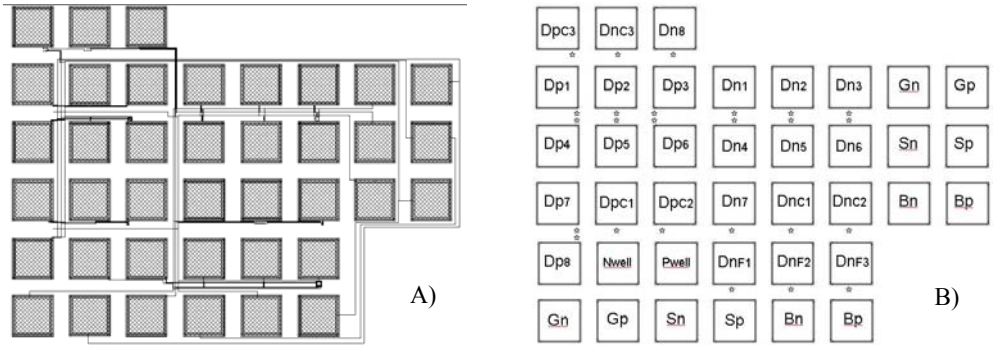


Fig. 2 – (A) Micro-pads layout and (B) Micropads configuration and transistor location.

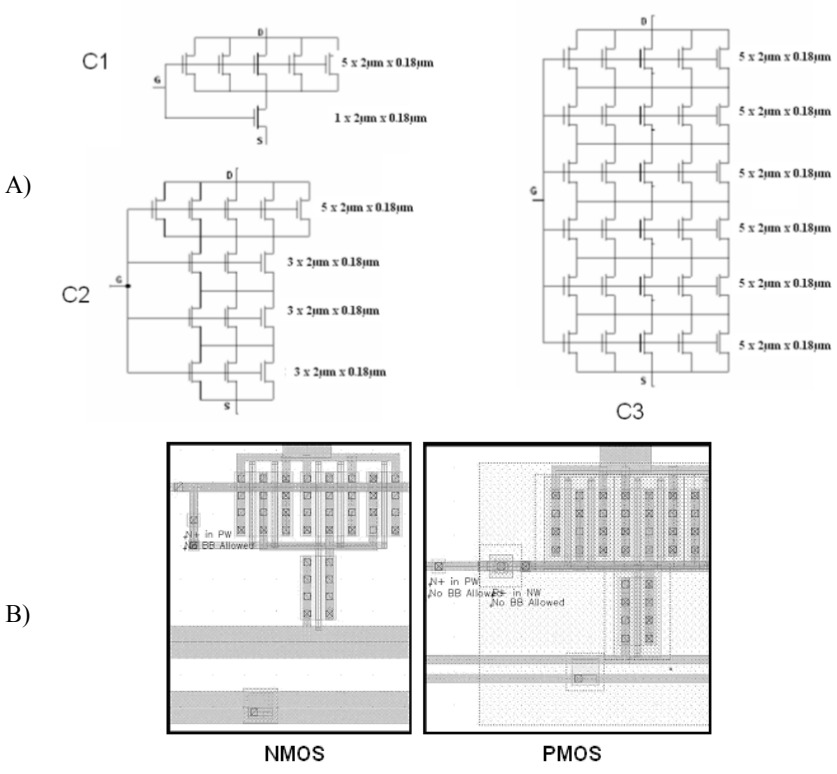


Fig. 3 – (A) Electrical configuration of the prototyped TAT's and (B) NMOS and PMOS layout of C1 TAT.

3. Test Characterization Procedure

The test and characterization procedure will follow two main steps. The first step is the characterization of the target process. Using a parameter analyzer and a microprobe station the basic DC measurements was obtained, such as *Id x Vd*, *Id x Vg* and *Id x Vs*. With these curves many important parameters can be obtained.

Therefore, the process in use can be fully characterized and the parameters of CMOS simulation models (such as ACM and EKV models) can be obtained. This can be used to fine tune the basic design curves and validate the designed blocks performance. Drain current, gate transconductance, output conductance and noise can be obtained direct from the MOS transistors structures measurements. The maximum frequency for this target process can also be obtained, using the Ring Oscillator structure. Fig. 4 shows some preliminary results for the MOS transistors characterization.

The next step is the fully characterization and validation of each block. In order to verify the gate delay for this process, a ring oscillator was designed and characterized. To verify the resulting average delay time a 19 stage ring oscillator circuit was implemented. The final layout is presented in the Fig.5 and the transistor sizes in the ring oscillator are Wp = 0.6µm, Wn = 0.4µm and Ln = Lp = 0.18µm.

We measured at a typical power supply voltage (± 1.8V) the oscillation frequency at about of *630 MHz*, resulting in an average gate delay of *41.8ps*. Fig. 6a illustrates the oscillation frequency dependency of the power supply and Fig. 6b the average delay versus supply voltage. These results were obtained using a spectrum analyzer through measurement of five samples.

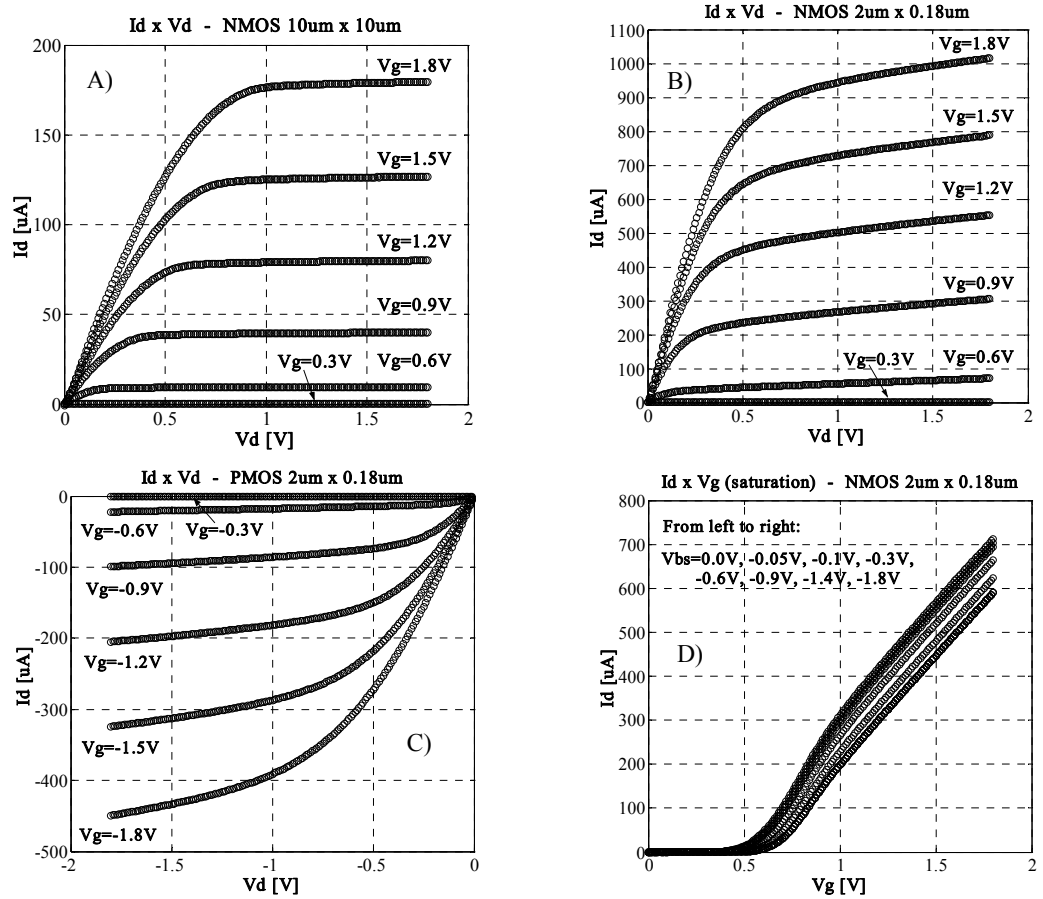


Fig. 4 – (A) Id vs. Vd measured curves for a single NMOS Transistor 10µm x 10µm; (B) Id vs. Vd measured curves for a single NMOS Transistor 2µm x 0.18µm; (C) Id vs. Vd measured curves for a single PMOS Transistor 2µm x 0.18µm and (D) Id vs. Vg measured curves for a single NMOS Transistor 2µm x 0.18µm.

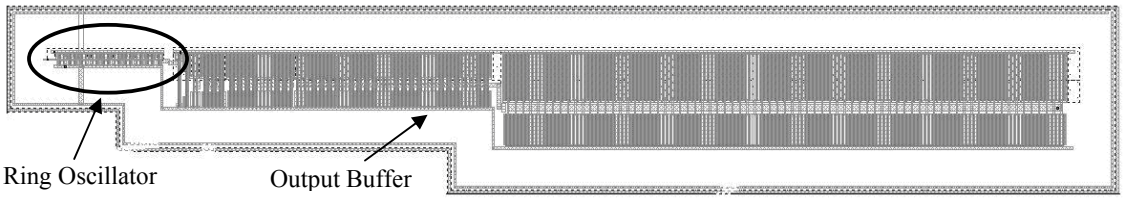


Fig. 5 – Ring Oscillator final layout.

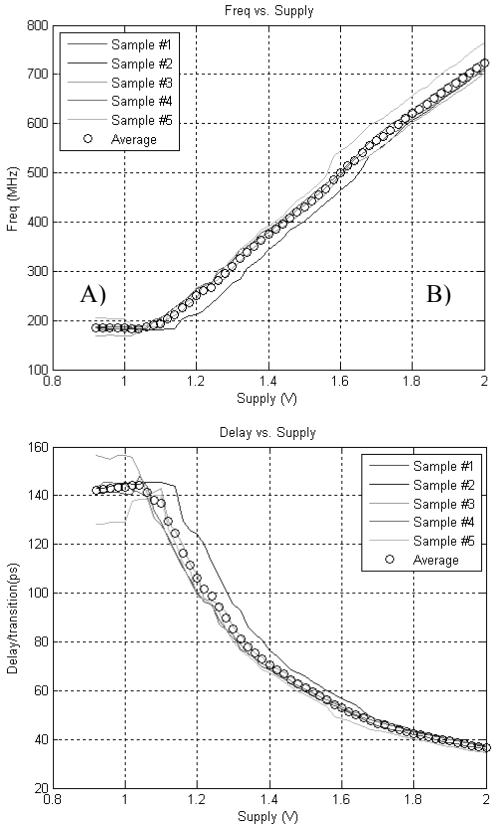


Fig. 6 – Ring Oscillator results: (A) Frequency vs. Supply Voltage and (B) Delay vs. Supply Voltage.

4. Conclusions and Future Work

Electrical characterizations based on measurements are fundamental for the complete analysis of an analog integrated circuit. Only measurements can determine the real circuit functionality. The prototyped test chip is currently under testing, and preliminary measurements of the designed blocks and test vehicles were presented, showing good results performance. As future work, we intend to perform a fully characterization and modeling optimization through electrical measurements to fine tune the basic design curves and validate the blocks performance.

5. Acknowledgements

The support of FAURGS, FINEP, CNPq and CAPES Brazilian agencies with scholarships and PDI-TI Program grant are gratefully acknowledged.

6. References

[COR 03] CORTES, Fernando Paixão; FABRIS, Eric; BAMPI, Sergio. **Applying the g_m/I_D Method in the Analysis and Design of a Miller Amplifier, a Comparator and a Gm-C Band-Pass Filter.** IFIP VLSI Soc 2003: Darmstadt, Germany, December 2003.

[COR 08] CORTES, Fernando Paixão; BAMPI, Sergio. **Analysis, Design and Implementation of Baseband and RF Blocks Suitable for a Multi-Band Analog Interface for CMOS SOCs.** PHD Thesis, PGMICRO – UFRGS, Porto Alegre, RS, 2008.

[GIR 03] GIRARDI, Alessandro; BAMPI, Sergio. **LIT - An Automatic Layout Generation Tool for Trapezoidal Association of Transistors for Basic Analog Building Blocks.** In: Design, Automation and Test in Europe 2003 - DATE'03. Munich, Germany, March 2003.

[GIR 03] GIRARDI, Alessandro; CORTES, Fernando Paixão; BAMPI, Sergio. **Analog IC Modules Design Using Trapezoidal Association of MOS Transistors in 0.35µm Technology.** In: 16th Symposium on Integrated Circuits and Systems Design (CHIP IN SAMPA), SBCCI 2003. Proceedings. São Paulo, Brazil, September 2003. 311-316 p.

[PAU 07] PAULA, Luciano S. de; Fabris, E., Bampi, S. **A High Swing Low Power CMOS Differential Voltage-Controlled Ring Oscillator**. In: 14th IEEE International Conference on Electronics, Circuits and Systems – ICECS 2007. December, 2007.

Accurate Method for Subthreshold and Gate Leakage Current Estimation in CMOS Complex Gates

[1]P. F. Butzen, [1]L. S. da Rosa Jr, [1]E. J. D. Chiappetta Filho, [1]D. S. Moura, [2]A. I. Reis, [1]R. P. Ribas

{pbutzen, leomarjr, ejdcfilho, dsmoura, rpribas}@inf.ufrgs.br, are@nangate.com

[1]Instituto de Informática – UFRGS, Porto Alegre, RS, Brazil
[2]Nangate Inc., Menlo Park, CA, USA

Abstract

This paper proposes a new method to estimate static power dissipation in digital circuits by evaluating simultaneously subthreshold and gate oxide leakage currents. The estimation method is performed over logic cells, including CMOS complex gates with multi-level series-parallel devices. Experimental results have been carried out on different fabrications processes, and good correlation with HSPICE™ simulator was obtained. The algorithm presents a speed up of 80x when compared to HSPICE™.

1. Introduction

Static power consumption is nowadays a crucial design parameter in digital circuits due to emergent mobile products and leakage current effects. Different leakage mechanisms have been discussed and modeled in literature [ROY 03]. The threshold voltage scaling increases subthreshold current exponentially. The gate oxide tunneling also appeared in sub-100nm as a result of the drastic scaling down of gate oxide thickness, although high-κ dielectric are being considered as an efficient solution to overcome such drawback [ITR04].

To face this new challenge, a great effort has been done in developing models and estimators for design support. The 'stack effect' observed with off-transistor (i.e., devices which are turned off) in series arrangement is quite important for subthreshold current prediction [GU 96]. Differently from subthreshold leakage, gate oxide tunneling currents are observed in both on- and off-devices, according to the transistor biasing [RAO 03]. Moreover, on-transistors in off-networks are usually considered by authors as ideal short-circuits in terms of subthreshold current prediction [YAN 05], however this can lead to inaccuracies [BUT 07].

Furthermore, the interaction among leakage mechanisms cannot be ignored in the analysis of static consumption. It was firstly calculated by Mukhopadhyay et al. [MUK 03], and a numerical solver, based on Matlab™, has been developed to evaluate leakage in simple logic gates by solving the Kirchoff's Current Law (KCL) at intermediate nodes, through the use of Sum of Current Source (SCS) transistor model.

In [LEE 03], Lee et al. pointed out the importance and complexity of subthreshold and gate oxide leakage simultaneous analysis, but dealing it through the use of look-up tables composed by SPICE pre-characterized values for a set of distinct transistor biasing. In that work, series-parallel transistor structures are mentioned for only maximum two levels of logic depth, for instance AOI and OAI gates, replacing parallel non-conducting transistor by a single device of equivalent size. It has been already demonstrated in [BUT 07] to be unsuitable for complex gate leakage evaluation.

The particular case of a 3-input NAND gate with the input state where only the device in the middle of the NMOS stack is conducting was discussed in [LEE 03] and [RAO 04]. In the first one, individual leakage results from SPICE simulation. In [RAH 04], the analytical method presented considers the internal nodes, in such biasing condition, quite closed to the power supply potential and the gate oxide leakage value is simply neglected. Moreover, the backward gate leakage current in non-conducting NMOS transistor placed other than on the top of the stack is calculated by considering source-gate voltage equals to power supply voltage minus the device threshold voltage. It is not true for devices placed in different positions in a more representative complex gate, resulting in inaccurate analysis.

No analytical model related to the interaction of leakage currents for multi-level series-parallel complex gates has been found in the literature. The main contribution of this work is to provide a straightforward and accurate method for fast prediction of static power dissipation by interacting different leakage mechanisms in CMOS complex gates. Multi-level series-parallel (SP) device arrangements are considered. An algorithm based on current-source association is proposed to calculate the voltage at internal cell nodes, according to the topology analysis. This avoids the adoption of more complex nonlinear equation solvers [MUK 03].

2. Proposed Estimation Method

The subthreshold current is usually modeled by the equation [SHE 87]:

$$I_S = I_0 W e^{\frac{V_{gs} - (V_{t0} - \eta V_{ds} - \gamma V_{bs})}{nV_T}} \left[1 - e^{\frac{-V_{ds}}{V_T}} \right] \quad (1)$$

where $I_0 = \frac{\mu_0 C_{ox} V_T^2 e^{1.8}}{L}$ and $V_T = \frac{kT}{q}$. V_{gs}, V_{ds} and V_{bs} are the gate-, drain- and bulk-to-source voltages of the transistor, respectively. V_{t0} is the zero bias threshold voltage, while W and L are the effective transistor width and length, respectively. γ is the body effect coefficient and η is the DIBL coefficient. C_{ox} is the gate oxide capacitance, μ_0 is the mobility and n is the subthreshold swing coefficient.

The gate oxide leakage, in turn, is modeled by the following equation [ROY 03]:

$$I_{gate} = W.L.A\left(\frac{V_{ox}}{t_{ox}}\right)^2 \exp\left(\frac{-B\left(1-\left(1-\frac{V_{ox}}{\phi_{ox}}\right)^{3/2}\right)}{\frac{V_{ox}}{t_{ox}}}\right) \quad (2)$$

where $A = q^3 / 16\pi^2 h\phi_{ox}$, $B = 4\pi\sqrt{2m_{ox}}\phi_{ox}^{3/2} / 3hq$, W and L are the effective transistor width and length, respectively, m_{ox} is the tunneling particle effective mass, ϕ_{ox} is the tunneling barrier height, V_{ox} is the potential drop across the gate oxide, t_{ox} is the oxide thickness, h is $1/2\pi$ times Planck's constant and q is the electron charge.

To the purpose of this work, whose goal is to estimate the total leakage current in complex CMOS gates, both equations (1) and (2) can be simplified to the following equations (3) and (4), respectively:

$$I_S = I_{S0} W e^{\frac{V_{gs}+\eta V_{ds}+\gamma V_{bs}}{nV_T}}\left[1-e^{\frac{-V_{ds}}{V_T}}\right] \quad (3)$$

where $I_{S0} = \frac{\mu_0 C_{ox} V_T^2 e^{1.8\frac{V_{t0}}{nV_T}}}{L}$, and

$$I_g = I_{g0}.W.e^{\frac{-K}{|V_{ox}|}} \quad (4)$$

where I_{g0} represents the gate leakage current for $V_{ox} = V_{dd}$, and K is a calibration constant. Due to the transistor bias state dependence, the proposed model considers the gate leakage currents for transistor turned ON (I_{g_ON}) and turned OFF (I_{g_OFF}). The parameters I_{S0}, I_{g_ON}, I_{g_OFF}, K, η, and γ are extracted through electrical simulation.

2.1. Cell Leakage Estimation Heuristic

In order to calculate each leakage current component, the voltage at cell internal nodes must be known. However, every current contributes on that by interacting each other. The proposed algorithm consists in three basic steps:

(1) On- and off-plane identification and off-network extraction – It consists in identifying the state of conduction of the logic planes (called here as on- and off-planes), for a certain input state condition. Notice that, in on-planes, the subthreshold leakage mechanism is not observed, while the gate oxide leakage is verified at switched-on transistors. Moreover, all nodes at the pull-up PMOS or pull-down NMOS conducting plane attain the power supply voltage (*Vdd*) or ground voltage (*gnd*), respectively.

The off-network is extracted from the off-plane by evaluating the conducting of devices in this plane. On-devices have their source and drain terminals short-circuited, and a current source representing the gate oxide current contribution is associated to such node. Note that, nodes short-circuited by on-transistors can eventually remove other devices, but again the contribution of these devices must be represented by current sources at the node in analysis.

(2) Internal node voltages calculation – The unknown internal node voltages have to be determined. For that, these nodes are treated in a sequential way, calculating firstly the nodes nearest to the cell output terminal, while the other unknown nodes are temporarily considered at the reference *Vdd* or *gnd*, according to the plane type. This procedure avoids the use of complex and time consuming nonlinear solvers, like Matlab™ and SPICE, without significant lost in accuracy.

The currents from all devices connected to a certain node in evaluation are then associated using the KCL principle, including also the current sources added when replacing the removed devices, mentioned before. Considering this procedure, the evaluated voltage represents in fact the drain-to-source voltage of the transistors connected at the supply reference. To compute the unknown internal nodes values, a roughback procedure from supply reference will compute the drain-to-source voltage of each transistor.

(3) Total leakage current estimation – Once all internal node voltages have been determined, the leakage currents are then estimated taking into account these voltage values. To account the total cell leakage, the subthreshold currents of the devices connected to the ground terminal (power supply terminal for PMOS off-plane) are summed, as well as the gate oxide leakage resulted from off-devices in the off-network. On-devices

connected to ground terminal in the off-plane and removed during the generation of the off-network also contribute to the final total leakage through the on-gate leakage. Finally, the gate leakage of the on-devices in the on-plane must be added to the final value

2.2. Case Study: Method Demonstration

To illustrate the procedure described in the previous sub-section, the complex gate depicted in Fig. 1a is considered. Taking into account the input vector [a,b,c,d,e,f,g,h,i] = [1,0,1,0,0,0,1,0,1], as shown in Fig. 1b, the pull-up PMOS plane represents the conducting one, while the pull-down NMOS plane is the off-plane, it means the output node presents the power supply voltage *Vdd*. The off-network extracted from the off-plane is shown in Fig. 1c. For this particular example, some conditions discussed before can be identified:

(a) the NMOS transistor with 'a' input ('ta'), placed at the top of the off-plane, is turned on, resulting in a voltage drop over it;
(b) the NMOS transistor 'tc' is short-circuited. It leads to the stack of two off-devices separate by an on-transistor, as pointed out in other works as a special case [LEE 03]. Both transistor 'tc' and 'td' are replaced by source currents;
(c) the NMOS transistor 'ti' is short-circuited. 'ti' and 'th' are replaced by source currents and 'tf' is connected to the top terminal;
(d) the transistor 'tg' is also short-circuited and replaced by a current source.

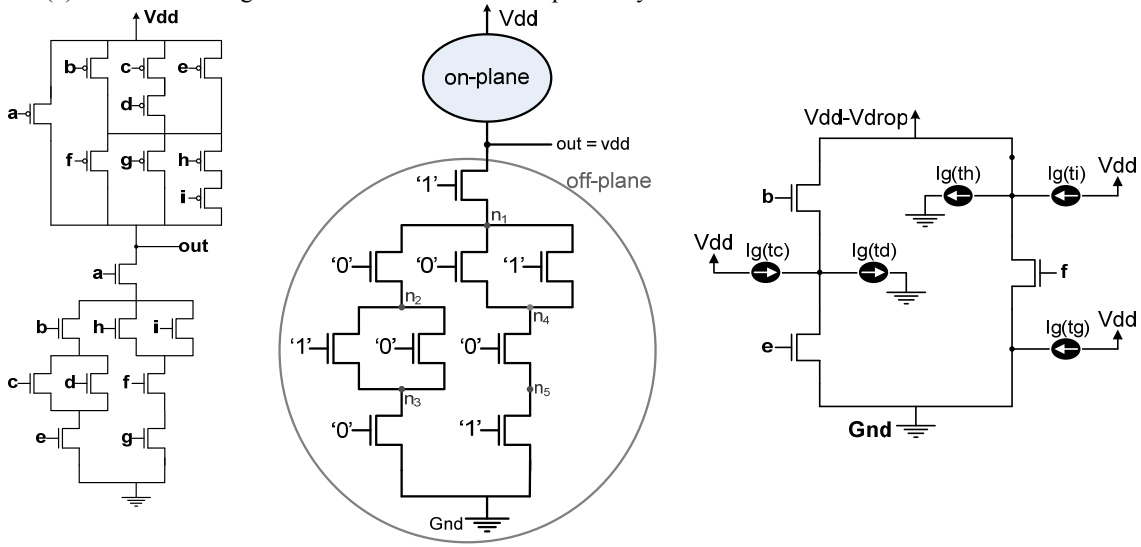


Fig. 1 – Complex CMOS logic gate: (a) original topology, (b) indication of steady state condition, (c) off-network from the off-plane.

In this case study only one node in the off-network is unknown, corresponding to the original 'n2' and 'n3' joined. To calculate it, the subthershold current from devices 'tb' and 'te', as well as the gate oxide leakages from the off-devices 'tb', 'td' and 'te' and the on-device 'tc' are associated according to their direction.

The total cell leakage current is then estimated by summing all current going out from the logic gate or, in other words, the currents flowing to the ground potential. It includes the gate leakage in the on-PMOS transistors and the off-NMOS ones, whose gate terminal is connected to low logic level. Moreover, the subthreshold current of the transistors connected to the *gnd* terminal, as well as the gate current of the on-NMOS 'tg' connected to the same ground node must also be added to the final value.

3. Experimental Results

To validate the proposed leakage estimation method in CMOS technology, the results obtained analytically were correlated with HSPICE™ simulation data, considering 65nm and 45nm CMOS Predictive Technology Model (PTM) process parameters [PTM 08].

First of all, the complex gate presented in Fig. 1a was evaluated to all input combinations, as shown in Fig. 2. The method was also evaluated at circuit level, taking into account ISCAS85 benchmark circuits [BRG 85] mapped with GenLib_44-6 library. The steady state of each internal cell in the circuit is pre-evaluated for a certain primary input vector. The leakage current in the cells is then calculated individually, and summed for the total current estimation. Results obtained at circuit level are shown in Tab. 1. The results present a good correlation to HSPICE™ simulation and a speed up of 80X.

The major contributor for the prediction error has been identified to be the switched-on devices voltage drop present in the off-plane. On-devices have been treated by other authors as ideal short-circuits. Besides it has significant impact in the leakage current estimation when they are placed at the top of the off-plane, it also

presents different values according to the number of off-stacked-transistor bellow that. Although it suggests a small voltage difference, the impact in the subthreshold and gate oxide leakages is quite significant.

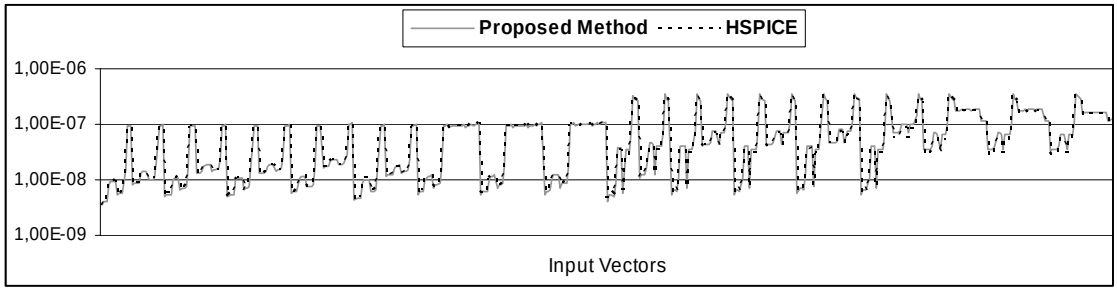


Fig. 2 – Leakage current in the logic gate from Fig. 1a using 45nm PTM process, for all 512 input combinations.

Tab. 1 – Total leakage current estimation for ISCAS85 benchmark circuits [BRG 85].

I_leak (µA)	PTM 65nm			PTM 45nm		
	Method	HSPICE™	Error	Method	HSPICE™	Error
C432	11.8	11.3	4.4%	22.0	20.9	5.3%
C499	37.2	35.6	4.5%	69.8	65.6	6.4%
C888	19.8	18.8	5.3%	36.3	34.2	6.1%
C1355	39.9	38.0	5.0%	73.6	69.1	6.5%
C1908	30.3	29.0	4.5%	54.9	51.5	6.6%
C3540	65.3	61.9	5.5%	122	114	7.0%
C6288	200	191	4.7%	370	350	5.7%

4. Conclusions

A new leakage current estimation method for CMOS circuits was presented, providing clear improvement in the state-of-the-art. Subthreshold and gate oxide leakages are both considered in the analysis by interacting such mechanisms. The proposed heuristic defines a sequential procedure, making some assumptions which simplify the current source equations calculation, without penalty in accuracy. The main factor of difference to HSPICE™ simulation data, taken here as the golden reference, is the variation in the drain-to-source voltage of switched-on transistors, usually ignored by other authors which consider it as an ideal short-circuit. Finally, the estimation method computation is significantly faster when compared to electrical simulation.

5. Referências

[BRG 85] BRGLEZ, F. et al. **A Neutral Netlist of 10 Combinatorial Benchmark Circuits**. ISCAS 1985, pp. 695-98.

[BUT 07] BUTZEN, P. F. Butzen et al. **Modeling and Estimating Leakage Current in Series-Parallel CMOS Networks.** GLSVLSI 2007, pp. 269-274.

[GU 96] GU, R. X. et al. **Power Distribution Analysis and Optimization of Deep Submicron CMOS Digital Circuit.** IEEE J. Solid-State Circuits, vol.31, no.5, May 1996, pp.707-713.

[ITR 04] **International Technology Roadmap for Semiconductors**, 2004 Edition. Available at http://public.itrs.net.

[LEE 03] LEE, D. et al. **Simultaneous Subthreshold and Gate-Oxide Tunneling Leakage Current Analysis in Nanometer CMOS Design.** ISQED 2003, pp. 287-292.

[MUK 03] MUKHOPADHYAY, S. et al. **Accurate Estimation of Total Leakage in Scaled CMOS Logic Circuits Based on Compact Current Modeling.** DAC 2003, pp. 169-174.

[PTM 08] **Berkeley Predictive Technology Model.** Available at http://www-device.eecs.berkeley.edu/~ptm.

[RAH 04] RAHMAN, H. et al. **A Leakage Estimation and Reduction Technique for Scaled CMOS Logic Circuits Considering Gate-Leakage.** ISCAS 2004, vol.2, pp. 297-300.

[RAO 03] RAO, R. M. et al. **Efficient Technique for Gate Leakage Estimation.** ISLPED 2003pp. 100-103.

[RAS 07] RASTOGI, A. et al. **An Efficient Technique for Leakage Current Estimation in Sub 65nm Scaled CMOS Circuits Based on Loading Effect.** IEEE VLSID 2007, pp. 583-588.

[ROY 03] ROY, K. et al. **Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits.** Proceedings IEEE, vol. 91, no. 2, Feb. 2003, pp. 302-327.

[SHE 87] SHEU, B. J. et al. **BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors.** IEEE J. Solid-State Circuits, vol. SC-22, no. 4, Aug. 1987, pp. 558-566.

[YAN 05] YANG, S. et al. **Accurate Stacking Effect Macro-modeling of Leakage Power in Sub-100nm Circuits.** Proc. Int. Conf. on VLSI Design, Jan. 2005, pp. 165-170.

A 65nm CMOS Analog Processor for Mobile Terminals Software Radio Front End

[1]François Rivet, [1]Yann Deval, [1]Domnique Dallet, [1]Jean-Baptiste Begueret,
[2]Didier Belot, [2]Philippe Cathelin
francois.rivet@ims-bordeaux.fr, didier.belot@st.com

[1]IMS Laboratory, University of Bordeaux, Talence, France
[2] STMicroelectronics R&D, Crolles, France

Abstract

The forth generation of mobile terminals is the place of multimedia convergence. Audio, video and data are told to be broadcast on the same device. Microelectronics industry is challenging to integrate all these applications. The Software Radio concept is one of the key to overcome technological bottleneck implied by mobile devices constraints. This paper presents the architecture and the design of a 65nm CMOS analog component in the RF receiving chain. Behavioural and post-layout simulations are depicted to validate the design of a RF analog processor.

1. Introduction

The Software Radio (SR) concept aims at bringing closer the antenna to a reprogrammable component in the RF receiving chain. Researches are more and more concentred on this topic since the last decade [MIT 95] because of the emergence of various RF standards and the need to broadcast them all together. That is why telecommunications industry is looking for solutions to integrate low cost, low power chips in mobile devices to answer this problematic. Multi-standard, multi-functional systems were studied using the concept of Software Defined Radio [ABI 06]. Whereas this concept allows handling several radio communication standards, SDR architectures are still limited to a few defined RF standards. Cognitive Radio (CR) emergence asked for more flexible architectures, like Software Radio.

This paper presents a Software Radio processor for mobile terminals. The Software Radio concept is ideally implemented in a receiving chain with an antenna directly connected to an Analog-to-Digital Converter (ADC) which displays the RF digital converted signal to a DSP. Then, the DSP can process digitally any RF signals and thus demodulate any kind of communication standard whatever its frequency, its channel bandwidth, its modulation. Such a system is not feasible according to mobile terminals constraints. The power consumption and the resolution of an ADC at high frequencies are the technological bottleneck. To overcome this problem, it is proposed to design an analog processor which process analogically RF signals at an acceptable power consumption for mobile terminals (Fig. 1). This component is called Sampled Analog Signal Processor (SASP). The analog operations executed by the processor can be done at high frequencies and at low power consumption.

A Processor is a component which implements an algorithm to process data (Fig. 2). The SASP is divided into 2 main parts: one to address the algorithm, one to receive and send data by the mean of a RF signal (Fig. 2). Each part is detailed in the next paragraphs.

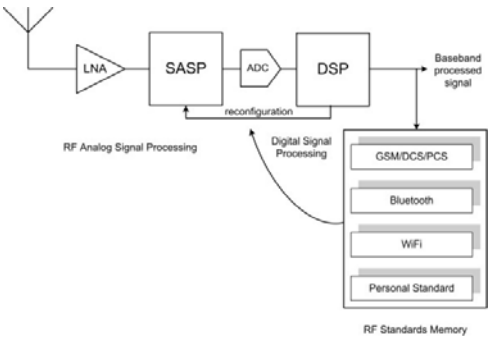


Figure 1: A SR receiving chain

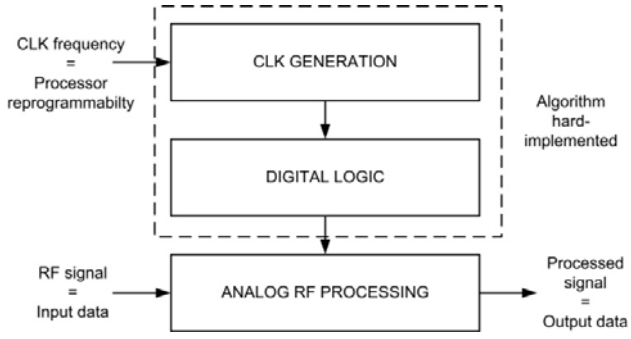


Figure 2: Processor Architecture

2. A Sampled Analog Signal Processor

2.1 The Butterfly Algorithm

The Sampled Analog Signal Processor implements the algorithm of a Discrete Fourier Transform (DFT) (Fig. 2). It goals at processing the spectrum of any RF signal in a range from 0 GHz to 5 GHz. Once the spectrum processed analogically, it is delivered through voltage sample (Fig. 3). It is just a matter to select the voltage sample representing the spectral envelope of the desired RF signal. Only this selected envelope is converted in digital. The selection of voltage samples participates to the reduction of the working frequency of the ADC. At a constant frequency, the SASP enables to lower the processing speed and the data rate of the ADC and the DSP. Thus, the power consumption is dramatically reduced and a Software Radio architecture can be integrated in a chip destined to mobile terminals.

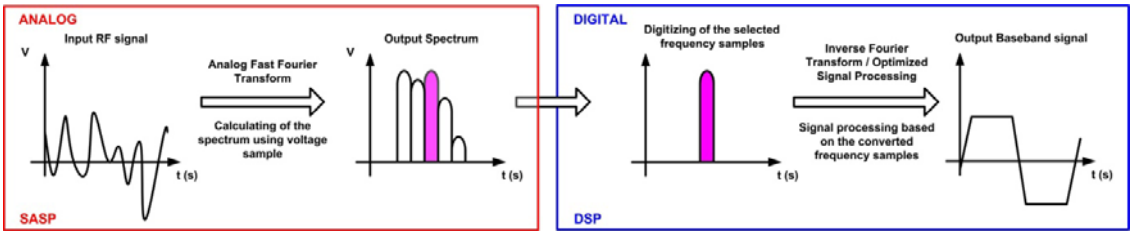


Figure 3: Processor principle

To implement the DFT, a radix-4 Butterfly algorithm is chosen (Fig. 4) [SWA 84]. It is a pipelined FFT. It is composed by $n=\log_4(N)$ generic modules, with N the number of samples taken into account for the FFT processing. Each module is detailed into 3 parts, extracted from the FFT formula [RIV 08]. It is structured by a delay line, a Processing Unit and a delay line feedback. The implementation of these modules is analog and controlled by a digital part [RIV 07].

Here is depicted the example of a 16-sample FFT (Fig. 4). The input vector is x(1..N) and the output vector is X(1..N). The input vector is made of voltage samples, representing the discretized RF signal. The output vector is made of voltage samples, representing the spectrum of the RF signal.

The order of the output voltage is a base-4-reversed order. It means that the base-4 bits are inverted (Fig. 4). It determines the position of expected voltage samples. This order implies a decimation by 4 of the output frequency because 2 neighbor voltage samples X(i) and X(i+1) are thus spaced by 4^{n-1} voltage samples.

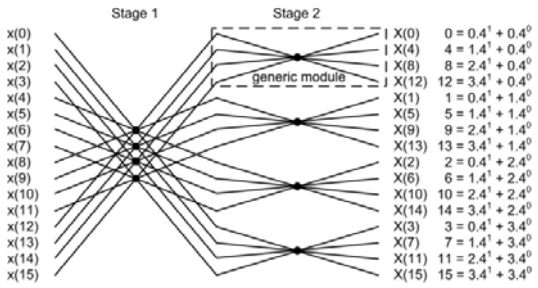


Figure 4: 16-sample FFT

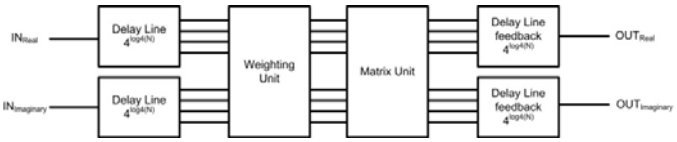


Figure 5: An analog implementation of the generic module

2.2 Analog Signal Processing

Analog signal processing induces errors based on technological mismatch such as in transistors sizing, current variations and others. The impact of these errors has to be taken into consideration. That is why two major sources of errors are identified in the Processing Unit (PU) composed by the Matrix Unit (MU) and the Weighting Unit (WU) implemented in the generic module (Fig. 5). To evaluate the influence of those errors, the normal law is used as a model for technological matching. Gaussian law is defined by its mean value and its standard deviation. Figure 2 depicts the standard deviation bounded by dynamic of error.

If σ is the standard variation and ε the induced error (percentage), it is reasonable to say that: $-3\sigma < \varepsilon < 3\sigma$

A standard deviation of 0.05 is considered. It implies errors under 15%. Figure 6 exhibits the distortion of a processed signal into 3 modules (named “stage”) of a 64-sample FFT. Errors are injected on each part of a stage, then each stage and finally on all the FFT circuit.

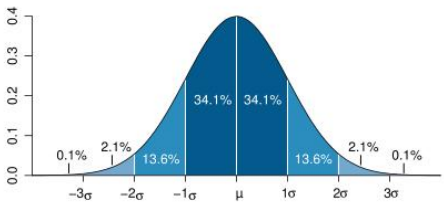


Figure 5: The normal law

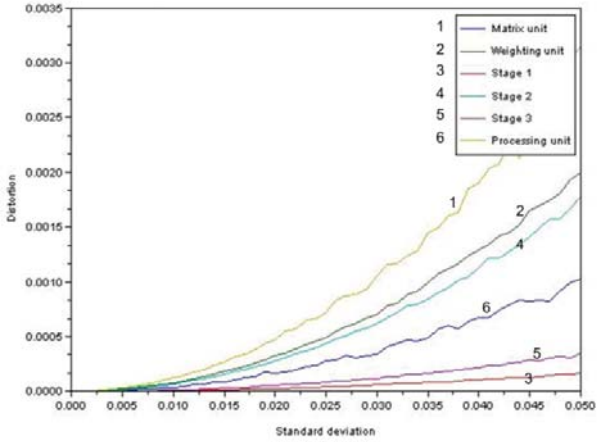


Figure 6: Distortion observed on a processed RF signal

It is easy to notice that the Processing Unit is the most sensitive part of the circuit. The Weighting Unit is the responsible for error introduction in the PU. The design of this module should be done carefully in order to limit the impact on all other parts of the circuit.

3. Analog Signal Processor Schematics

A SASP is designed to process 64 samples (N=64). 3 modules are pipelined. The schematic is ruled by specifications enforced by a differential structure with a dynamic range of 200mV at a working DC voltage of 800mV. The maximal frequency of this SASP is a sampling frequency of 1GHz. It can thus process RF signals from 0 to 500MHz. Analog building blocks are designed using the 65nm CMOS technology from STMicroelectonics to realize the analog basic operations of the processor [RIV 07]. One of them is an analog discrete multiplication of an RF discrete signal with coefficients within the interval [0,1]. It aims at windowing the RF signal with a Hamming window, depicted by the equation $W(t) = 0.54 + 0.46.\cos(2\pi.t/T_p)$.

The principle used here is a voltage/current/voltage conversion. Voltage samples are sent on gates of a series of transistors (Fig. 7). The current crossing each transistor is proportional to the input voltage equal to V_{gs}. A switch network (S_x) selects the input voltage of each transistor gate (Fig. 7). The input voltage can be either a voltage sample to be weighted or the DC reference voltage (800mV). The voltage seen at the drain of the transistors is proportional to the sum of the input voltages. The configuration of the switch network determines the coefficient to be applied. Every transistor (M_x) has a different size (Fig. 7). It implies that the current crossing each transistor is no more proportional to the input voltage but to the width of the transistor. Switches are controlled by a hard-implemented digital circuit which is synchronized on the SASP main clock. A simulation is depicted with a constant input voltage of 100mV peak to peak (Fig. 8). The Hamming window can be noticed at the ouput, as the multiplication with a constant signal and amplifying the signal by a factor of 1.45.

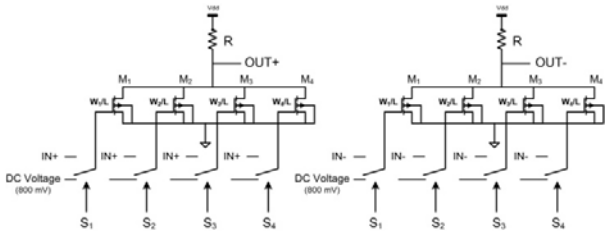


Figure 7: Analog discrete multiplication circuit

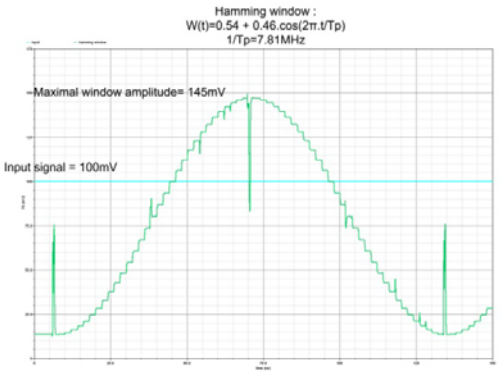


Figure 8: Hamming window simulation

4. Modeling results

A Post Layout Simulation is done to validate the 65nm CMOS technology circuit behavior (Fig. 9). The SASP carries out analog operations on voltage samples of a sine wave with a frequency equal to $7.25.f_{sampling}/64 = 56.64MHz$ ($f_{sampling}$ is equal to 500MHz). Only the 7th voltage samples processed by the processor is selected

and displayed. It contains the phase and amplitude information of the input signal. As expected, the spectrum is periodic every 4 processed FFT and output at a low frequency of 500/64=7.81MHz. The whole power consumption is 352mW and the die area is 0,96mm² as shown in a layout view (Fig. 10). An A/D conversion is thus possible at a lower frequency and further versions of the SASP with 4096, 16384, 65536 samples can be envisaged to process RF signals such as GSM or WiFi.

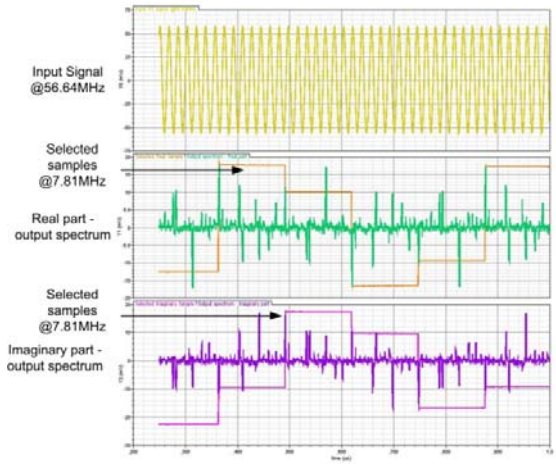


Figure 9: A Post Layout Simulation

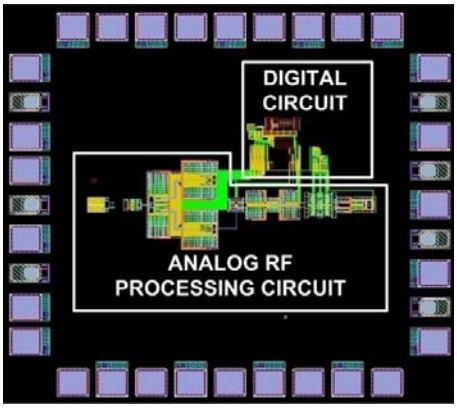


Figure 10: SASP Simplified Layout

5. Conclusion

This paper exhibited the principle, behavior simulations, schematics and layout design of a 65nm CMOS analog processor dedicated to RF applications in order to address the concept of Software Radio. It validates the principle of calculation based on analog voltage samples at RF frequencies. The SASP aimed at being part of the solution to access to a fully reconfigurable mobile terminal working on the Cognitive Radio approach in the radio spectrum range from 0 to 5GHz.

6. Reference

[MIT 95] J. MITOLA. **The Software Radio Architecture.** In: IEEE Communications Magazine, vol 33, May 1995, pp 26-38.

[ABI 06] A.A. ABIDI. **Software-defined radio receiver: dream to reality**. In: IEEE Communications Magazine, 2006, pp. 111-118.

[SWA 84] E. SWARTZLANDER, W. YOUNG, and S. JOSEPH. **A radix 4 delay commutator for fast fourier transform processor implementation.** In: IEEE Journal of Solid-State Circuits, vol.19, pp. 702-709, Oct. 1984.

[RIV 08] F. RIVET, Y. DEVAL, D. DALLET, JB. BEGUERET, P. CATHELIN and D. BELOT. **Disruptive Receiver Architecture Dedicated To Software Defined Radio.** In: IEEE TCAS-II, to be published, April 2008.

[RIV 07] F. RIVET, Y. DEVAL, D. DALLET, JB. BEGUERET, P. CATHELIN and D. BELOT. **A Disruptive Software-Defined Radio Receiver Architecture Based on Sampled Analog Signal Processing**. In: Proceedings IEEE RFIC, Honolulu, USA, June 2007, pp 197-200.

NETWORKS-ON-CHIP

Functional Test of Networks-on-Chip: Test of Routers

[2]Pedro Almeida, [1]Marcelo Lubaszewski, [2]Marcos Hervé,
[2]Fernanda Lima Kastensmidt, [2]Érika Cota

{prvalmeida, fglima, erika}@inf.ufrgs.br, luba@ece.ufrgs.br, marcos.barcellos@ufrgs.br

[1]Electrical Engineering Department, [2]Computer Science Department
Federal University of Rio Grande do Sul, Porto Alegre, Brazil

Abstract

In this work, a functional-based test method is presented for the test of Network-on-Chip routers. The proposed approach is scalable to any size of network. Experimental results show that fault coverage can reach up 92.75% of router stuck-at faults, with yet affordable test sequence lengths.

1. Introduction

The advent of system-on-chip (SoC) technology and integration of multiple cores and Network-on-Chip (NoC) on a single die brought new challenges in terms of testability [BEN 02, GEN 07, ANG 06]. Defining an efficient test strategy of such a new kind of system architecture is still a challenge to be overcome by the research community. It is well-known that the test of NoC-based multicore chips is generally divided on the test of cores and on the test of the communication infrastructure (network). The test of cores is usually based on the reuse of the NoC as Test Access Mechanism (TAM) to reduce area overhead and test time [COT 04]. As a result, the testing of the communication infrastructure is essential to guarantee the reliability of the entire system.

This work focuses on the functional testing of the NoC infrastructure. An improved version of a functional-based method originally proposed for the detection of interconnect shorts is used to detect stuck-at faults in the routers. Then, new configurations are used to reach maximum coverage. A functional-based approach is preferred, to reduce NoC re-design costs and to provide at-speed testing. However, scan and BIST-based approaches may be required to enhance both fault coverage and test application time.

2. Test Strategy

Our test strategy is defined by the following steps:

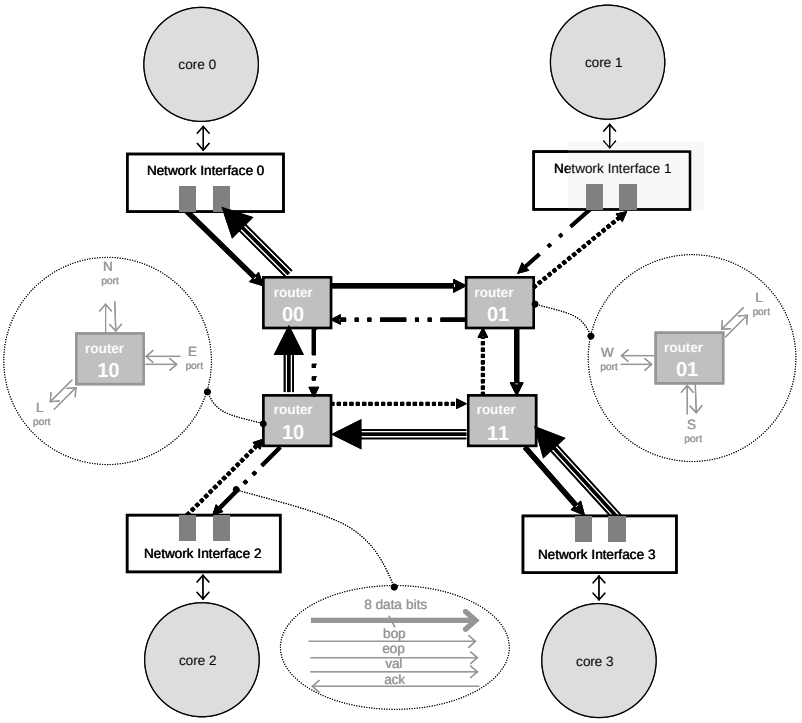
1) the interconnections are tested using an improved version of the functional-based method originally proposed in [COT 04] (many router faults are inherently detected in this step also);
2) additional test configurations are applied to the NoC in functional mode, such that the fault coverage of interconnects and routers is increased to the maximum affordable test application time and network interface area overhead;

The proposed method consists in detecting stuck-at faults in the routers of a 2x2 NoC by sending packets through the network (fig. 1(a)). To implement this test strategy, test patterns must be applied through and test responses collected from the network interfaces (NIs). The router is a VHDL soft core, parameterized in three dimensions: communication channel width, input buffer depth, and routing information width and was configured with five ports—north (N), south (S), west (W), east (E), and local connection between router and core (L)—a 3-position depth and a 10-bit width input buffer. Each communication channel actually has 12 bits: eight bits for data, two extra wires for framing (stored in the input buffer) called framing bits or beginning of packet (bop) and end of packet (eop) bits, and two signals for the communication handshake between routers (val and ack). After synthesizing the RASoC router according to these parameters, 959 logic stuck-at locations come out, resulting in a model that contains 1,918 faults, including stuck-at 0 and stuck-at 1 faults.

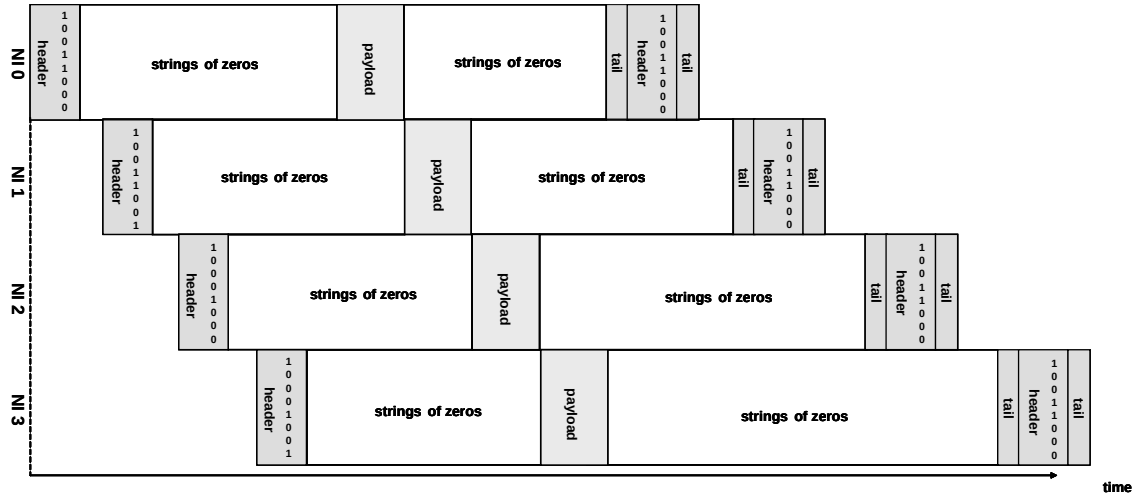
The packet payload must contain all test vectors necessary to detect the faults. Payload and time-out faults may occur. Without loss of generality, we will exemplify our test approach by using a walking-one sequence, and evaluate its coverage for the stuck-at fault models. This sequence can be easily generated by hardware or software and presents maximum fault detectability of shorts. Figure 1(a) illustrates the application of the test sequence in the basic NoC topology. Considering the XY routing strategy, this means that 4 packets can be sent across the 2x2 mesh network. Since all network interfaces use the same test sequence (48 vectors to fill all four 12-bit channels), the payload containing the test sequence should be shifted in time and channels should be filled in with strings of zeros to ensure the detection capabilities of the walking-one sequence (Fig. 1(b)). The second packet has only the header and tail flits, since all other faults have been detected by the first packet. Extra strings of zeros must be added at the end of the first packet of each NI to guarantee that a single tail or header is traversing the network at a time.

In order to demonstrate the effectiveness of the proposed approach, a 2x2 SoCin NoC was implemented in VHDL together with a fault injection mechanism capable of inserting stuck-at faults between all pair of wires in the design (data, framing –bop and eop- and handshake bits). Test sequence generators and test response analyzers were implemented as testbenches. We note that although our experiments were performed for the Walking One Sequence, similar analysis can be performed for any test sequence generator and its corresponding analyzer, according to the defined fault model. The simulations were all performed using the ModelSim tool, where internal signals can be connected and disconnected using the command signal spy *force* and *release* in the testbench file. The entire simulation time takes only few minutes.

Analyzing fig. 1(a), we note that two ports of each router are not exercised (N and W in router 00, for example). To solve this problem, we expanded the network and the number of test rounds, as shown in fig. (2). Note that we just implemented a set of basic test configurations running in parallel. Nine test configurations are applied in four test rounds. It can be shown that, for 3x3 and larger NoCs, the number of test rounds is kept constant and equals 4.

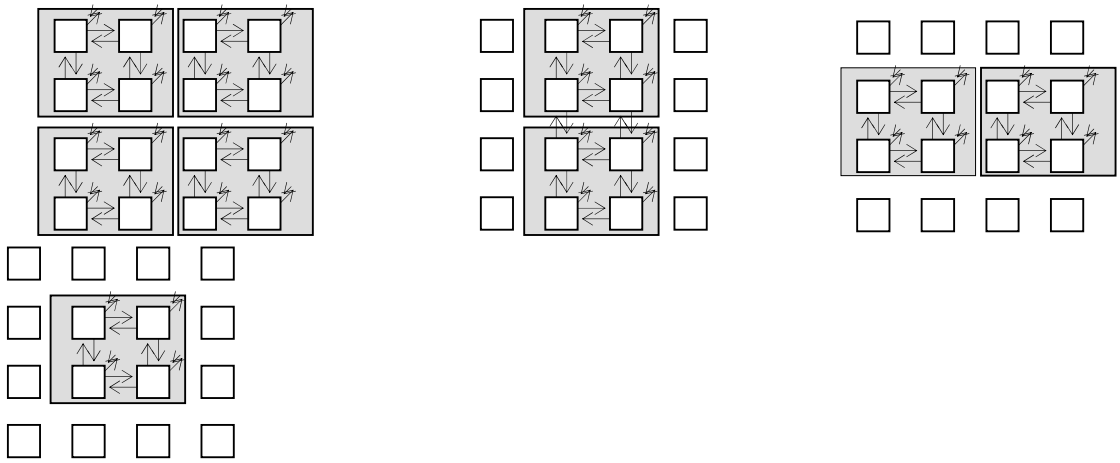


(a) Test strategy for the basic NoC topology



(b) Application of the shifted in time test sequence to the 2x2 NoC.

Figure 1 - Interconnect testing approach

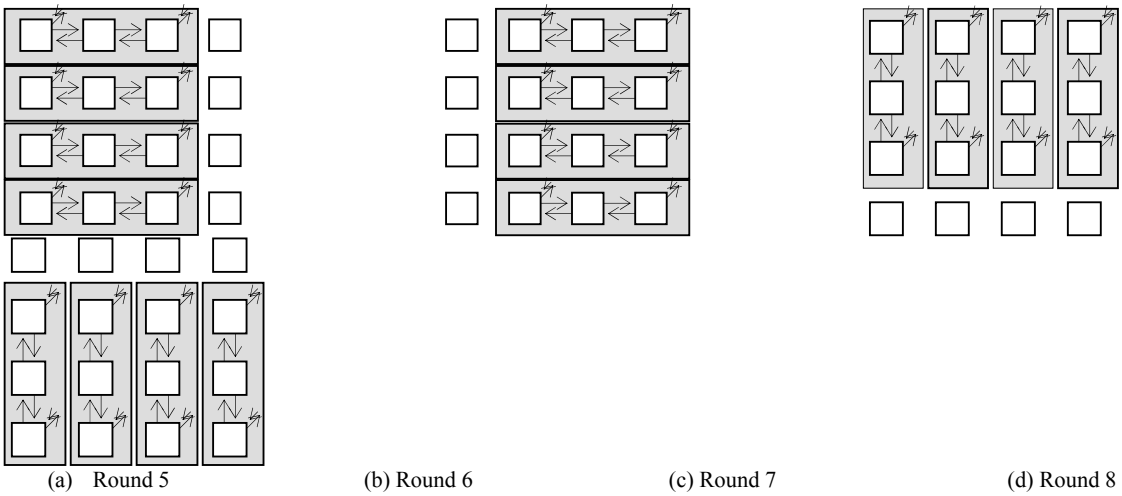


(a)Round 1 (b) Round 2 (c) Round 3 (d) Round 4

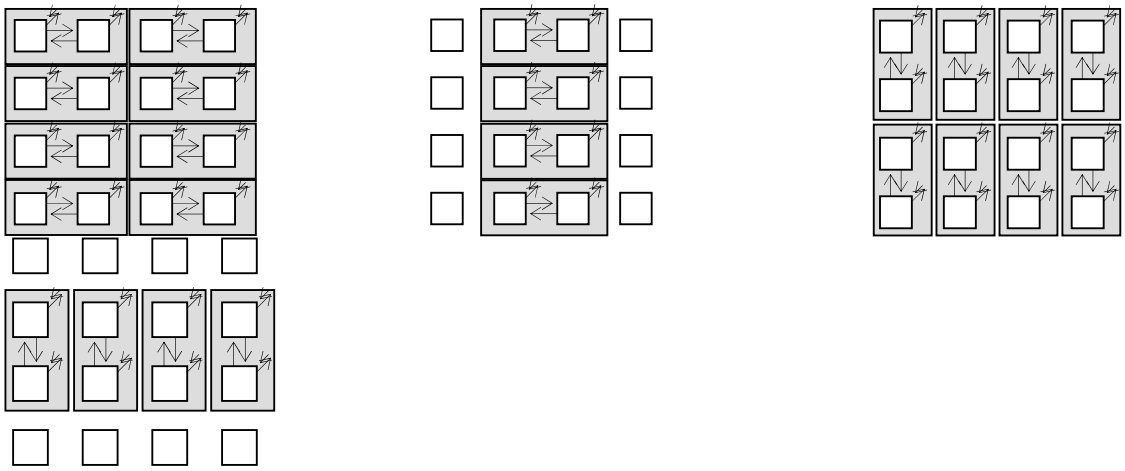
Fig. 2 - The test rounds and respective interconnect test configurations for a 4x4 mesh NoC.

3. Experimental Results

First, we wanted to find out the fault coverage that the same sequence for testing interconnects was able to achieve considering the router internal faults. Then, stuck-at faults were exhaustively injected in all flip-flops and gates of the central router of the 3x3 NoC under test, while the 4 rounds of the interconnect test sequence were applied. According to the first line of Table 1, 1,467-out-of-1,918 stuck-at faults were detected, resulting in a 76.49% fault coverage. As expected, many FIFO stuck-at 0 faults remained undetected due to the dominance of zeros in the applied sequence. However, by adding a test packet with three all-1 flits to the time-shifted walking-one sequence, all FIFO stuck-at faults could be detected. Note that no additional test round was needed and the test sequence length was just slightly increased. As pointed out in line 2, the router fault coverage increased to 78.05%, but 421 faults still remained undetected.



(a) Round 5 (b) Round 6 (c) Round 7 (d) Round 8



(e) Round 9 (f) Round 10 (g) Round 11 (h) Round 12

Figure 3 - Additional test rounds and respective routing logic test configurations for a 4x4 mesh NoC.

To further increase the fault coverage, new test rounds (fig.3) that exercise other router functionalities are needed, some to test the routing logic and others to test the arbitration logic. Firstly, rounds 9-11 were applied using a test packet with one all-0 and one all-1 flit and only the 421 undetected faults were injected in the central router of the 3x3 NoC. For each round, 4 clock cycles (header + 2 flits payload + tail) plus a latency of 3 cycles are needed to apply such short sequence. According to line 3 of Table 1, 189 new faults were detected and the total fault coverage increased to 87.9%. Still, 232 faults remained undetected. Finally, the arbitration logic test rounds were applied using data packets with different payloads (one all-0 flit, one all-1 flit, one half-0+half-1 flit and one half-1+half-0). According to line 4, applying all these rounds to the 3x3 NoC, additional 93 faults are detected and the fault coverage increases to 92.75% after the application of a 1486 clock cycles test sequence. Finally, 139 faults remained undetected.

Tab. 1. - Test Fault Coverage Results Analyzed by Simulation-induced Fault Injection Techniques

Experiment Setups	**Test sequence length (# clock cycles)**	**# detected faults (% fault coverage)**
Router testing: faults in the logic of the central router of a 3x3 NoC (1,918 injected faults)		
Time-Shifted Walking-One Sequence (Rounds 1, 2, 3 and 4) Stuck-at faults	**1,144**	1,467 (76.49%)
FIFOs Test Sequence : faults in the logic of the central router of the 3x3 NoC (1,918 injected faults)		
Time-Shifted Walking-One followed by Test packet with 3 All-1 flits (Rounds 1, 2, 3 and 4) Stuck-at faults	**1,164**	1,497 (78.05%)
Faults in the logic of the central router of the 3x3 NoC (421 injected faults)		
Test packet with 1 All-0 and 1 All-1 flit (Rounds 5, 7, 9, 10, 11 and 12) ***Stuck-at faults***	**28**	189 (9.85%)
Arbitration Logic Test Rounds: faults in the logic of the central router of the 3x3 NoC (232 injected faults)		
All-0, All-1, Half-0+Half-1 and Half-1+Half-0 flits ***Stuck-at faults***	**294**	93 (4.85%)

4. Conclusions and Future Works

The achieved fault coverage can still be increased by adding to the test sequence other non-covered network functional configurations. Many arbitration possibilities were not applied in our approach, for example. However, as it can be concluded by comparing lines 3 and 4 of Table 1, insisting in a purely functional test approach will require an ever increasing number of test configurations and rounds, for an ever decreasing number of detected faults. The tradeoff seems to be not in favor of pushing further a functional approach. The 139 undetected faults are located in the router control logic. From the point of view of a functional test, for example, about 10% of those faults are undetectable because they are related to unreachable control states, such as invalid routing (N→E, N→W, S→E and S→W) or non-valid X and Y header addresses for a given router location in the network topology. Then, considering that few registers with few bits are used to implement the router control logic, it is definitely worthy looking at the implementation of short scan chains to further improve

the router fault coverage. Since FIFOs are fully covered by the functional test proposed here, the scan costs related to silicon overhead, test data volume and test application time reported in [AMO 05] will be greatly reduced.

5. References

[BEN 02] Benini, L.; De Micheli, G.; "Networks on Chips: A New SoC Paradigm, IEEE Comp. Vol.35 no.1, Jan. 2002.

[GEN 07] Genko, N.; Atienza, D.; De Micheli, G.; Benini, L., NoC Emulation: A Tool and Design Flow for MPSoC. IEEE Circuits and Systems Magazine, Volume 7, Number 4, Fourth Quarter 2007, pp. 42-51.

[ANG 06] Angiolini, F.; Meloni, P.; Carta, S.; Benini, L.; Raffo, L.; Contrasting a NoC and a Traditional Interconnect Fabric with Layout Awareness, Design, Automation and Test in Europe, DATE, 2006, pp. 1 - 6

[COT 04] Cota, E.; Carro, L.; Lubaszewski, M., Reusing an on-chip network for the test of core-based systems, ACM Transactions on Design Automation of Electronic Systems, TODAES, v.9 n.4, 2004, pp.471-499.

[COT 07] Cota, E.; Kastensmidt, F. L.; Cassel, M., Meirelles, P., Amory, A., Lubaszewski, M., Redefining and Testing Interconnect Faults in Mesh NoCs. International Test Conference (ITC), 2007.

[GRE 05] Grecu, C.; Pande, P.; Baosheng Wang; Ivanov, A.; Saleh, R.; Methodologies and algorithms for testing switch-based NoC interconnects, 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT, 2005, pp. 238 – 246.

[AMO 05] Amory, A. M.; Briao, E.; Cota, E.; Lubaszewski, M.; Moraes, F.G.; A scalable test strategy for network-on-chip routers, IEEE International Test Conference, ITC, 2005. 9 pp.

[GRE 06] Grecu, C.; Pande, P.; Ivanov, A.; Saleh, R.; BIST for network-on-chip interconnect infrastructures, 24th IEEE VLSI Test Symposium, VTS, 2006, 2006, 6 pp.

[ZEF 03] Zeferino, C.A.; Susin, A.A.; SoCIN: a parametric and scalbale network-on-chip, 16th Smposium on Integrated Circuits and Systems Design, SBCCI, 2003, pp.169 – 174.

[ZEF 04] Zeferino, C.A.; Kreutz, M.E.; Susin, A.A.; RASoC: a router soft-core for networks-on-chip, Design, Automation and Test in Europe Conference, DATE, 2004, pp.198 – 203.

Soft Cores for Performance Evaluation of NoCs in FPGA

Thiago Felski Pereira, Cesar Albenes Zeferino
{felski, zeferino}@univali.br

Universidade do Vale do Itajaí
Centro de Ciências Tecnológicas da Terra e do Mar
Grupo de Sistemas Embarcados e Distribuídos

Abstract

The use of an application in a real system model, with processors, memory and input/output devices to evaluate and validate a project of NoCs (Networks-on-Chip) is complex and time costly. Alternatively, they can be used traffic generator models which inject traffic at specific rates and for pre-defined destinations. Such approach allows to verify the correctness of a network design, and, furthermore, to extract performance metrics. In this paper, it is presented the design of synthesizable soft-cores of a traffic generator and a traffic meter which will be used in a platform for performance evaluation of NoCs in FPGA.

1. Introduction

The increase in the integration level of silicon components has enabled the building of complex systems on a single chip with multiple processors, memories, peripherals and input/output controllers. Such systems are named System-on-Chip (SoCs). In order to deal with the complexity in the design of SoCs, designers reuse pre-defined and pre-verified hardware blocks, which are called cores [GUP 97].

The interconnection between cores of a SoC is usually done by means of buses. However, future SoCs, with dozens of cores, will demand communication architectures with better performance. In this scenario, Networks-on-Chip – NoCs [JAN 03] emerge as a solution for intra-chip interconnection. They provide scalable performance and parallelism in communication, meeting the requirements identified for future SoCs.

The design of a NoC involves several steps, like specification, architectural design, implementation, validation and performance analysis. The last two steps can be done in models of real systems or by using emulating the traffic profile of a given application.

Traffic generators are often used in the evaluation and validation of simulation models. However, thet also have applicability in the physical validation, and in measuring performance. To accomplish with this, it is necessary that they could be synthesizable on chip. In this direction, two implementation approaches can be used: (*i*) software-based (a code running on an embedded GPP – General Purpose Processor); or (*ii*) hardware-based (a non-programmable SPP – Single Purpose Processor). The same approaches can be used to implement a core responsible to collect performance metrics (the traffic meter).

In this paper, we present the design of synthesizable hardware-based cores for traffic generation and measurement. Firstly, in Section 2, they are described some related works. After that, in Section 3, it is presented the proposed architecture for on-chip traffic evaluation. In Section 4, we present the design of a traffic generator and a traffic meter, and, since this is still an ongoing work, in Section 5, they are discussed current results regarding the implementation and validation of the traffic meter component. Concluding, in Section 6, they are presented some final remarks.

2. Related Works

In the literature, there is a few number of works describing synthesizable traffic generators for on-chip performance evaluation.

In [ZEF 03a], it is presented a synthesizable traffic generator designed to be used in the validation process of a NoC named SoCIN (SoC Interconnection Network) [ZEF 03a][ZEF 03b]. This traffic generator allows the configuration of multiple flows originated from a source node for multiple destination nodes. Each traffic generator (named TG) is composed by sub-modules (named STG – Simple TG), and each one is responsible for the generation of a single communication flow, which is configured by a set of parameters.

In [WOL 07], it is presented a platform for on-chip performance evaluation of a NoC. The traffic generator is implemented by software running on an ARM9 processor, which injects traffic into a FPGA, where a NoC is implemented. Only a single router is synthesized on this FPGA, and the full NoC is emulated in a sequential way. According to the authors, the proposed approach makes possible to perform the evaluation of large NoCs, since the network size is not constrained by the logic density of the FPGA. Also, the on-chip evaluation is from 80 to 300 times faster than the evaluation of a SystemC model running on a computer.

3. On-Chip Platform for Performance Evaluation

The proposed platform is derived from the one originally used to evaluate performance of SoCIN by using SystemC, as described in [ZEF 07]. It will be composed by synthesizable models of the following components: (*i*) ParIS (Parameterizable Interconnect Switch) router [ZEF 04]; (*ii*) TG (Traffic Generator); and (*iii*) TM (Traffic Meter). As it is shown in Fig. 1, a TG-TM pair is attached to a terminal of a router. TG injects and collects packets into/from the network, while TM collects data from the packets received by TG, including the information necessary to calculate performance metrics, such as the average latency and the traffic accepted by the network. A set composed by a ParIS router, a TG and a TM can be seen as system node.

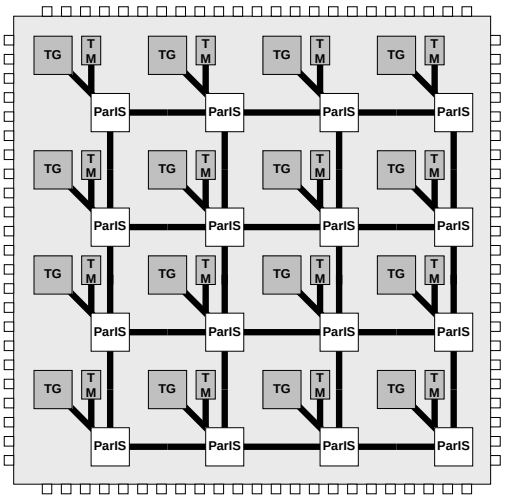


Fig. 1 – Platform for on-chip performance evaluation of NoCs

The platform will also include a centralized core (named Controller), which will be responsible to configure TG instances and collect data from TM instances. This core will have an interface for communication with a software (named Supervisor) running on a PC.

Supervisor will offer the following functionalities for the user: (*i*) configuration of parameters for the network and for the traffic; (*ii*) interface with EDA tools in order to compile the platform and synthesize it into the FPGA; (*iii*) configuration of TGs; (*iv*) control of experiments; (*v*) gathering of data collected by TMs; and (*vi*) analysis of experiments.

In Supervisor, traffic will be described by using the model proposed in [TED 05]. A bit stream will be generated for each TG, and all the bit streams will be downloaded to the FPGA by means the control core, which will send data by a chain of TGs. At the end of an experiment, data collected by TMs will be gathered and uploaded by Controller to Supervisor. TMs will also be interconnected to Controller by means of a chain.

4. Organization of TG and TM

4.1 Traffic Generator

When a source TG sends packets to a destination TG under a given traffic configuration, it is said that these packets are part of a *communication flow*. A TG can inject more than one communication flow into the network, for a same or for different destinations. A flow is described by a set of parameters that defines its *descriptor word*, shown in Fig. 2. This configuration word is composed by 7 fields. The required bandwidth (or injection rate) field is 7-bit wide (for values from 1 to 100 %), and the other ones have parameterizable width. The second field is the network address of the destination node. The following three fields describe the number of packets to be sent, their size and the number of wait cycles between the end of a packet and the beginning of the next packet in the flow. Finally, for bursting flows, the last two fields describe the number of packets to be sent in the burst and the size of the last packet in the burst (which is smaller or equal to the packet size field).

Required Bandwidth	Destination Address	Number of Packets	Packet Size	Wait Cycles	Burst Size	Last Packet Length
7 bits	max = 8 bits	max = 20 bits	max = 10 bits	max = 17 bits	max = 10 bits	max = 10 bits

Fig. 2 – Descriptor word for communication flows

The organization of TG is shown in Fig. 3, where it is presented a simplified version of its datapath. It has a shift in/shift out interface used to build a chain of TGs, and by which Controller configures the traffic generators. It still has an input to receive the current value of a counter of clock cycles (named Global Cycle Counter – GCC), and an output channel to be attached to a network terminal. Internally, it is composed by Flow Generator (FG) blocks, one arbiter and one output multiplexer. Each FG has a Descriptor Register, an NPC Register and a comparator. The first one stores the descriptor word of the flow. The second one holds the ideal cycle to inject the next packet (named NPC – Next Packet Cycle). The third one compares NPC with the current value of GCC, and sets a request to the arbiter when the latter is equal or greater than the former. Arbiter receives requests from FGs and selects one of them by using a round-robin scheduler. The selected FG is connected to the output channel by the multiplexer and sends its packet (or burst of packets). Each packet includes information necessary for traffic evaluation: its required bandwidth (in the flit) and NPC value (in the last flit). Since NPC means the ideal cycle that the packet should have been sent, when a packet is sent after this cycle, it means that there was congestion at TG (or at the network) in the moment its creation.

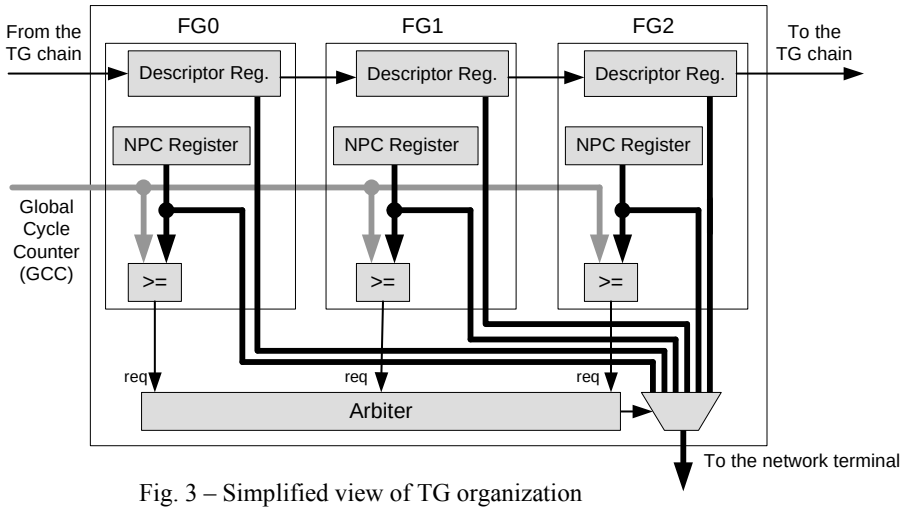


Fig. 3 – Simplified view of TG organization

The organization of TG is more complex than the one shown in Fig. 3. It still includes the control unit and functional units responsible to calculate NPC and to update the number of packets and flits to be sent .

4.2 Traffic Meter

Fig. 4 depicts the organization of TM, showing its data path. It has an interface to GCC, an input channel to collect data from the network terminal, and a shift in/shift out interface. This one is used to build a chain of TMs, and to send information to the Controller. Internally, it has three registers which share two functional units in order to calculate the following information: (*i*) accumulated packet latency; (*ii*) number or received packets; and (*iii*) accumulated required bandwidth. Packet latency is determined by subtracting NPC (extracted from the last payload flit) from the current value of GCC. This information can be used by Supervisor to determine the average required latency and the average required bandwidth (the offered load).

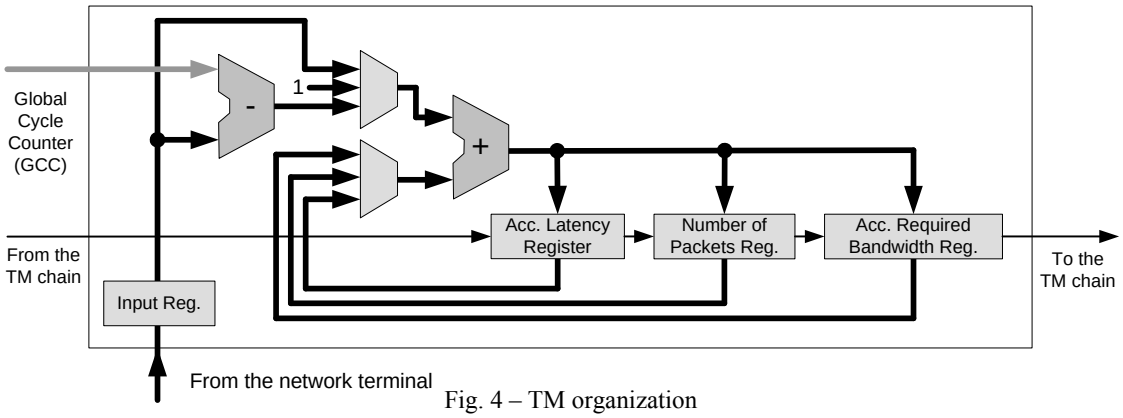


Fig. 4 – TM organization

5. Results

This work is part of an ongoing project. The VHDL model of TM is already implemented and validated by simulation. Fig. 5 presents a waveform that shows two packets being received at the input channel of TM. The

first packet (cycles 15 to 18) is 4-flit long and includes information about the required bandwidth (30) and NPC (5), as can be seen at *in_data* channel. As long as the packet is received, registers (0 to 2) are updated to accumulate the required bandwidth (30), the packet latency (14 = 19-5) and the number of packets already received (1), respectively. The second packet (cycles 23 to 26) requires 20% of the channel bandwidth and its NPC equals 10. After it is received, registers equal 50 (= 30+20), 31 (= 14 + (27-10)) and 2. Considering these two packets, average latency equals 15,5 cycles and average required bandwidth equals 25%.

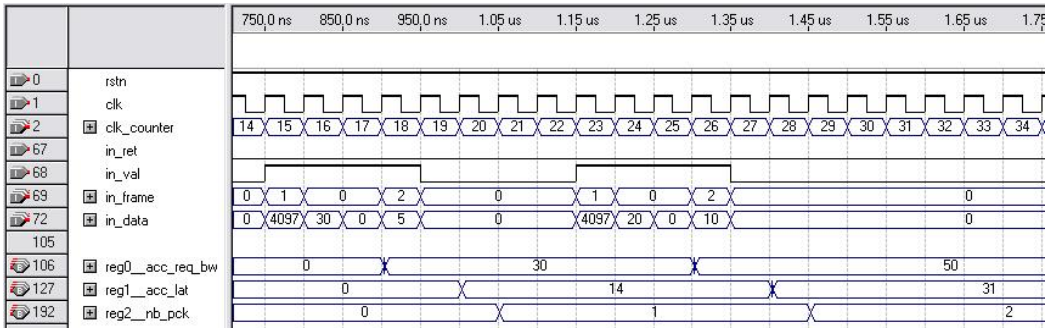


Fig. 5 – Simulation of TM operation

TM VHDL model is totally parameterized. Their costs depend on the size of its channels, functional units and registers. In a standard configuration, multiplexers and functional units are 64-bit wide, the input channel is 32-bit wide, and they are spent 148 bits in registers. For this configuration TM operates at 107,5 MHz and spends 392 LUTs and 149 flip-flops of a Cyclone II EP2C35F672C FPGA (less than 1% of the FPGA capacity).

6. Conclusion

In this paper, they were presented issues related with the design of a platform to be used for on-chip performance evaluation of NoCs in FPGA. They are being developed two soft-cores for traffic injection and measurement. The TM core is already modeled and validated, while the TG core is still being modeled in VHDL.

7. Acknowledgments

This work was supported by ProBIC research program of University of Vale do Itajaí – Univali.

8. References

[GUP 97] GUPTA, R. K.; ZORIAN, Y. Introducing core-based system design. IEEE Design & Test of Computers, [S.l.], v. 14, n. 4, p. 15-25, Oct.-Dec. 1997.

[JAN 03] JANTSCH, A.; TENHUNEN, H. (Eds.). Networks on Chip. Boston: Kluwer Academic Publishers, 2003. 303p.

[TED 05] TEDESCO, L. P. Uma Proposta para Geração de Tráfego e Avaliação de Desempenho para NoCs. Dissertação de Mestrado, PPGCC-PUCRS, Porto Alegre, 2005.

[WOL 07] WOLKOTTE, P. T.; HÖLZENSPIES, K. F.; SMIT, J. M. . Fast, accurate and detailed NoC simulations. In: INTERNATIONAL SYMPOSIUM ON NETWORKS-ON-CHIP (NOCS'07), 1., 2007. Proceedings… [S.l.: s.n.], 2007. p. 323-333.

[ZEF 03a] ZEFERINO, C. A., Redes-em-Chip: arquiteturas e modelos para avaliação de área e desempenho. Tese de Doutorado, PPGC-UFRGS, Porto Alegre, 2003.

[ZEF 03b] ZEFERINO, C. A., SUSIN, A. A. : SoCIN: A Parametric and Scalable Network-on-Chip. In: SBCCI, 16, 2003, São Paulo, IEEE CS Press, 2003. p. 168-174.

[ZEF 04] ZEFERINO, C. A., SANTO, F. G. M. E., SUSIN, A. A. "Paris: A Parameterizable Interconnect Switch for Networks-on-Chip", In: SBCCI, 17., 2004, Porto de Galinhas. Proceedings. ACM Press, 2004. p. 204-209.

[ZEF 07] ZEFERINO, C. A, BRUCH, J. V, PEREIRA, T. F, KREUTZ, M. E, SUSIN, A. A., "Avaliação de Desempenho de Redes em Chip Modelada em SystemC. In: WPERFORMANCE, 2007, Rio de Janeiro, SBC, 2007. p. 559-578.

A SystemC-based Environment for Performance Evaluation of Networks-on-Chip

Jaison Valmor Bruch, Rafael Luiz Cancian, Cesar Albenes Zeferino
{jaison, cancian, zeferino}@univali.br

Universidade do Vale do Itajaí
Centro de Ciências Tecnológicas da Terra e do Mar
Grupo de Sistemas Embarcados e Distribuídos

Abstract

Networks-on-Chips (NoCs) appear as the best alternative of communication architecture for future systems with dozens of cores integrated on a single chip. They are reusable, their performance scales with the system size, and they provide high parallelism in communication, requirements foreseen for such systems. The design space of NoCs is very large, and there are several architectural alternatives for implementation. In order to select the best configuration of a NoC for a given application, it is necessary to have tools which aid the designer in the evaluation of each configuration. In this context, this work presents an integrated environment based on SystemC which allows evaluating different configurations of a NoC by simulation at the register transfer level.

1. Introduction

One of the problems in the design of future multi-core Systems-on-Chip (SoCs), with dozens of cores, will rely on the providing communication performance compatible with the requirements of the multiple processing cores. As it has been largely discussed in the literature, bus-based architecture will no be able to deliver the communication performance required for future Systems-on-Chip (SoCs) with dozens of cores integrated onto a single chip [GUE 00]. As a solution for this problem, it is a consensus in scientific and industrial communities that on-chip switched networks will be the best communication architecture for futures SoCs, because they can meet the requirements of these systems. These architectures are largely refereed as Networks-on-Chip or NoCs.

A NoC can be described by its topology and by the techniques used for routing, switching, scheduling, buffering and flow control, each of them with several alternatives for implementation. Therefore, the design space of NoCs is very large, and finding the best configuration of parameters for a given application is a very hard task.

In this context, this paper presents an integrated environment that allows the performance evaluation of a NoC for different configurations. This environment, named X Gsim, is based on a RTL (Register Transfer Level) SystemC model of SoCIN (SoC Interconnection Network) [ZEF 03], and on a set of tools developed to automate tasks related to the simulation of SoCIN network operation under different traffic scenarios. It has a visual interface, which was developed in order to ease the configuration and the execution of experiments for performance evaluation.

This paper is organized in six sections. Section 2 shortly describes a related work, while Section 3 presents the architecture of SoCIN. X Gsim environment is described in the fourth section, and its application in a set of experiments is illustrated in Section 5. Concluding, Section 6 presents the final remarks.

2. Related Works

Several studies about NoCs have been developed by different research groups around the world. Many of these works use some kind of simulation-based method to evaluate their proposals. In general, simulation models are built by using some HDL (Verilog or VHDL) and/or SystemC (at Transaction Level or RTL). However, there is a few number of simulation-based environment specially designed for NoCs described in the literature. One of them is presented in [MEL 07] and is named Atlas. Developed by researchers of PUCRS (Pontifícia Universidade Católica do Rio Grande do Sul, Brazil), it is an integrated environment with visual interface designed to automate several processes related with the design flow of Hermes NoC [MOR 03]. It integrates tools like: (i) *Network Generator*, for configuration and generation of VHDL models of Hermes; (*ii*) *Traffic Generator*, for configuration and generation of traffic patterns to be applied on the network; (*iii*) *NoC Simulation*, for simulation execution; and (*iv*) *Performance Evaluation*, for the analysis of results. For traffic generation, Atlas uses the model proposed by Leonel Tedesco [TED 05], which allows modeling different patterns of traffic and offers alternatives for spatial and temporal distributions.

3. SoCIN Architecture

SoCIN [ZEF 03] is a NoC developed at Federal University of Rio Grande do Sul (UFRGS). It uses a 2-D mesh topology, where each router is addressed by a pair of coordinates (see Fig. 1). The first router of SoCIN was named RASoC (Router Architecture for SoC). Current router is named ParIS (Parameterizable Interconnect Switch) and was developed at University of Vale do Itajaí (Univali) [ZEF 04]. It is a soft-core with several parameters which allow generating SoCIN instances based on different configuration of its design space.

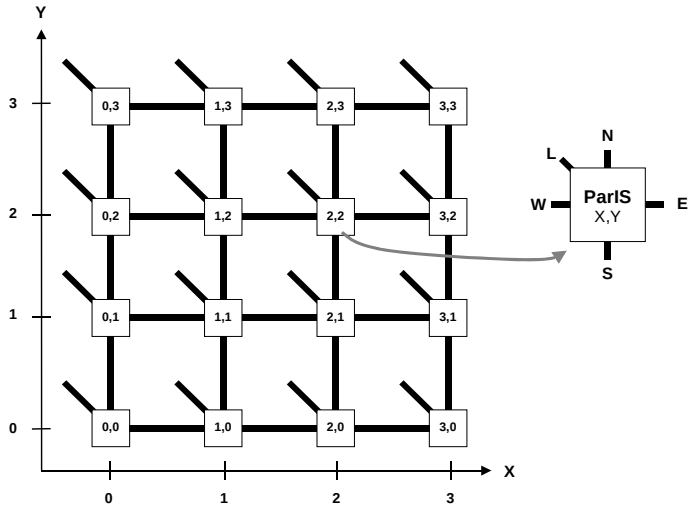


Fig. 1 – SoCIN network and ParIS router

As shown in Fig. 1, each router has up to 5 communication ports. One of them is intended to be used as a terminal to attach a core (the *Local* port, or *L*), and the other ones (*N*, *E*, *S* and *W*) are used for connection with neighbor routers. Communication among cores attached to the network terminals is done by the means of packets of unlimited length. To be routed along the network, each packet must include the destination coordinates in a header. This information is used by each router in the packet path to determine the communication port to be used to forward the packet.

4. X Gsim architecture

X Gsim software architecture is composed by a visual integrated environment and a set of tools, originally presented in [ZEF 07]. X Gsim works with a SoC platform based on SystemC models of a router (ParIS), a traffic generator (named TG) and a traffic meter (named TM). ParIS router is modeled in SystemC RTL, and TG and TM are modeled in SystemC TL. A pair of TG-TM is attached at each network terminal and is responsible to inject packets into the network and to collect data for performance evaluation. X Gsim does the interface between user and its integrated tools, which are summarized below:

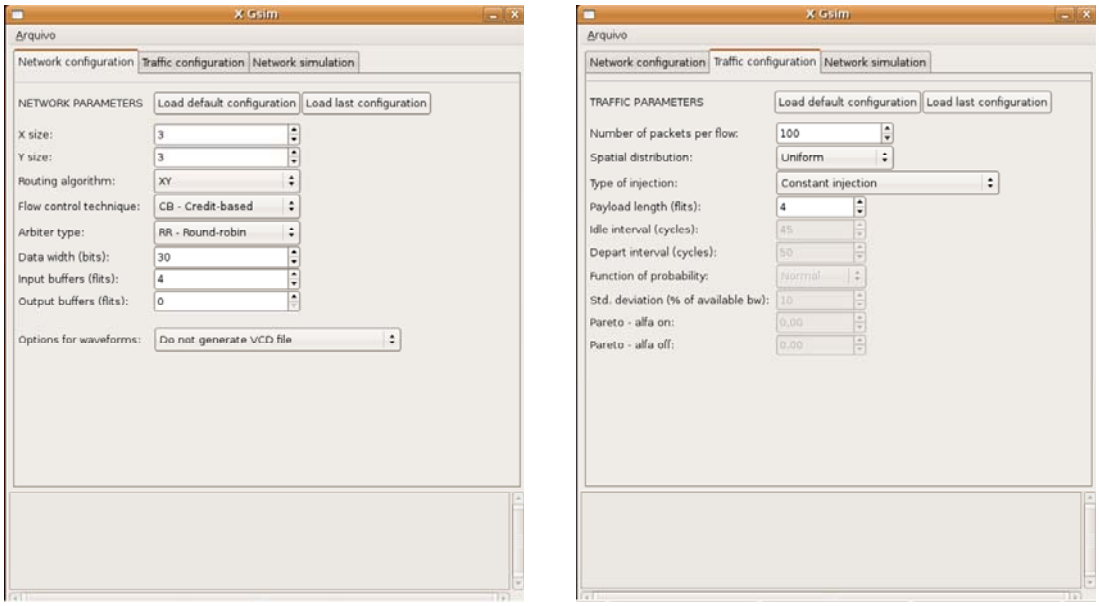
- GNoC: it generates a SystemC RTL model of SoCIN (a network of ParIS instances);
- GSoC: it generates a SystemC model of a SoC composed by SoCIN and instances of TM and TG;
- GCC (C compiler): it is invoked to generate a system simulator based on the models generated by previous tools;
- GTR: it configures the traffic pattern to be emulated by the network generators at the moment of a simulation experiment;
- ATR: it analyses data collected by traffic meters and generates summary files used by X Gsim to show the performance metrics results to the user;
- GNUPLOT: a GNU tool used to show the simulation results in a graphical format;
- GTKWAVE: a GNU tool used to show waveforms with the packets sent and received at the network terminals.

For traffic modeling, the same model used in Atlas [TED 05] is applied. The major parameters for traffic generation used by GTR tool include: (*i*) bandwidth required by the flows; (*ii*) number of packets to be sent by each flow; (*iii*) spatial distribution used to define the destination of each flow; and (*iv*) type of injection rate. They are available the following spatial distributions: uniform, non-uniform, local, complement, matrix transpose, perfect shuffle, and bit-reversal. The injection rate can be constant or variable. The last one includes five alternatives based on the variation of packet size, interval between packets, interval between arrivals, and

burst size. For variable injection rate, one can use one of three probability distributions: Normal, Exponential or Pareto On-Off.

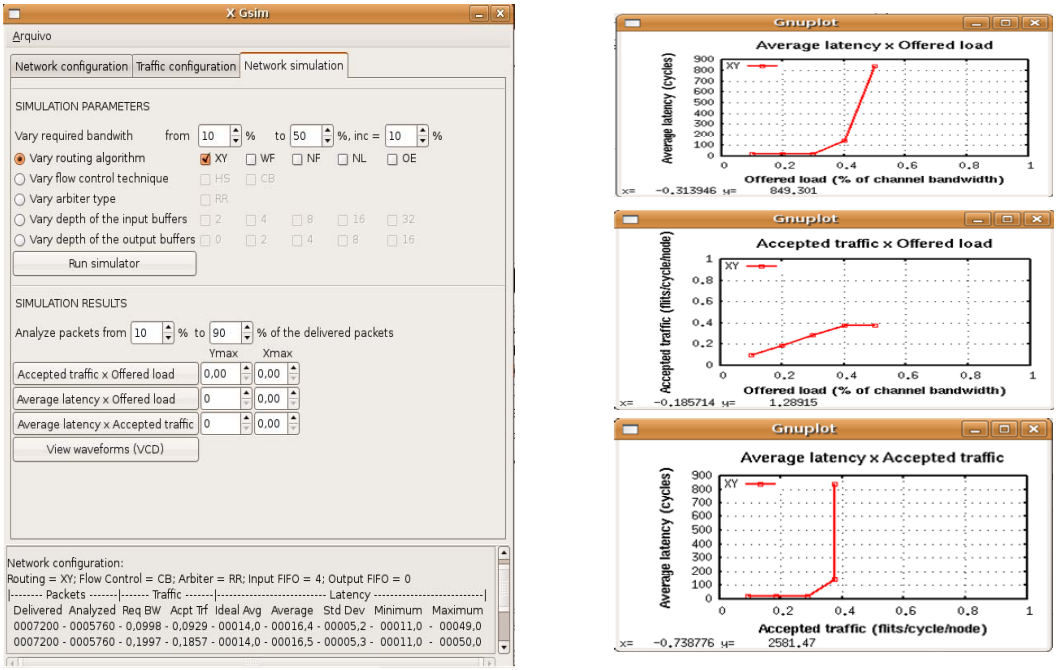
In the development of X Gsim interface, they were used GTK+ and GLADE, a RAD (Rapid Application Development) framework for the development of visual interfaces based on GTK+ and GNOME. X Gsim was built to run over Linux systems.

X Gsim interface is organized in three tabs: *Network configuration* (Fig. 2.a), *Traffic configuration* (Fig. 2.b) and *Network simulation* (Fig. 3.a). The first two are used to enter the network and traffic parameters, respectively. The third one allow to configure a single or a multiple simulation experiment in order to evaluate performance of a single or multiple parameters set under one or more required bandwidths (see Fig 3.a). After simulation, the results can be seen in a text table presented in the message window (bottom area in Fig. 3.a), or in a graphical format (Fig. 3.b).



(a) (b)

Fig. 2 – X Gsim interface: (a) *Network configuration* tab; (b) *Traffic configuration* tab.

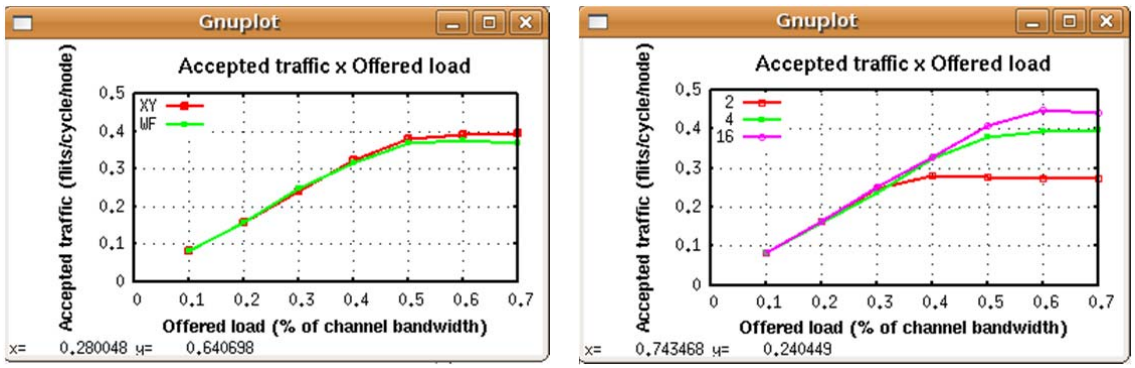


(a) (b)

Fig. 3 – X Gsim: (a) *Network simulation* tab; (b) *graphical view*

5. Using X Gsim in NoC evaluation

This section presents examples of usage of X Gsim to compare the performance of different networks configurations. For the illustrated experiments bellow, the following parameters were fixed: (*i*) network size: 4 × 4; (*ii*) flit width: 30 bits; (*iii*) arbiter type: round-robin (RR); (*iv*) flow control technique: credit-based (CB); (*v*) packet length: 16 *flits*; (*vi*) spatial distribution: non-uniform, (*vii*) traffic type: constant; (*viii*) required bandwidth: ranging from 10 to 100% of channel bandwidth (with increment of 10%); (*ix*) number of packets per flow: 1000. In the first experiment, whose results are summarized in Fig. 4.a, we set the input and output buffers depth to 4 and 0 flits, respectively and varied the routing algorithm (XY and WF). Results show that, for the selected network and traffic configurations, XY algorithm is able to accept more traffic than WF algorithm. This occurs because WF is a partially adaptive algorithm and tends to concentrate traffic in the center of the network. In the second experiment, we set the routing algorithm to XY, and the output buffers depth to 0 (no buffer). Three input buffer depths were compared: 2, 4 and 16 flits. Results are summarized in Fig 4.b and show that, for the selected configurations, as deeper the input buffers are, more traffic is accepted by the network.



(a) (b)

Fig. 4 – Experiments comparing: (a) XY and WF routing algorithms; and (b) 2-, 4-, and 16-flit buffers.

6. Conclusions

This paper presented an integrated environment for performance evaluation of NoCs by simulation using SystemC. This environment improved the usability of tools it integrates, because they were originally invoked by command line. Also, it automated the execution of multiple experiments. Compared to Atlas, X Gsim is different in the sense that simulation is done by using only SystemC models. In Atlas, the network is modeled in VHDL and the other components in SystemC. X Gsim has a major limitation regarding the lack of an interface to edit each flow independently from the others, and this feature will be added in a future work.

7. Acknowledgments

This work was supported by Art.170 research program of Santa Catarina State Governement.

8. References

[GUE 00] GUERRIER, P.; GREINER, A. A generic architecture for on-chip packet-switched interconnections. In: DATE, 2000, Paris., IEEE CS Press, 2000. p. 250-256.

[MEL 07] MELLO, Aline V. de. Qualidade de serviço em redes intra-chip: implementação e avaliação sobre a rede Hermes. Dissertação de Mestrado, PPGCC-PUCRS, Porto Alegre, 2007.

[MOR 03] MORAES, F. et al. A low area overhead packet-switched Network-on-Chip: architecture and prototyping. In: IFIP WG 10.5 VLSI-SOC, 2003, Darmstadt, 2003. p. 318-323.

[TED 05] TEDESCO, L. P. Uma Proposta para Geração de Tráfego e Avaliação de Desempenho para NoCs, Dissertação de Mestrado, PPGCC-PUCRS, Porto Alegre, 2005.

[ZEF 03] ZEFERINO, C. A., SUSIN, A. A. : SoCIN: A Parametric and Scalable Network-on-Chip. In: SBCCI, 16, 2003, São Paulo, IEEE CS Press, 2003. p. 168-174.

[ZEF 04] ZEFERINO, C. A., SANTO, F. G. M. E., SUSIN, A. A. "Paris: A Parameterizable Interconnect Switch for Networks-on-Chip", In: SBCCI, 17., 2004, Porto de Galinhas. Proceedings. ACM Press, 2004. p. 204-209.

[ZEF 07] ZEFERINO, C. A, BRUCH, J. V, PEREIRA, T. F, KREUTZ, M. E, SUSIN, A. A., "Avaliação de Desempenho de Redes em Chip Modelada em SystemC. In: WPERFORMANCE, 2007, Rio de Janeiro, SBC, 2007. p. 559-578.

Automatic Code Generation for Embedded Applications from Verified Alloy Models

Ronaldo Ferreira, Emilena Specht, Lisane Brisolara, Érika Cota, Luigi Carro
{rrferreira, emilenas, lisane, erika, carro}@inf.ufrgs.br

Embedded Systems Lab – Informatics Institute
Federal University of Rio Grande do Sul (UFRGS) – Porto Alegre - Brazil

Abstract

Software development for embedded systems has to cope with the hard-constrained nature of these sorts of devices. Non-functional requirements, such as memory footprint, energy consumption, target platform and mainly, time-to-market, are essential in this specific software domain. Due to the embedded software development complexity, and time-to-market pressure windows, techniques that address the gap between modeling and code generation in an abstract fashion are needed. To address this problem, we propose an approach for automatic code generation from verified Alloy models. Applying such technique, we aim the overall time-to-market reduction, eliminating the coding and testing phases of the development cycle.

1. Introduction

Embedded systems are everywhere in the people's daily life. According to Nokia [NOK 05], by 2009 almost half of the world population is going to own a cell phone. This sort of systems is characterized by the offering of computing services not as its main activity. Some examples are cell phones, microwave ovens, DVD and MP3 players. Embedded systems are also characterized by its hard-constrained nature of resources, such as memory footprint, energy consumption and, mainly, time-to-market. In order to properly cope with such requirements and to obtain high quality products in the expected time window, it is necessary to increase the design abstraction level and to formally verify its properties.

Embedded systems may be life-critical, where reliability and safety are more important than performance and embedded applications are increasingly being composed of software portions. [GRA 03] points out that European embedded industry employs only general Software Engineering practices – even UML modeling was not widely adopted, they use it just for document illustration purposes. Only one company employed formal methods on a very critical application. In this way, our approach aims to be easily adopted by the reduction of time-to-market, while bringing more quality to the applications and products.

There are some techniques for automatic code generation for the embedded domain, which are based on semi-formal UML models. [SCH 04] proposes an integrated environment where the structural portion of the application is modeled with class diagrams, and the behavior is modeled with StateCharts and a graphical formalism similar to sequence diagrams. This approach uses several formalisms to model the application, while ours uses only Alloy, being thus more orthogonal and completely formal.

ForSyDe [SAN 04] is an environment for software and hardware for embedded domain. It uses Haskell as description language, where the application code is obtained through refinement. Although the adoption of a language with well defined semantics, one of the know problem to the adoption of formal techniques by the embedded community is the learning and adoption effort of such techniques. Our approach, in addition of being more abstract than Haskell – this language contains explicit structures, offers a relational objected oriented syntax, while offering optional design space exploration of the data structure that better fits a given application.

We propose the automatic code generation targeting embedded applications from verified Alloy [JAC 06] models, and present the transformation rules to Java code. The proposed approach provides software automation, high level of abstraction and formal verification. We have chosen Alloy as the formal method due to its object-oriented syntax, so reducing the initial learning effort. In addition, Alloy uses the *Kodkod* [TOR 06] engine to simplify the SAT solving input, reducing the analysis time and the state explosion problem.

This paper is organized as follows: section two briefly introduces the Alloy modeling and the study case adopted to present the approach; section three shows the translation rules between Alloy and Java through the use of a study case; section four presents the results obtained with the code generation process for three applications; and, finally, section five draws conclusions and future works.

2. Alloy

This section briefly presents the Alloy formal method through the use of a drink vending machine study case. This machine has the following requirements: i) there are three kinds of drinks: water, tea and soft drink; ii) every drink have a price; iii) the machine works by inserting coins in it; iv) a sell is only completed when the needed amount of coins was previously inserted; v) the machine does not sell drinks whose storage is empty.

The first step is the structural modeling, which is performed by the declaration of signatures.

```
1.  open util/ordering[State] as ord
2.  abstract sig Drink {price: Int}
3.  sig Water extends Drink {}
4.  sig Softdrink extends Drink {}
5.  sig Tea extends Drink {}
6.  sig Coin {}
7.  sig VendingMachine{
8.    coinStorage:   set Coin,
9.    drinkStorage:  set Drink
10. }
11. sig State { machine: VendingMachine }
```
(2.1)

After the structural modeling, the application properties are written; they hold for the entire Alloy model. These properties or axioms are declared as facts.

```
1.  fact InitialState {
2.    let s0 = ord/first.machine | no s0.coinStorage
3.    let s0 = ord/first.machine | all d:Drink | d in s0.drinkStorage
4.    some Tea and some Water and some Softdrink
5.  }
6.  fact DrinkBuying {
7.    all s: State, s': ord/next[s] | some d: Drink | some c: Coin |
8.      d.price <= #s.machine.coinStorage => buyDrink[s.machine,s'.machine,d]
9.        else addCoin[s.machine,s'.machine,c]
10. }
11. fact Prices {
12.   all w: Water      | w.price = 1
13.   all s: Softdrink  | s.price = 2
14.   all t: Tea        | t.price = 3
15. }
16. fact MaximunCoinAmount { #Coin = (sum d : Drink | d.price) }
17. fact AlwaysDoSomething{ all s:State,s': s.ord/next |
18.   s.machine.coinStorage != s'.machine.coinStorage }
19. fact AllMachineInSomeState{ all m:VendingMachine | some s: State |
20.   m in s.machine }
```
(2.2)

It is possible to build operations within the model, modeling the application's behavior, being done by the use of predicates and functions. These operations represent model constraints which may hold or not in the specified scope.

```
1.  pred buyDrink[m, m': VendingMachine, d: Drink] {
2.    m'.drinkStorage =  m.drinkStorage - d
3.    #m'.coinStorage  = #m.coinStorage  - d.price
4.  }
5.  pred addCoin[m, m': VendingMachine, c: Coin] {
6.    m'.coinStorage  = m.coinStorage + c
7.    m'.drinkStorage = m.drinkStorage
8.  }
```
(2.3)

After the structural, behavioral and properties modeling, it is possible to simulate the modeled application within a specified scope. This is done by calling a *run* statement. The defined scope creates a specified amount of atoms of every signature type declared, unless otherwise specified by any fact.

```
1.  run { no ord/last.machine.drinkStorage and no ord/last.machine.coinStorage }
2.    for 14
3.  run addCoin
4.  run buyDrink
```
(2.4)

For a detailed introduction to Alloy, as well as to its logic and more examples, see [JAC 06].

3. Translation of Alloy Models into Java Code

The tool for translating Alloy models into Java code, with the aid of the CUP and JFlex parser and scanner generators was coded in Java also. Classes and its attributes are extracted from the declared signatures. Signatures having fields (lines 7 to 9 – 2.1) are translated into classes encapsulating the fields as attributes; the empty ones are translated into the Java's String type. As there are no explicit data structures in Alloy, it is needed to create them in Java. In our approach, the designer selects the structure that will be generated in the Java code. This approach enables the choosing between *Vector, ArrayList, LinkedList, HashSet, TreeSet, HashMap, Hashtable, LinkedHashMap* and *AbstractCollection*.

Methods are extracted from the declared predicates and functions on the Alloy model. The ones extracted from predicates are typed as *void*; the ones extracted from functions are typed accordingly the function's type. Together with the generated methods, there is a static library which implements the Alloy's operations, in order to guarantee the modeled behavior on the Alloy model. The inference of method's class membership is done from its formal parameters list. As the state change modeling is performed by means of distinct atoms of the

same type, holding different data, there are always at least two such atoms in the method's parameters list (lines 1 and 5 – 2.3). By this way, we infer that such method is member of the atom's type class. If there is no such pattern in the parameters list, the method is created as a static one within the Main Java class.

```
1.    public class VendingMachine {
2.      protected LinkedList<Drink> drinkStorage;
3.      protected LinkedList<String> coinStorage;
4.      public VendingMachine() {
5.        this.drinkStorage = new LinkedList<Drink>();
6.        this.coinStorage   = new LinkedList<String>();
7.      }
8.    public void addCoin(String c) {
9.        AlloyOperations.alloyUnion(this.coinStorage,c);
10.     }
11.     public void buyDrink_Water(Water d) {                                    (3.1)
12.       AlloyOperations.alloySubtraction(this.drinkStorage,newWater(d));
13.     }
14.     public void buyDrink_Softdrink(Softdrink d) {
15.       AlloyOperations.alloySubtraction(this.drinkStorage,new Softdrink(d));
16.     }
17.     public void buyDrink_Tea(Tea d) {
18.       AlloyOperations.alloySubtraction(this.drinkStorage,new Tea(d));
19.     }
20.   }
21.   ...
```

The code 3.1 shows the generated *VendingMachine* class created from the signature with this same name (line 7 – 2.1). The attributes are all collections due to the multiplicity *set* declared in the model (lines 8 and 9 – 2.1). Clearly, the designer chose as data structure a linked list from the pool of structures available by the translation tool. From the predicate *buyDrink* (line 1 – 2.3), three methods were created, in order to remove parameter polymorphism from the generated code. This helps the translation tool to disambiguate between possible branches that can be taken by the state machine.

The program state machine is automatically synthesized from the model. We adopted the reactive model of computation due to its broad adoption by embedded applications. In this model, the application waits for an internal or external action, performing a command depending on the received action. The designer tells the translation tool which operations will be inserted by the run commands (lines 3 and 4 – 2.4) containing predicates or functions as its only argument. From now, all the synthesis process is automatic. When there is a polymorphic type to be inserted on the state machine, such as the predicate *buyDrink* (line 1 – 2.3), all possible methods are generated, in order to enable the generation of a non-ambiguous reactive state machine. We show below a portion of the generated reactive state machine code.

```
1.    public class Main {
2.      public static void main(String[] args) {
3.      MaquinaVendas maquina = new MaquinaVendas();
4.      Menu menu_enum = Menu.done;
5.      InputStreamReader stdin = new InputStreamReader(System.in);
6.      BufferedReader console = new BufferedReader(stdin);
7.      try {
8.        while(true){
9.          if(menu_enum.ordinal()==Menu.adicionaficha.ordinal()) {
10.           try {                                                            (3.2)
11.             System.out.println("Enter Ficha:");
12.             String c = console.readLine();
13.             machine.adicionaFicha(c);
14.           }
15.           catch (IOException e) { e.printStackTrace(); }
16.           finally { menu_enum = Menu.done; }
17.         }
18.   ...
```

The options beginning on lines 9, 18 19 and 20 were extracted from the run commands wrote by the designer (lines 3 and 4 – 2.4). The command's bodies were omitted, but are very similar to the one shown on line 9. The state machine behavior is simple: after the end of every synthesized command from run constructions, the state machine goes to *done* state. Within this state the next command to be executed is chosen. Finally, based on the formal parameters list for each command synthesized, there is the insertion of Java code that captures the needed data from standard input. (lines 11 to 13 – 3.2).

4. Results

In order to analyze the proposed approach, as well as the generated Java code, three applications are modeled in Alloy, and the code translator ware applied in them. The address book and the vending machine mixes control and data-flow portions; the elevator controller is totally control-flow.

Table 1 shows the quantitative analysis between the Alloy and Java constructions existing in the model and in the generated Java code. It can be seen from data that data-flow applications benefit more of the code generation in terms of the amount of lines of code. This is due to the absence of explicit data structures in the Alloy models, but they must be generated in the Java code in order to enable its execution; this is evident when considering the less intensive use of diverse Alloy constructions by the address book, but yet the generated code is the largest among the presented applications. The column *options* in the table 1 tells the number of declared run commands for the code generation of the program state machine, considering the existence of inheritance and method polymorphism.

Tab. 1 - Quantitative analysis between Alloy and the generated Java code for three applications

Alloy	**Signatures**	**Predicates**	**Facts**	**Lines of Code**
Vending Machine	7	2	5	39
Address Book	4	3	2	33
Elevator Controller	4	1	4	37
Java Generated	**Classes**	**Methods**	**Options**	**Lines of Code**
Vending Machine	7	18	3	198
Address Book	4	16	2	205
Elevator Controller	1	1	1	53

The number of generated Java methods superior to the Alloy predicates is due to the fact that data-flow applications have more attributes encapsulating data, needing thus *get* and *set* methods, in addition to its more elaborate behavior on the data (vending machine and address book). In the control-flow application, the only behavior is the state machine, implemented by the *main* method. It is worth noting that none of the applications analyzed presented errors, indicating that the translation rules are all consistent.

5. Conclusions and Future Work

This paper presented an approach and a tool for automatic code generation from verified Alloy models targeting embedded applications. This approach has the advantage of being based on a simple formal method, with relational and object-oriented syntax. In addition, it may contribute to the potential reduction of time-to-market windows needed to launch a product where the modeled application is embedded on, obtaining thus more quality in the overall design process. Also, we analyzed quantitatively the generated Java code for control and data-flow applications, showing the good results and the potential of the proposed approach. As future works we cite the translation of Alloy facts and the modeling of a more complex application, qualitatively comparing it to the respective UML model and the hand-written Java code. Without the fact translation, the application's properties are currently modeled within its behavior, making the modeling process less flexible.

6. References

[GRA 03] GRAAF, Bas, LORMANS, Marco, TOETENEL, Hans. **Embedded Software Engineering: The State of the Practice**. IEEE Software, Vol. 20 (6), November, p. 61-69.

[JAC 06] JACKSON, Daniel. **Software Abstractions – Logic, Language and Analysis**. The MIT Press, Cambridge, MA, USA. First Edition.

[NOK 05] NOKIA. **The Road to Three Billion Subscribers: Nokia outlines strategy to reduce total cost of mobile phone ownership for consumers in new growth markets**. Press Releases. June.

[SAN 04] SANDER, Ingo, JANTSCH, Axel. **System Modeling and Transformational Design Refinement in ForSyDe**. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23 (1), January, p. 17-31.

[SCH 04] SCHINZ, Ingo, TOBEN, Tobe, MRUGALLA, Christian, WESTPHAL, Bernd. **The Rhapsody UML Verification Environment**. Proceedings of the 2nd Int. Conf. on Soft. Eng. and Formal Methods. IEEE Comp. Soc., Washington, DC, USA, p. 174-183.

[TOR 06] TORLAK, Emina, JACKSON, Daniel. **The Design of a Relational Engine**. MIT-CSAIL Technical Report. MIT, Cambridge, MA, USA, MIT-CSAIL-TR-2006-068, 29th September.

CAD TOOLS I

Performance-Driven Routing and Interconnect Delay Models

T. J. Reimann, G. B. V. Santos, M. O. Johann, R. A. L. Reis
{tjreimann,gbvsantos}@inf.ufrgs.br

Instituto de Informática – Universidade Federal do Rio Grande do Sul

Abstract

Critical Sink Routing Trees (CSRT) are the most efficient structures for performance-driven single net routing. In this work we have developed the Steiner Elmore Routing Trees with identified critical sink (SERT-C) algorithm by Boese et. al., from 1995, and compared it with the maze-routing-based algorithm from Hentschke et. al., called AMAZE. We recreated the scenario shown by Boese et. al. where the algorithm is described to validate our implementation. Nevertheless, first we have implemented and validated the Elmore and the Two-pole delay models by comparing its results with SPICE-computed delay approximations. After that, we can observe how SERT-C provides results considerably more efficient in earlier technology parameters process while both algorithms generate quite similar results in a 90nm process. Also, the results suggest that scaling provides a considerable impact in both delay models accuracy.

1. Introduction

Since the early 90´s the use of small cost (say small capacitance) interconnect instances was reported to be NOT the best choice for performance-driven routing [PRA 90] [CON 92] [CON 93] [ALP 93]. In fact, from that long, different properties of the routing structures have been explored for delay reduction, such as shortest paths between source and sinks [RAO 92] [CON 93] radius bounding [CON 92] [ALP 95], and the prioritized optimization of the critical sink [KAH 95] [BOE 95] [BOR 97]. This last one comprises the Critical Sink Routing Trees (CSRT). In this work we address algorithms for generating such structures. We developed the SERT-C from [BOE 95] and compared it with the AMAZE from [HEN 07].

In order to validate our SERT-C implementation we also implement and validated the Elmore [ELM 48] and Two-Pole [ZHO 92] Delay models, what is described in section two. The Elmore is used inside of the SERT-C algorithm, and Two-Pole was the estimator used in the original work experiments. In section three we validate the implementation of SERT-C by reproducing the experiments scenario of [BOE 95]. And finally we present both algorithms comparisons followed by the conclusions.

2. Elmore and Two-pole Delay Models

Elmore delay does not intrinsically correspond to any delay time (it is the first moment of the impulse response in a distributed RC tree), but can be said to correspond to a 63.2% delay criterion of distributed RC tree [BOE 95]. Thus, we compare Elmore delay results to a 63.2% threshold voltage delay given by SPICE simulation.

Two-pole delay is based on the first and second moments of the impulse response in a distributed RC tree to estimate interconnect tree delay. The nature of Two-pole approximation makes it more suited to a 90% rise time criterion [ZHO 92]. Therefore, we compare it to a 90% threshold voltage delay given by SPICE.

The comparison was made over 250 random nets for each net size (3, 4, 5, 6, 7, 10 and 15) and each technology parameter set described in tab. 1. Using SERT-C to connect them, the comparison was made over 15.750 routing topologies, and results are shown in the last two columns of tab. 1, that shows the ratio between the average results of SPICE and both estimators.

Tab. 1 – Technology parameters and average ratios between SPICE and both Elmore and Two-pole estimators. R_d is driver resistance, R_w and C_w are unit wire resistance and capacitance and C_l is loading capacitance

Technology	R_d(ohm)	R_w(ohm/µm)	C_w(fF/µm)	C_l(fF)	SPICE/Elmore	SPICE/Two-pole
2.0µm [CON 96]	371	0.0206	0.054	5.175	0,03%	2,47%
2.0µm [ALP 95]	164	0.033	0.019	5.7	0,05%	2,31%
1.5µm [CON 96]	178	0.015	0.0042	6.21	0,03%	2,37%
1.2µm [CON 96]	160	0.0164	0.0053	4.416	0,03%	2,36%
1.2µm [BOE 95]	212.1	0.073	0.083	7.06	0,09%	2,33%
0.8µm [BOE 93]	100	0.03	0.352	15.3	0,07%	2,40%
0.5µm [CON 96]	195	0.112	0.0391	1	0,17%	2,32%
0.5µm [ALP 95]	270	0.112	0.039	1	0,10%	2,39%
90nm[1]	1	0.292	0.16	10	1,07%	13,72%

[1]Technological parameters based on a 90nm process

As we can observe form tab. 1 the results for ratios between SPICE simulation and Elmore delay are consistently close to 0% for earlier technology parameters and close to 1% for 90nm based parameters. These results validate our Elmore delay implementation. Nevertheless, Two-pole ratios are close to 2.35% for earlier technology parameters and 13.7% for 90nm. As observed in [KAM 95], Two-pole delays are within 13% of SPICE-computed delays, so these results validate our implementation. However, the 90nm results suggest that Elmore and Two-pole delay suffer a higher impact on their accuracy in newer technologies. This impact is probably caused by the smaller R_d/R_w ratio in this technology compared to the others.

3. SERT-C Algorithm Validation

To ensure the correction of our SERT-C implementation we reproduced the comparison scenario shown in [BOE 95]. This scenario includes Minimal Spanning Tree (MST), AHHK[1] [ALP 92], 1-Steiner[2] [KAH 92], and the ERT, SERT and SERT-C algorithms. We also implemented ERT and SERT algorithms, which are simplifications of the SERT-C (an euclidean e a rectilinear version of it, respectively, which optimizes the whole set of terminals delay, and not just the critical one).

In tab. 2 we can observe the results of comparison over 100 random nets for each net size. All choices for critical sink are random. Technology parameters called IC1, IC2 and IC3 are obtained from [BOE 95]. Delay values are calculated with Two-pole estimator.

As we expected, we can observe that tab. 2 results show SERT-C superiority over the others algorithms for critical sink delay in all technologies and all net sizes. It also shows that SERT-C produces worst results for other no critical sinks delays and wirelength, as well.

The great similarity of this results with the results observed in [BOE95] validate our SERT-C implementation.

Tab. 2 – Two-pole simulation results. ERT constructions are compared with MST and AHHK. SERT and SERT-C constructions are compared with 1-Steiner. ERT, AHHK, SERT and SERT-C results are reported as percent differences from corresponding MST or 1-Steiner results.

		IC1		IC2		IC3	
		|N| = 5	|N| = 9	|N| = 5	|N| = 9	|N| = 5	|N| = 9
Critical Sink Delay (ns)	MST	1,099	1,890	0,670	1,195	0,440	0,790
	AHHK	-18,49%	-35,84%	-22,78%	-47,21%	-24,45%	-51,62%
	ERT	-7,65%	-19,62%	-19,91%	-41,38%	-24,32%	-49,52%
	1-Steiner	0,974	1,556	0,589	0,963	0,385	0,630
	SERT	5,62%	0,90%	-3,88%	-13,97%	-7,00%	-19,13%
	SERT-C	-6,42%	-18,27%	-19,93%	-38,90%	-24,63%	-46,19%
Max. Delay (ns)	MST	3,146	5,287	2,017	3,470	1,348	2,324
	AHHK	-24,75%	-40,64%	-30,12%	-51,50%	-32,08%	-55,50%
	ERT	-12,91%	-24,53%	-24,77%	-44,39%	-28,66%	-51,10%
	1-Steiner	2,665	4,297	1,679	2,763	1,114	1,833
	SERT	4,13%	-2,61%	-4,24%	-16,32%	-6,74%	-20,63%
	SERT-C	5,27%	7,45%	-0,04%	3,51%	-1,47%	2,54%
Average Wirelength (cm)	MST	1,62	2,48	1,62	2,48	1,62	2,48
	AHHK	-6,26%	-4,98%	-6,26%	-4,98%	-6,26%	-4,98%
	ERT	18,73%	23,74%	18,73%	23,74%	18,73%	23,74%
	1-Steiner	1,48	2,22	1,48	2,22	1,48	2,22
	SERT	26,01%	30,56%	26,01%	30,56%	26,01%	30,56%
	SERT-C	20,63%	20,26%	20,63%	20,26%	20,63%	20,26%

Different that was seen in [BOE 95], AHHK results shows a great improvement compared to MST results. This can be explained because we used a recent framework which implements AHHK construction.

4. Comparison Experiments

The former CSRT's algorithms like SERT-C [BOE 95] have in its heuristics *RC* delay calculations. On the other hand, AMAZE [HEN 07] provides more efficient critical sink isolation, with a maze-routing-based approach and no *RC* delay calculation. This work also aims to evaluate AMAZE inside a CSRT approach.

[1] We did used the C-TREE framework for buffered trees, with the proper parameters in order to achieve the AHHK results – check http://dropzone.tamu.edu/~cnsze/GSRC/ctree.html

[2] Which is available at http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/Robins_BI1S.html.

The comparison between AMAZE and SERT-C algorithms was performed over a 500x500 grid size, 250 random nets were generated for each net size. While AMAZE results are the same for all technology parameters, SERT-C provides different result according to technology parameters, which are used as an input for the algorithm.

In tab. 3 we observe the relative percent difference of AMAZE to SERT-C results for critical sink delay. Tab. 4 shows the relative percent differences of AMAZE to SERT-C results for wirelength. We have seen that SERT-C constructions provides greatly improved critical sink delay with earlier technology parameters which have greater driver resistance and wire unit capacitance (2.0, 1.2, 1.2B, 0.8, 0.5 and 0.5B technologies of tab. 1). This superiority is because these particular parameter sets makes wirelength cost represents a higher impact on delay, and the isolation of critical sink provided by AMAZE results in higher total wirelength values.

Tab. 3 – Percent differences of AMAZE compared to SERT-C average critical sink delay for each net size and each technology parameter

	2.0	**2.0A**	**1.5**	**1.2**	**1.2B**	**0.8**	**0.5**	**0.5B**	**90nm**	**Average**
3	24,00%	13,55%	6,91%	26,05%	22,08%	26,05%	25,56%	26,90%	0,00%	19,01%
4	21,30%	10,94%	5,34%	22,99%	18,85%	22,99%	22,15%	23,74%	0,83%	16,57%
5	18,39%	8,54%	4,01%	19,58%	15,47%	19,58%	18,21%	20,16%	1,40%	13,93%
6	17,04%	7,40%	3,34%	18,18%	14,02%	18,18%	16,65%	18,76%	1,57%	12,79%
7	14,29%	5,70%	2,49%	15,12%	11,34%	15,12%	13,61%	15,61%	1,84%	10,57%
10	10,90%	3,44%	1,34%	11,22%	7,80%	11,22%	9,76%	11,59%	1,01%	7,59%
15	7,67%	1,62%	0,51%	7,58%	4,65%	7,58%	6,12%	7,84%	2,08%	5,07%
Average	16,23%	7,31%	3,42%	17,25%	13,46%	17,25%	16,01%	17,80%	1,25%	12,22%

Tab. 4 - Percent differences of AMAZE compared to SERT-C average wirelength for each net size and each technology parameter

	2.0	**2.0A**	**1.5**	**1.2**	**1.2B**	**0.8**	**0.5**	**0.5B**	**90nm**	**Average**
3	34,00%	34,00%	34,00%	34,00%	34,00%	34,00%	34,00%	34,00%	0,04%	30,23%
4	31,94%	31,89%	31,81%	31,89%	31,89%	31,89%	31,81%	31,89%	0,17%	28,36%
5	28,90%	28,71%	28,54%	28,67%	28,70%	28,76%	28,58%	28,76%	-0,12%	25,50%
6	27,86%	27,46%	26,91%	27,36%	27,46%	27,64%	27,23%	27,49%	0,16%	24,40%
7	24,26%	23,96%	23,27%	23,87%	24,05%	24,19%	23,62%	24,13%	-0,10%	21,25%
10	20,23%	19,73%	19,11%	19,62%	19,91%	20,06%	19,66%	20,01%	0,24%	17,62%
15	16,06%	15,46%	14,81%	15,31%	15,68%	15,83%	15,50%	15,77%	0,15%	13,84%
Average	26,18%	25,89%	25,49%	25,82%	25,96%	26,05%	25,77%	26,01%	0,08%	23,03%

As we observe the difference between the two algorithms reduces close to 1.5% in 90nm process, suggesting an improve in AMAZE results for more recent technology parameters.

5. Conclusions and Future Work

In this work, we validated ours implementations of Elmore and Two-pole delay models and SERT-C algorithm. Elmore and Two-pole are validated comparing its delays approximation results to SPICE-computed delays. We also perform a comparison of SERT-C and AMAZE router algorithms with several technology parameters. This comparison shows that SERT-C has better results than AMAZE with earlier technology parameters. On the other hand, they both show quite similar results for parameters based on a 90nm process.

As future work we intent to evaluate routing and delay modeling using recent, or even predictive, process technologies. Also, a re-evaluation of the fidelity property of Elmore delay established in [BOE 95] in the current context of nowadays parameters.

6. References

[ALP 92] ALPERT, C. J.; T. C. HU; HUANG J. H.; KAHNG, A. B. **A direct combination of the Prim and Dijkstra constructions for improved performance-driven global routing.** University of California, Los Angeles, Dept. Computer Science, Tech. Rep. CSD-92005 1, 1992.

[ALP 93] ALPERT, C. J.; HU, T. C.; HUANG, J.H.; KAHNG, A. B. **A direct combination of the Prim and Dijkstra constructions for improved performance-driven global routing.** IEEE International Symposium on Circuits and Systems, ISCAS, 1993.

[ALP 95] ALPERT; HU; HUANG; KAHNG; KARGER. **Prim-Dijkstra tradeoffs for improved performance-driven routing tree design**. IEEE Transactions on Computer-Aided Design, 1995.

[BOE 93] BOESE, K. D.; KAHNG, A. B.; MCCOY; ROBIN, G. **Towards Optimal Routing Trees**. 4th ACM/SIGDA Physical design Workshop, 1993.

[BOE 95] BOESE, K. D.; KAHNG, A. B.; MCCOY; ROBINS, G. **Near-optimal critical sink routing tree constructions**. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1995.

[BOR 97] BORAH; OWENS; IRWIN. **A fast algorithm for minimizing the Elmore delay**. IEEE Trans. on CAD, 1997.

[CON 92] CONG, J.; KAHNG, A. B.; ROBINS, G.; SARRAFZADEH, M.; WONG, C. K. **Provably Good Performance-Driven Global Routing.** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992.

[CON 93] CONG, J.; LEUNG, K. S.; ZHOU, D. **Performance-Driven Interconnect Design Based on Distributed RC Delay Model**. In: ACM/IEEE Design Automation Conference. Proceedings. p. 606-611, 1993.

[CON 96] CONG, J.; LEUNG, K. S. **On the Construction of Optimal or Near-Optimal Steiner Arborescence.** Tech. Report CSD-960033 – UCLA Computer Science, Los Angeles, CA, 1996.

[ELM 48] ELMORE, W.C. **The Transient Response Of Damped Linear Networks with Particular Regard to Wideband Amplifiers**. Journal of Applied Physics, 1948.

[HEN 07] HENTSCHKE; NARASIMHAN; JOHANN, REIS. **Maze Routing Steiner Trees with Effective Critical Sink Optimization.** ISPD, 2007.

[KAH 92] KAHNG, A. B.; ROBINS, G. **A new class of iterative Steiner tree heuristics with good performance.** IEEE Trans. Computer-Aided Design, vol. 11, pp. 893-902, July 1992.

[KAH 95] KAHNG, A. B.; ROBINS, G. **On Optimal Interconnects for VLSI**. Kluwer, 1995.

[KAM 95] KAHNG, A. B.; MUDDU. **Two-pole Analysis of Interconnection Trees**. Proceedings IEEE Multi-Chip Module Conference, 1995.

[PRA 90] PRASITJUTRAKUL, S.; KUBITZ, W. J.. **A timing-driven global router for custom chip design Computer-Aided Design**. ICCAD-90. Digest of Technical Papers, IEEE International Conference on 11-15 Nov. 1990 p 48-51.

[RAO 92] RAO, Sailesh K.; SADAYAPPAN, P.; HWANG, Frank K.; SHOR, Peter W. **The Rectilinear Steiner Arborescence Problem**. Algorithmica 7', 1992.

[ZHO 92] ZHOU D.; SU S.; TSUI F.; GAO D. S.; CONG J. **Analysis of Trees of Transmission Lines**, technical report UCLA CSD-920010, 1992.

Improving Pathfinder using Iterated 1-Steiner Algorithm

Luca Bochi Saldanha, Charles Capella Leonhardt, Adriel Mota Ziesemer Junior, Ricardo Augusto da Luz Reis
{lbsaldanha,ccleonhardt,amziesemerj,reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul

Abstract

The primary goal of this work is to improve the current implementation of the routing algorithm used in our automatic layout generator tool. Routing, as well as placement, determine the length of interconnects, so it is crucial to have extremely efficient algorithms in order to reduce the circuit wirelength and, consequently, the circuit area. The present implementation is based on Pathfinder algorithm. It was chosen because different nets have to be routed in a same layer of the layout, so they may have to compete for the use of some shared nodes. This algorithm proposes a heuristic method to resolve this competition. Our work is focused mainly in improve the routing result of each individual net, leaving the negotiation issues aside. A new approach is proposed and implemented: the insertion of Steiner points to reduce the wirelength of each net and, therefore, the wirelength of the circuit.

1. Present Implementation – Pathfinder Algorithm

This algorithm was first designed to perform routing in FPGAs [MCM 95] but it can be employed to solve others routing issues. The present implementation is based on two semi-independents parts. The First one performs interconnections between nodes that belong to the same net – that is called "signal router" and employs the method proposed by Lee [LEE 61]. The other one is the "global router". It uses the signal router to connect all the nets and solve their conflicts.

Signal router

The signal router was implemented using a method proposed by Lee [LEE 61] to find the shortest path between two nodes of the same net. In our model, all nets are represented by graphs where the nodes are vertices and the connections are edges. So, to connect two given nodes we must perform a breadth-first search, starting in one node until we find the other one.

Beginning in a root node, his neighbors are visited. If the terminal node is not among them, another search is performed from each one of the neighbors. This is done repeatedly until the terminal node is found. In order to produce optimal results (considering only two nodes), these subsequent searches are made regarding the distance from the root: the closest neighbors are visited first. After the minimum path between two nodes is known, the search continues taking all this nodes as root for the subsequent iterations in case of there are unrouted nodes inside the net.

The final result is not optimal. Different results can be obtained starting the algorithm from different nodes. In our case, the order is arbitrarily decided.

Global router

The global router is used to route graphs with multiple nets by negotiating the occupation of congested nodes. This router was designed to balance the goals of both performance and routability using an heuristic method.

The global router uses the signal router in several iterations to solve the conflicts between different nets. It does that by adjusting the cost of interconnections that lead to congested nodes. This adjustment is done considering the following formula:

$$C_n = (B_n + H_n) \times P_n$$

It expresses the cost of using a determined node n (C_n) in terms of the base cost (B_n), the congestion history of n in past iterations (H_n) and the amount of nets currently using n (P_n).

In the first iteration, all nodes have P_n set to one. Subsequently, its value is adjusted accordingly to the node occupation in the current iteration. The key of the algorithm is H_n. It makes the cost of the overcrowded nodes constantly increase. As more iterations are performed, it gets more expensive to use congested paths. It makes the algorithm look for alternative paths with unused resources and converge to a feasible solution.

2. Iterated 1-Steiner algorithm

A Steiner tree is the Minimum Spanning Tree (MST) formed by the original net with the insertion of Steiner points. Steiner point is any point that when added to the original net, it reduces the total wirelength. Solve the Steiner tree problem is NP-complete, so, heuristics are used to solve restricted cases. In our approach we used the Iterated 1-Steiner algorithm heuristic that iteratively calculate optimum Steiner points and include them into the net [KAH 92].

Since our approach is graph-based, all nodes that don't belong to any net are considered candidates for being Steiner points. Because the graph structure, it's not possible to use methods for selecting Steiner candidates that are designed for rectilinear Steiner trees. These methods would reduce the amount of Steiner point candidates and so would decrease the time employed in the task of evaluating the points that are Steiner points.

After the global router find a feasible solution for all the nets, an optimization step using the Steiner points is performed. The algorithm remove one net each time and route it again with the inclusion of the Steiner points. This continue until all nets have been evaluated for the possibility of adding Steiner points and the points that bring an improvement in wirelength are inserted. The result is always better or equal to the global router result.

3. A Proposal of Implementation

Since the beginning of this work, it was clear that Pathfinder's approach was the best one face the problem we had in hands. Thus we have decided not to touch the main functionality of the algorithm – the negotiation idea.

At first it appeared to be a good idea to search for Steiner points on each net one by one. So the following was implemented:

```
1. Route net N;
2. best_cost = wirelength(N);
3. Repeat
4.   Create a list L of steiner candidates
5.   Repeat
6.     best_candidate := Null;
7.     Insert front(L) in net N, creating a new net N';
8.     Route net N';
9.     If wirelength(N') < best_cost;
10.       best_candidate := front(L);
11.       best_cost := wirelength(N');
12.    Take out front(L) from net N';
13.    Take out front(L) from list L;
14. Until L is empty;
15. If best steiner candidate != Null
16.   Add best_candidate permanently to net N
17.Until best_candidate == Null
```

Fig. 1 shows, in a simplified way, the steps proceeded to achieve the final routing result. At first, there are two nets to route, net 1 represented by circles and net 2 represented by squares (Fig. 1a). The routing is made one net at a time, not considering the other. Fig. 1b is the routing result without the addition of any Steiner point in net 1 (Fig. 1d for net 2, respectively). Then Fig. 1c shows net 1 already with Steiner points included, represented by filled nodes (Fig. 1e for net 2, respectively). This is where our implementation stops. After that, the conflicts that appear clearly in Fig. 1f are solved by the global router, resulting in the final routing result (Fig. 1g).

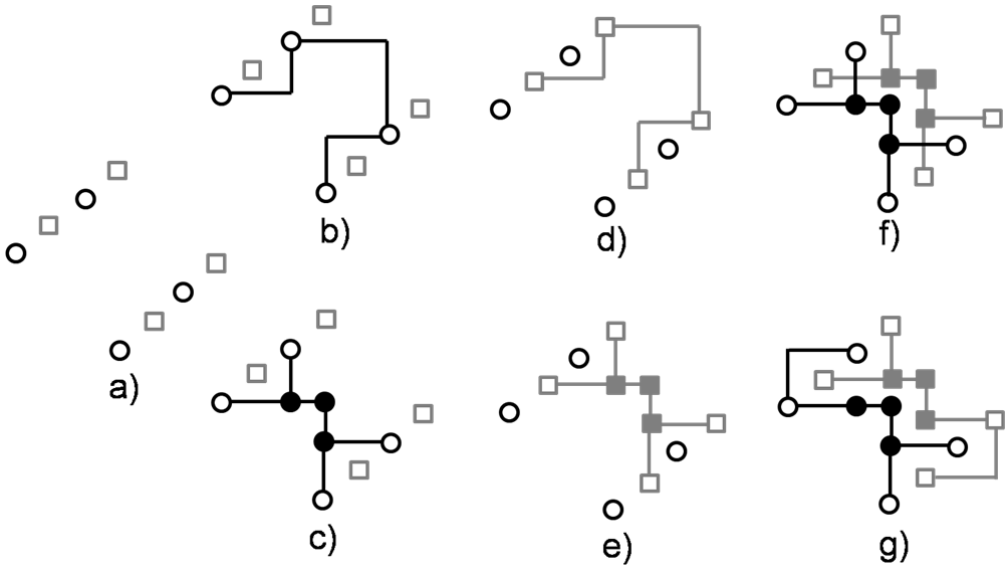


Fig. 1 – Steps to achieve the routing result through this approach

The great advantage of this approach is that neither the signal nor the global router had to be modified. This implementation preserves the tool modularity and this change is transparent to other modules. The only thing that had to be done is to call the sub-routine described above before route the circuit using Pathfinder algorithm with no change.

Unfortunately this approach didn't work presenting many problems. One, and least of them, is the execution time of the sub-routine. By "list of steiner candidates" we mean all nodes that don't belong to any net. There is a really large number of them. This had to be done because in the graph used in the tool it is not possible to know where the nodes are physically, so there is no way to discover which ones are good or bad choices. This increase the number of iterations in the exterior loop.

The second problem was surprising: the wirelength of the circuit increased in most of the tests. Investigations showed that by adding more nodes to the nets we were making more difficult to solve the negotiation problems. Steiner points were making all nets tend to use the center of the circuit and, as alternative paths were chosen, those Steiner points became useless, increasing the size of the net. This effect can be seen on Fig. 2f, where the inclusion of Steiner points caused a little "confusion".

The source of this problem is that other nets are not taken into account when searching for Steiner points. In order to bypass this difficulty, the signal router had to be modified. Instead of routing the nets one by one, not considering all of them at a time nor "remembering" previous routing attempts, we now intend to preserve some information (and use it) when searching for Steiner points.

First a regular routing is made, without any change, including the solution of the congestion problems. Then we call the sub-routine described above with one change. In steps 1 and 8 we use another signal router, identical to the other but with one little change. This router does not try to use nodes that already belong to other nets. In other words, we erase the previous routing result one net at time and route it again. This will certainly not cause any conflict and we may search for Steiner points with no worry.

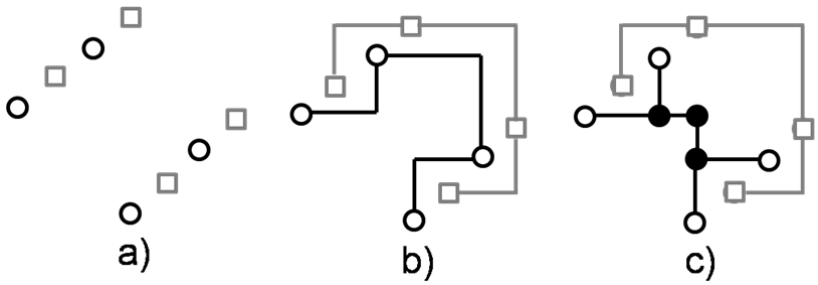


Fig. 2 – Steps to achieve the routing result through the new approach

Fig. 2 shows how the change inserted in the signal router affects the routing result. In Fig. 2b we can see that a normal routing is made, without the inclusion of any Steiner point. After that, the routing of net 1 is unmade and the looking for Steiner points begin. Three of them are found and the routing of this net is remade considering them. Notice that when we try to find Steiner points for net 2, none is found. That happens because the way is blocked by net 1.

4. Results

In order to compare the results of the global router and the Iterated 1-Steiner approach, two measures were used for comparison: wirelength and time, as shown in Tab.1.

From the tab.1 we can notice that our approach presented better wirelength than the reference global router in all test cases. On the other side, it also took considerable more time to finish. It was expected because the additional steps, in comparison to the original algorithm, needed to calculate the Steiner nodes.

For intracell routing, our algorithm presented an average improvement of about 1% in wirelength. It is excellent because the runtime was keep low even for larger cells and produce smaller cells is far more important than execution time.

For detailed routing the results in wirelength were better because the increased number of arcs and nodes, leading to more Steiner point candidates.

Tab. 1 – Comparison between the algorithms intracell and global routing

	Global Router		Iterated 1-Steiner		Comparison	
Circuit	Wirelength	Time(s)	Wirelength	Time(s)	Diff(Wirelength) %	Diff(fime) %
ADD22	3599	6	3578	11	-0.58	83.33
NAND2	2190	1	2186	1	-0.18	0
XOR41	5032	2	5021	14	-0.22	600
OAI421	6076	34	6004	42	-1.18	23.53
Circuit (nrNets, netSize)	Wirelength	Time(s)	Wirelength	Time(s)	Diff(Wirelength) %	Diff(fime) %
(5, 4)	887	1	862	9	-2.81	800
(9, 4)	1672	4	1604	7	-4.07	75
(12, 4)	2497	311	2375	313	-4.89	0.64
(5, 5)	1072	1	1066	10	-0.56	900
(15,3)	2508	287	2436	289	-2.87	0.69
(3, 7)	818	1	801	10	-2.07	900

5. Conclusion and future works

The algorithm presented good results in decreasing the wirelength. The best results were obtained during the detailed routing because more Steiner nodes were found. Beyond that, this technique also increased the execution time. For this reason we are going to replace the Lee router by A* algorithm that should decrease the time necessary for the algorithm execution. This happens because in A*, all distances between nodes are known and the signal routing can be performed smartly.

Another future work is to test the algorithm using academy benchmarks, to better compare our results with other tools.

6. References

[KAH 92] KAHNG, Andrew. **A New Class of Iterative Steiner Tree Heuristics with Good Performanc**e. Los Angeles: IEEE, c1992.

[MCM 95] MCMURCHIE, Larry ; EBELING, Carl. **PathFinder: A Negotiation- Based Performance-Driven Router for FPGAs.** In International Symposium on Field Programmable Gate Arrays, Monterey, Ca.,Feb.1995.

[LEE 61] LEE, C. Y.; **An Algorithm for Path connections and its applications.** In: IRE Transactions on Eletronic Computer, 1961.

[ZIE 07] ZIESEMER JUNIOR, Adriel Mota ; LAZZARI, Cristiano ; REIS, Ricardo **. Transistor Level Automatic Layout Generator for non-Complementary CMOS Cells.** In: VLSI-SOC, 2007, Atlanta. Very Large Scale Integration - System on Chip 2007, 2007.

Timing-Aware Placement Algorithm to 3D Circuits

[1]Felipe Pinto [1]Guilherme Flach, [1]Renato Hentschke, [1]Ricardo Reis
{fapinto,gaflach,renato,reis}@inf.ufrgs.br

[1]Universidade Federal do Rio Grande do Sul

Abstract

This paper presents a novel technique for path-based placement of 3D VLSI circuits. The Critical Star 3D technique changes the circuit netlist by grouping cells of a same critical path using a special star net model. The technique has shown its effectiveness in eliminating 3D-Vias while reducing the total wirelength of the critical paths without disturbing the total circuit wirelength. The algorithm was tested over the 3D version of ISCAS 99 benchmarks. In such circuits, the technique presents an efficiency of almost 100% on keeping all cells, belonging to a same critical path, in the same tier. Beside this, the critical wirelength is improved in 4.6% and the total wirelength is increased by just 0.03%.

1. Introduction

The stacking of 2D VLSI circuits composes a 3D Circuit. The stacked circuits, called tiers, are composed by an active area, metal layers and insulator layers. The 3D fabrication opens many new possibilities. With 3D integration we can arrange elements of the system in a chip and avoid common design problems, such as noise for analog circuitry, memory to processor bandwidth and others.

Depending on how the circuits are arranged, the integration between a pair of adjacent tiers can be classified as face-to-face, face-to-back or back-to-back. The integration modifies how a CAD tool treats 3D circuits. Tiers are connected through 3D-vias [DAV05, DAS04, ABA05, KAY04 and OBE99].

The face-to-face (f2f) strategy has the minor size of 3D-Via between the three strategies. If manufactured by Tezzaron\ just 0.5 uM, and manufactured by MITLL 5 uM. The face-to-back and back-to-back strategy will support fewer cells because of the 3D-Vias spaces. In f2b and b2b the 3D-Vias use active layer area.

The 3D-vias in f2f strategy does not use active layer area, but when using the f2f strategy, is necessary to use other strategies to stack more than 2 tiers. Experimental results show that one can improve the wire length compared to a 2D solution from 15% up to 30% in average. Note that the face-to-back Through Vias must dig a hole in the Bulk and for that they require active space that cannot be used by the circuit logic. On the other hand, a face-to-face 3D-Via has to go through all the metal layers of both tiers consuming more routing resources than face-to-back. For this reason, it is reasonable to allocate larger space for face-to-face 3D-Vias than face-to-back ones. Additionally, a maximum of two consequent tiers can be integrated in face-to-face.

Although 3D-vias are important to the improvement of wirelength in 3D circuits, they could be prohibitive in critical path, depending on the integration strategy. There are several issues when using 3D-Vias that would make them not attractive to critical wires. For instance, in the f2f strategy, to connect two cells in adjacent tiers, regular vias going through all the metal layers on both tiers are required. A regular via is around have 10 times more resistance than a wire unit [DAV05], then keeping the cells on the same tier is important to avoid regular vias through critical paths and do not use all metal layers.

The 3D-via capacitance vary widely, depending on whether the 3D-via are isolated or shielded with surrounding wires, making it difficult to equate via capacitance with a length of wire [DAV05]. In this work, we exploit the problem of keeping critical paths with no 3D-Via connections. Another point is the electrical characteristics of the 3D-Via itself could be harmful. From the CAD perspective, the tools and algorithms for 3D circuits are still on the early years. There is a huge new research space to be explored. New algorithms for the various CAD tasks need to be proposed in order to explore the potential advantages of the 3D technology.

In order to address the proposed problem, we define the 3D-Via avoidance problem formally in section 2. Explain the quadratic placement approach to 3D circuits in section 3. Propose an algorithm that will work under our placer on section 4. Experimental results are presented on section 5 and conclusions on section 6.

2. Problem Definition

Let be C the set of cells, P the set of the k most critical path and Kc the set of all critical path, which the cell c belongs to. The cell placement problem takes each cell c ∈ C and place them on the (cx, cy, cz) coordinate with t tiers such that cz ∈ {1, 2, ..., t}. The 3D-Via avoidance problem is defined as a constraint to the cell placement result defined by equation (1).

$$\forall c_1, c_2 \in \mathrm{C} \left| \left(\mathrm{K}_{\mathbf{c_1}} \cap \mathrm{K}_{\mathbf{c_2}} \right) \neq \varnothing \rightarrow c_{1_z} = c_{2_z} \right. \quad (1)$$

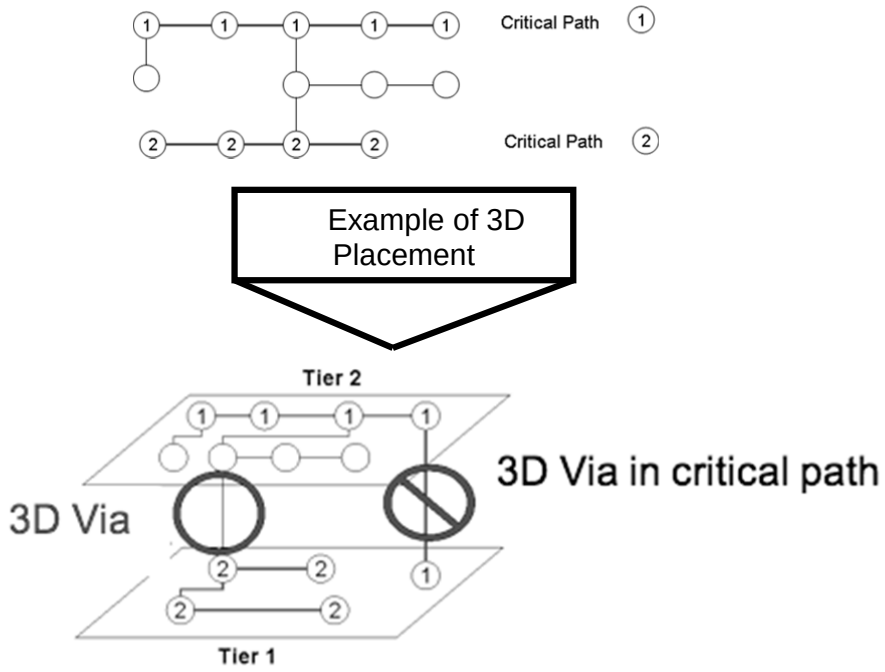


Fig. 1, 2 – A example of 3D placement with two critical paths

3. Quadratic Placement

We start applying a hybrid net modeling and weighting the connections, as suggested by [VIS05]. Nets of size 2 or 3 are modeled as a complete graph, while every net larger than three pins is translated into a star. After transforming the netlist, we apply the quadratic wirelength function $\varphi(\mathbf{x},\mathbf{y},\mathbf{z})$ as described by equation (2). It can be observed that the three coordinates **x**, **y** and **z** can be optimized independently with $\varphi(\mathbf{x}) + \varphi(\mathbf{y}) + \varphi(\mathbf{z})$. Applying the derivative on each of the optimization equation, the optimal solution can be found at the minimum point (derivative equals to 0). After some algebraic transformations, each coordinate is solved by the system of equations $\mathbf{Q} \times \mathbf{x} = \mathbf{d_x}$, where Q is the part of connectivity matrix related to the movable cells, x is the vector of unknowns which represents the position of movable cells and dx is the right-hand-side vector. The details of how these matrices can be obtained are found in the review provided in [ALP 97].

$$\Phi(\mathbf{x},\mathbf{y},\mathbf{z}) = \frac{1}{2}\sum_{i,j=1}^{n}\left[\sqrt{w_x\left(x_i - x_j\right)^2 + w_y\left(y_i - y_j\right)^2 + w_z\left(z_i - z_j\right)^2}\right]^2 =$$

$$\underbrace{\frac{1}{2}\sum_{i,j=1}^{n} w_x\left(x_i - x_j\right)^2}_{\Phi(\mathbf{x})} + \underbrace{\frac{1}{2}\sum_{i,j=1}^{n} w_y\left(y_i - y_j\right)^2}_{\Phi(\mathbf{y})} + \underbrace{\frac{1}{2}\sum_{i,j=1}^{n} w_z\left(z_i - z_j\right)^2}_{\Phi(\mathbf{z})} \quad (2)$$

The system of equations is solved using a preconditioned Conjugate Gradient method with incomplete Cholesky factorization [KER75] of matrix **Q** as the pre-conditioner. After solving the system of equations, all the cells are placed within the sliced cube, but probably the cells will be concentrated in the middle of the cube and most of them are placed at invalid *z* coordinates. The next section describes our methodology for spreading the cells and fixing their *z* coordinate.

4. Critical Star 3D Method

The algorithm for 3D-Via avoidance is applied into the global placement phase, more specifically on the Quadratic Placement engine [ALP97]. The algorithm was implemented over our placer. The technique is called Critical Star 3D. In order to keep the cells on the same critical path together, the netlist is updated by inserting an artificial node called critical star for each critical path. Every cell $c \in C$ such that $cp(c) = i$ is connected to csi with a special kind of artificial connection called Z-Grouping. All cells in a Z-Grouping are connecting for the Critical Star cell.

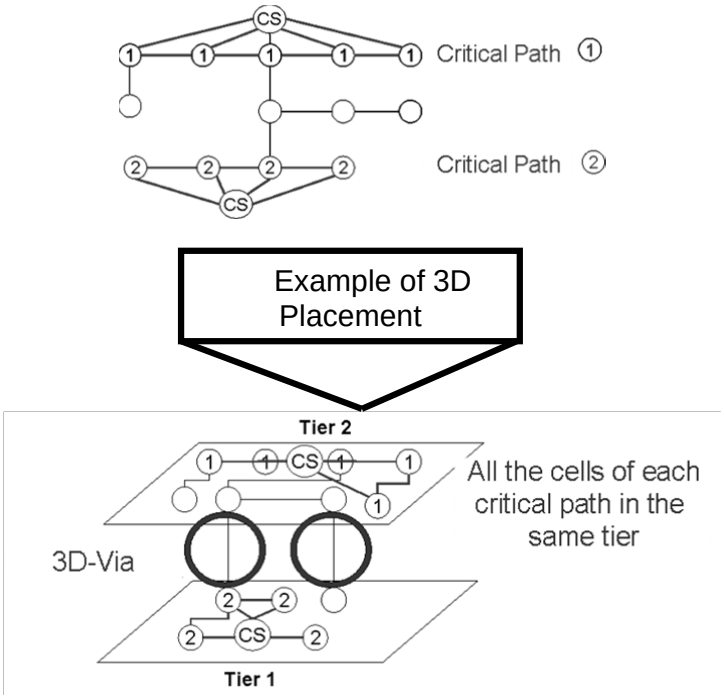


Fig. 3, 4 – The solution for example showed in Figure 1 and 2.

A Z-Grouping connection in Critical Star has a distinct weight for the Z axis in contrast with the weight on X or Y. In fact, the weight on Z axis is set to a very high number, such as 500 times a regular wire, while on the X and Y it could be set to 0 or some small number (such as 0.5 times a regular wire) in order to address the critical wire length reduction problem as well. Note that that for the 3D-Via avoidance problem the weight on X and Y can be set to any value, but it might affect wire length and number of 3D-Vias.

5. Experimental Results

In order to verify the effectiveness of the proposed algorithm, the following experimental setup was proposed. We have tested our approach under ISCAS`99 benchmark [ISC99]. The benchmark set with timing information was generated with the aid of commercial from the VHDL files and mapped to a logic level netlist. A timing analyzer for the same commercial tool set was used to identify the 100 most critical paths at the logic level. The details of the generated benchmarks can be observed in table. Initially, a baseline execution of our placer did not include any Z-Grouping net. After that, 100 Z-Grouping nets were introduced with 0.5 weight on X and Y axes and 500 on the Z.

6. Conclusions

The proposed method is very effective to avoid critical 3D-Vias, and all experimental results (except for the circuit b13a) resulted in none critical 3D-Via. The circuit b13a is a small benchmark. Note that smaller benchmark is more constrained than larger ones. On our experiments, 100 critical paths might too much forth is circuit, making the problem unfeasible. The wirelength is not affected (0.03% affect in average). Number of 3D-Vias is only 2.36% increased in average and critical wirelength is improved by 4.6%.

This article proposes a method to avoid the critical 3DVias by adding artificial nets in the circuit and making them with a strong weight only for the Z axis. The method reduces the number of 3D-Via to zero in all tested cases (except for one small circuit) with no effect to circuit wire length and number of 3D-Vias. The method explores a degree of freedom existing in the placement problem, where many solutions are equivalently good in terms of wire length and other classical metrics. Our algorithm actually introduced a new metric by the number of critical 3D-Vias and demonstrated experimentally that this objective does not conflict with 3D wire length optimization in all benchmarks

7. Acknowledge

The authors acknowledge the Brazil government and CNPq for support of this work.

8. References

[DAV05] Davis W., Wilson J., Mick S., Xu J., Hua H., Mineo C., Sule A., Steer M., and Franzon P., “Demystifying 3d ics: The pros and cons of going vertical,” Design and Test of Computers, pp. 498–510, Nov-Dec 2005.

[DAS04] Das S., Fan A., Chen K.-N., Tan C. S., Checka N., Reif R., “Technology, performance, and computer-aided design of three-dimensional integrated circuits,” in ISPD ’04: Proceedings of the 2004 international symposium on Physical design. New York, NY, USA: ACM Press, 2004, pp. 108–115.

[ABA05] Ababei C., Mogal H., Bazargan K., “Three-dimensional place and route for fpgas,” in ASP-DAC ’05: Proceedings of the 2003 conference on Asia South Pacific design automation. IEEE Press, 2005.

[DEN01] Deng Y. , Maly W. P., “Interconnect characteristics of 2.5-d system integration scheme,” in ISPD ’01: Proceedings of the 2001 international symposium on Physical design. New York, NY, USA: ACM Press, 2001, pp. 171–175.

[LIU05] Liu G., Li Z., Zhou Q., Hong X., Yang H. H., “3d placement algorithm considering vertical channels and guided by 2d placement solution,” in ASICON 2005: 6th International Conference On ASIC, 2005, pp. 24–27.

[KAY04] Kaya I., Salewski S., Olbrich M., Barke E., “Wirelength reduction using 3-d physical design,” in Integrated Circuit
and System Design -Power and Timing Modeling, Optimization and Simulation; Proceedings of 14th International Workshop, PATMOS 2004, 2004.

[OBE99] Obenaus S., Szymanski T., “Gravity: Fast placement for 3-d vlsi,” ACM Transactions on Design Automation of Electronic Systems, vol. 8, pp. 69–79, March 1999.

[ALP97] Alpert C. J., Chan T., Huang D. J.-H., Markov I., Yan K., “Quadratic placement revisited,” in DAC ’97: Proceedings of the 34th annual conference on Design automation. New York, NY, USA: ACM Press, 1997, pp. 752–757.

[ISC99] ISCAS `99 – Benchmarks, 1999 - http://www.fm.vslib.cz/~kes/asic/iscas/

[VIS05] Viswanathan, N; Chu C.C.-N - FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement,and a Hybrid Net Model. IEEE Transactions on CAD, Volume 24, Issue 5, pp 722-733, May 2005.

[KER75] Kershaw, D. The incomplete cholesky-conjugate gradient method for the iterative solution of systems of linear equations. Journal of Computational Physics, 26:43–65, 1975

Tab. 1 – From left to right: the benchmark, the placement of benchmark with a different number of tiers and the Z-place baseline and with the Z-grouping technique. Where V is the number of 3D-Vias, CV is the number of critical 3D-Vias, CWL is the critical wirelength and the WL is the total wirelength.

		Baseline				Z-Grouping			
Benchmarks	Tiers	#V	#CV	CWL	WL	#V	#CV	CWL	WL
b11a	2	206	544	4727	49843	225	0	4520	48540
b11a	3	413	1269	3640	42967	407	0	4207	42373
b11a	4	385	716	3286	41113	405	0	4312	40395
b12a	2	461	299	2698	78557	460	0	3290	77013
b12a	3	501	175	3438	80853	474	0	4456	83664
b12a	4	395	255	3674	76486	364	0	3015	73409
b13a	2	88	29	2673	17758	68	0	3636	18635
b13a	3	149	84	2662	17559	143	34	2529	17255
b13a	4	110	91	2415	16344	125	17	2637	17300
b14a	2	1384	1065	8299	379860	1451	0	9734	372845
b14a	3	2020	2307	6096	328408	2186	0	6753	321635
b14a	4	1472	1167	7425	348034	1913	0	7276	337371
b15a	2	3110	687	5807	1.04E+06	3143	0	6059	993602
b15a	3	4142	881	6035	946681	4291	0	7067	917239
b15a	4	3314	981	6967	901196	3959	0	4679	819516
b17a	2	9061	989	16941	3.71E+06	8894	0	16769	3.64E+06
b17a	3	12586	1550	13329	3.34E+06	12260	0	12824	3.23E+06
b17a	4	10457	1172	10640	3.19E+06	10996	0	12825	2.93E+06
b20a	2	3221	1925	11725	912468	3275	0	13264	948918
b20a	3	4605	2272	9300	816211	4234	0	11921	913147
b20a	4	3980	1930	8993	825921	4215	0	12705	958814

Manhattan Placement by Linear Programming

Guilherme Flach, Marcelo Johann, Ricardo Reis
{gafach, johann, reis}@inf.ufrgs.br
UFRGS – Universidade Federal do Rio Grande do Sul

Abstract

In this paper, we present the idea of a Manhattan placer. Placement is the physical design step where components of the integrated circuit, such as logic gates, memory blocks, are placed over the circuit area. This problem is NP-Complete, but can be solved efficiently when we relax overlap constraints. The Manhattan placement minimizes the total Manhattan wirelength between connected cells. We use a linear program performs the minimization of the total Manhattan wirelength with regardless overlapping. We briefly compare the Manhattan placement with others placement approaches: linear, quadratic and fourth order. This paper's first intend is to show that the Manhattan objective can be modeled as a convex program and in this way be efficiently solved.

1. Introduction

Placement is the physical design step where the components of the integrated circuit are placed over the circuit area. At first glance this seems to be a simple problem - a children game where the objective is to place pieces side-by-side. However when we start looking for minimizing the wire, which interconnects the components, and not allowing cells to overlap each other we start to play with a NP-Complete problem.

Since the placement is a NP-Complete problem, designers have proposed many heuristics. Among them the most successful are min cut [CAL 91] and quadratic placement [ALP 97]. The first one acts by recursively partitioning the components into regions inside the placement area. The second models the connections between cells as spring and so solve a linear system, which minimize the quadratic of total wirelength.

Quadratic placement has received great attention due to their ability to place a large amount of cells in little computational time and yielding as good result as min cut-based placers. The main difference among quadratic placers is the way in which they spread cells since cells are allowed to overlap. Fastplace [VIS 05] presented a spreading technique in which it adds new spring to push cells away. This technique is straightforward with the idea of quadratic placement it self, since spring are used to both minimize wirelength and spread cells.

This paper presents how to model the placement as a linear programming to minimize the Manhattan wirelength instead of the quadratic placement used in quadratic placement. Our modeling also relaxes the non-overlapping constraint and do not eliminate the need of spreading techniques.

2. Manhattan Placement

In this section we present how to model the problem of minimizing the Manhattan distance between connected cells by a linear program. The connections between cells are defined in the circuit net list. Net list can be viewed as a hypergraph where nodes represent the circuit components to be placed and the hyperedges represent the connections between a set of nodes (components). Some nodes, such as logic cells, are free to be placed. On the other hand, there is a set of nodes with fixed position as I/O pads. To model the placement using linear programming, we first must break hyperedges into edges connection. For that we create an edge between each pair of nodes connected by the hyperedge.

The sum of 2D Manhattan distance of connected cells as in given in equation (2.) where $\|.\|_1$ is the 1-norm which is mathematically equivalent to the absolute function.

$$\sum \left[\left\| x_i - x_j \right\|_1 + \left\| y_i - y_j \right\|_1 \right] = \underbrace{\sum \left\| x_i - x_j \right\|_1}_{\text{I}} + \underbrace{\sum \left\| y_i - y_j \right\|_1}_{\text{II}} \tag{2.1}$$

Equation (2.) can be expressed in matrix form as $\|\mathbf{Qx}\|_1 + \|\mathbf{Qy}\|_1$ where $\mathbf{Q}$ is the edge matrix with rows representing edges and the columns representing the nodes. The vectors $\mathbf{x}$ and $\mathbf{y}$ are the node positions. Each row in the matrix $\mathbf{Q}$ has just two non-zero entries: +1 and -1. The out coming node of an edge is set to +1, and the incoming node is set to -1. Since an edge has no direction, we define the out coming and incoming nodes as follow. If the edge connect two free nodes, the incoming and out coming nodes are chosen arbitrarily. If the edge connects a free node and a fixed one, we define as the out coming node the free node. Edges connecting two fixed nodes do not make sense for placement purposes and do not affect the result of the linear programming since they are constants, consequently they can be ignored.

Since we can minimize equation (2.) by minimizing $\|\mathbf{Qx}\|_1$ and $\|\mathbf{Qy}\|_1$ separately, hereafter we pay attention only to *x* dimension. To deal with free and fixed nodes we split the $\mathbf{x}$ vector and $\mathbf{Q}$ accordingly. We start by setting $\mathbf{x} = [\mathbf{x}_m \; \mathbf{x}_f]^T$ where $\mathbf{x}_m$ and $\mathbf{x}_f$ are related to free and fixed nodes respectively. Note that $\mathbf{x}_m$ is a

variable vector while $\mathbf{x}_f$ is a constant vector. **Q** is split as shown in equation (2.1) and must follow the partition done in **x** vector.

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q_{mm}} & \mathbf{Q_{mf}} \\ \mathbf{Q_{fm}} & \mathbf{Q_{ff}} \end{bmatrix} \tag{2.1}$$

Following the definition of in and out coming nodes for edges stated above and the splitting of **x** vector, we observe that:

- $\mathbf{Q}_{mm}$ is related to edges connecting two free nodes and therefore has two non-zeros in each row, i.e., the in and out coming nodes are is $\mathbf{Q}_{mm}$.
- $\mathbf{Q}_{mf}$ is a zero-matrix.
- $\mathbf{Q}_{fm}$ is related to the out coming nodes of edges connecting a free and a fixed node. Due to the definition of in and out coming nodes, this matrix has just one non-zero entry per row valued +1.
- $\mathbf{Q}_{ff}$ is related to the in coming nodes of edges connecting a free and a fixed node. Due to the definition of in and out coming nodes, this matrix has just one non-zero entry per row valued -1.

Now, by performing the multiplication $\|\mathbf{Qx}\|_1$, we have the equation (2.2).

$$\begin{aligned} \left\|\mathbf{Q}\times\mathbf{x}\right\|_1 &= \left\| \begin{bmatrix} \mathbf{Q_{mm}} & \mathbf{0} \\ \mathbf{Q_{fm}} & \mathbf{Q_{ff}} \end{bmatrix} \times \begin{bmatrix} \mathbf{x_m} \\ \mathbf{x_f} \end{bmatrix} \right\|_1 = \left\| \begin{bmatrix} \mathbf{Q_{mm}}\times\mathbf{x_m} \\ \mathbf{Q_{fm}}\times\mathbf{x_m} + \mathbf{Q_{ff}}\times\mathbf{x_f} \end{bmatrix} \right\|_1 = \\ &= \left\|\mathbf{Q_{mm}}\times\mathbf{x_m}\right\|_1 + \left\|\mathbf{Q_{fm}}\times\mathbf{x_m} + \mathbf{Q_{ff}}\times\mathbf{x_f}\right\|_1 \end{aligned} \tag{2.2}$$

The goal of the linear programming is to find values for $\mathbf{x}_m$ such that equation (2.) – the total Manhattan wirelength – is minimized. The minimization problem so can be written as in (2.3).

$$\text{minimize } \left\|\mathbf{Q_{mm}}\times\mathbf{x_m}\right\|_1 + \left\|\mathbf{Q_{fm}}\times\mathbf{x_m} + \mathbf{Q_{ff}}\times\mathbf{x_f}\right\|_1 \tag{2.3}$$

However (2.3) is not a standard linear programming. The standard linear program derivate from (2.3) is shown in (2.4).

$$\begin{aligned} &\text{minimize } s \\ &\text{subject to} \\ &\mathbf{Q_{mm}}\times\mathbf{x_m} \le \mathbf{t_1} \qquad \mathbf{Q_{fm}}\times\mathbf{x_m} + \mathbf{Q_{ff}}\times\mathbf{x_f} \le \mathbf{t_2} \\ &-(\mathbf{Q_{mm}}\times\mathbf{x_m}) \le \mathbf{t_1} \qquad -(\mathbf{Q_{fm}}\times\mathbf{x_m} + \mathbf{Q_{ff}}\times\mathbf{x_f}) \le \mathbf{t_2} \\ &sum(\mathbf{t_1}) + sum(\mathbf{t_2}) - s \le 0 \end{aligned} \tag{2.4}$$

A linear program is a convex program and can be solved efficiently. However, this linear program is a constrained program, which is not so efficient as a non-constrained problem and there is no necessarily unique solution for this program. For instance, in a quadratic formulation there is just one solution and it can be find simply solving a linear system of equations, which is more efficient than solving a general convex program.

Example

In this subsection we present a little example of the Manhattan placement. As we have mentioned, the Manhattan placement can be solved separately for dimensions *x* and *y*. Therefore we take just a one-dimensional case. The placement example is shown in the Fig. 3 – Manhattan Placement Example.Fig. 3.

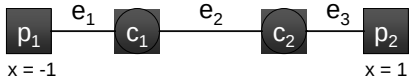


Fig. 3 – Manhattan Placement Example.

For this example, the matrix **Q** is given by (2.5).

$$\text{where } \mathbf{Q_{mm}} = \begin{bmatrix} 1 & -1 \end{bmatrix},\ \mathbf{Q_{fm}} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix},\ \mathbf{Q_{ff}} = \begin{bmatrix} -1 & \\ & -1 \end{bmatrix} \tag{2.5}$$

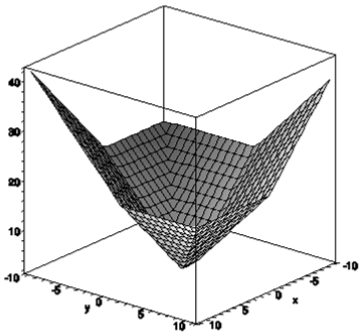


Fig. 4 – Objective function for this section example.

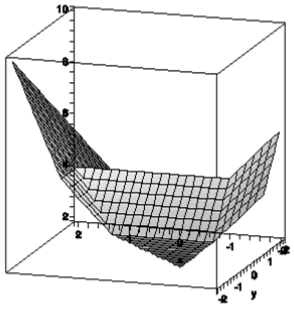


Fig. 5 – The optimum solution is inside the flat bottom region on the graphic.

Fig. 4 shows the graphic of the objective function, which models this section's example. As we can see it is a convex function. Fig. 5 presents the same function, now highlighting the optimum region.

3. Placement Comparison

In this section, we compare the placement with four different approaches: Manhattan, linear, quadratic and fourth order placement. The three last approaches are obtained from Convex Optimization book [BOY 04]. As we can see, Manhattan, quadratic and fourth-order placements tend to spread more the cells than linear placement. Quadratic and fourth order penalizes more long connections

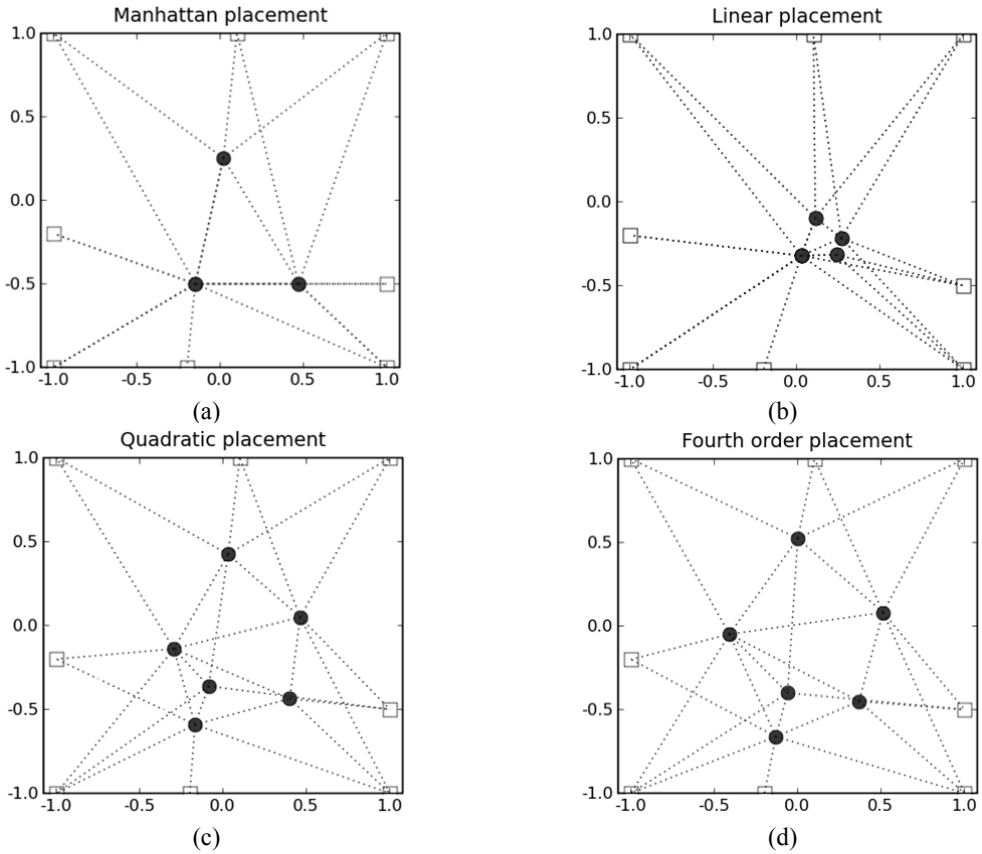


Fig. 6 – Placement Approaches Comparison

4. Conclusions

In this paper, we have presented a modeling for Manhattan placement through linear programming. The paper's first intend was to present a convex (linear) modeling for minimizing the Manhattan wirelength between connected cells. As we have mentioned, the linear modeling is not so efficient as the quadratic modeling, however the Manhattan can be express more accurately the final routing of the circuit.

5. References

[CAL 91] CALDWELL, A. E.; KAHNG, A. B.; MARKOV, I. L. **Can recursive bisection produce routable placements.** In Proc. ACM/IEEE Design Automation Conf., pages 477–482, 2000.Gordian: VLSI placement by quadratic programming and slicing optimization. IEEE Trans. Computer-Aided Design, 10(3):356–365, 1991.

[VIS 05] VISWANATHAN, N.; CHU, C C-N. **FastPlace: Efficient Analytical Placement using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model.** IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems, Vol 24, Issue 5. May 2005.

[ALP 97] ALPERT, C. J.; CHAN, T.; HUANG, D. J.-H.; MARKOV, I.; YAN, K**. Quadratic Placement Revisited.** In: DAC '97: Proceedings. ACM Press, 1997.

[BOY 04] BOYD, S. P.; VANDENBERGHE, L. **Convex Optimization.** Cambridge University Press 2004.

RAMAGT - Radix-2^m Array Multipliers Automatic Generation Tool

Diego P. Jaccottet, Jonatas M. Roschild, Leandro Z. Pieper, Eduardo A. Costa, Sérgio J. de Almeida

diego_porto_j@yahoo.com, jonatasroscild@bol.com.br
leandrozaf@pop.com.br,{ecosta,smelo}@ucpel.tche.br

Universidade Católica de Pelotas - UCPel

Abstract

In this paper it is presented a new Automatic Generation Tool (RAMAGT) that generates Radix-2^m Array Multipliers. With the increase of digital systems complexity, textual descriptions of the circuits became very hard to be made by hand. This problem can be solved by using tools that can automatically generate textual description of digital circuit. As multiplication is a common operation in many digital circuits such as Digital Signal Processors for example, it has motivated us to create a tool that generates multiplier circuit descriptions for VHDL and BLIF formats automatically. This tool generates n bit wide multiplier circuits and binary and hybrid encoding are used in group of m bits of the Radix-2^m Array Architecture. To validate the multipliers generated by the RAMAGT tool we have used Quartus II and SIS environments.

1. Introduction

Multiplier modules are common to many DSP applications and general purpose processors. The state of the art has pointed to multipliers that can operate on 64-bit wide, which is very hard to describe by hand due to the complexity of this type of circuit. As circuit designers do not have any time to spend making the circuit textual description by hand, mainly due to the time-to-market aspect, thus it is very important the development of tools that can automatically generate state of the art digital circuits. In this work we propose a RAMAGT tool that can generate descriptions of radix-2^m array multiplier circuits automatically.

The multipliers generated by the RAMAGT tool were proposed by [COS 02] and consists of pure array multipliers that can operate on 2's complement and use a radix-2^m encoding to reduce the partial product lines. The RAMAGT tool can generate BLIF (Berkeley Logic Interchange Format) [BER 07] and VHDL descriptions of *n*-bit wide multipliers automatically. Besides, binary and hybrid encoding can be used on group of *m* bits. The hybrid encoding can reduce the glitching activity along the array multiplier and this encoding can be set in the RAMAGT tool when low power multipliers design are taken into account. Another important aspect of the tool is the possibility of enabling different group of *m*-bit operation in the array multipliers easily. This aspect enables the array multiplier to reduce significantly the number of partial product lines. In this work, we have limited the generation of multipliers to group of m=6 and thus, the binary and hybrid multipliers can operate on the maximum of radix-64.

While the BLIF description enables the design of the multipliers at the gate level, the VHDL description enable the multipliers to be synthesized onto FPGAs. With this flexibility proposed by the RAMAGT tool, the designer can choose the most adequate design environment depending on the metrics to be considered (area, performance or power). Besides, another important issue is that commercial tools such as Mentor Graphics [GRA 08] accept BLIF and VHDL as input descriptions in its flow [OLI 05].

2. Related Works

A lot of work has been put in the development of automatic generators of multiplier circuits. In [FAD 93] it is proposed a M-bit by N-bit Booth encoded parallel multiplier generator. In [CHE 01] a new low-power constant multiplier with a generator written in C++ is used to generate technology-independent VHDL code of the constant multiplier for different input specifications. In [TSO 02] an automatic hardware generator is developed for computing multiplier-based arithmetic functions where verilog codes are generated. In [QIA 03] a regularized multiplier generator is proposed creating VHDL descriptions for signed and regular multiplier. One of the most significant publication is [TSO 05] where a module generator called Mullet is presented to produce a near-optimal parallel multipliers, creating multipliers architectures that broken down into a partial product generator (PPG) and a partial product summer (PPS). The tool can also be used to explore tradeoffs between architectures. Although the great amount of tools to generate automatic descriptions of multipliers present in the literature, none of them has taken into account the flexibility of generating another textual description rather than VHDL or Verilog. Besides, neither of them considers another operand encoding rather than binary. In our tool radix 2^m hybrid array multipliers are generated by RAMAGT tool as a way of reducing the switching activity along the array, and thus enabling low power design.

3. Overview of the radix-2^m array multiplier

In this section we summarize the methodology of [COS 02] for the generation of regular structures for arithmetic operators using signed radix-2^m representation.

For the operation of a radix-2^m multiplication, the operands are split into groups of m bits. Each of these groups can be seen as representing a digit in a radix-2^m. Hence, the radix-2^m multiplier architecture follows the basic multiplication operation of numbers represented in radix-2^m in 2's complement.

This operation is illustrated in fig. 1, for a radix-16 multiplication example. For the W-m least significant bits of the operands unsigned multiplication can be used. The partial product modules at the left and bottom of the array need to be different to handle the sign of the operands.

For this architecture, three types of modules are needed. Type I are the unsigned modules. Type II modules handle the m-bit partial product of an unsigned value with a 2's complement value. Finally, Type III modules that operate on two signed values. Only one Type III module is required for any type of multiplier, whereas $2^{\frac{W}{m}}$ -2 Type II modules and $(\frac{W}{m} - 1)^2$ Type I modules are needed.

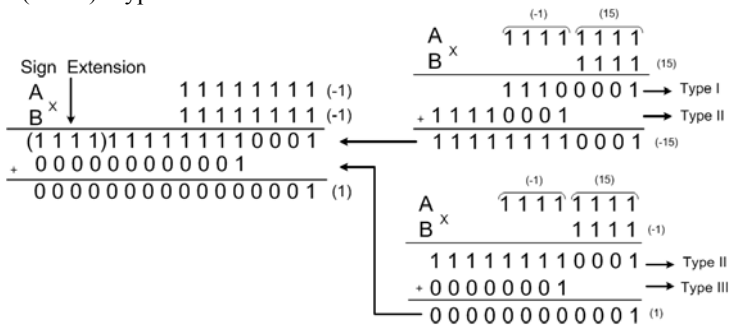


Fig. 7. Example of a 2's complement 8-bit wide radix-16 multiplication

3.1 Hybrid operand encoding

The example show in fig. 1 can be easily adapted for another operand encoding. In our work we have used a different encoding into the group of m-bits. This different scheme is named Hybrid encoding and it represents a compromise between the minimal input dependency presented by the binary encoding and the low switching characteristic of the Gray encoding. The idea of the Hybrid code is to split the operands in groups of m-bits, encode each group using the Gray code and use the Binary approach to propagate the carry between the groups. In this manner, the number of transitions inside each group is minimized and a regular structure can be easily built. Tab. 1 exemplifies the Hybrid encoding for 4-bit numbers and for m=2. The Hybrid multipliers generated by the RAMAGT are limited in this work to m =6.

Tab. 1 - Hybrid code presentation for m=2

Dec	Hybrid	Dec	Hybrid	Dec	Hybrid	Dec	Hybrid
0	0000	4	0100	8	1100	12	1000
1	0001	5	0101	9	1101	13	1001
2	0011	6	0111	10	1111	14	1011
3	0010	7	0110	11	1110	15	1010

4. Methodology

In this section we will briefly show how the RAMAGT tool was developed. The tool was implemented by using two programming languages. One of them is C++ that generates executable files in a faster way. The other is the CSL (C Scripting Language) [KOC 02] that is a powerful and easy programming language that follows the C syntax very closely and it is used like an interpreter. A language that uses scripts has the advantage that the code does not need to be compiled [OUS 98], so that the projects can be easily generated. The C++ language is used to link the interface of the projects requirements with the user. It allows that many different projects can be done at the same time.

In fig. 2 we show the methodology used to generate a multiplier description. The main program executes the secondary one with the language format as parameter (BLIF or VHDL), allowing multiple secondary programs with different parameters and languages to be opened at the same time. The secondary program lists the tree of projects for the given language and it executes the selected project from the tree with the given parameters set. It will execute according to the configuration of the language (a configuration file on the language folder includes the way it will execute, its extension and the description for the language). It will also send the locations for the project to execute appropriately. The project folders contain the files to be executed

and any extra files (the main CSL executable and its companions may require to execute appropriately). It also contains the notes on the project and a file describing the parameters and initial values for them.

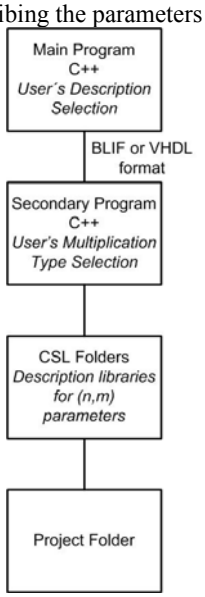


Fig. 2 - Methodology used to create the RAMAGT tool

5. The RAMAGT Tool

In this section we present the steps in the RAMAGAT tool to generate multiplier descriptions of radix-2^m array architectures. The main screen of the RAMAGT tool is showed in fig. 3. In this screen we can select the format of the multiplier circuit description (BLIF or VHDL). After the selected format, a new window will be opened as can be shown in fig 4. It should be observed that a good characteristic of this tool is that we can make many projects at the same time.

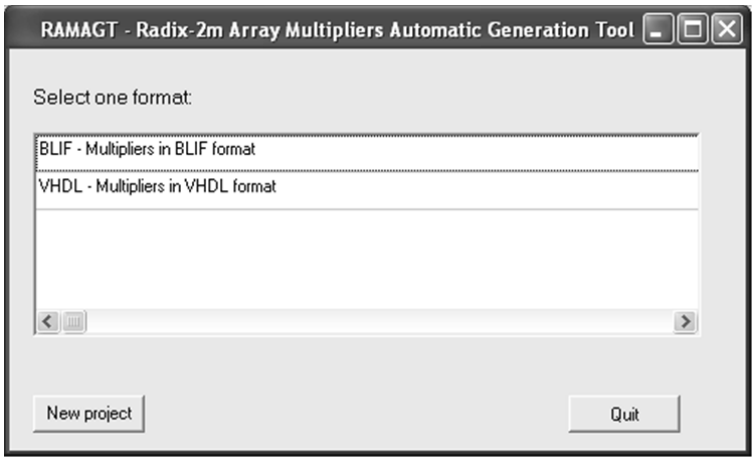


Fig. 3 - RAMAGT main screen

In the next step the type of operand encoding of the multiplier can be selected (Binary or Hybrid encoding), as can be observed fig 4. Once we have selected the encoding for the multipliers, another window is opened and then we will able to select the number of groups of bits (m) and the length of the operands, as is shown in fig. 5. After that we just click on "Generate" button, and it will be created a folder in the current path called "Projects" where will be the files containing the textual description of the multiplier. To validate the descriptions created by the tool we introduce the VHDL files in Quartus II from Altera or the BLIF format in the SIS environment.

6. Conclusions and Future Work

We have presented a new RAMAGT tool for the automatic generation of radix-2^m array multiplier. The tool generates automatically BLIF or VHDL descriptions of n bit and m group of bits of array multipliers. The circuit descriptions enable that the multipliers can be synthesized to FPGAs (VHDL description) or at gate level by using SIS environment (BLIF description). As future work we intend to introduce in the RAMAGT new multipliers architectures that use the low power techniques proposed by [PIE 08].

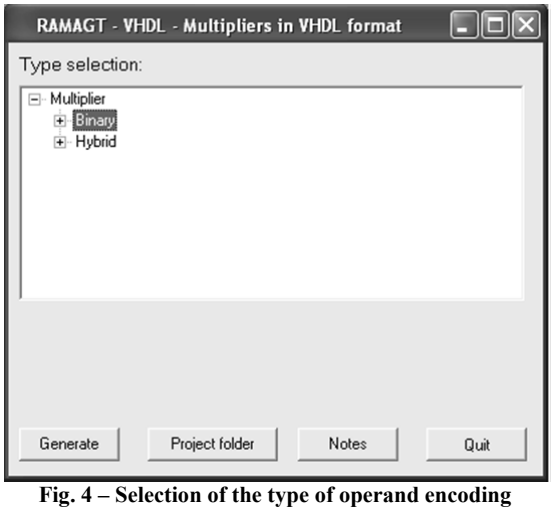


Fig. 4 – Selection of the type of operand encoding

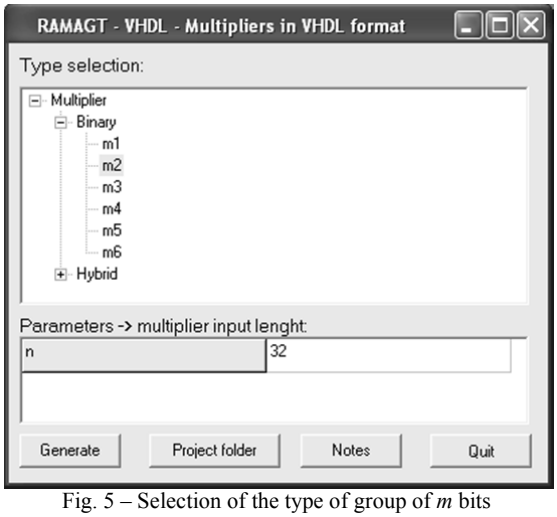


Fig. 5 – Selection of the type of group of *m* bits

7. References

[BER 07] BERKELEY, U. **BLIF - Berkeley Logic Interchange Format**. University of California. At http://embedded.eecs.berkeley.edu/ . 2007.

[CHE 01] CHENG-YU, P.; AL-KHALILI, A.J.; LYNCH, W.E; **Low-Power Constant-coefficient multiplier generator.** ASIC/SOC Conference, 2001. In *14th Annual IEEE Int. 12-15* Sept. 2001 Page(s):185 – 189

[COS 02] COSTA, E.; MONTEIRO, J.; BAMPI, S. **A New Architecture for Signed Radix 2^m Pure Array Multipliers**. In *IEEE International Conference on Computer Design*, pages 112-117, 2002

[FAD 93] FADAVI-ARDEKANI, J. **M×N Booth encoded multiplier generator using optimized Wallace trees**. *Very Large Scale Integration (VLSI) Systems*, IEEE Trans. on Volume 1, Issue 2, June 1993 Page(s):120 – 125

[GRA 08] GRAPHICS, M.; **QuickPath User's and Ref. Manual**. *Mentor Graphics*. At http://www.mentor.com . 2008

[KOC 02] KOCH, P. **C Scripting Language –Reference Manual V. 4.4.0.** At http://csl.sourceforge.net/csldoc/index.htm Last Revision: 2002

[OUS 98] OUSTERHOUT, J. K. **Scripting: Higher Level Programming for the 21st Century.** IEEE Computer Magazine. March 1998

[PIE 08] PIEPER, L.; COSTA, E.; ALMEIDA, S.; BAMPI, S.; MONTEIRO, J. **Efficient Dedicated Structures for Radix-16 Multiplication.** In XIV Iberchip – IWS, 2008.

[QIA 03] QIAN, Yu.; WANG, Dong-Hui;. **A design of regularized multiplier generator.** ASIC, 2003. Proceedings. In *5th International Conference* on Volume 2, 21-24 Oct. 2003 Page(s):1269 - 1272 Vol.2

[TSO 02] TSO-BING, J.; JENG-HSIN, J.; MING-YU, T.; SHEN-FU H.. **A high performance function generator for multiplier-based arithmetic operations.** ASIC, 2002. Proceedings. 2002 IEEE Asia-Pacific Conference on 6-8 Aug. 2002 Page(s):331 - 334.

[TSO 05] TSOI, K. H.; LEONG, P.H. GENDEREN, A. **Mullet - a parallel multiplier generator.** In *Field Programmable Logic and Applications*, 2005. Int. Conference on 24-26 Aug. 2005 Page(s):691 - 694.

Dynamic Power Dissipation in Pass Transistor Circuits

Reginaldo da N. Tavares
regi@unipampa.edu.br

Universidade Federal do Pampa

Abstract

Pass transistors are used to design dedicated circuits such as: transmission gates, flip-flops, multiplexers, selectors and three state buffers. However, they are not commonly used to implement random logic circuits. First, design of pass transistors is more difficult than conventional static CMOS gates, and secondly, few techiniques are available to estimate circuit performance. This paper presents a method to estimate dynamic power dissipation in pass transistor circuits. To estimate power dissipation two different contributions are introduced, they are dynamic control power dissipation and dynamic pass power dissipation.

1. Introduction

Static CMOS gates perform 1 to 0 and 0 to 1 logic transitions. These logic transitions either charge or discharge the output load capacitance. In both cases the power demanded from the power source is dissipated [RAB 03][CHA 95]. Dynamic power dissipation of a CMOS logic gate can be estimated by equation ***Pdyn = a.Cl.f.Vdd^2* (1)**. *Pdyn* is the dynamic power dissipation, *a* is the switching activity of a CMOS gate, *Cl* is the load capacitance, *f* is the clock frequency, and *Vdd* is the voltage of the power source. This equation shows that voltage is an important contribution of power disssipation. Therefore, voltage reduction is effective to design low power circuits. Pass transistors circuits can be used to design low power circuits because pass circuits are able to implement logic circuits with reduced swing voltage. This paper present a method to estimate dynamic power dissipation of pass transistor circuits. The method is based on the equation described before. To perform the method two dynamic power contributions namely dynamic *control* and dynamic *pass* are described.

2. Pass Transistor Circuits

Pass transistor circuits and static CMOS logic gates evaluate Boolean functions in different ways. While a static CMOS gate connects the output node to *Vdd* or *GND*, where *Vdd* corresponds to 1 and *GND* corresponds to 0, a pass transistor circuit connects a primary logic input to the primary output node. The output value depends on the primary input variable. In this paper a pass transistor circuit is defined as follows:

- A pass transistor circuit is a collection of paths whith a single common output node.
- An ON path is a pass path between a single primary input and a primary output.
- A pass transistor circuit has only one path for each combination of its primary input variables.

Figure 1 shows the pass transistor and static CMOS circuit design. The leftmost draw shows the basic idea of a pass transistor circuit. It shows the input variables divided in two sets of variables, i.e. the pass input variables and the control input variables. An internal circuit path will connect a single input variable to the output node of the circuit. On the other side, the static CMOS gate has a pull down and a pull up transisitor network.

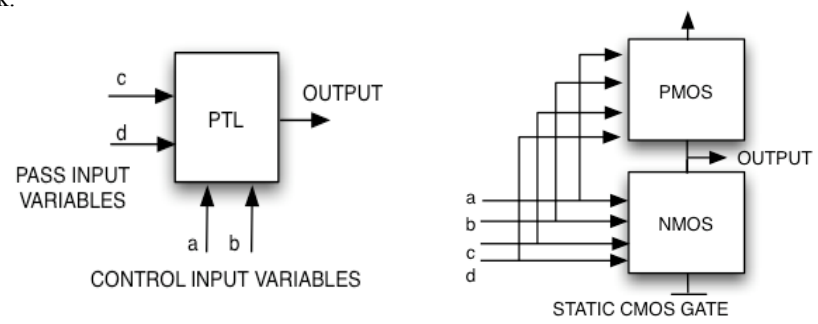


Fig. 1 – A pass transistor circuit structure and a static CMOS gate.

Although only one path is necessary to evaluate a logic function, pass circuits can have multiple paths able to transmit input signals. Considering the definitions 1, 2 and 3, a pass circuit with *n* control input variables and

m input variables works like a multiplexer of *n* control variables. Any Boolean function can be implemented by a multiplexer. However, a large function implemented by a monolithic multiplexer is unpractical circuit design techinique. Therefore, in order to implement large and complex pass transistor circuits, large multiplexers may be decomposed in a network of multiplexers. Multiplexer decomposition is well done by binary decision diagrams (BDDs) [AKE 78].

3. Pass Transistor Circuits and BDDs

Some important features on BDDs pass transistor circuits are:

- Large and complex logic functions can be implemented directly from BDDs.
- Pass transistor circuits based on BDDs evaluate the logic function correctly since BDDs represent Boolean functions.
- The function can be evaluated mapping each BDD node to 2-input multiplexer.
- Circuits based on BDDs avoid sneak paths since there is a single path between a primary input and a primary output.
- The BDD can be used to estimate area, delay and power dissipation.

A BDD circuit may be implemented with a set of 2-input multiplexers. The multiplexers can be shared by several paths. It decreases the total number of muxes, but, on the other hand, their fanout increases frequently. Figure 2 shows a 2-input multiplexer based on NMOS transistors.

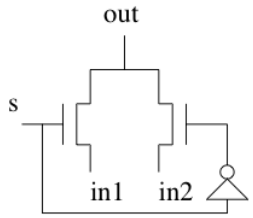


Fig. 2 – 2-input NMOS multiplexer.

Input pass variables are also used as control variables in BDD pass transistor circuits. In this case only 1 and 0 are propagated through the pass circuit. Since 1 and 0 are implemented by *Vdd* and *GND* respectively, a pass path have to connect the *Vdd* or *GND* to the output node. It avoids to load each pass input variable with large capacitances. Several works that combines BDDs and pass transistors were presented. For example, [KAZ 96][BUC 97] introduced methods to synthesize practical pass transistors circuits directly from BDDs .

4. Pass Circuits Based on NMOS Transistors

In pass transistor circuits logic signals cannot always be transmitted without voltage degradation. It is due to the characteristics of the MOS transistors. For instance, when a high logic signal passes through a NMOS transistor, the output node starts to switch off at voltage *Vdd-Vtn*, and no current is conducted. On the other hand, when a low logic signal passes through a PMOS transistor, at the voltage *Vdd+Vtp* the PMOS transistor switches off and no current is conducted. *Vdd*, *Vtn* and *Vtp* are the voltage of the power source, the threshold voltage of the NMOS transistor, and threshold of the PMOS transistor respectively. Low level signals are well transmitted by NMOS transistors and high level signal are well transmitted by PMOS transistors. Clearly, pass transitors degrade one logic level. Since pass paths are composed by single or series of pass transistors, voltage degradation is an important problem in pass transistor design.

In order to avoid voltage degradation pass switches could be implemented by transmission gates. Transmission gates are switches implemented by a parallel association of a NMOS and a PMOS transisitor. Transmission gates remove the voltage degradation problem because at least one transistor can transmit the input logic signal. However, the input and output capacitances increase significantly. Pass paths with several transmission gates may have large capacitances, and, therefore, delay and power dissipation are affected. A 2-input multiplexer based on transmission gate is shown in figure 3.

If a pass path is implemented with single switches, normally NMOS transistors are used. NMOS transistors are used because they are smaller and faster than PMOS transistors.

Considering low power target implementations, NMOS transistors have an important advantage. They can be used to transmit logic signals with reduced voltage. In NMOS pass paths the voltage swings between *GND* and *Vdd-Vtn*, instead of *GND* and *Vdd* voltages.

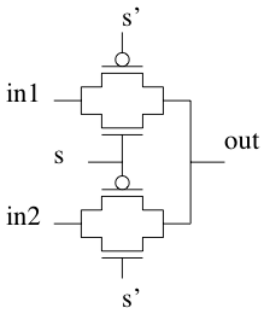


Fig. 3 – A 2-input transmission gate multiplexer.

5. Dynamic Power Dissipation in Pass Transistor Circuits

Dynamic power dissipation in pass circuits are related to charge and discharge all capacitances associated to the circuit. Two different capacitances can be seen in pass transisitor circuits. They are the pass switched capacitances and the gate capacitances. Pass switched capacitances are capacitances related to the connections between transistors. The gate capacitance are related to the transistors gate capacitance. Therefore, the dynamic power dissipation depends on the dynamic pass power dissipation and the dynamic control power dissipation described by the equation ***Pdyn = Ppass + Pcontrol*** (2).
Dynamic Pass Power Dissipation (Ppass) can be estimated by the equation ***Ppass = pi.Cpi.f.(Vdd-Vtn)^2*** (3). *Ppass* is the dynamic pass power dissipation, *pi* is the switching activity at node *i*. *Cpi* is the switched pass capacitance associated to node *i, f* is the clock frequency, and *Vdd-Vtn* is the maximum swing voltage. The difference between equations 1 and 3 is the circuit voltage. Since the maximum voltage transmitted by a pass transistor is *Vdd-Vtn*, the swing voltage is reduced, and it decreases the power dissipation.

Pass switched Capacitance (Cp) are created due to the drain-source connection between pass transistors. *Cp* is the contribution of three different capacitances as shown by equation ***Cp = Cj + Cw + Cg*** (4). *Cj* is the drain junction capacitance associated to the node. *Cw* is the interconnection capacitance, and *Cg* is the gate capacitance due to inverters and buffers eventually placed between pass transistors. *Cp* can be estimated for each node when the BDD is traversed. To compute the fanout all incoming edges of each node are taking into account. The fanout is used to multiply the drain junction capacitance of a pass transistor.

Dynamic Control Power Dissipation (Pcontrol) is related to the power dissipated to drive all switches in the circuit. The primary inputs are directly connected to the transistor gates. Dynamic control power dissipation is an important factor and it can be the dominant dynamic power in the circuit. The fanout of a primary input driver may be large, because each input variable is represented by several nodes in a BDD. Gate capacitance is normally larger than the drain junction capacitance. There is no voltage reduction to drive any transistor gate in the circuit. The swing voltage of the input primary variables is between *GND* and *Vdd*. Dynamic Control Power Dissipation can be estimated by the equation ***Pcontrol(x) = pt(x).Clx.f.Vdd^2*** (5). *Pcontrol(x)* is the dynamic control power dissipation of the primary input variable *x*. *pt(x)* is the switching activity of a primary input variable *x*. *Clx* is the parasitic capacitance to be driven by the primary input variable *x*. *f* is the clock frequency. *Vdd* is the maximum input voltage.

6. Experiments

In order to verify which power dissipation is dominant, both control and pass power dissipation were estimated for a set of well known benchmark circuits. Each pass circuit was represented by a monolithic BDD. Buffers were not inserted in the circuit. Gate and diffusion capacitance were considered the same value for a generic 0.25um CMOS process. The supply voltage is 2.5V. The input signal probability is 0.5. The threshold voltage is 0.5V. Then, for all circuits, control and pass power were estimated.

All nodes in the circuit were considered during the computation of the power estimation because all paths will switch. For example, figure 4 shows a circuit example that implements the function *F=a.b'.c' + a.b'.c.d' + a'.c.d' + a'c'*. For the input vector [abcd]=[0000], all paths in the circuit are charged. However, only the ON path contributes to evaluate the function. The results are shown in table 1. Table 1 shows the name and the size of the circuit. The column **Pcontrol/Ppass** shows the ratio of Dynamic Control Power Dissipation to Dynamic Pass Power Dissipation for each circuit.

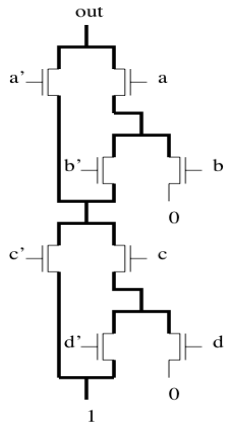


Fig. 4 – Pass transistor circuit.

Tab.1 – Dynamic control and pass power dissipation.

Circuit	In/Out	Pcontrol/Ppass	Circuit	In/Out	Pcontrol/Ppass
alu4	14/8	2.0	apex7	49/37	2.5
alu2	10/6	1.9	cordic	23/2	5.0
apex2	39/3	5.5	example2	85/66	3.8
apex4	9/19	1.9	frg2	143/139	4.8
apex5	117/88	3.0	x1	51/35	3.6
apex6	135/99	2.5	x3	135/99	2.5
count	35/16	2.5	pcler8	27/17	2.5
c8	28/18	2.2	cc	21/20	2.6
decod	5/16	5.6	tcon	17/16	2.1
f51m	8/8	1.7	unreg	26/16	2.1
pcle	19/9	2.4	x4	94/71	2.4
ttt2	24/21	2.4	i1	25/16	3.7

7. Conclusion

This paper shows that power dissipation in pass transistors based on BDDs is given by two different contribution: the dynamic power dissipation, and the dynamic pass power dissipation, i.e., ***Pdyn = Pcontrol + Ppass***. The results from the experiments show that the dynamic control power dissipation is the principal source of power dissipation. It occurs because the transistor gates are important capacitances in the circuit. If the main contribution of the dynamic power dissipation can be found, then strategies to reduce its impact can be developed.

8. Referências

[AKE 78] AKERS, Sheldon B.; **Binary Decision Diagrams.** IEEE Transactions on Computers, Vol.c-27, NO.6., June 1978, pp.509-516.

[KAZ 96] KAZUO, Yano; YASUHIKO, Sasaki; KUNIHITO, Rikino; KOICHI, Seki. **Top-Down Pass Transistor Logic Design.** IEEE Journal of Solid State Circuits. Vol.31,NO.6.,June 1996,pp.762-803.

[BUC 97] BUCH, Premal; NARAYAN, Amit; NEWTON, A.Richard; VINCENTELLI, A.S.; **On Synthesizing Pass Transistor Networks.** In Proc. International Symposium on Logic Synthesis, 1997.

[RAB 03] RABAEY, Jan M.; CHANDRAKASAN, A.; NIKOLIC, B.; **Digital Integrated Circuits: A Design Perspective.** Prentice-Hall, 2003.

[CHA 95] CHANDRAKASAN, A.; BRODERSEN, R.; **Low Power Digital CMOS Design.** Kluwer Academic Publishers, 1995.

FAULT TOLERANCE

Asymmetric and Symmetric Transistor Sizing to Reduce SET Sensitivity in Integrated Circuits

Cristiano Lazzari [1,2], Thiago Assis [1], Fernanda Lima Kastensmidt [1], Gilson Wirth [1], Lorena Anghel [2], Ricardo Reis [1]

{clazz,thiago.assis,fglima,wirth,reis}@inf.ufrgs.br, lorena.anghel@imag.fr

[1] PGMICRO-PPGC - Federal University of Rio Grande do Sul

[2] TIMA Laboratory - Institut National Polytechnique de Grenoble
Grenoble - France

Abstract

This work presents a new transistor sizing algorithm based on asymmetric and symmetric transistors sizing to reduce combinational logic circuit sensitivity to single event transients. The sizing strategy takes into account the logical and electrical masking of the transient voltage pulses to analyze the sensitivity of each node of the circuit. A novel propagation modeling finds the maximum acceptable transient voltage pulse duration in each node to independently size pull-up and pull-down transistors. Results show that this asymmetric sizing involving PMOS and NMOS transistor is more efficient in area overhead and power consumption in comparison with a symmetric methodology.

1. Introduction

In deep submicron technologies, decreasing feature sizes and lower voltage levels cause an increase in the soft error rate (SER) in integrated circuits. When a particle strikes a sensitive region of a semiconductor device with a particular energy, the resulting electron-hole pair generation may change the logical state of the circuit node.

When this temporary current disturbance occurs in a combinational logic circuit, the effect is known as single event transient (SET). SETs may lead a system to an unexpected response whether it propagates to a memory element or a primary output (PO) of a circuit. If a particle directly hits a memory element, the logic value stored may be changed causing the erroneous functioning of the circuit. This changing state in memory elements is known as single event upsets (SEU).

Historically, memories have been concerned for single event upsets. Efficient solutions to memory protection are presented in [BES 94], [CAL 96], [ROC 88]. However, since the transition time of the logic gates is getting shorter and clock frequencies are significantly increasing in nanometric technologies, errors in combinational logic parts are increasing and error rates will reach the same levels as in memories in the near future.

A recent work predicts SERs in combinational logic circuits comparable to memory elements by 2011 [WHI 91]. For this reason, the design of combinational logic tolerant to transient faults is mandatory. This paper proposes a new transistor sizing technique for soft errors protection in combination logic circuits. The main characteristic of the proposed methodology is to find the smallest accepted transistor widths to attenuate SETs in the nodes of a combinational circuit.

Another important point is that pull-up and pull-down transistors are independently sized, minimizing the area overhead and the power consumption. In other words, we apply asymmetric transistor sizing to attenuate SETs with minimized area overhead. Works presented in the literature are based in symmetric models to size pull-up and pulldown blocks. At the following sections a brief discussion about the problem and the results of the algorithm.

2. RelatedWorks

Radiation harden combinational logic has been discussed in the last years due to the importance that SET effects have shown in submicron technologies. Some of this techniques are shown in this section. A gate duplication methodology is presented in [SHI 02]. Results show a 50% area overhead for 90% soft error failure rate reduction. It is clear that the area overhead is significantly smaller than other techniques such as TMR, although special attention has been given in the last years to find better solutions.

A SER analysis in combinational logic circuits and a partial gate multiplication methodology for soft error protection is proposed in [MOH 03], [HEI 06]. These works present a very important contribution in relation to

SET analysis. The SET propagation is analyzed with respect to logical and electrical masking. A drawback of this partial multiplication method is the complexity to test cells sharing the same inputs and outputs.

A gate sizing method is proposed in [NIE 06]. This technique presents smaller area overhead than the multiplication methodology without compromising the circuit test due to the absence of redundancy. This gate sizing technique targets soft error failure rate reduction by selectively sizing the most sensitive nodes. The sensitivity analysis in combinational logic circuits has been discussed in some researches such as the works presented in [ZHO 06] and [CAZ 05].

An accurate and computer efficient model for SET is presented in [DHI, 05]. The model evaluates the behavior of a SET in a circuit node as function of the driving gate resistances and the capacitances involved with the faulting node. The work presented in this paper uses this model to analyze the sensitivity in circuit nodes. Section 4 gives further details about the model.

3. The Transistor Sizing Strategy

The transistor sizing strategy proposed in this paper consists of finding the smallest transistor widths of each circuit gate for SET attenuation. We consider logical and electrical masking in our sizing strategy. The *logical masking* occurs when a SET provoked by a particle is not propagated to a primary output (PO) due to the logic of the circuit. In other words, the pulse is masked as function of the vector applied in the primary inputs (PI) of the circuit. Controllability and observability techniques are used to define the logical masking of a node.

Controllability in combinational logic circuits denotes the ability to a state be set in a node. Observability is a measure for how well a state in a internal node can be known at the primary outputs (PO). A pulse in one of the gate inputs is propagated through the gate only if a non-controlling value is applied at the other input. In the OR gate, the same situation is considered, where the pulse propagates through the gate only if the noncontrolling value is applied at the other input.

Electrical masking can be defined as the electrical attenuation of a pulse in a node by the gates in a path to the point that the SET does not affect the results of the circuit.

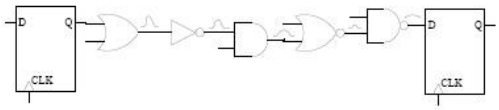


Figure 1: The Electrical Masking

Figure 1 shows an example of SET degradation. This degradation is the base to the electrical masking, where the pulse is degraded as function of the electric characteristics of the gates in the path. The pulse can be captured by the memory element if it is not enough degraded. The model presented by [WIR 07a] and [WIR 07b] was used to calculate electrical masking factor at this tool.

We consider the logical and electrical masking as the sensitivity of a circuit. The logical masking represents the probability of a transient pulse be masked by the logic function of the circuit, and the electrical masking describe if a transient pulse in a node is not propagated to the POs. Thus, the sensibility of a circuit is given by

$$S_{circuit} = \sum_{n=1}^{N} (1 - L_n) \cdot (1 - E_n) \quad (1)$$

where L_n is the logical masking and E_n is the electrical masking. The logical masking L_n is a probability value. Bigger logical masking means smaller probability of a transient pulse be detected in the circuit outputs. The electrical masking E_n is a binary value where "0" indicates that the transient pulse is totally attenuated and "1" indicates that the transient can be seen in the outputs.

Algorithm 1 The transistor sizing for SET attenuation.

```
Require: Set of gates G, Set of Nets N, Set of outputs O,
    Maximum sensitivity M, Max critical charge Q_c, De-
    sired circuit sensitivity S_desired
Ensure: Set of gates with sized transistors G_new
 1: G_new ⇐ ∅
 2: for all n ∈ N do
 3:    L_n ⇐ calculateLogicalMasking( n );
 4:    E_n ⇐ calculateElectricalMasking( n, Q_c );
 5:    S_n ⇐ (1 − L_n) · (1 − E_n)
 6: end for
 7: V ⇐ O        {Nets to visit, starting from the outputs.}
 8: while V ≠ ∅ do
 9:    for all n ∈ V do
10:       g ⇐ getFaninGateConnectedToNet( n );
11:       if S_n > M then
12:          τ_n ⇐ getMaximumSET( n, g );
13:          g_new ⇐ sizeTransistors( s, g, τ_n );
14:          G_new ⇐ G ⋃ {g_new} \ {g}
15:       end if
16:       I ⇐ getGateInputs( g );
17:       V ⇐ V ⋃ I \ {n}
18:    end for
19: end while
```

Algorithm 1: Transistor Sizing

The proposed transistor sizing strategy is presented in Algorithm 1. First lines (2-6) define the cicuit sensitivity as shown in (1). The transistor sizing strategy starts at line 8, where every node n of the circuit is visited in order to find the minimum transistor width to each gate g connected to this node. It is important to note that only nodes with the sensitivity bigger than the maximum defined sensitivity M are evaluated (line 11). Function getMaximumSET(n,g) (line 12) finds the maximum pulse duration τ_n in the node n that is suppressed before the primary outputs. The transistor sizing algorithm to a gate g is function of this SET duration τ_n.

Function sizeTransistors(s,g, τ_n) (line 13) continuously increase the transistors width until the SET in the node n be smaller than τ_n. When this situation is reached, we consider the transistors of the gate *g* are sized as expected to the charge Q_c.

Other lines of the strategy shown in Algorithm 1 give some idea about the navigation in the nets. The algorithm evaluates every node of the combinational logic, from the primary outputs (PO) to the primary inputs (PI). This is done because the delay of the gates is changed after sizing. When transistors of a gate are sized, the delay usually becomes smaller and a transient pulse propagates with smaller degradation.

Erroneous interpretation concerning the SET propagation must happen if the transient pulse is evaluated before the sizing of the gates in the path to the POs. Thus, when the SET is evaluated in a node n, we guarantee that every gate in the path between this node n and the POs, were already sized.

4. Results

Table 1 shows the results obtained by the proposed transistor sizing strategy. Results include a comparison between a symmetric and asymmetric sizing methodologies. A 180nm technology process was used [ASU 07]. The transient pulse propagation parameter k was defined by hspice simulations as 0.8 for this technology. We present results concerning circuits with 50% sensitivity and 100% sensitivity reduction.

A study presented in [ZHO 06] shows that the charge of very few particles if higher than 0.3pC at ground level. We use this value in our experiments by considering as the worst case deposited charge.

The first important point shown by these results is the small overhead presented by the proposed methodology. The worst case was a 115% area overhead for complete protection against particles with charge Q = 0.3pC. Results show an 83% area overhead for the symmetric sizing and 61% for the asymmetric sizing. Power consumption presents 70% average overhead for the symmetric sizing against 43% for the asymmetric. Timing penalties were the most important results where protected circuit were 13% slower.

The asymmetric transistor sizing resulted in smaller area, power consumption and timing in comparison with the symmetric sizing. Despite the penalties when designing radiation hardened circuits, results shown the asymmetric sizing efficacy.

Table 1. Results show the area, timing and average power overhead for symmetric and asymmetric sizing techniques for particles with charge Q = 0.3pC**.**

Combinational Circuit	Number of Gates	Sensitivity $S_{circuit}$	Symmetric Sizing			Asymmetric Sizing		
			Area (%)	Power (%)	Timing (%)	Area (%)	Power (%)	Timing (%)
C432	227	50%	47.4	63.8	0.0	35.5	50.7	2.0
		0%	69.8	105	1.2	50	59.7	0.0
C880	365	50%	88.0	72.4	0.0	69.2	51.6	0.0
		0%	115.3	88.7	12.3	86.9	59.1	13.2
C1355	464	50%	62.4	38.6	16.0	50.6	29.5	15.8
		0%	80.0	61.6	24.8	58.6	37.2	17.1
C1908	423	50%	47.0	35.5	12.0	37.0	29.0	8.8
		0%	69.2	20.89	13.0	49.2	17.4	10.16
Average overhead		50%	61.2	52.7	7.0	48.0	40.2	6.65
		0%	83.5	69.0	12.82	61.1	43.3	10.11

5. Conclusions

This paper presents a new transistor sizing algorithm aiming at protecting combinational logic circuit to single event transients. The transistor sizing strategy is based on logical and electrical masking in order to independently size pull-up and pull-down transistors.

Results show small area and power consumption overhead, mainly if compared with traditional techniques such as TMR, which result is more than 200% area overhead. The reduced timing penalties presented by the sizing methodology allows the development of high frequency circuits.

On going works target the insertion of a sizing step at the end of the process to reduce timing penalties.

6. References

[ASU 07] ASU**. Predictive technology model. Nanoscale integration and modeling (nimo) group**. Available in http://www.eas.asu.edu/_ptm/, 2007.

[BES 94] BESSOT, D.; VELAZCO, R. **Design of seu-hardened cmos memory cells: The hit cell.** In *Proceedings of the 1994 RADECS Conference*, pages 563–570, 1994.

[CAL 96] CALIN, T.; CALIN, M; et al. **Upset hardened memory design for submicron cmos technology.** *IEEE Transactions on Nuclear Science*, 43:2874–2878, December 1996.

[CAZ 05] CAZEAUX, J. M.; ROSSI, D.; et al. **On transistor level gate sizing for increased robustness to transient faults.** In *IOLTS '05: Proceedings of the 11th IEEE International On-Line Testing Symposium*, pages 23–28, Washington, DC, USA, 2005. IEEE Computer Society.

[DHI, 05] DHILLON, Y. S.; DIRIL, U.; CHATTERJEE, A. **Soft-error tolerance analysis and optimization of nanometer circuits.** In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 288–293, Washington, DC, USA, 2005. IEEE Computer Society.

[HEI 06] HEIJMEN, T.; NIEULAND, A. **Soft-error rate testing of deep-submicron integrated circuits.** In *ETS '06: Proceedings of the Eleventh IEEE European Test Symposium (ETS'06)*, pages 247–252, Washington, DC, USA, 2006. IEEE Computer Society.

[MOH 03] MOHANRAM, K.; TOUBA, N. A. **Cost-effective approach for reducing soft error failure rate in logic circuits.** *itc*, 00:893, 2003.

[NIE 06] NIEULAND, A. K.; JASAREVIC, S.; JERIN, G. **Combinational logic soft error analysis and protection**. In *IOLTS '06: Proceedings of the 12th IEEE International Symposium on On-Line Testing*, pages 99–104, Washington, DC, USA, 2006. IEEE Computer Society.

[ROC 88] ROCKETT, L. **An seu hardened cmos data latch design.** *IEEE Transaction on Nuclear Science*, NS-35(6):1682–1687, Dec 1988.

[SHI 02] SHIVAKUMAR, P; KISTLER, M.; KEC, S. W.; BURGER, D.; ALVISI, L. **Modeling the effect of technology trends on the soft error rate of combinational logic.** In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 389–398, Washington, DC, USA, 2002. IEEE Computer Society.

[WHI 91] WHITAKER, S.; CANARIS, J.; LIU, K. **SEU Hardened memory cells for a ccsds reed solomon encoder.** *IEEE Transaction on Nuclear Science*, NS-36(6):1471–1477, December 1991.

[WIR 07a] WIRTH, G.; VIEIRA, M.; KASTENSMIDT, F.L. **Accurate and computer efficient modelling of single event transients in cmos circuits.** *IET Circuits, Devices & Systems*, 1:137–142, April 2007.

[WIR 07b] WIRTH, G.; VIEIRA, M.; NETO, E.H.; KASTENSMIDT, F.L. **Modelling the sensivity of cmos circuits to radiation induced single event transients.** *Microelectonics Reliability*, 47(3), March 2007.

[ZHO 06] ZHOU, Q.; MOHANRAM, K. **Gate sizing to radiation harden combinational logic.** *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25:155– 166, Jan 2006.

Modeling of a NMOS 90 nm device to Multiple Event Transient Simulation

Thiago Assis[1], **Gilson Wirth** [1], **Fernanda Kastensmidt**[1], **Ricardo Reis**[1]
{ thiago.assis,fglima,wirth,reis}@inf.ufrgs.br

[1] PPGC - Federal University of Rio Grande do Sul

Abstract

This work presents the results of the modeling of a NMOS 90 nm device. The device was developed using data extracted from 90 nm predictive models and the validation was done comparing the results of current and voltage behavior of the model with a 90nm Predictive Technology Model (PTM) from the Arizona State University (ASU) and data extracted from ST 90 nm design rule document. The results show an agreement from the model with PTM simulation and ST document.

1. Introduction

With the continue scaling of Metal Oxide Semiconductor Field Effect Transistor (MOSFET) to nanometer technologies the susceptibility of these devices to soft errors increase significantly [JOH 00]. The space industry and government agencies have been working to mitigate soft and permanent errors but with the continue evolution of MOSFET technologies there is an increase of the necessity of research fault tolerance techniques to mitigate and understand better these effects [TRI 06].

Radiation effects on Integrated Circuits (IC) are generally classified in permanent and transient effects. Permanent effects are those that damage the circuit and their effect will persist over the device life. Transient effects are characterized by a duration that will not persist over the device life time and will eventually disappear [SCH 07]. The kind of effects that radiation will cause into the device depends of many factors and the classification of these also takes into account the kind of the integrated circuit. The classification of radiation effects and other issues about programmable devices will not be discussed at this article, for detailed information about this issue consult [BOU 07] and [KAS 06].

This article will present the modeling of a NMOS 90 nm device that will be used to simulate transient effects over different layout topologies. Besides analyze different layout topologies into a 3D model the radiation simulation will also take into account the incident angle of the radiation particle.

The main contribution of this article is expose to community the modeling of a NMOS 90 nm device in a 3D simulation environment using the tool Davinci from Synopsys [SYN 06].

2. Predictive 90 nm device

Data extracted from [MIT 08] and [HU 95] suggest the topology showed at figure 1. The model presents a NMOSFET using a Super Steep Retrograde (SSR) channel doping, source/drain halo and a Light Doped Drain (LDD) structure. The SSR technique described at [SRI 05] and [TIA 94] consist of a channel engineering technique used to prevent short channel effects on nanometer MOS devices. The technique proposes increase the doping of the substrate (prevent short channel effects) and create a thin layer with a lower doping near the surface of oxide and silicon to control the threshold of the device.

The halo implant is also a channel engineering technique used to control short channel effects. This structure will not be used in this work, for more details about halo implants consult [SRI 05]. The LDD structure consist of create a small active region that will be overlapped by the channel using a thickness smaller that the active region (Source\Drain) and with a smaller doping concentration [TSI 99].

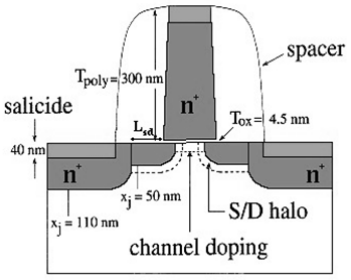


Fig.1- NMOSFET device [MIT 07]

At the figure 1 the structure of the NMOSFET device is shown. Notice the different thickness form the LDD and source\drain active regions. The source\drain halos are not shown in details and just their positions are indicated. The dash lines over the channel indicate the SSR channel doping.

Another work that presents a 90 nm device is [DAS 07]. At this work a device is created to evaluate the soft error collection mechanism and to propose a technique on process level to reduce soft errors effects. The doping profiles and devices regions dimensions where analyzed with the data from [MIT 08] to create the device proposed on this work. At the next section the developed model is presented.

3. Modeled Device

The first step to create a device model is to define a strategy to build the MESH. The MESH is a structure where the device regions are created and their sections and points define where physic calculations are done to evaluate the simulations. If the MESH is defined with too much simulation points will leads to many days of simulation, but if this number is small the results of the simulation will be too far from a real device operation. A good strategy to define the MESH is presented at [DAS 07] where the author suggest the creation of a MESH divided on regions with higher and lower concentration of simulation points and with the refinement of the points on areas on which the simulation is more important. In this work these regions would be the active regions and channel.

With the data extracted from [MIT 07] and [DAS 07] the device where constructed. After simulations the doping profiles and dimensions where calibrated to achieve the desired behavior and electrical characteristics. At figure 2 the created model.

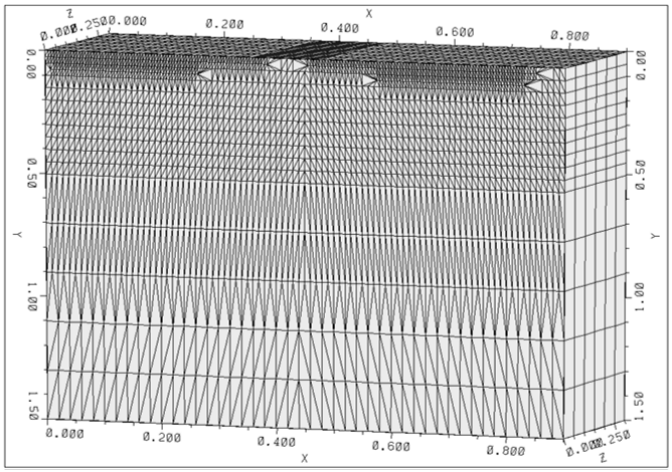


Fig. 2 - Developed NMOS 90nm device

Notice at figure 2 that from *0.0* to *0.5um* the MESH has more division's that from *0.5* to *1.5 um*. Over active regions and channel the MESH is refined to receive more simulation points. This kind of refinement is done during simulations to achieve the desired accuracy.

At table 1 the dimensions of the LDD, SSR, active regions, gate and oxide. The doping profile is also presented.

Region	Width (x)	Width (y)	Width (z)	Doping
LDD	*115 nm*	*30 nm*	*240 nm*	*n-type. Peak: 1e19*
Source\Drain	*240 nm*	*60 nm*	*240 nm*	*n-type. Peak:1e20*
SSR	*90 nm*	*5 nm*	*240 nm*	*n-type. Peak:1.5e18*
Gate	*136 nm*	*300 nm*	*240 nm*	*n-type. Peak:2e20*
Oxide	*136 nm*	*1.4 nm*	*240 nm*	*None*
Substrate	*0.9*	*1.5*	*0.3 um*	*p-type. Peak:5.5e18*

Table 1: Dimensions and doping profiles

The doping has Gaussian distribution. The difference of doping between Substrate and SSR is used to control the threshold of the device. Using an uniform substrate doping the threshold of the device was too high almost near *VDD* (1.2 V for this device) and the SSR was used to reduce it value. The oxide was not adjusted to control the threshold voltage and this value is the same that [DAS 07] had used to 90 nm high performance devices.

4. Simulation Results

To validate the NMOS transistor simulations were conducted to analyze the behavior of the device when a bias is applied over the terminals (substrate\source\gate\drain). Over the simulations of figure 3 and 4 the substrate and source where grounded (Vs=Vb=0V). At figure 3a the Vds is fixed on 1.2 V and a bias is applied on the gate terminal. At figure 3b the Vgs is fixed on 1.2 V and the drain bias is varied.

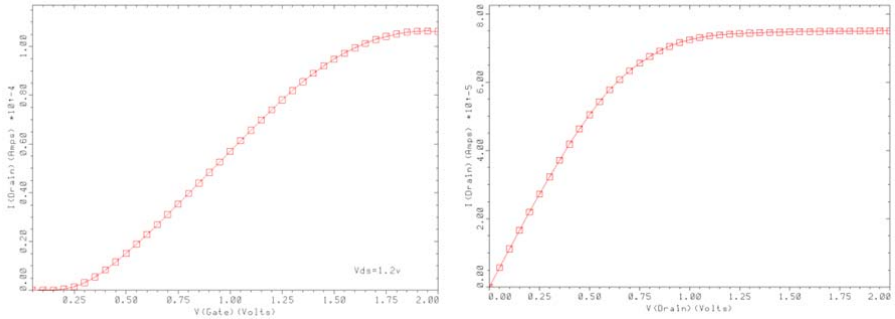


Fig. 3 – a) *Ids vs Vgs* b) *Ids vs Vds* simulation

To validate the agreement of the model with electrical parameters of a 90 nm device, simulations where conducted using a PTM available at [ASU 08]. The threshold voltage of the PTM model was about *Vt=0.29 V*. The document of design rules from ST 90nm [ST 02] indicates a *Vt=0.32 V*. The threshold voltage of the developed model was obtained by extrapolation method and is about *Vt=0.29* V. The scale of the simulation (*Ids vs Vds*) of the PTM model and [DAS 07] also fits with the results of the developed device.

At Figure 5 the bias of regions of the device during operation (*Vgs=1.2 V*).

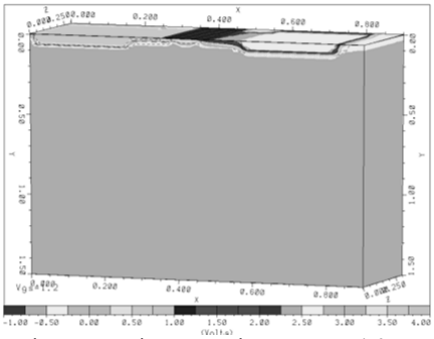


Fig. 5 - Device operation at *Vgs=1.2 V*.

The Figure 5 confirms the desired operation of the NMOS transistor over the simulation inputs. Its possible to notice at the device the difference between the voltage of the substrate and source.

5. Conclusions

This paper presents the modeling of a NMOS 90nm device in Davinci tool from Synopsys. Results comparing the device with PTM simulations and values extracted from [DAS 07] and ST 90nm shows a good agreement with the model. The next step of this work is to simulate ionizing radiation over the device and evaluate the results.

6. References

[ASU 08] ASU. **Predictive Technology Model** . Nanoscale Integration and Modeling Group. Arizona State University (ASU). <Available at: http://www.eas.asu.edu/~ptm/ >. 2008.

[BOU 07] BOUDENOT, J. Radiation Space Environment. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.) **Radiation Effects on Embedded Systems.** Netherlands: Springer, 2007. P. 1-9

[DAS 07] DASGUPTA, S. **Trends in Single Event Pulse Width and Pulse Shapes in Deep Submicron CMOS.** Thesis from the Faculty of the Graduate School of Vanderbilt University. Electrical Engineering. Vanderbilt University. Nashville, Tennessee. USA. 2007.

[HU 95] HU. H. et al., IEEE TED-42(4), p. 669, 1995

[JOH 00] JOHNSTON, A. H. **Scaling and Technology Issues for Soft Error Rates.** 4th Annual Research Conference on Reliability, Stanford University, October 2000. Available at <http://parts.jpl.nasa.gov/docs/Scal-00.pdf >.

[KAS 06] KASTENSMIDT, F, Carro, L, Reis, R. **Fault-Tolerance Techniques for SRAM-Based FPGAs**. Editora Springer. 2006.

[MIT 08] MIT. **Well-Tempered Bulk-Si NMOSFET Device.** Microsystems Technology Laboratory, Massasuchets institute of technology (MIT). <Available at: http://www-mtl.mit.edu/researchgroups/Well/>. 2008.

[SCH 07] SCHRIMPF, R. D. Radiation Effects in Microelectronics. In: VELAZCO, R. FOUILLAT, P. REIS, R. (Ed.) **Radiation Effects on Embedded Systems.** Netherlands: Springer, 2007. P. 11-29.

[SRI 05] SRINIVASAN, R.; BHAT, N. **Impact of Channel Engineering on Unity Gain Frequency and Noise-Figure in 90nm NMOS Transistors for RF Applications.** International Conference on VLSI Design. 2005.

[ST 02] STMicroelectronics**. HCMOS9_GP Design Rules Manual 013 MICRON CMOS PROCESS.** Date 11-Apr-02.

[SYN 06] SYNOPSYS. **Taurus Medicis Davinci User Guide.** Version Y-2006.06 June 2006.

[TIA 94] TIAN, H. HULFACHOR, R.B. et al. **An Evaluation of Super-Steep-Retrograde Channel Doping for Deep-Submicron MOSFET Applications.** IEEE TRANSACTIONS ON ELECTRON DEVICES, VOL. 41, NO.10. OCTOBER 1994.

[TRI 06] TRIPATHI, R.K; WILSON, J.W; YOUNG, R.C.; **Electrostatic Active Radiation Shielding – Revisted.** IEEE AEROSPACE CONFERENCE, 2006.

[TSI 99] TSIVIDIS, Y. **Operation and Modeling of The MOS Transistor.** Second Edition. New York: Oxford University, 1999, 620 p.

A Two-Step SET Sensitivity Estimation Technique

[1] Matheus P. Braga, [2] Guilherme Corrêa, [2] Luciano Agostini, [3] José Luís Güntzel

matheus.braga@inf.ufrgs.br, {gcorrea_ifm, agostini}@ufpel.edu.br, guntzel@inf.ufsc.br

[1]Grupo de Microeletrônica (GME) – II – UFRGS
[2]Grupo de Arquiteturas e Circuitos Integrados (GACI) – DINFO – UFPel
[3]Laboratório de Automação do Projeto de Sistemas (LAPS) – INE – UFSC

Abstract

The SET sensitivity analysis is becoming an important issue for circuits fabricated with nanometer CMOS technologies. This paper presents a method to help estimating circuits' sensitivity due to SETs divided in two steps: a timing-aware step that identifies all pulses arriving at the outputs out of the latching window and a second step that identifies the pulses that are logically masked. Although the second step has the advantage of being independent of circuit timing characteristics, the experimental results show that a significant speedup is achieved when both steps are used.

1. Introduction

Current state-of-the-art CMOS technologies, commonly referred to as "nanometer technologies", feature extremely small transistors and low threshold and operating voltages. Therefore, circuits designed with these technologies tend to be more vulnerable to soft errors [BAU 05].

A soft error may be induced when a charged ionizing particle strikes a sensitive region of an integrated circuit, i.e. the reverse-biased p-n junctions of the transistors that are off [MES 82]. If the particle strikes a storage element in the circuit, the stored data is changed and can be corrected only if it is read again. This kind of soft error is known as Single-Event Upset (SEU). If the particle hits a combinational block, the voltage pulse resulting at the gate output may propagate through the circuit and may eventually be latched by a storage element, also causing a soft error. This phenomenon is referred to as Single-event Transient (SET) [ALE 04]. In recent past years SEU has received more attention from industry and academia because memory elements tend to be more susceptible to radiation-induced soft errors [NIE 06]. However, it has been predicted that by the year 2010 the soft error rate (SER) due to SET will be as great as the soft error rate in unprotected memories [SHI 02]. Therefore, SET vulnerability analysis is becoming relevant for the design of reliable electronic systems.

The SET sensitivity of a circuit may be estimated by analyzing the propagation of SET-induced pulses in order to determine the percentage of pulses that may achieve any of the primary outputs (POs). To accomplish that many of the proposed techniques use circuit simulation (e.g., [ALE 04]) or rely on probabilistic values to compute the SET sensitivity.

This paper presents a SET sensitivity estimation technique that is based on SET propagation analysis. The technique has two steps. In the first step Topological Timing Analysis is used in order to identify the SET-induced pulses that will not achieve any PO. The second step uses the PODEM (Path-Oriented Decision Making) algorithm [GOE 81] in order to identify existing statically sensitizable paths between the SET site and any PO. This step analyzes the SET propagation just for the nodes remaining from the first step, thus reducing the overall execution time.

2. SET Propagation Analysis

The effect of an SET may propagate to all gates belonging to the logic cone that begins at the gate that was hit by the particle. If at least one of the resulting pulses achieves the output register within the latching window, then an erroneous value is stored. However, there are three factors that can inhibit the effect of the SET (and thus are called masking factors). These are logic masking (or logic blocking), electrical masking (or attenuation) and timing masking (or latching window masking). Fig. 1 illustrates these three factors.

Attenuation is resulted from electrical characteristics as gate delays and connection capacitances. Timing masking occurs when a pulse arrives at the output register out of the sampling window and therefore, is not captured. Logic masking may be evaluated by using the concept of gate controllability: a pulse will propagate through a gate only if all of its inputs except the one that has the pulse present the gate non-controlling value.

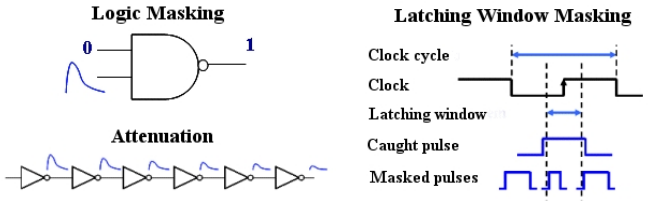


Fig. 1 – Factors that inhibit the propagation of a transient pulse through the circuit

Circuit simulation-based approaches tend to provide the most accurate estimates for pulse propagation analysis, but at a high computational cost. Considering a circuit with *n* primary inputs (PIs) and *m* gates, the complete analysis will demand the injection of 2x*m* faults (each circuit node may undergo a 1→0→1 SET or a 0→1→0 SET), each fault being simulated for all 2^n possible input vectors. Unfortunately, for large combinational circuits, such approach will demand a prohibitively long execution time. Therefore, it is necessary to avoid circuit simulation. One possible approach to cope with this problem is to rely on circuit timing information, in order to identify all faults that cannot achieve the circuit POs within the latching window.

Consider **td** as the time by which net *nt* settles to its stable value. Consider also that $\mathbf{D_{max}}$ is the maximum time for a signal at *nt* to propagate to the circuit POs. Suppose a transient voltage pulse occurring at *nt*, beginning at the instant **ti** and with duration **d**. According to Alexandrescu, Anghel and Nicolaidis [ALE 04], there are two conditions that may prevent this pulse to be captured by the output register: the transient pulse has occurred at *nt* before this net settles to its stable logic value and the pulse achieves the inputs of the output register before its setup time. These conditions may be respectively expressed by the following equations:

$$\mathbf{ti} + \mathbf{d} < \mathbf{td} \quad (1)$$

$$\mathbf{ti} + \mathbf{D_{max}} > \mathbf{t_{setup}} \quad (2)$$

By using such conditions and considering values for the beginning of the pulse (**ti**) and pulse duration (**d**) it is possible to identify the nets at which the occurrence of a SET will not become a soft error because none of the resulting pulses will be captured by the output register.

3. Latching Window Masking Analysis

The Topological Timing Analysis (TTA) procedure proposed in this work to analyze SET propagation can be divided into three steps. In the first step, the combinational block is modeled as a Direct Acyclic Graph (DAG) in which nodes represent (CMOS) gates and edges represent circuit connections. The second step performs a forward topological processing of the circuit DAG, beginning by node *s* and going forward until node *t* is reached. A node *v* is processed only after all its predecessors have been processed. In our implementation, processing node *v* consists on determining the maximal delay from the source node *s* to node *v* (**mds**). As long as we considered different falling and rising gate delays two **mds** values are computed for a given node *v*: **mdsf(***v***)** and **mdsr(***v***)**. These values are calculated by the following equations:

$$\mathbf{mdsf}(v) = \mathbf{tdhl}(v) + \max\{\mathbf{mdsr}(u_i)\} \quad (3)$$

$$\mathbf{mdsr}(v) = \mathbf{tdlh}(v) + \max\{\mathbf{mdsf}(w_i)\} \quad (4)$$

where **tdhl**(*v*) and **tdlh**(*v*) are the falling and rising gate delays of *v*, respectively, $u_i \in$ **predf**(*v*) (the list of predecessors of *v* for a falling transition) and $w_i \in$ **predr**(*v*) (the list of predecessors of *v* for a rising transition). **mdsf**(*v*) and **mdsr**(*v*) give the maximum of the delays of all possible paths starting at node *s* and ending at node *v*, for a falling transition and for a rising transition at the output of *v*, respectively.

In the third step a backward topological processing of the circuit DAG is performed, beginning by node *t* and going backwards until node *s* is reached. This procedures is similar to the one performed in step 2, except that it traverses the DAG in the opposite direction and uses the following equations:

$$\mathbf{mdtf}(v) = \mathbf{tdhl}(v) + \max\{\mathbf{mdtr}(u_i)\} \quad (5)$$

$$\mathbf{mdtr}(v) = \mathbf{tdlh}(v) + \max\{\mathbf{mdtf}(w_i)\} \quad (6)$$

where $u_i \in$ **succ**(*v*) (the list of successors of *v* for a falling transition) and $w_i \in$ **succ**(*v*) (the list of successors of *v* for a rising transition).

Once the **mdt** and **mds** values have been computed for all nodes, it is possible to use equations 1 and 2 to identify all circuit gates at which a SET cannot influence on the circuit sensitivity because the transient pulse generated does not achieve any PO within the latching window. For a given node *v*, **mdsf**(*v*) and **mdsr**(*v*) coincide respectively to **tdf** and **tdr**, i.e., the time by which *v* settles to its stable value for a falling and for a rising transition. Moreover, **mdtf**(*v*) and **mdtr**(*v*) are the maximum time for a signal to propagate from *v* to the POs (i.e., **Dmax**), considering a falling and a rising transition at *v*, respectively.

Now, given a pulse with duration **d** that begins at time **ti**, it is possible to identify the set of gates (or equivalently, the nodes that fan out from these gates) that must be considered for fault injection simulation in SET propagation analysis. The other gates (nodes) of the circuit do not need to be considered in SET propagation analysis.

Although the goal of this work is to avoid circuit simulation in SET propagation analysis, a previous work has showed that the proposed TTA-based method is fast enough to be used as pre-processing step for circuit simulation-based SET propagation analysis [BRA 07].

4. Logic Masking Analysis

To consider logic masking, transient pulses may be modeled as digital pulses A Single-Event Transient has a temporary effect in combinational part of the circuit, in such a way that after a certain time the signal in the affected gate output will be restored to its previous (thus faulty-free) value. However, each circuit node may undergo a 1→0→1 SET or a 0→1→0 SET. Therefore, the transient pulse can be modeled as a stuck-at-fault. For instance, a line with a stuck-at-1 fault will always have a logic state 1 independently of the correct logic output of the gate driving it. By assuming such modeling, the SET propagation problem is translated to a test generation problem and hence, ATPG techniques may be applied.

As already mentioned, in this work we use the PODEM algorithm [GOE 81] to analyze logic masking in SET propagation. As any other structural ATPG algorithm, PODEM divides the test generation process into two steps: fault activation and error propagation. The PODEM algorithm starts from an initial objective, which is to set line L to logic value v', where L is a line stuck at value v (L, s-a-v). In the first step a backtrace routine is called in order to choose one of the PIs and to assign a value to this PI. PODEM uses the length of paths between its initial objective and primary outputs (POs) to measure the difficulty of sensitizing a path. Objectives are selected by its logic level to pick the easiest object to achieve.

After that, the value to be assigned is determined using controllability/observability measures [GOL 79]. If all gate inputs have to be set to achieve the objective, PODEM backtraces through the hardest-to-control input first. In the same way, if only one logic gate input needs to be set to achieve the objective, PODEM backtraces through the easiest-to-control input.

In the second step the values assigned during the backtrace must be implied through the circuit. These two steps iterate over until the objective is satisfied (a test is found) or a conflict occurs. If the objective is satisfied, the PODEM ends successfully. Otherwise, more inputs may be assigned or, if the implication produced a conflict, the backtrack procedure should be called. The backtrack restores the state of the network by flipping the logic value of the last PI assignment.

If a test cannot be generated with a certain number of assigned PIs, the PODEM algorithm tries to assign a new value to an unassigned primary input or tries an untried combination of values on assigned primary inputs. If a test can be generated, but the fault does not appear in the primary outputs of the circuit, the fault is not propagated. However, in some cases, even after all primary input vectors are tried, a test is not generated and then, the fault is untestable.

5. Experiments and Results

The proposed SET sensitivity analysis technique was implemented in C language as a prototype tool called GILDA. To test and validate the technique, we have run the prototype on the circuits of the MCNC benchmark suite. The circuits were first mapped to use only inverters, 2-input NAND and 2-input NOR gates. After that, for each circuit the rising and falling delays of each gate were obtained through Spice simulations assuming the 90nm Predictive Technology Model (PTM) [CAO 00] and by using Spice Opus simulator [SPI 07] along with in-house developed scripts.

In the first step of the method the initial time of a transient pulse (**ti**) is specified in terms of percentages of the minimum clock cycle for the circuit under consideration. The minimum clock cycle, by its turn, is assumed as being equal to the critical topological delay of the circuit (**D**). The transient pulse duration (**d**) is specified in picoseconds. For instance, supposing that **D** is the critical delay of circuit **C**, a pair {d=50, ti=0.2} means that **C** will be evaluated for a transient pulse generated by an SET with duration equals 50 ps and beginning at instant 0.2x**D** from the initial clock edge. The analysis considers that a pulse with these given characteristics may occur at each circuit gate. Therefore, the tool prototype provides a list of the circuit nodes from which a SET can propagate to the POs (i.e., the nodes that are sensitive to the considered SET).

For the current work we have assumed transient pulses with duration of 100ps (as suggested in [DHI 06]), and initial time varying by 0.1x**D**. Tab. 1 shows the results of the prototype execution for the MCNC circuits. The fourth and the fifth columns show respectively the SET sensitivity and the execution time when the tool runs only the second step, that is, the PODEM-based logic masking analysis (logic masking only). The GILDA prototype was run in a computer equipped with a 64-bit 2.0 GHz AMD Turion processor with 1 GB of main memory.

The results showed a significant speedup when the pre-processing step was applied before the logic masking analysis (more than 15 times for circuit alu4, for example). The exception is circuit C17. However, it is necessary to note that this circuit is really too small and its execution times are too small to be accurately measured.

It is worth to note that the sensitivity values reported by the use of the logic masking (second step only) serves as an upper bound on circuit sensitivity to SETs. The results for the two-step technique show that circuit sensitivity greatly depends on the initial instant of the pulse: pulses that are closer to the end of the clock period

are rarely filtered. On the other hand, it is possible to notice a tendency on lower circuit sensitivity when pulses begin very early.

Tab. 1 – SET sensitivity and GILDA running time for the circuits of the MCNC benchmark suite

Circuit			No pre-processing		ti = 0%		ti = 20%		ti = 40%		ti = 60%		ti = 80%	
Name	**# gates**	**# inputs**	**sens**	**Time (s)**	**sens**	**Time (s)**	**sens**	**Time (s)**	**sens**	**Time (s)**	**sens**	**Time (s)**	**sens**	**Time (s)**
C17	13	5	76.92	< 0.001	42.31	< 0.001	42.31	< 0.001	42.31	< 0.001	65.38	0.015	65.38	< 0.001
bw	279	5	78.32	0.234	6.63	0.093	24.19	0.14	43.01	0.171	60.57	0.203	75.63	0.218
C432	297	36	65.15	0.406	11.11	0.156	30.13	0.25	46.97	0.328	54.38	0.39	62.29	0.421
9sym	331	9	70.24	0.812	9.67	0.218	54.08	0.812	66.16	0.859	67.82	0.796	69.64	0.796
C880	512	60	72.66	0.859	19.53	0.390	44.82	0.593	56.35	0.687	65.33	0.781	68.46	0.796
C499	638	41	84.09	1.578	11.91	0.484	32.68	0.921	42.95	1.266	54.62	1.328	67.95	1.515
C1908	696	33	75.95	1.296	8.24	0.437	31.2	0.734	42.35	0.906	55.39	1.203	70.63	1.234
alu2	704	10	69.32	3.421	5.68	0.609	45.38	2.734	55.89	3.031	63.07	3.359	67.47	3.609
C1355	709	41	65.87	0.859	16.01	0.484	33.22	0.64	38.43	0.671	45.06	0.718	61.42	0.796
C2670	1128	157	73.36	5.171	17.51	1.937	31.69	2.781	54.3	4.218	67.91	5.125	71.76	5.296
C3540	1525	50	70.56	9.546	7.7	2.437	47.21	8.156	58.92	8.812	65.87	9.156	69.44	9.562
T481	2201	16	72.94	89.593	3.04	11.562	35.98	58.718	64.56	84.11	69.92	88.187	72.56	89.08
C6288	2624	32	61.26	37.89	3.22	7.296	26.71	24.203	38.64	32.67	50.72	36.515	58.96	37.55
C5315	2967	178	73.64	28.171	10.09	10.5	54.5	23.046	67.66	26.27	71.92	27.453	72.87	27.33
C7552	3459	206	71.73	62.687	8.17	16.937	48.53	47.734	61.78	57.78	69.89	61.328	71.36	62.11
alu4	5796	14	69.82	1287.6	0.71	51.296	44.16	932.3	58.61	1151.6	68.57	1270.7	69.44	1280.3

6. Conclusions and Future Work

The experimental results obtained for the proposed SET sensitivity analysis technique showed that a significant speedup may be achieved if the timing-aware step is used before the logic masking analysis. The results obtained for the complete technique demonstrated that circuit sensitivity to SETs greatly depends on timing aspects. On the other hand, the logic masking analysis step may also be used solely when circuit timing information is not available or is not reliable (for example, due to process variations).

Future works include the application of the proposed method to evaluate different mappings for the same circuits, in order to derive synthesis rules for SET-tolerant design.

7. References

[ALE 04] ALEXANDRESCU, D. et al. Simulating Single Event Transients in VDSM ICs for Ground Level Radiation. JETTA, Vol. 20, pp. 413-421, 2004.

[BAU 05] BAUMANN, R. C. Radiation-Induced Soft Errors in Advanced Semiconductor Technologies. IEEE Trans. Devices & Materials, v.5, n.3, pp. 305-316, Sep. 2005.

[BRA 07] BRAGA, M. et al. Timing Aware Pre-Analysis of Single Event Transient Propagation. In: XXII South Symposium on Microelectronics, Porto Alegre, Brazil, 2007.

[CAO 00] CAO, Y.; SATO, T.; ORSHANSKY, M.; SYLVESTER, D.; HU, C. New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Simulation. In: CICC'00, pp.201–204.

[DHI 06] DHILLON, Y.; DIRIL, A.; CHATTERJEE, A.; SINGH, A. Analysis and Optimization of Nanometer CMOS Circuits for Soft-Error Tolerance. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.14, n.5, p.514-524, maio de 2006.

[GOE 81] GOEL, P. An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits. In IEEE Transactions on Computers, Piscataway, NJ, v. C-30, n. 3, p. 215-222, 1981.

[GOL 79] GOLDSTEIN, L. H. Controllability/Observability Analysis of Digital Circuits. IEEE Transactions on Circuits and Systems, v. CAS-26, n. 9, p. 685-693, 1979.

[MES 82] MESSENGER, C. Collection of Charge on Junction Nodes from Ion Tracks. IEEE TNS, v. NS-29, pp. 2024-2031, Dec. 1982.

[NIE 06] NIEUWLAND, A. et al. Combinational Logic Soft Error Analysis and Protection. In: Proceedings of the 12th IEEE International Symposium on On-Line Testing, 2006.

[SHI 02] SHIVAKUMAR, P. et al. Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. in ICDSN, 2002, pp. 389-398.

[SPI 07] SpiceOpus: SPICE with integrated OPtimization UtilitieS. Available at <http://fides.fe.uni-lj.si/spice/>, 2007.

Fault-Tolerant Fast Adders Implemented in FPGAs

[1] Guilherme Corrêa, [1] Helen Franck, [1] Eduardo Mesquita, [1] Luciano Agostini, [2] José Luís Güntzel

{gcorrea_ifm, hfranck.ifm, emesquita.ifm, agostini}@ufpel.edu.br, guntzel@inf.ufsc.br

[1]Grupo de Arquiteturas e Circuitos Integrados (GACI) – DINFO – UFPel
[2] Laboratório de Automação do Projeto de Sistemas (LAPS) – INE – UFSC

Abstract

As long as advanced FPGA devices are fabricated with nanometer CMOS technologies, it is also expected that FPGA-based systems may be affected by transient faults. While the cost of hardened FPGA devices are prohibitively high, designers must rely on traditional protection techniques, as triple-modular redundancy (TMR), temporal redundancy (TR) or a combination of both. In this work we have investigated the application of three fault tolerance techniques to fast adder circuits: carry-select (CSA), carry lookahead (CLA) and a new type of adder named re-compute with inverse carry-in (RIC). Synthesis and simulation results obtained for Altera Stratix III FPGAS have showed that the RIC adders are the ones offering the best performance at a reasonably low cost.

1. Introduction

In order to allow the implementation of high performance complex electronic systems, state-of-the-art FPGA devices are fabricated with nanometer CMOS technology. Due to the extremely small geometries and low operation voltage, circuits fabricated with such technologies tend to be vulnerable to transient faults [COH 99], also known as soft errors.

A soft error may occur when an ionizing particle hits a sensitive region of a circuit causing a transient voltage pulse [MES 82][BAU 05]. Radiation-induced soft errors may be originated in memory elements or in the combinational logic. The former kind of soft error is known as Single-Event Upset (SEU) [BAU 05], while the latter kind is referred to as Single-Event Transient (SET). Although the soft error rate due to SEUs is currently higher than the soft error rate due to SETs, it has been predicted that, due to the continuous shrinking of the CMOS devices, by the year 2010 these rates will become comparable, considering unprotected circuits [SHI 02]. As hardened FPGA devices have a high unitary cost, the adoption of techniques to mitigate SET effects on the user's programmable logic will be mandatory in forthcoming designs.

Among arithmetic circuits, adders play a very important role because they serve as the basis for practically all other arithmetic operations. This way, it is necessary to investigate the architectural choices to implement robust adders and to evaluate each solution in terms of resource use, speed and degree of protection.

This paper presents an evaluation of four different adder architectures implemented using three different tolerance techniques. We have described the Ripple-Carry, Carry-Select, Carry-Lookahead [HWA 79] and the Re-computing the Inverse Carry-in [MES 07] adders in VHDL, using the Triple Modular Redundancy, Time Redundancy and Duplication With Comparison associated with Time Redundancy techniques.

2. Fast Adders

Because the hardware can only perform a relatively simple set of Boolean operations, arithmetic functions are based on a hierarchy of them. As speed, power dissipation and used area in the chip are the most used measures of efficiency there is a strong relationship between an architecture and the technology used in its implementation.

The philosophy behind the Carry Lookahead Adder (CLA) is to compute simultaneously several carries. The CLA, just like like other basic adders, follows the structure of the algebraic addition, performing the addition of the bit S_i as a function of the corresponding bits in the input operands (A_i and B_i) and the resulting carry of the addition in the previous stage (referred to as C_{i-1}).

The CLA differs from the other adders because the C_i, which is the carry bit used to compute the addition bit of the next stage (S_{i+1}), is not generated based on the addition bit S_i. Instead, two auxiliary signals are generated: G_i and P_i. The first one, called *Generate*, indicates the generation of a carry bit in the stage i (i.e., $C_i = 1$) and can be calculated by the following equation: $G_i = A_i \cdot B_i$. The signal P_i, called *Propagate*, indicates that the carry coming from stage i-1 (C_{i-1}) is propagated to the stage i (i.e., $C_i = C_{i-1}$). This situation happens when one of the operands is equal to '1' and the other is equal to '0'. This way, C_i does not depend on C_{i-1}. The signal *Propagate* is calculated by the following equation: $P_i = A_i \oplus B_i$.

The CLA block is divided into four sub-blocks: the GAP, which computes the signals *Generate* and *Propagate*; the CLA, which generates the intermediate carry bits; the SUM, which computes the addition itself

and the BGP, which computes the carry-in bits for the next adder blocks and the final carry-out, when the n-bit CLA basic block is instantiated to build the higher order CLA.

In the Carry-Select Adder (CSA) the addition is divided into a given number of sections that are computed in parallel. This speeds up significantly the carry propagation, resulting in high operation speed. Each section performs two additions at the same time: one considering a '0' as carry-in and another considering a '1' as carry-in. This way, the carry chain propagation is broken to be accelerated. After all the addition sections finish their operation, for each section the correct addition value is selected based on the resulting carry-out of the previous section.

The drawback of the CSA relies on its high cost in terms of hardware resources. Resource overhead comes mainly from the need for using two adders in each section, in order to allow the parallel computation of the additions. The high cost of CSA may prevent its use in several applications where hardware resources are limited. However, its redundancy may be explored in order to derive fault-tolerant adder versions.

In a previous work we have developed a new adder architecture inspired in the CSA that was named Re-computing the Inverse Carry-in Adder (RIC) [MES 07]. Although presenting the same philosophy of a CSA, the RIC adder demands less hardware because one of the two adders used in one adder section is replaced by a less expensive combinational block, called Re-computing Block. As shown in Figure 1, in each section there is one adder that generates the result assuming that the carry-in is equal to '0'. The Re-computing Block (RB in this figure) receives this result R and converts it to R+1. This re-computation is based on the exploration of some binary addition properties, as proposed by Kumar and Lala [KUM 03].

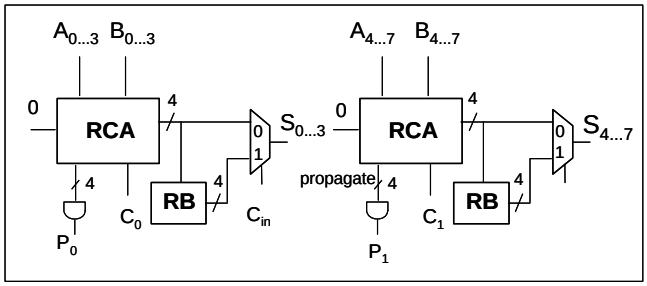


Fig. 1 – An 8-bit RIC adder is composed by two sections with one RCA and one Re-Computing Block.

3. Applying Fault-Tolerance Techniques to Adders

Among the existing soft error protection techniques, the Triple Modular Redundancy (TMR) is the simplest and most used one [CAR 01]. It also serves as reference, since its overhead in terms of hardware resources is known to be near 200% (masked IC implementations). The high resource overhead comes from the fact that TMR requires the use of three exemplars of the block to be protected plus a voter circuitry. The outputs of these three blocks are connected to a voter that decides, by means of a majority election, which is the correct result. The voter is the solely block susceptible to error. If an error occurs in the voter, it may take a wrong decision.

The RIC adder protected with TMR demands only three adders per section, in contrast to the CSA adder protected with TMR that needs six adders per section. Note also that, as in the unprotected version, the delay of the RIC adder is expected to be higher than the delay of the CSA.

In the Temporal Redundancy (TR) technique the output of the block to be protected is sampled in three different moments (clk, clk+d and clk+2d). The time difference between two consecutive resulting samples must guarantee the disappearance of the fault, if it occurs. To implement this technique it is necessary to triplicate the storage elements in order to store the three samples that are used as inputs to the majority voter. In this technique the hardware overhead is smaller than in the TMR case. However, the TR version is more complex because it requires different clocks for the three storage elements. Thus, this technique presents disadvantages concerning the time necessary to perform the whole computation, which is approximately equal to clk+2d+tp, where tp is the voter delay and d is the time required to perform each computation [LIM 03].

The Duplication With Comparison + Time Redundancy technique (DWC+TR) is an alternative to TMR. It explores time and hardware redundancy in an attempt to reduce the hardware overhead. Instead of using three instances of the block to be protected, as in the TMR case, it demands only two instances (CLA0 and CLA1 in Fig. 2). The output of each instance is latched into two registers, being one triggered at instant clk and the other, at instant clk+d. Hence, four samples of the block output value are available, two per block instance. These four samples are fed to the error detection block, which is responsible for verifying if an error has occurred in one of the two instances. If that is the case, the error detector delivers the right value to be used as the third input of the majority voter. The voter also receives the output of each block instance. Its output is stored in a register at instant clk+2d. It is worth to notice that the amount of time "d", which is the difference between two clock edges, must be long enough to allow a transient pulse to propagate through the logic that is placed between registers. Otherwise, the probability of sampling an error becomes higher.

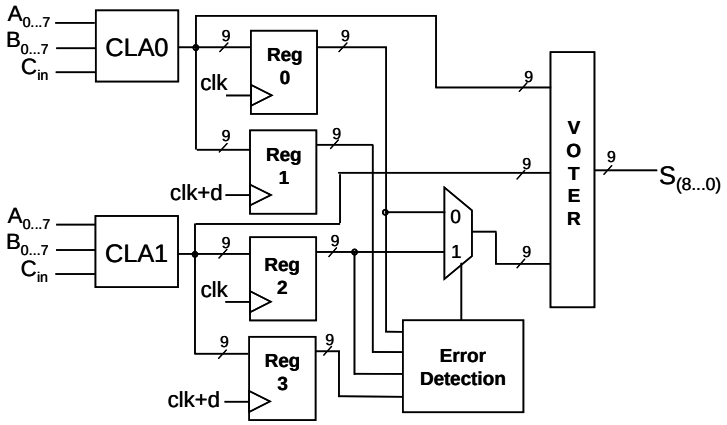


Fig. 2 – 8-bit CLA protected with DWC+TR.

4. Experiments and Synthesis Results

In this work we have described in VHDL unprotected and protected versions of the adders for six different data lengths, from 4 to 128 bits. To serve as a reference for the comparison, we have also described in VHDL protected and unprotected versions of the Ripple-Carry Adder (RCA) with the same data lengths. For each of the four adder architectures (RCA, CLA, CSA and RIC) three protected versions were designed: TMR, TR and DWC+TR.

All adder versions were synthesized for the EP3SE50F484C2 device (from Stratix III FPGA family) and validated through functional simulation with delay using Quartus II software from Altera (version 7.2 SP2). Table 1 shows the number of ALUTs reported by Quartus II for each adder version and the critical delay of each version, according to the Altera Classic Timing Analyzer.

Tab.1 – Number of ALUTs and critical delay for each adder version synthesized and simulated for Stratix III.

Adder	Technique	Number of ALUTs						Topological critical delay (ns)					
		4 bits	8 bits	16 bits	32 bits	64 bits	128 bits	4 bits	8 bits	16 bits	32 bits	64 bits	128 bits
RCA	unprotected	33	62	120	236	468	932	1.65	2.71	5.3	9.59	18.87	37.68
	TMR	87	173	337	665	1321	2633	2.62	4.03	6.96	12.93	23.95	46.62
	TR	73	138	260	571	1123	2227	10.39	14.68	24.36	47.92	93.62	184.7
	DWC+TR	150	280	530	1030	2031	4288	9.73	14.08	21.41	38.72	71.97	149.8
CLA	unprotected	63	120	230	468	924	1916	3.59	3.98	4.26	5.07	6.34	6.96
	TMR	196	327	637	1259	2512	5089	4.47	5.48	5.95	6.66	8.23	9.19
	TR	129	231	433	838	1646	3343	12.25	20.94	19.85	23.02	28.93	31.32
	DWC+TR	237	436	826	1616	3178	6450	13.59	13.31	18.02	21.62	25.09	27.69
CSA	unprotected	65	142	280	556	1110	2215	2.39	3.05	3.53	4.31	4.38	6.4
	TMR	184	371	733	1465	2915	5813	3.18	3.96	4.9	5.44	6.29	8.46
	TR	105	200	384	823	1619	3220	10.64	14.17	15.2	22.26	27.12	29.88
	DWC+TR	182	402	776	1532	3090	6145	10.38	11.05	14.98	15.96	19.58	24.77
RIC	unprotected	55	118	232	460	918	1831	2.42	3.09	3.66	3.91	4.38	5.49
	TMR	160	323	637	1273	2530	5044	3.7	4.01	4.46	5.31	6.26	7.99
	TR	97	184	352	758	1490	2964	11.44	13.4	14.68	17.48	24.2	27.99
	DWC+TR	166	362	696	1370	2770	5504	11.72	11.57	12.57	14.49	18.24	23.39

From the data of table 1 it is possible to conclude that the protection with TMR results in resource overheads ranging from 161.3% to 211.1%, being 175% the average overhead. In theory, the increase of TMR should be higher than 200% for full custom (masked) implementations. However, the synthesis algorithms and the granularity of FPGA programmable elements may influence this overhead.

Also from the first part of Table 1 it is possible to compute the resource overhead needed to protect adders with TR. It ranges from 37.1% up to 141.9%, with an average of 81.3%. However, a careful examination of resource overheads allows us to conclude that the impact of the extra registers (and voter) to protect with TR is higher for the RCA and for the CLA, with these overheads ranging from 116.7% to 141.9% and from 74.5% to 104.8%, respectively. This is because unprotected CSA and CLA adders require less ALUTs to be implemented than unprotected CSA or RIC adders. Therefore, the impact of the extra resources is not so prominent for CSA and RIC adders.

The resource overhead needed to protect the adders with DWC+TR ranges from 175.5% to 354.5%, with an average overhead of 245.1%. These results are in conflict with the original purpose of DWC+TR that is to reduce resource overhead, as claimed by some authors (see [LIM 03], for example). However, it is necessary to

remark that our implementation of DWC+TR is quite conservative, in the sense that we use four registers, while other authors use only two.

The examination of the second part of Table 1 leads us to conclude that the protection of adders with TMR resulted in increases of critical delay ranging from 21.9% up to 58.8%. In the average, TMR results in an increase of 35.5% of the critical delay. On the other hand, the use of TR resulted in an increase of delay that ranges from 241.2% to 529.7%, with an average increase of 382.1%. These results were expected, since the implemented TR needs four clock cycles: three to sample the block output and one to vote the correct result. The increase in critical delay resulting from the use of DWC+TR ranges from 234.4% to 489.7%, with an average of 312.2%.

Comparing the number of ALUTs and critical delays of the fast adders (CLA, CSA and RIC) protected with TMR, one may note that the CSA and the RIC present similar values. Particularly, for data lengths between 16 and 128 bits the RIC is faster or as fast as the CLA, at a lower or comparable cost.

5. Conclusions and Future Work

In order to evaluate the robustness of the proposed protection scheme against single-event transients (SETs), fault injection campaigns were performed for all 4-bit adders by means of logic-level simulation using Mentor Graphics ModelSim simulator. The voter was not considered in the robustness analysis because it was known beforehand that it is susceptible to faults. Also, we have limited the evaluation to pulses with 100 picoseconds, since it appears as a typical SET duration for the 65nm CMOS technology, as the one used in the fabrication of Stratix III devices [DHI 06].

The fault injection and simulation procedure showed that all the protection techniques used are efficient in protecting 4-bit adders against SETs. Since higher order adders are built up from such modules, it is expected that also higher order adders can be efficiently protected.

The data obtained from synthesis for Altera Stratix III FPGAs have shown that the TR technique required less extra hardware resources to implement protected adders, while the DWC+TR was the technique that has demanded more hardware. The functional simulation results have showed that adders protected with TR have the poorest performance, since the average critical delay increase was the highest: 382.1%. On the other hand, the adders protected with TMR presented an average increase in critical delay of only 35.5%, thus corresponding to the lowest increase among the three protection techniques.

As future work we intend to re-run these experiments for FPGA devices from other companies. It is also intended to compare full custom nanometer CMOS implementations of protected and unprotected fast adders.

6. References

[BAU 05] BAUMANN, R. C. Radiation-**Induced Soft Errors in Advanced Semiconductor Technologies**. In: IEEE Trans. on Devices and Materials Reliability, v.5, n.3, pp.305-316, Sep. 2005.

[CAR 01] CARMICHAEL, C. **Triple Module Redundancy Design Techniques for Virtex FPGA**. In: Xilinx Application Notes 197, San Jose, USA, 2001.

[COH 99] COHEN, N. et al.. **Soft Error Considerations for Deep-Submicron CMOS Circuit Applications**. In: Internacional Electron Devices Meeting, pp. 315-318, July 1999.

[DHI 06] DHILLON, Y. S. et al. **Analysis and Optimization of Nanometer CMOS Circuits for Soft-Error Tolerance**. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.14, n.5, p.514-524, 2006.

[HWA 79] HWANG, K. **Computer Arithmetic: Principles, Architecture, and Design**. New York: Wiley, 423 p., 1979.

[KUM 03] KUMAR, B. **On-line Detection of Faults in Carry-Select Adders**. In: International Test Conference, pp. 912, 2003.

[LIM 03] LIMA, F. et al. **Designing Fault Tolerant Systems into SRAM-based FPGAs**. In: International Design Automation Conference, DAC, 2003, Proc. New York: ACM, 2003, pp. 650 – 655.

[MES 82] MESSENGER, C. **Collection of Charge on Junction Nodes from Ion Tracks**. In: IEEE Transactions on Nuclear Science, v. NS-29, pp. 2024-2031, Dec. 1982.

[MES 07] MESQUITA, E. et al. **RIC Fast Adder and its SET Tolerant Implementation in FPGAs**. In: 17th International Conference on Field Programmable Logic and Applications, Amsterdam, Netherlands, 2007.

[SHI 02] SHIVAKUMAR, P. et al.. **Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic**. In: International Conference On Dependable Systems And Networks, pp. 389 – 398, 2002.

DIGITAL DESIGN

An experimental analysis of the lockstep architecture

[1]Cristina Meinhardt, [2]Massimo Violante, [2]Matteo Sonza Reorda, [1]Ricardo Reis
{cmeinhardt, reis}@inf.ufrgs.br, {massimo.violante, matteo.sonzareorda}@polito.it

[1]Universidade Federal do Rio Grande do Sul
[2] Politecnica di Torino

Abstract

Lockstep is an interesting architecture to implement safe systems starting from commercial processors, where the basic error-detection mechanism is duplication and comparison. In this paper we designed a lockstep system using two Leon processors, a master and a slave, and we implemented a checker unit comparing the behavior of the two processors at every clock cycle. The implemented system was then used to practically evaluate the effectiveness of the lockstep approach.To assess the robustness of the resulting system to soft errors, we focused on the processors' internal registers, and in particular the pipeline, using fault injection. Results show that all errors observed without using the lockstep architecture are detected when the lockstep is active. After evaluating the impact of fault injection in the registers of the pipeline stages, it is possible to note that most of the faults provoke address mismatches.

1. Introduction

Fault detection capabilities are likely to be necessary even for several general-purpose systems, as the newest manufacturing technologies are particularly sensitive to radiations. When considering processor-based systems, fault detection mechanisms can be classified as software-based and hardware-based. Software-based approaches detect faults within the program's control flow or in the program's data, and also cover of those faults that affect the memory elements the processor embeds (e.g., the processor's state register, or temporary registers used by the arithmetic and logic units)[OH 02][CHE 00]. Fault detection is achieved by introducing instructions and information redundancies, followed by consistency checks among replications computations. This technique also introduces a memory overhead, because of data and instructions replication, while the consistency checks can affect the performance. This overhead may not be acceptable in certain type of applications.

Hardware-based techniques consist in introducing in the system some redundant hardware. Some researches proposed to adopt special-purpose hardware modules known as watchdogs which work in close cooperation with the processor in order to monitor the control-flow execution, the data accesses patterns [DUP 02], and to perform consistency checks [MAH 88], while letting the software running on the processor mostly untouched. Although watchdogs have limited impact on the performance of the hardened system, they may require non-negligible development efforts. Moreover, watchdogs are barely portable among different processors.

To combine the benefits of software-based approaches with those of hardware-based ones, a hybrid fault detection solution was introduced in [BER 06]. This technique combines the adoption of software techniques in a minimal version, for implementing instruction and data redundancy, with the introduction of an Infrastructure-Intellectual Property core (I-IP) attached to the processor, for running consistency checks. The I-IP behavior does not depend on the application the processor executes, and therefore it is widely portable among different applications.

The recent introduction of processors inside FPGA devices allows implementing on a single device complex hardware/software systems where the software runs on the embedded processor, which communicates with custom hardware implemented by the FPGA's logic resources. Due to their characteristics, hardware based approaches, as that introduced in [PIG 06][NG 04], are well suited for hardening the processor cores FPGAs embed. These researches explored the possible benefits stemming from the duplication of the system's processor and the insertion of special monitoring modules that check whether the duplicated processors execute the same operations. These approaches are particularly attractive when processor duplication does not impact severely the hardware costs. Moreover, since they do not require modifications to the software running on the duplicated processors, as conversely happens for the software-implemented approaches, commercial off the shelf software components can be hardened seamlessly.

The purpose of this paper is to analyze the concept of lockstep [NG 04] when applied to a complex pipelined processor. In particular, we are interested in analyzing how a lockstep system, where two processors (a master and a slave) execute the same software concurrently, while a checker unit compares their results every clock cycle, behaves in presence of soft errors affecting the most sensitive processor's memory elements, i.e., the pipeline's registers. The paper is organized as follows. Section 2 presents the basic concepts about the lockstep error-detection mechanism. The architecture analyzed in this paper is described in Section 3. Analyses and results are exposed in Section 4.

2. Lockstep Technique

The lockstep technique is a fault tolerance mechanism that adopts two identical processors (a *master* and a *checker*) with different roles [NEL 90]. The master and the checker execute the same program but just the master interfaces with the other system hardware components (such as memory, peripherals, and other devices). The outputs of both CPUs are connected to a *checker logic* that compares the results produced and possibly rises up a mismatch signal. The name lockstep is due to the fact that both processors have to lock their execution at the same time and then they restart together in synchronous way; in this way the two processors state is always the same for each clock period in which the application is executed.

Fig. 1 shows the high level scheme of a fault tolerant system based on lockstep. The inputs are connected to both processors so that both processors always receive the same input. The master outputs are connected to the system resources and to the checker unit, while the checker processor outputs are connected to the checker unit, only. The checker unit compares the results and signals possible mismatches; if the results are equal, the program continues the execution normally.

Using the lockstep mechanism it is possible to achieve high fault detection capabilities with respect to faults happening in both the processors, but not with respect to both transient and permanent faults in the system memory. In fact, if the two processors are completely identical, they will share the same data and instructions fetched from the same memory segments addressed by the master processor.

Xilinx [NG 04] proposed a lockstep architecture with two identical processor (PPC405) included in a wrapper structure with a comparator inside that gives in output eventual errors and output results. The IBM PowerPC 750GX processor presents an integrated lockstep facility (LSF). Mismatches are signaled by the checkstop-out pin of the checker processor, allowing mismatches diagnosis and giving the system designer a simple mismatch signal that the system processor or system logic can use [ELS 05]. HP proposed the Itanium2 NonStop Advanced Architecture (NSAA), where the key physical component is the LSU (off-chip Logical Synchronization Unit). LSU implements lockstep for the HP NS-series Integrity Itanium 2 servers. These LSUs process sequences of instructions as a comparison unit, and compare the results [HP 03].

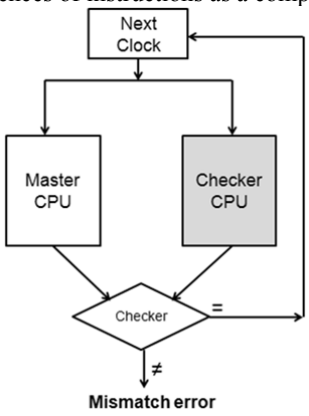


Fig. 1 - Lockstep scheme

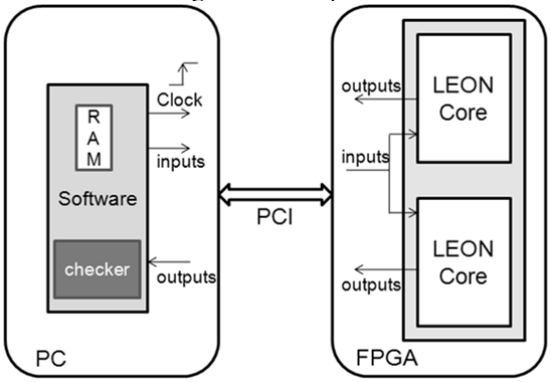


Fig. 2 - Adopted lockstep scheme

The most important advantage of the lockstep mechanism is the fast error detection (i.e., the very low error latency). This characteristic could be essential in real time systems and it certainly improves reliability. It is also important to note that a faster error detection mechanism is not only able to provide a more reliable system, but it also decreases the probability of producing wrong outputs. The main disadvantage is the significant reduction (nearly 50%) in the achieved performance with respect to the available computational power. In fact, all the instructions are replicated and executed in both processors. Anyway, the safety is guaranteed for what concerns soft errors affecting the processors.

3. Experiments

The experimental analysis we performed aims at evaluating the impact of soft errors inside the pipeline's registers of a complex processor. For this purpose we selected the Leon processor, we developed a prototypical environment allowing injecting faults in the locations of interest, and we investigated the robustness of the lockstep architecture when applied to Leon [LEO 04].

The prototypical environment used in this experiment is composed of 2 main parts: a hardware component and a software component, showed in fig. 2. The communication between the parts is performed by one PCI connection. The hardware component is implemented in a Xilinx's XVC1000E FPGA [XIL 02] and adopts two identical Leon 32-bit processors included in a wrapper structure designed to allow the fault injection and the access to internal signals and buses of the processors. The summary of resources used to implement our Leon-based lockstep systems, plus the resources needed for fault injection are presented in tab. 1.

The communication between the FPGA and the software is executed resorting to the ADM-XRC SDK [ADM 06]. This is a set of resources including an API intended to assist the user in creating an application using one of Alpha Data's ADM-XRC range of reconfigurable coprocessors. The API library takes care of open, close and device I/O control calls to the driver.

The software component controls the CPUs synchronism, implements the RAM memory, controls the access to the RAM and includes the checker module. This part accesses pre-compiled Leon programs stored in the RAM memory implemented in software. Moreover, it sends the signals to the processors and receives the outputs from both processors. The outputs are compared by the checker unit. The checker unit is responsible for comparing the data, the control and the address busses of the master and checker CPU, and for signaling any kind of mismatch when found. If one mismatch is reported, the program is aborted. If no mismatch is observed, the program enables the CPUs to execute the next instruction. Moreover, the program detects software and hardware abnormal behavior.

The checker unit is able to identify seven kinds of mismatches after a fault injection: *Address mismatch,Data mismatch*, *Control mismatch*, *Address, Data and Control mismatch*, *Address and Control mismatch, Address and Data mismatch*, and *Data and Control mismatch*. It is important to note that this seven kinds are report just for analyze the effects of one fault on the buses. From a practical point of view, it is not necessary to verify all the seven possibilities, but just the three first ones. For example, after the observation of one address mismatch, the mechanism could abort the current execution or initialize one routine to correct the system.

The hardware part of the adopted environment implements two Leon 32-bit processors. Each processor is coded in VHDL and models the IEEE-1754 (SPARC V8) architecture provided under Lesser GNU Public License (LGPL). The Leon core has the following features: separate instruction and data cache interface, support for 2 –32-registers windows, configurable multiplier, optional 16x16 bit MAC with 40-bit accumulator, radix-2 divider and 5-stage instruction pipeline.

4. Results and Conclusions

All results were obtained executing the experiment with four benchmarks. The benchmarks are: Ellipf, FIR filter, Kalman and Matrix multiplication. Details of the benchmarks are presented in tab. 2. The fault injection system allows defining how many faults have to be injected and in which stage of the pipeline they have to be injected. For each program, the experiment consisted in injecting faults in one pipeline stage and observing the results with and without the lockstep mechanism.

Without lockstep each fault can produce some effect belonging to the following categories:

- *Silent*: the program finishes correctly. The injected fault does not interfere in normal execution of the program and the output of the program is equal to the output default.
- *Timeout*: the fault injected affects the program in such a way that the program is not able to finish in a pre-defined time. Timeout is defined for each program taking in account the number of clock cycles required by the program to finish correctly.
- *Detected*: the fault triggered some error detection mechanism the processor embeds (i.e., the illegal instruction exception).
- *Wrong Answer*: the program finishes but the output is different from the expected one.

When lockstep is active a further result is possible: *Mismatches*. Mismatches correspond to cases in which one injected fault provokes a difference in value between one of the CPUs buses, i.e., the checker signals one mismatch.

Fig. 3 presents the results which have been gathered. For each program 1785 faults were injected. It is possible to observe that with lockstep we never observed any Timeout, Detected and Wrong Answer case. This is because these errors are previously detected as mismatch. These experiments showed that the fault detection mechanism needs more cycles of clock to detect one of these errors than to indicate that a first mismatch has been occurred. Therefore, it can be extremely benefic to improve the time for error detection.

Fig. 4 shows that 56.78% of wrong answers detected without lockstep are previously detected as Address Mismatch with lockstep, 23.27% as Data Mismatch and 15.86% have had mismatches among all buses.

Fig. 5 presents the same analysis for the detected results without lockstep. In this case, most of detected errors correspond to an Address Mismatch (40, 8%) or a Control Mismatch (39, 47%). Moreover, about 10% have reported Address and Control Mismatch first error.

However, some of the found mismatches do not imply a future error. This explains why the number of silent faults decreases. Fig. 6 details this problem.Approximately 14% of silent results obtained without Lockstep correspond to a mismatch when the Lockstep mechanism is adopted. Most of them are mismatches between the address buses.

Moreover, the experiments executed allow evaluating the effects in each pipeline stage of the fault injection. Faults are injected in internal registers of the pipeline stages and the Checker unit detected if each fault provokes any change in address, data or control buses comparing the buses of Master CPU and Checker CPU. The results show that most of the faults injected in the registers provoke Address mismatches, independently on the pipeline stage.

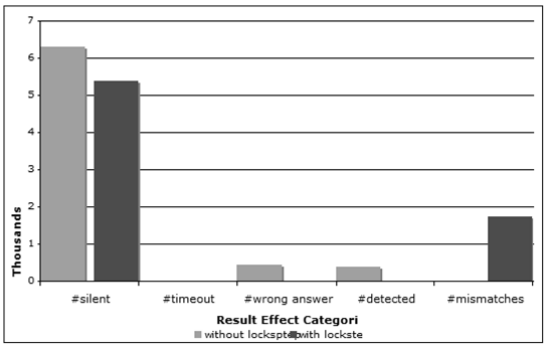


Fig. 3 - Events reported after the fault injection

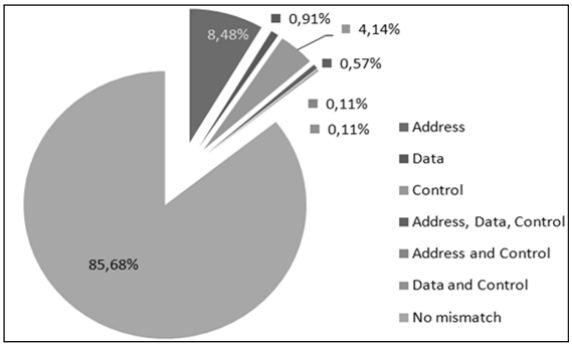


Fig. 4 - Wrong answers detected with lockstep

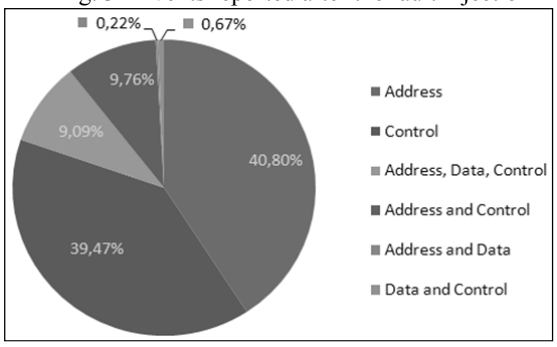


Fig. 5 - Errors detected with lockstep

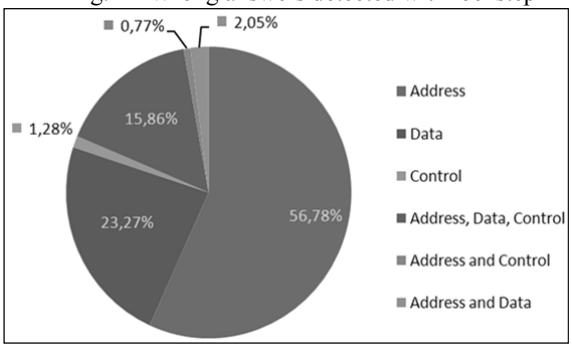


Fig. 6 - Silent results with mismatches

5. References

[ADM 06] ADM-XRC SDK 4.7.0 User Guide Win32 Document version: 4.7.0.1, 2006.

[BER 06] P. BERNARDI, L. BOLZANI, M. REBAUDENGO, M. SONZA REORDA, F. VARGAS, M. VIOLANTE. **Hybrid Fault Detection Techniques in Systems-on-a-Chip.** In: IEEE Trans. on Computers, Vol. 55, No. 2, Feb. 2006, pp. 185- 198

[CHE 00] P. CHEYNET, B. NICOLESCU, R. VELAZCO, M. REBAUDENGO, M. SONZA REORDA, M. VIOLANTE. **Experimentally evaluating an automatic approach for generating safety-critical**

Tab. 1 - Xilinx utilization summary

	Used	Available	%
Slice Flip Flops	4.476	24.576	18
Occupied Slices	10.714	12.288	87
4-LUTs	19.321	24.576	78
Bonded IOBs	61	404	15
Block RAMs	20	96	20

Tab. 2 - Features of the adopted benchmarks

Program	Clock cycles	Code Size	Data Size
Ellipf	9854	736 Bytes	168 Bytes
FIR	6235	872 Bytes	0 Bytes
Kalman	32811	1232 Bytes	2376 Bytes
Matrix	10713	376Bytes	312 Bytes

software with respect to transient errors. In: IEEE Trans. on Nuclear Science, Vol. 47, No. 6, December 2000, pp. 2231-2236

[DUP 02] E. DUPONT, M. NIKOLAIDIS, AND P. ROHR. **Embedded Robustness IPs for Transient-Error-Free ICs**. In: IEEE Design and Test of Computers, Vol. 19, no. 3, pp. 56 – 70, May/June 2002

[ELS 05] D. ELSON. **PowerPC processor tips: Two PowerPC 750GXs are better than one.** IBM library, Oct, 2005.

[HP 03] HP NonStop Advanced Architecture White Paper, 2003.

[LEO 04] LEON2 Processor User's Manual Version 1.0.30, 2004.

[MAH 88] A. MAHMOOD AND E.J. MCCLUCKEY. **Concurrent Error Detection Using Watchdog Processors-A Survey**. IEEE Trans. on Comp., vol. 37, no. 2, pp. 160-174, Feb. 1988

[NEL 90] V. P. NELSON. **Fault-tolerant computing: fundamental concepts.** In: Computer, v.23, Jul, 1990, pp 19-35.

[NG 04] H. H. NG. **PPC405 Lockstep System on ML310.** In: Virtex-II Pro Family, Oct. 2004

[PIG 06] M. PIGNOL**. DMT and DT2: two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers.** In: Proc. IEEE Int.l On-Line Testing Symposium 2006, 2006, pp. 10-12

[OH 02] N. OH, P. P. SHIRVANI, E. J. MCCLUSKEY. **Control-Flow Checking by Software Signatures**. In: IEEE Trans. on Reliability, Vol. 51, no. 2, pp. 111 – 122, Mar. 2002

[XIL 02] XilinxVirtex-E Data Sheet, 2002.

Hardware Implementation of a Base-2 Common Factor FFT Algorithm With Frequency Decimation

[1]Guilherme Mauch de Freitas, [1]Bruno Zatt, [1]Sergio Bampi
{gmfreitas,bzatt,bampi}@inf.ufrgs.br

[1]UFRGS – Informatics Institute (II)

Abstract

This work describes a hardware implementation of a base-2 common factor FFT algorithm with frequency decimation to 16 input points with 16 bits resolution, considering only real numbers. Three implementation versions are presented: with input and output registers, pipelined and multiplexed. The algorithm is described in VHDL language and was synthesized to Xilinx Virtex-II Pro xc2vp30 FPGA. The target application is the channels modulation for digital TV signal transmission.

1. Introduction

The fast Fourier transform (FFT) is an efficient class of algorithms for the discrete Fourier transform (DFT) processing. The FFT algorithms are typically projected to minimize the number of additions and multiplications, keeping a simple form in their structure.

There is a series of algorithms for the FFT calculation. The most popular class of these algorithms is named common factor FFT. These algorithms are also known as Cooley-Tukey because it has been popularized by their researchers. The common factor algorithms which presents the most computational structure simplicity are the base 2 and base 4. However, algorithms with base higher than four present reduced computer efficiency.

This work focus on base 2 common factor algorithm with frequency decimation. In this algorithm, the sample values in frequency are decimated in each FFT stage. The operations are made in a pair of input signals. The basic structure of this FFT algorithm is the butterfly, showed below on fig. 1.

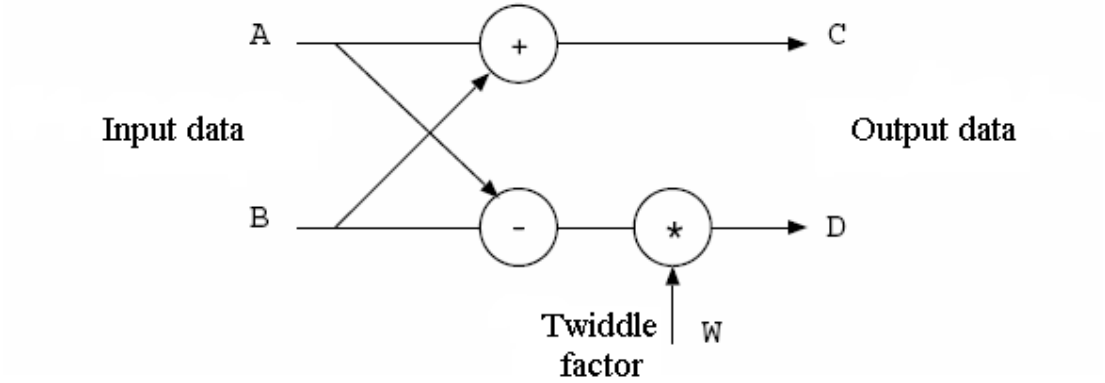


Fig. 1 – Diagram of a butterfly with frequency decimation of common factor on the base 2 [COS 02].

In the butterfly structure, the samples are multiplied by fixed coefficients (twiddle factor). These coefficients represent multiple values of a factor $2\pi/N$, where N is the number of input points of the FFT.

2. Structure

The purpose of this work to offer different solutions for the implementation of the base 2 FFT algorithm, with frequency decimation, to resolve the real DFT, unidimensional, with 16 inputs of 16 bits. The structure of this algorithm is established through the correct arrangement of the butterflies. This structure is divided into 4 stages, where each stage contains 8 butterflies and the number of stages follows the $\log_2 N$ relation.

The complete calculation structure can be observed on fig. 2. From the knowledge of the operation and data flow of the algorithm, it is possible to realize the basic implementation and to do a few pertinent variations.

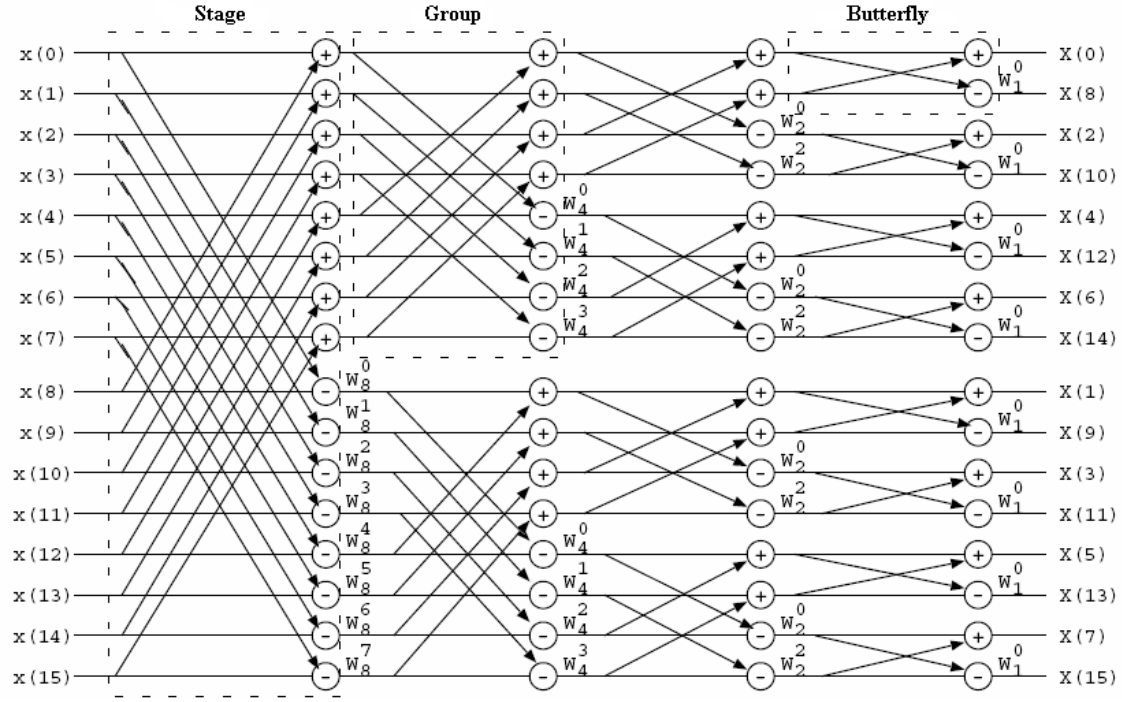


Fig. 2 – Common factor FFT data flow on base 2 with frequency decimation [COS 02].

3. Implementation

The implementation was realized using VHDL hardware description language, using the ISE software (Xilinx). The code was synthesized for xc2vp30 device, of Virtex-II Pro family. The first step of description was to describe an efficient code for butterfly implementation, because this is the base structure of all algorithm. The butterfly is formed by one adder, one subtractor and one multiplier, all of them with 16 bits. The multiplier output data is represented in 32 bits, however this data must be truncated to 16 bits, discarding the least significant 16 bits, because the output of one butterfly is used as the input of next butterfly stage. This process can be observed at the diagram showed on fig. 3. On the initial implementation, this butterfly is instantiated 32 times, eight for each stage. The interconnection is realized with internal signals, and the input and output signals are directly connected to input/output pins. The initial implementation purpose was to prove the general operation of butterfly code.

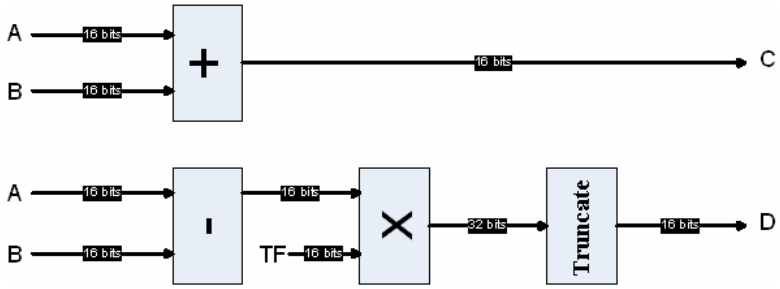


Fig. 3 – Butterfly´s implementation block diagram.

3.1. Register Implementation

In this implementation, all the input and output data including the twiddle factors are stored at 16 bits registers. This process allows the inclusion of this block on a digital signal processor (DSP) or on a system on chip (SoC), for example. The registers described in the code are type D flip-flops sensible to the rising edge of clock signal, with cell enable signal sensible to voltage level and asynchronous reset. The block diagram of this implementation is showed on fig. 4.

3.2. Pipelined Implementation

The pipelined implementation objective is to increase the FFT processing rate, becoming useful to applications with constant data input to be processed. Each structure stage was isolated with a registers bank in

their output, as shown in fig. 5. So, it is possible to process 4 samples at the same time, because the stages are not interdependent.

This pipelined version has a great applicability in signal modulation techniques. One of these techniques, the OFDM modulation (Orthogonal Frequency Division Multiplexing) has large application in data communications, being employed in wired communications, like ADSL (Asymmetric Digital Subscriber Line), or in wireless communications, like WI-FI and WIMAX.

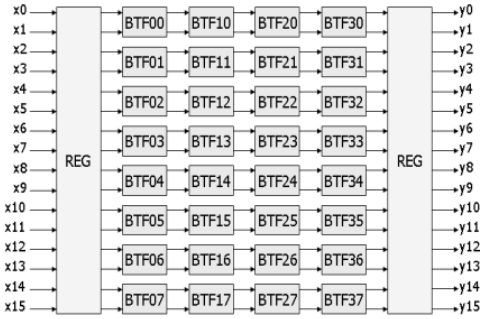


Fig. 4 – Register version.

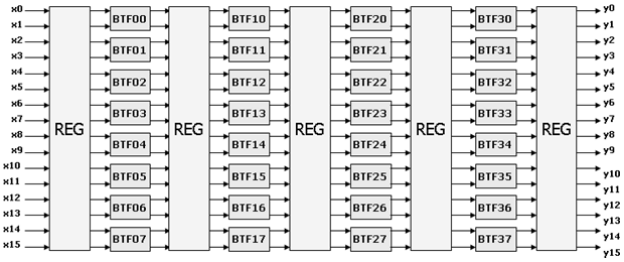


Fig. 5 – Pipelined version.

3.3. Multiplexed Implementation

This is another implementation with great applicability, which the main objective is the great reduction on the number of multipliers. This is an important fact that must be observed for the resolution of FFT's with most number of points. As the number of used multipliers in the FPGA is reduced, on greater applications probably it would not be possible to implement all the butterflies because each butterfly uses one multiplier. The propose solution is to use only one stage of the main structure, with 8 butterflies. For that it is necessary to make use of multiplexers in the input of each butterfly.

As the proposed structure has 4 stages of butterflies and in this implementation is used only one stage, each butterfly is used 4 times. The first step receives the input data of the system and the three later steps receive data from others butterflies. To make the control of input data in each butterfly, is used one multiplexer (MUX) with 4 inputs and 2 control signals. Besides multiplexing the input data, it is necessary to multiplex the twiddle factors signals. The control signals are established through a finite state machine, which on the first state the control variable is set to "00", on the second state to "01", on the third state to "10" and on the fourth state to "11". The state machine has an initial state, in which is realized the variables startup and the system initial configuration. Butterflies output data are also registered, but when the process is over, an output pin of the module is turned on to '1'.

The difficulty of this implementation is to correctly define the input signals of multiplexers, because the butterflies' structure does not follow a linear sequence, they are crossed. This is a factor that demands a lot of time on initial structure analysis. To facilitate the signals definition is advisable to analyze the butterflies that are on the same level of different stages.

An example of the diagram of the third level of the butterfly stage is presented in fig. 6. In this case, the lower inputs of the butterfly are: the input data x10 (state 0); the higher output of seventh butterfly (state 1); the lower output of third butterfly (state 2); the higher output of fourth butterfly. The higher inputs are also showed on the figure below.

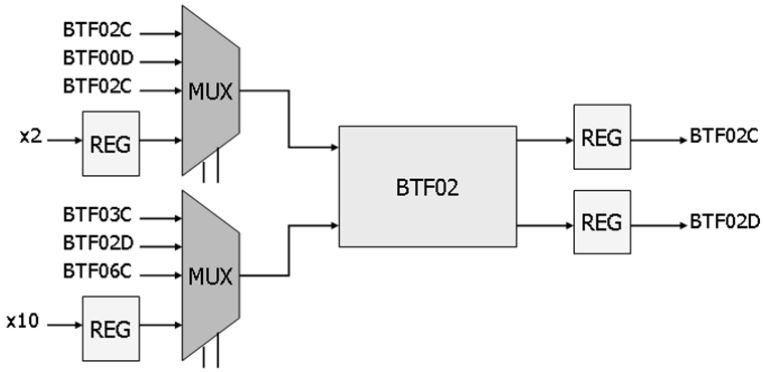


Fig. 6 – Multiplexed implementation.

4. Results

The results were obtained through VHDL code synthesis, using ISE software (Xilinx), for xc2vp30 device of Virtex-II Pro family. This specific device was chosen because of the availability to use a board of microelectronic UFRGS laboratory. Analyzing the synthesis results is possible to evaluate the advantages and disadvantages of each structure. All the analysis data are showed on the tab. 1.

Two of the four implementations presented most significantly results. The pipelined version allows the input of a new data in each 8.376 ns, operating with a frequency of 119.389 MHz. The multiplexed version reduced in 25% the number of multipliers, becoming easier to use another modules in the same FPGA. The disadvantage of the multiplexed version is the total time to realize a complete FFT calculation.

Tab. 1 – Synthesis results for xc2vp30 device.

	Slices	**Slice FF**	**4-Input LUTs**	**Mult 18x18**	**Delay (ns)**	**Max. Freq. (MHz)**
Without registers	512	0	1024	32	31.862	31.385
With registers	733	512	1024	32	28.513	35.072
Pipelined	1167	1280	1024	32	8.376	119.389
Multiplexed	883	853	1112	8	10.685	93.588

5. Conclusion

The main application of this FFT algorithm is to realize the channels modulation for transmission of digital TV signal. As many algorithms will work simultaneously on the same FPGA device, the most applicability implementation is the multiplexed version. The reason to use this implementation is the reduced multipliers utilization, because there are few of them on FPGAs but are very useful to signal processing applications. In the case of an application in which the number of FFT points are not high and there is a constant input data flow, the pipeline version is most indicated.

6. Future works

This project has a very high flexibility and high possible applications. However, a limiting factor is that only real number inputs are considered. Then, the next step is adapting the code to work with complex numbers, allowing real applications of this algorithm. Other possible modification for the channel modulation is to expand the algorithm for 1024 points.

At the moment the algorithm operates with complex numbers, it will be possible to make the code validation with mathematic softwares. With the conclusion of these next steps, the algorithm will be able to be used with another modules on transmission of digital TV signal.

7. References

[COO 65] COOLEY, J.; TUKEY, J. **An Algorithm for the Machine Calculation of Complex Fourier Series.** In: MATHEMATICS OF COMPUTATION, Washington, v.19, p.297-301, Apr. 1965.

[COS 02] COSTA, E. **Operadores Aritméticos de Baixo Consumo para Arquiteturas de Circuitos DSP.** Porto Alegre: PPGC da UFRGS, 2002.

[HAY 01] HAYKIN, S.; VEEN, B. V. **Sinais e Sistemas.** Trad. José Carlos Barbosa dos Santos. Porto Alegre: Bookman, 2001.

[OPP 89] OPPENHEIM, A.; SCHAFER, R. **Discrete-Time Signal Processing.** London: Prentice Hall Signal International, 1989.

[SUN 04] SUNKARA, D. **Design of Custom Instruction Set for FFT Using FPGA-Based NIOS Processors.** Florida: The Florida State University – College of Engineering, 2004.

Architectural Specification of a Microcontroller by Using ArchC

Maicon Carlos Pereira, Cesar Albenes Zeferino
{maicon, zeferino}@univali.br

Universidade do Vale do Itajaí
Centro de Ciências Tecnológicas da Terra e do Mar
Grupo de Sistemas Embarcados e Distribuídos

Abstract

In this paper, it is presented the architectural specification of a basic microcontroller, named µBIP, developed to be used in the teaching of concepts related to the design of embedded systems, at the software and the hardware levels. The microcontroller architecture is derived from a family of basic processors (BIP – Basic Instruction-set Processor) specially designed focusing on students' learning. In this paper, it is described the architecture of µBIP, which was specified by using ArchC, an architecture description language based on SystemC.

1. Introduction

The increasing demand for products based on embedded systems has carried out to the need for engineers able to work on the different phases of the design of such computing systems. However, some of the basic concepts in this field have not been discussed in most of undergraduate courses in Computer Science [HEN 07].

Computer Architecture and Organization disciplines play a major role in the development of abilities and skills of students of Computer Science and Computer Engineering courses, especially for the ones which will work on the design of embedded systems. However, many students have difficulties in understanding the first concepts taught in the introductory disciplines in this domain. To minimize this problem, teachers continuously work in the search of new methods to help the students in understanding such concepts, by making a link with other disciplines related to programming, compilers, and operating systems [MOR 06].

In this way, researchers of the Embedded and Distributed Systems Group (GSED) of University of Vale do Itajaí (Univali) made a specification of a processor family named BIP – Basic Instruction-set Processor. This family aims at facilitating the understanding of introductory concepts on Computer Architecture and Organization in the first terms of an undergraduate course. A second goal of BIP family is to promote an interdisciplinary approach, allowing teachers of other disciplines to use BIP specification in practical activities.

Current BIP specification is based on the simplest architectural alternatives in order to ease the programming and the building of a processor: (*i*) it is an accumulator-based architecture; (*ii*) all the instructions are based on a single instruction format; and (*ii*) there are only two addressing modes (Immediate and Direct). The first version of BIP (BIP I) is an "as simple as possible" architecture, with a very small Instruction Set Architecture (ISA), including only load, store and arithmetic instructions (add and subtract). The second version (BIP II) adds conditional and unconditional branch instructions.

BIP I and BIP II were already applied in several disciplines in undergraduate courses in the Computer Science field, allowing the students to improve their comprehension about several concepts. However, due to their simplicity, the applicability of these CPUs shown to be limited to be used in more advanced disciplines, because of the lack of several features of real processors applied in embedded systems, like I/O ports, timers, and other peripherals.

This paper presents results of an ongoing project that aims at the design of a simplified microcontroller named µBIP (micro BIP). This microcontroller is based on the BIP II specification and extends its architecture by adding new instructions. It also includes support for the integration of peripherals to the CPU. In this paper, it is described the architectural specification, which was done with the aid of ArchC, an Architecture Description Language – ADL developed at the Instituto de Computação of the University of Campinas (IC-UNICAMP) [ARC 07a].

This text is organized in seven sections. Section 2 presents some concepts on ArchC, while Section 3 describes the µBIP architecture. The ArchC µBIP model is presented is Section 4, followed by the description of the method used to validate the specified architecture, in Section 5. Concluding, in Section 6, is done a briefly discussion about the use of µBIP in Education, and, in Section 7, they are presented the final remarks.

2. ArchC

An ADL is a computer language used to describe software and/or system architectures. As defined by [BAL 05], given a description of a processor architecture, an ADL allows to automatically generate software tools (ISA simulator, compiler, assembler, linker and debugger). ArchC is an ADL which allows describing the processor architecture and the memory hierarchy. In its current version (v.2.0), it also allows generating ISA simulator, linker and debugger tools.

The description of an architecture using ArchC is composed by two main blocks [ARC 07b]: Architecture Resources (AC_ARCH) and Instruction Set Architecture (AC_ISA). The first one includes the description of the amount and type of memory, the amount of registers and the pipeline. The second one includes information about the instruction set, as: (*i*) instruction width; (*ii*) instruction format; (*iii*) opcodes; (*iv*) decoding; and (*v*) instruction behavior.

The modeling of a processor in ArchC can be done at different levels of abstraction. At the functional level, a model includes only a few information, but they allow generating tools to analyze the instruction set details about timing or pipeline operation. At the cycle-accurate level, it is possible to build a model closer to a realistic implementation, including information about timing and pipeline.

3. µBIP Architecture

In µBIP microcontroller, instructions and data are 16-bit wide. There is only one instruction format, shown in Fig. 1. This instruction format considers one implicit operand, the ACC (accumulator) register, and one explicit operand (the *operand* field), which can be a constant (in immediate addressing mode), a variable (in direct addressing mode) or a vector index (in indirect addressing mode).

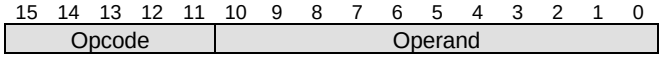


Fig. 1 – Instruction format

The instruction set includes 29 instructions (BIP II had only 15), which are organized in the nine classes shown in Tab. 1.

Tab. 1 – Instruction-set

Class	**Instructions**
Store	`STO`
Load	`LD, LDI`
Arithmetic	`ADD, ADDI, SUB, SUBI`
Logical	`AND, OR, XOR, ANDI, ORI, XORI, NOT`
Control	`HLT`
Branch	`BEQ, BNE, BGT, BGE, BLT, BLE, JMP`
Logical Shift	`SLL, SRL`
Vector access	`STOV, LDV`
Procedures	`RET, RETI, CALL`

µBIP has Harvard architecture, program and data memories. Address space is organized as a 2-Kword program space and a 2-Kword data space. I/O is memory mapped and occupies half of the data memory space.

The first version of µBIP uses a mono-cycle organization and includes the following peripheral and hardware features: (a) two 16-bit I/O ports with individual direction control for each pin; (b) a 16-bit timer; (c) an interrupt controller; and (d) hardware support for procedure calls.

4. Modeling µBIP with ArchC

Since µBIP uses a mono-cycle organization, functional and cycle-accurate models in ArchC are equivalent. Its ArchC description is composed by the following files: (a) *ubip.ac*; (b) *ubip_isa.ac*; (c) *ubip_isa.cpp*; and (d) *ubip_address.h*. The first file (*ubip.ac* file) describes the architecture resources: memories, registers, word size and endianness (see Fig. 2).

```
AC_ARCH(ubip){
  ac_mem memRAM:4K; //4Kbyte/2K-Words
  ac_mem memROM:4K; //4Kbyte/2K-Words
  ac_reg PC, ACC, STATUS;
  ac_wordsize 16;
  ARCH_CTOR(ubip) {
    ac_isa("ubip_isa.ac");
    set_endian("big");
  };
};
```

Fig. 2 – Description of architecture resources (*ubip.ac*)

The second file (*ubip_isa.ac*), partially shown in Fig. 3, is where the instruction format and its fields (*op* and *operand*) are specified. Following, the instructions set is declared and it is defined a map (*ac_asm_map*) describing the allowed values for the instruction operands. Two functions are used to define the syntax (*set_asm*) and the opcode (*set_decoder*) of a given instruction. Another function (*pseudo_instr*) allows defining pseudo-instruction which, at the assembling process, is replaced by a sequence of native instructions.

```
AC_ISA(ubip){
  ac_format INS_FORMAT  = "%op:5 %operand:11";

  ac_instr<INS_FORMAT> add, addi, sub, subi;

  ac_asm_map ubipSFR {
    "$"[0..2047]  = [0..2047];
    "$port0_dir"  = 1024;
    "$port0_data" = 1025;
    "$tmr0_config"= 1040;
    ...
  }
  ISA_CTOR(ubip){
    //--- ASSEMBLY ---
    add.set_asm("add %ubipSFR", operand);
    add.set_asm("add %exp", operand);
    add.set_decoder(op=0x04);
    ...
    //Pseudo-Instructions
    pseudo_instr("psto %ubipSFR, %imm"){
      "ldi %1";
      "sto %0";
    }
    ...
  };
};
```

Fig. 3 – Description of Instruction-Set Architecture (*ubip_isa.ac*)

The third file (*ubip_isa.cpp*) describes the behavior of instructions and peripherals. Fig. 4 and Fig. 5 present part of this file, where they are shown the behavior of *ADD* instruction (*behaviour*(*add*)) and part of the

common behavior for all the instructions (*behavior*(*instruction*)). Two functions (*DataMemoryRead* and *DataMemoryWrite*) were implemented to deal with word addressing, since ArchC addressing is byte-oriented. The same problem had to be solved in the functions related with PC register (*inc_PC*, *set_PC* and *get_PC*). A number of constants used in *ubip_isa.cpp* are defined in the last of the four files which compose the ArchC µBIP model.

```
void ac_behavior( add ){
  ac_word operand1 = ACC.read();                        //Fetching operand 1
  ac_word operand2 = DataMemoryRead(memRAM,operand);    //Fetching operand 1
  ac_word result   = operand1 + operand2;               //Executing the operation
  ACC.write(result);                                    //Writtng result into ACC
  set_status(STATUS, Z, check_zero(result));            //Updating STATUS flags
  set_status(STATUS, N, check_negative(result));
  set_status(STATUS, C, check_carry_out(operand1, operand2));
  inc_PC(ac_pc);                                        //Updating PC
}
```

Fig. 4 – Behavior of ADD instruction (*ubip_isa.cpp*)

```
// For all the instruction do:
void ac_behavior( instruction ){
   //Check for interrupt requests on PORT0 and on Timer0
   bool bIntr = false;
   bIntr = bIntr | CheckInterruptPA0(memRAM, ac_pc, tempMem[PORT0_DATA],
                            DataMemoryRead(memRAM,PORT0_DATA));
   bIntr = bIntr | (trm00_inc(memRAM, ac_pc, ac_instr_counter)); //Update Timer0
   if (bIntr) ac_annul();
   (...)
}
```

Fig. 5 – Common behavior for all the instructions (*ubip_isa.cpp*)

5. Validation

For the validation of the developed ArchC model, it was defined a test plan which were applied by using software tools (ISA simulator, assembler and linker) automatically generated by ArchC. This test plan included a set of unitary tests for each instruction, which verified the correctness of these instructions regarding the operation to be executed and the state of registers and data memory, before and after the instruction execution.

The validation process will be extended with the use of test bench codes based on the benchmarks available in the Dalton Project site [DAL 01].

6. Discussion

While BIP I and BIP II were specified to be used in introductory classes on Computer Architecture, µBIP is intended to be applied in courses focusing in Digital System Design and Embedded System Design. We envision that it could become a useful tool to aid students in learning issues related to these disciplines.

Advanced studies can be carried out, like, for instance, adding new peripheral or implementing µBIP in a pipeline-based organization.

7. Conclusions

In this paper, it was presented some results of an ongoing project which aims at the development of a basic microcontroller to be used in Education, especially in advanced disciplines of undergraduate courses in Computer Science and Computer Engineering. But it could also be used in graduate courses.

In the first phase of this development, ArchC ADL was used to aid the architecture specification and validation. The software tools automatically generated by ArchC were very useful in this process. In the next phase, it will be implemented the VHDL model for the monocycle organization of μBIP, which will be synthesized and validated in FPGA.

8. References

[ARC 07a] THE ARCHC TEAM. **The ArchC architecture description language v2.0: reference manual**. Campinas: The ArchC Team, 2007.

[ARC 07b] THE ARCHC TEAM. **The ArchC project home page**. Disponível em: <http://www.archc.org>.

[BAL 05] BALDASSIN, Alexandro. **Geração automática de montadores em ArchC**. 2005. Dissertação (Mestrado em Ciência da Computação)-Programa de Pós-Graduação em Ciência da Computação, Universidade Estadual de Campinas, Campinas, 2005.

[DAL 01] THE DALTON PROJECT TEAM. **The UCR Dalton Project**. University of California – Riverside, 2001. Disponível em: < http://www.cs.ucr.edu/~dalton/ >.

[HEN 07] HENZINGER, Thomas A; SIFAKIS, Joseph. **The Discipline of Embedded Systems Design**. Computer. v. 40, n.10, p. 32-40. out. 2007.

[MOR 06] MORANDI, Diana; PEREIRA, Maicon Carlos; RAABE, André Luis Alice; ZEFERINO, Cesar Albenes . **Um processador básico para o ensino de conceitos de arquitetura e organização de computadores**. Hífen, Uruguaiana, v. 30, p. 73-80, 2006.

New Dedicated Architectures for the Radix-16 Multiplication

[1]Leandro Zafalon Pieper, [1]Eduardo A. C. Costa, [1]Sergio J. M. de Almeida, [2]Sérgio Bampi, [3]José C. Monteiro
leandrozaf@pop.com.br,{ecosta, smelo}@ucpel.tche.br, bampi@inf.ufrgs.br, jcm@inesc-id.pt
[1]Universidade Católica de Pelotas - UCPel
[2] Universidade Federal do Rio Grande do Sul - UFRGS
[3] Instituto de Engenharia e Sistemas de Computadores – INESC-ID

Abstract

In this work are presented two new dedicated blocks for the radix-16 multiplication. The blocks compose the structure of the 16 bit binary array multiplier proposed by [COS 02]. In the array multiplier of [COS 02] the blocks that perform the radix-16 multiplication are automatically synthesized by using SIS environment. In this work we tested two other new structures by using less complex multiplication blocks and more efficient adders such as Carry Save (CSA). We have compared area, delay and power consumption of the 16 bit array multiplier using the new dedicated modules, by using SIS and SLS (Switch Level Simulator) tools. The results presented show that with the new radix-16 structures it is possible to save area, delay and power consumption when compared against the previous original structure.

1. Introduction

In this work we improve the radix-2^m binary array multiplier architecture that was proposed in [COS 02], by using different schemes for the dedicated modules that perform the radix-16 multiplication. The multiplier of [COS 02] uses radix-2^m encoding for the operands. This aspect enables the array multiplier to reduce the number of partial lines by keeping the same level of regularity presented by a conventional array multiplier [HWA 79]. In the binary array multiplier, the radix-2^m multiplication is performed by dedicated multiplication blocks. For the radix-4 operation (m=2) these blocks are easily obtained. However, for the operation performed by blocks with radices higher than 4, the construction of the dedicated multiplication blocks become more complex. In the multiplier of [COS 02], the radix-16 blocks are synthesized automatically by using SIS environment [SEN 92]. As these blocks appear in the binary array multiplier circuit (W/m)-1 times, where W is the number of bits, it is important to reduce the complexity of these blocks. In this work we have proposed two new structures for the radix-16 dedicated blocks, using less complex radix-4 blocks, Ripple Carry (RCA) and Carry Save (CSA) adders.

The dedicated blocks were all tested in the 16-bit binary array multiplier of [COS 02] and the results we have found show that we can reduce area, delay and power consumption of this multiplier by using the new proposed blocks.

2. Overview of 2's Complement Array Multiplier

For the operation of a radix-2^m multiplication, the operands are split into groups of m bits. Each of these groups can be seen as representing a digit in a radix-2^m. Hence, the radix-2^m multiplier architecture follows the basic multiplication operation of numbers represented in radix-2^m. The radix-2^m operation in 2's complement representation is illustrated in Figure 1, for a radix-16 multiplication example.

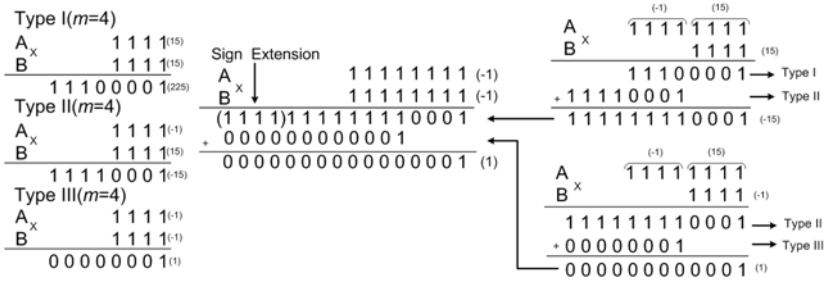


Fig. 8 - Example of a 2's complement 8-bit wide radix-16 multiplication

For the W-m least significant bits of the operands unsigned multiplication can be used. The partial product modules at the left and bottom of the array need to be different to handle the sign of the operands.

For this architecture, three types of modules are needed. Type I are the unsigned modules. Type II modules handle the m-bit partial product of an unsigned value with a 2's complement value. Finally, Type III modules

that operate on two signed values. Only one Type III module is required for any type of multiplier, whereas $2\frac{W}{m}$ -2 Type II modules and $(\frac{W}{m} - 1)^2$ Type I modules are needed. We present a concrete diagram for W=16 bit wide operands using radix-16 (m=4) in Figure 2.

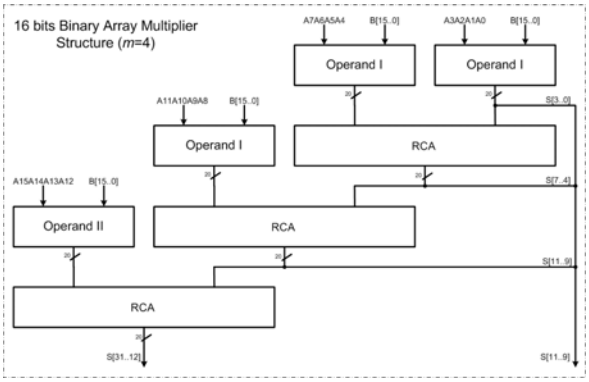


Fig. 2 - 16 bit wide binary array multiplier architecture (m=4)

Figure 3 shows the internal structure of the Operand I (illustrated in Figure 2). As can be observed in Figure 3, the structure of this operand is composed by m=4 Type I multiplier blocks and adders to produce the partial line result. In this work we have proposed three new alternatives to optimize the m=4 Type I multiplier block.

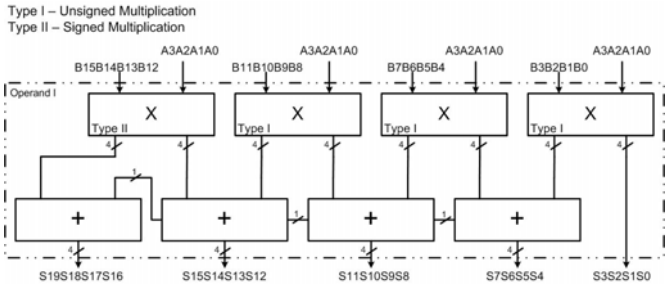


Fig. 3 - Operand I structure

3. Dedicated blocks for the radix-16 multiplication

In the work of [COS 02] the dedicated blocks for the radix-16 multiplication are all described in PLA format and automatically synthesized by SIS environment. In the PLA description all the inputs and outputs of the radix-16 multiplication are taken into account. The PLA description is mapped onto a gate level. The radix-16 dedicated blocks are finally generated in Berkeley Logic Interchange Format (BLIF). It was observed that these dedicated blocks automatically generated by SIS are complex and thus, in this work are proposed two new alternatives for the improvement of this block [PIE 08]. This is an important approach because in [COS 02] it was shown that the radix-16 binary array multiplier, that uses this dedicated block, is more efficient than the radix-4 architecture. This due to the less amount of partial lines used in the radix-16 operation.

3.1 The use of radix-4 blocks and RCA Adders

This first alternative uses dedicated radix-4 (m=2) blocks. This block is easily obtained by Boolean logic. For this block, only 8 logic gates are needed. Thus, the radix-16 block is arranged by using simple radix-4 blocks and RCA, as can be observed in Figure 4(b). Figure 4(a) shows an example of the multiplication of two 4 bit operands. As can be observed, two bits are multiplied at a time and groups of two bits of the partial terms are added in order to obtain the final result.

As can be observed in Figure 4(b), only one full adder, three half adders and three adders 2:1 (addition of 2 bits and the carry-in) are needed to compose the radix-16 block. In this block, the critical path is composed by one multiplication block (m=2), two half adders and one 2:1 adder, as can be observed in the dotted lines of Figure 4(b).

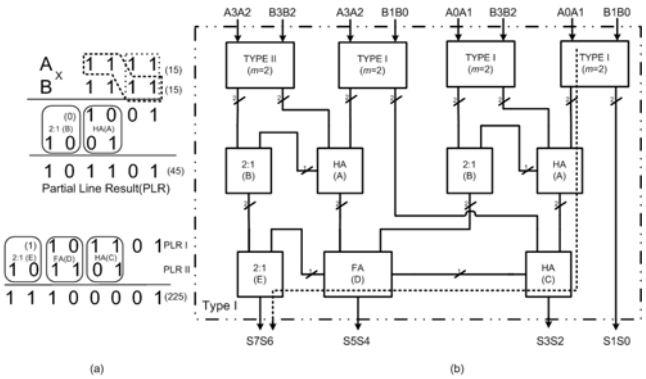


Fig. 4 - Radix-16 block composed by m=2 multiplier blocks and RCA

3.2 The use of radix-4 blocks and CSA Adders

In this alternative is used the same radix-4 block. But now with a more efficient adder called CSA [KIM 98] for the addition of the partial lines, as can be observed in the example of Figure 5(a). This example shows that by using CSA three numbers are partially added together. The basic idea is that three numbers can be reduced to 2, in a 3:2 compressor, by doing the addition while keeping the carries and the sum separate. In a previous work [FON 05], the CSA was applied to the partial product lines of the radix-4 multiplier and reductions on area, delay and power consumption were reported.

Figure 5(b) shows the use of CSA in the internal structure of the radix-16 block. As can be observed in the dotted lines of this figure, the critical path is composed by the sum of the delays of one multiplication block *m*=2, one CSA block, 1 half adder and one full adder.

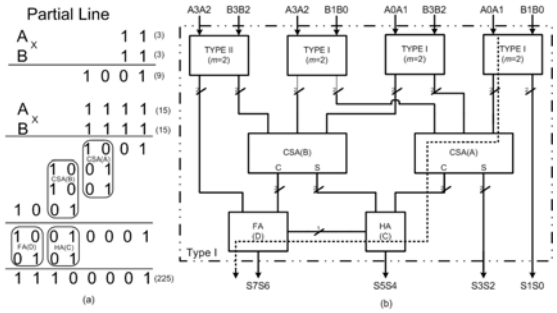


Fig. 5 - Radix-16 block composed by m=2 multiplier blocks and CSA

4. Performance Results

In this section is presented results for *W*=16-bit multiplier architectures on radix 16 (*m*=4) by using the original and the new alternatives for the dedicated radix-16 multiplication blocks presented before. Area and delay were obtained by using SIS tool and power results were obtained with SLS tool. Area results are presented in terms of number of literals. Delay results were obtained using the worst delay propagation between the input and output signals. Power results were obtained by using the average power value of the SLS tool [GEN 78]. For the power simulation we have applied a random pattern signal with 10000 input vectors.

4.1 Area results

As can be observed in Table 1, the two new alternatives present less area than the original one. This occurs because the new alternatives use more simple dedicated multiplication block. As can be also observed in Table 1, the first new alternative (radix-4 and RCA) presents less area value between all the alternatives. In fact, since this alternative uses more simple radix-4 multiplication block and more simple half adders and 2:1 adders, this alternative become more simple than the others.

Tab. 2 - Area Results

Alternatives	Area (literals)	%
original	15932	---
m=2 and RCA	7544	-111.2
m=2 and CSA	8612	-85.0

4.2 Delay results

The second new alternative (radix-4 and CSA) presents the less delay value, as can be seen in Table 2. This occurs because this alternative uses more simple radix-4 multiplication block and more efficient tree of CSA. The use of CSA in the radix-16 block contributes for the reduction of carry propagation along the structure and thus, the critical path is reduced. As can be also observed in Table 2, the two new alternatives are more efficient than the original one. In fact, it was observed that the complexity of the original block contributes for the higher critical path presented by its internal structure.

Tab. 3 - Delay Results

Alternatives	Delay (ns)	%
original	234.0	---
m=2 and RCA	204.8	-14.2
m=2 and CSA	199.3	-17.4

4.3 Power consumption results

As can be observed in Table 3, the binary array multipliers using the two new alternatives are more efficient in terms of power consumption when compared against the original one. This occurs because the new alternatives present more simple and more regular structures, which contributes for the less amount of glitching inside the multiplication blocks.

Tab. 4 - Power Consuption Results

Alternatives	Power (mW) - random input	%
original	214.09	---
m=2 and RCA	189.94	-12.7
m=2 and CSA	190.42	-12.4

5. Conclusions

This work described two new alternatives for the dedicated radix-16 multiplication block which composes the radix-2^m binary array multiplier of [COS 02]. As was observed, the binary array multiplier using the new alternatives become more efficient, because the new radix-16 blocks use more simple blocks of multiplication (radix-4). Besides, the inclusion of CSA into the dedicated radix-16 multiplication block, improves the performance and reduces the power consumption of the binary array multiplier. As future work we intend to test our new alternatives into higher radices dedicated multiplication blocks.

6. References

[COS 02] COSTA, E.; MONTEIRO, J.; BAMPI, S. **A New Architecture for Signed Radix 2^m Pure Array Multipliers**. In *IEEE International Conference on Computer Design*, pages 112-117, 2002

[FON 05] FONSECA, M. R.; COSTA, E.; BAMPI, S.; MONTEIRO, J. **Performance Optimization of Radix-2^m Multipliers using Carry Save Adders**. In XI Iberchip Workshop-IWS, 2005.

[GEN 78] GENDEREN, A. **SLS: an efficient switch-level timing simulator using min-max voltage waveforms.** Int. *Conf. on Very Large Scale Integration*, VLSI, 1989, Munich, pages 79-88, 1978.

[HWA 79] HWANG, K. **Computer Arithmetic – Principles, Architecture and Design. School of Electrical Engineering**, 1979.

[KIM 98] KIM, T.; JAO, W.; TJIANG, S. **Arithmetic Optimization using Carry-Save-Adders**. *In 35th Design Automation Conference*, pages 433-438, 1998

[PIE 08] PIEPER, L.; COSTA, E.; ALMEIDA, S.; BAMPI, S.; MONTEIRO, J. **Efficient Dedicated Structures for Radix-16 Multiplication.** In XIV Iberchip – IWS, 2008.

[SEN 92] SENTOVICH, E. **SIS: a System for Sequential Circuit Synthesis**. *Berkeley: University of California*, 1992.

IMAGE & VIDEO PROCESSING I

4:2 Adder Compressor for the T Block´s Forward 4x4 Hadamard Transform of the H.264/AVC Video Compression Standard

[1]André Marcelo C. da Silva, [1]Eduardo A. C. da Costa, [1]Sergio J. M. de Almeida, [2]Sergio Bampi

{andresilva, ecosta, smelo}@ucpel.tche.br; bampi@inf.ufrgs.br

[1]Universidade Católica de Pelotas - UCPel
[2]Universidade Federal do Rio Grande do Sul - UFRGS

Abstract

This paper presents an initial study on the use of 4:2 adder compressors circuits in transform architectures that are dedicated to the H.264/AVC video compression standard. The H.264/AVC application in mobile devices such as laptops, palmtops and mobile phones require many low power improvements. This paper presents a structural description of the Forward 4x4 Hadamard Transform, which is used in the (T) Block of the H.264/AVC encoder, and is the unit responsible for forward transform, such as FDCT (Forward Discrete Cosine Transform), Forward 4x4 Hadamard Transform and Forward 2x2 Hadamard Transform. In this implementation we proposed the use of 4:2 adder compressors that are more efficient than the conventional Ripple Carry (RCA) adders. Those compressors perform the sum of four operands at a time with no critical path penalties. In this work we present the comparison of area and delay results from the synthesis in the SIS environment and the power results from the synthesis in the SLS environment. We have used 4:2 adder compressor in 14 and 16 bit wide adder circuits.

1. Introduction

Digital videos have been used in several different applications, such as Web, videoconferencing, broadcast transmissions, and also in SBTVD (Brazilian Digital Broadcast TV) have been worked since last year.

To manipulate these digital videos so that they can be transmitted efficiently, it is usually necessary to make the video compression, which aims at reducing the number of bits required to represent a digital video without any loss of significant information for the video representation. This is done through the operation of the redundancy property in the video sequences. With the redundancy much of the data required to represent a digital video is superfluous. Thus, the compression video appears as the main alternative for the efficient transmission data. In this aspect, the H.264/AVC video compression standard [WIE 03] is more efficient than the other existing standards, such as MPEG-2 and H.263 [ITU 00]. The efficiency of compression in the H.264/AVC is obtained at cost of the computational expense. This occurs because the increase of computational complexity in some modules of the H.264/AVC standard.

Besides the use of the video compression for transmission over a network of computers, mainly the Web, another important point to be explored is the information storage. In fact, the efficient video compression helps to optimize the use of resources such as DVDs, CDs, HDs and so on, that are used for the storage of the digital information.

The challenges of the video compression and H.264/AVC standard are being widely studied by researchers from academic and industry and it has been observed that many significant contributions can still be given, mainly in terms of power consumption reduction. This aspect is related to the growth of mobile devices that use digital videos, such as palmtops, portables digital video cameras and mobile phones. In this sense, it is very important to have the blocks for implementations of the standard that meets the best requirements of performance, the adequate use of resources and power consumption reduction,

Thus, this work aims to study and use techniques for the power consumption reduction, that are present in the literature [COS 02], that can be applied to the H.264/AVC standard codec. The main focus is the use of 4:2 adder compressor circuit in the Forward 4x4 Hadamard Transform, which is contained in the T block on H.264/AVC standard encoder. We have presented some initial results of area and delay obtained in the SIS environment [SEN 92] and power results obtained in the SLS [GEN 89] environment.

2. The T Block

The H.264/AVC is the most recent standard for video compression whose final draft was approved in October 2003 [PUR 04]. Several innovations have been employed in the H.264/AVC in order to achieve an increase in the efficiency of compression with regard to older standards such as H.263 [ITU, 00]. Fig. 1 shows the block diagram of the standard H.264/AVC encoder [AGO 07].

The T block is highlighted in fig. 1. In the T block the following Transforms are processed: i) the FDCT (Forward Discrete Cosine Transform), ii) Forward 2x2 Hadamard Transform and iii) Forward 4x4 Hadamard Transform that is in the scope of this work. The T Block is in the critical path of the intra-frame prediction. This, because the next block of the frame will be encoded only after the current block is totality processed for

H.264/AVC encoder. After this encoding process, the actual block can be used as reference for the next block encoded in the intra-frame prediction.

Thus, an implementation that has reducted delay e low power consumption is very important to improve the performance in T Block and intra-frame prediction.

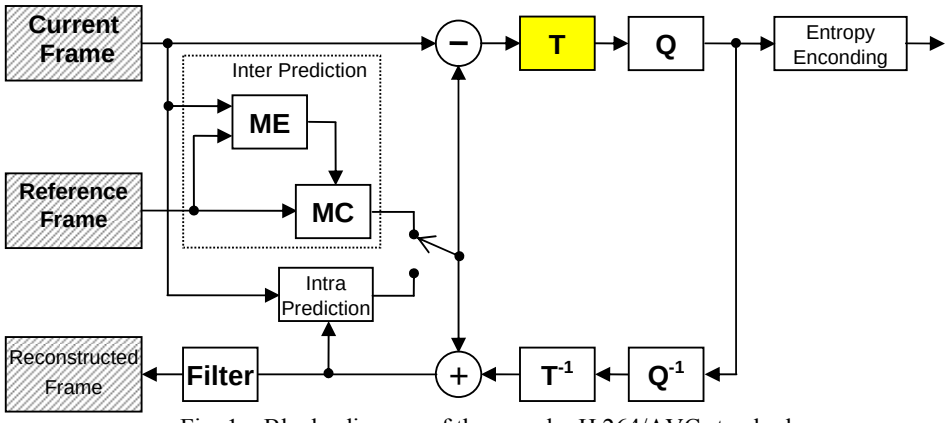


Fig. 1 – Blocks diagram of the encoder H.264/AVC standard.

The transforms has been studded in this work (Forward Hadamard 4x4 Transform) is mentioned as innovation of the H.264/AVC standard.

The Forward 4x4 Hadamard Transform is applied in the residual luma DC blocks that are generated by DCT-2D when 16x16 intra-frame prediction was selected.

The Forward 4x4 Hadamard Transform calculation is shown in equation (1). Note that in the matrix which composes the transform only values 1 and -1 are considered. Thus, only sums and subtractions are needed to the Transform calculation achievement [AGO 07]. The W_D in the equation (1) is the matrix formed by DC elements of the DCT outputs.

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \\ W_D \\ \\ \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2 \qquad (1)$$

3. 4:2 Adder Compressor Circuits

The main goal of this work is the application of efficient adder circuits to perform the calculation of sum and subtraction generated by matrix multiplication of the Hadamard Transform depicted in equation (1). In this context, the compressors circuits appear in the literature as efficient structures for the operation of addition, because these circuits have the ability to add numbers with a minimum spread of carry.

In [WEI 81] a structure of sum of 4-2 numbers was presented, whose scheme is named 4-2 Carry-Save module. In this structure it is possible the sum of four numbers simultaneously. The partial result is given by three terms represented by *Carry*, *Cout* and *Sum*. In the structure of the 4:2 compressors, the inputs and the sum output have the same weight. However, the terms *Cout* and *Carry* present a higher weight. Thus, the final sum result is given by the combination of the partial terms: *S* +2 (*Cout* + *Carry*). As an example, suppose all the 4 inputs and *Cin* equal to 1. In this case, the three terms are equal to 1. But by using the equation shown above, the correct result is pointed as: [1 +2 (1 +1)] = 5.

In this work we have used an improved 4:2 compressor structure proposed by [OKL 96] that uses multiplexers for carry generation (*Carry* and *Cout*), whose structure is shown in fig. 2. This structure has a critical path with a maximum delay of three XOR gates, which represents a gain compared against the previous structure proposed by [WEI 81], which presented four XOR gates in the critical path. Another advantage presented by this modified structure is the fact that the multiplexer circuit can be optimized in the transistor level by the use of transmission gates.

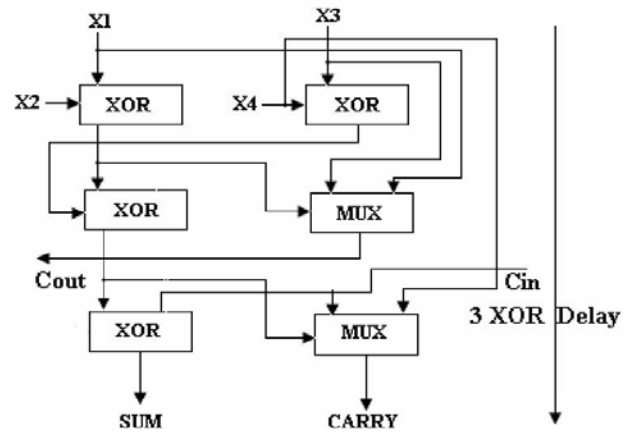


Fig. 2 – 4:2 Compressor structure proposed by [OKL 96].

4. The 4:2 Adder Compressor Circuit in the Forward 4x4 Hadamard Transform Architecture

The architecture proposed for the Forward 4x4 Hadamard Transform is shown in fig. 3. In this figure the 4:2 compressors are represented by 4:2 blocks. On the other hand, the small boxes with no information are responsible for the 2's complement operation which is used when a subtraction of two operands is required. The blocks marked with the sign ">>" are responsible for making the shift to the right in order to perform a division by 2 operation. This architecture was developed from the equation of the Forward 4x4 Hadamard Transform. However, in this work we have implemented the 14 and 16 bit wide 4:2 adder compressor circuit, and we have made a comparison against a block of 3 adders, that is normally used for add four inputs in the Forward 4x4 Hadamard Transform architecture.

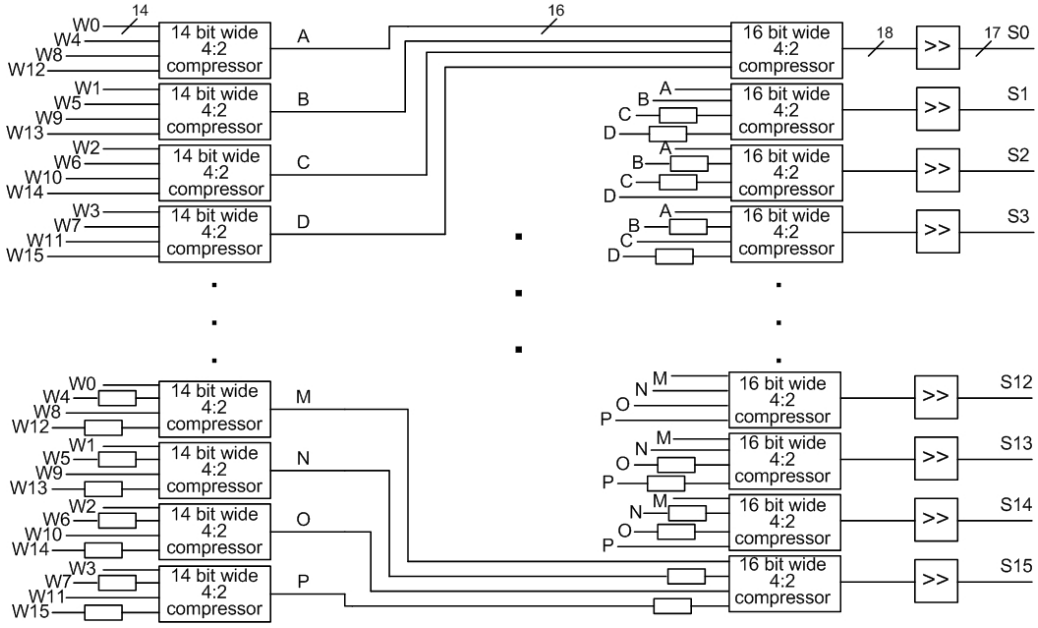


Fig. 3 – Forward 4x4 Hadamard Transform Architecture.

5. Results

In this work we have described four architectures BLIF format (Berkeley Logic Interchange Format) [BER 08], 14 and 16 bit wide 4:2 adder compressor circuit, and 14 and 16 bit wide blocks of 3 adder circuits. The circuits were simulated and validated in the SIS environment to obtain area and delay results. This tool considers 5 Volts voltage and 20MHz clock frequency. Power consumption results were obtained in the SLS environment by using a random input vector with 10.000 points. Tab. 1 shows area, delay and power results. As can be observed in this table, area results are expressed in terms of the number of literals, where each literal is normally equal to 2 transistors.

Tab. 1 – Results obtained with SIS and SLS tools synthesis

Parameters	14 bit wide block of 3 adders (A)	14 bit wide 4:2 compressor (B)	Difference (%) B vs. A	16 bit wide block of 3 adders (C)	16 bit wide 4:2 compressor (D)	Difference (%) D vs. C
Area (literals)	590	524	11.18	674	600	10.97
Delay (ns)	43.59	38.78	11.03	48.70	43.88	9.89
Power (mW)	17.23	9.73	43.52	19.80	11.84	40.20

The results presented by tab. 1 show that the 14 and 16 bit wide 4:2 compressor present less area, delay and power consumption than the block composed by 3 adders. The results are more expressive for the power consumption where 4:2 compressor can reduce more than 40% of this parameter. These results can be explained by the more simple structure presented by the 4:2 compressor with a reduced critical path. These results show that the use of 4:2 compressor can enable large gains in the performance of the Forward 4x4 Hadamard Transform, since this type of architecture use a large amount of these blocks.

6. Conclusions and Future Work

This paper presented an initial study on the use of 4:2 adder compressor circuits as a way of reducing power consumption and increase performance of the Forward 4x4 Hadamard Transform. The 14 and 16 bit wide were implemented and compared against blocks of 3 adders. The obtained results showed that the 4:2 compressor is more efficient, because this circuit can reduce area, delay and power values. This is an important issue, because this type of structure is widely used in the 4x4 Forward Hadamard Transform. As future work, we intended to implement all the 4x4 Forward Hadamard Transform and observe the effect of the use of 4:2 compressor in this architecture in terms of area, performance and power consumption.

7. References

[AGO 07] AGOSTINI, Luciano. **Desenvolvimento de arquiteturas de alto desempenho dedicadas à compressão de vídeo segundo o padrão H.264/AVC**. Tese de Doutorado. Porto Alegre : UFRGS, 2007.

[BER 08] `http://www-cad.eecs.berkeley.edu/~pchong/sis.html. Accessed in February, 2008.`

[COS 02] COSTA, Eduardo A. C. da. **Operadores Aritméticos de Baixo Consumo para Arquiteturas de Circuitos DSP**. Tese de Doutorado. Porto Alegre. UFRGS, 2002.

[GEN 89] GENDEREN, A. SLS: an efficient switch-level timing simulator using min-max voltage waveforms. International conference on very large scale integration, VLSI, 1989. New York: ACM, 1989.

[ITU 00] ITU-T. **Recommendation H.263: Video Coding for Low Bit Rate Communication,** 2000.

[OKL 96] OKLOBDZIJA, Vojin. G.; VILLEGER, D.; LIU, S. S. **A method for speed optimized partial product reduction and generation of fast parallel multipliers using and algorithmic approach**. IEEE Transaction on Computers. 1996.

[PUR 04] PURI, Atul; CHEN, Xuemin; LUTHRA, Ajay. **Video Coding Using the H.264/MPEG-4 AVC Compression Standard**. Elsevier B. V 2004.

[SEN 92] SENTOVICH, E. et al. **SIS: a system for sequential circuit synthesis**. Berkeley: University of California, 1992.

[WEI 81] WEINBERGER, A. **4:2 carry-save adder module**. IBM Technical Disclosure Bulletin. 1981.

[WIE 03] WIEGAND, Thomas; SULLIVAN, Gary J.; BJONTEGAARD, Gisle; LUTHRA, Ajay. **Overview of the H.264/AVC video coding standard** IEEE Transactions on Circuits and Systems for Video Technology, 2003.

Full-Custom 6T SRAM Design for Motion Compensation Module in H.264/AVC Video Decoder

[1]Guilherme Mauch de Freitas, [1]Cláudio Machado Diniz, [1]Bruno Zatt,
[2]Altamiro Amadeu Susin, [1]Sergio Bampi
{gmfreitas,cmdiniz,bzatt,bampi}@inf.ufrgs.br, altamiro.susin@ufrgs.br

[1]UFRGS – Informatics Institute (II)
[2]UFRGS – Electric Engineering Departament (DELET)

Abstract

This work proposes a full-custom SRAM memory design for the motion compensation module in H.264/AVC decoder. It was designed to occupy minimal area and to reach the performance to operate at 100 MHz. The complete memory was described in full-custom layout for TSMC 0.18 µm technology using Virtuoso (Cadence). Results showed that memory can work at 100 MHz and its final layout has an area of 2.7mm^2.

1. Introduction

Motion compensation is the module with highest complexity in an H.264/AVC video decoder. The video coding group from Federal University of Rio Grande do Sul proposed the motion compensation hardware architecture (MoCHA) [AZE 07] and its hardware description was synthesized for TSMC 0.18 µm *standard-cells* technology using Encounter (Cadence). MoCHA uses 20kB of SRAM memory to store neighbor macroblock information for inter prediction. Then, for an ASIC design of this module, a full-custom SRAM memory design is needed to implement this memory, obtaining high performance and area reducing.

The design is developed by the following methodology: i) Memory architecture definition; ii) Development of SRAM modules and electric simulations; iii) Layout design of the modules; iv) Memory integration. The design is implemented on TSMC 0.18 µm technology, using Virtuoso (Cadence). This paper is organized as follows. Section 2 presents the memory architecture definition. Sections 3, 4 and 5 show the design for SRAM cell, row decoder and sense amplifier, respectively. Section 6 presents some area and simulation results. Section 7 concludes the paper.

2. Memory architecture

The 20 kB SRAM memory is formed by four 5 kB blocks and a diagram of one 5 kB block is shown in Fig. 1. Each 5 kB block is organized into 512 words of 80 bits, divided in two 2.5 kB banks. Each bank is composed by 40x512 memory cells and 40 sense amplifiers, for reading the output word. Each column is associated to one sense amplifier The row decoder selects a word to be read or write, corresponding to the input address. It is placed between the two memory banks for reducing the word line delay.

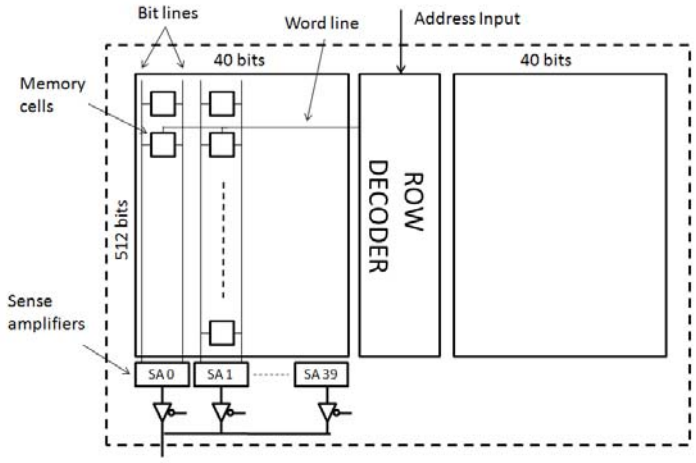


Fig. 1 – Memory block diagram.

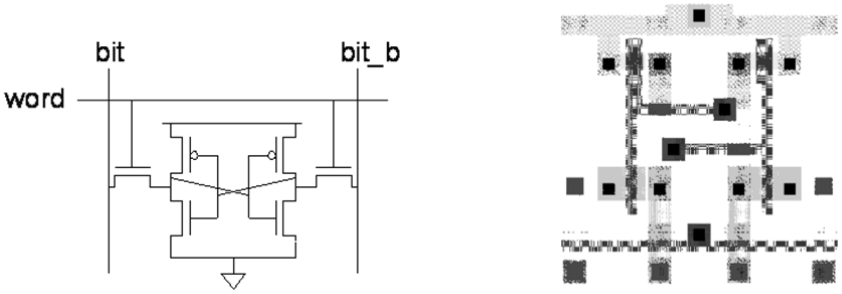
3. SRAM Memory Cell

The memory cell is responsible for most of the memory area, because of its high occurrence in an SRAM memory. By reducing its area, the whole memory area can be reduced drastically. In this design, each 1-bit

SRAM memory cell was formed by a set of two cross-coupled inverters and two pass transistors (known as 6T cell), as shown in Fig. 2a. We choose this memory cell because it is compact and shows a good performance.

The function of the cross-coupled inverters in Fig. 2a is to store a bit. Both pass transistors are connected to the bit-lines and word line and its function is to access the inverters through the bit-lines during a reading or writing operation. The writing operation is simple in this memory cell: pass transistors are selected by the row decoder through the word line and the value to be written is forced into bit-lines. The non destructive reading operation is more complicated because it needs a sense amplifier to obtain the value by the bit-lines and do not flip the value stored by the cross-coupled inverters. The sense amplifier operation is detailed in section 5.

The operation of this memory cell depends on adequate transistor dimensions. NMOS transistor in the inverter should be stronger than pass transistor to avoid bit flip during a reading operation. A weak PMOS transistor in the inverter guarantees a writing consistency [WES 04]. The transistor dimensions for PMOS and NMOS, in the inverters, are 0.27x0.27 μm and 0.18x0.54 μm, respectively. The dimension for both NMOS pass transistors is 0.18x0.36 μm. Before the reading operation, bit-lines are pre-charged and equalized, to guarantee a correct sense amplifier operation. The memory cell is implemented in a TSMC 0.18 μm technology using Virtuoso and the memory cell layout is shown in Fig. 2b. This layout was designed to allow repeatability and integration with row decoder and sense amplifier modules. The memory cell area is 23.16 μm^2.



(a) (b)

Fig. 2 – (a) Memory Cell [WES 04]; (b) Memory Cell Layout.

4. Row Decoder

The row decoder selects one word of memory to be read or write. It receives the input address and selects all pass transistors from the corresponding line of memory cells. This memory has 512 words, so the address is represented by 9 bits and the row decoder is a 9:512 decoder. It is formed by three stages of 3:8 NOR-based pre-charged decoders, arranged in a domino-logic style. In this logic style, all address lines are pre-charged. When the address is selected, only the word line selected is pulled down. It is very power effective on the address selection, because only one line changes its value. The complete decoder is formed by three stages composed by one, eight and sixty four 3:8 decoders at each stage. The decoder layout was designed and Fig. 4 shows a 3:8 decoder. Metal lines in the horizontal are part of the input address and the metal lines at the top of the figure are the output selection lines for the 3:8 decoder.

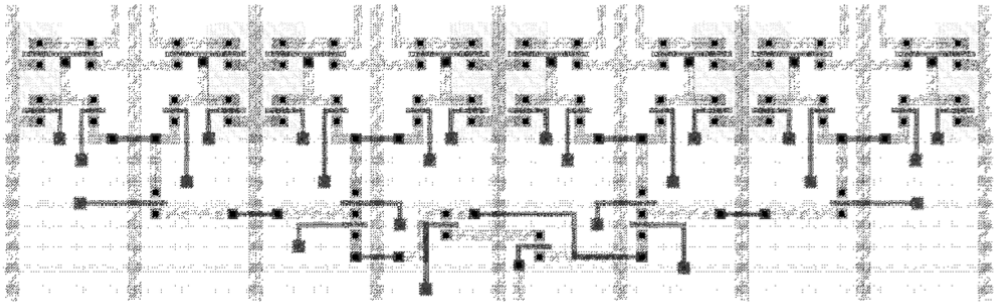


Fig. 4 – Row Decoder Layout.

5. Sense Amplifier

Sense amplifier, also known as differential amplifier, is a determinant block for memory performance and power consuming. A worse design of this module may cause a bit-flip on the memory cell. Because it is responsible for a high delay on the reading operation, its performance is important to ensure low delay reading. Due to design specification, a pure analog sense amplifier was designed, formed by a NMOS differential pair, a PMOS current mirror, a current supply and a driving transistor, as shown in Fig. 5. A tri-state buffer is connected to the sense amplifier output for connecting it to the output bus.

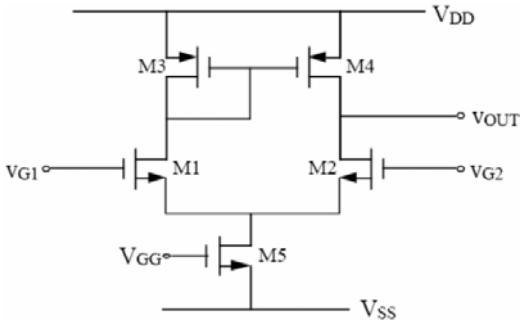


Fig. 5 – Sense Amplifier Schematic.

The main characteristics of differential amplifier are: high differential gain (A_{VD}); low common mode gain (A_{VC}); ease of cascading without need coupling capacitors. Its objective is to amplify the differential signals and to eliminate or minimize the common mode voltage on the input signal. CMOS process parameters,

obtained by the wafer measurements, are used for the dimensioning. The equations (1) to (6) show the calculations to determine the transistor dimensioning for the sense amplifier.

$$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} = \frac{3{,}9.8{,}854x10^{-14}}{4{,}5x10^{-7}} = 7.67347fF/\mu m^2 \quad (1)$$

$$k_n = \frac{(\mu_{0n}.C_{ox})}{2} = \frac{273{,}85.767{,}35n}{2} = 105.07\mu A/V^2 \quad (2)$$

$$k_p = \frac{(\mu_{0p}.C_{ox})}{2} = \frac{121{,}47.767{,}35n}{2} = 46.61\mu A/V^2 \quad (3)$$

$$Cl = 7{,}67.0{,}18.18 + 7{,}67.0{,}18.(18/1{,}5) = 41.44fF \quad (4)$$

$$Slew-rate(SR) = 100V/\mu s \qquad I_{BIAS} \geq Cl.SR = 100x10^{-15}.100x10^{6} \geq 10\mu A \quad \rightarrow \quad I_{BIAS} = 50\mu A \quad (5)$$

$$A_{VD} = 100V/V \qquad g_m = \frac{2(I/2)}{V_{GS} - V_t} = \frac{50\mu}{1{,}8 - 0{,}45} = 37\mu A/V \quad (6)$$

Considering the differential amplifier works on the interval of 0V (gnd) to 1,8V (Vdd) and the current supply provides 50µA (25µA for each side), we get a *W/L* ratio of 2.35 (*L* = 0.36 µm, *W* = 0.9 µm) for the differential pair, and a *W/L* ratio of 6.38 (*L* = 0.36 µm, *W* = 2.34 µm) for the current mirror.

In the layout of this sense amplifier were used techniques targeted to analog circuits design, which are: common centroid, interdigitize and orientation. Here, we also used a technique to protect the circuit against faults caused by collision of cosmic particles, know as guard rings. The layout of the sense amplifier is shown, with details, in Fig. 6.

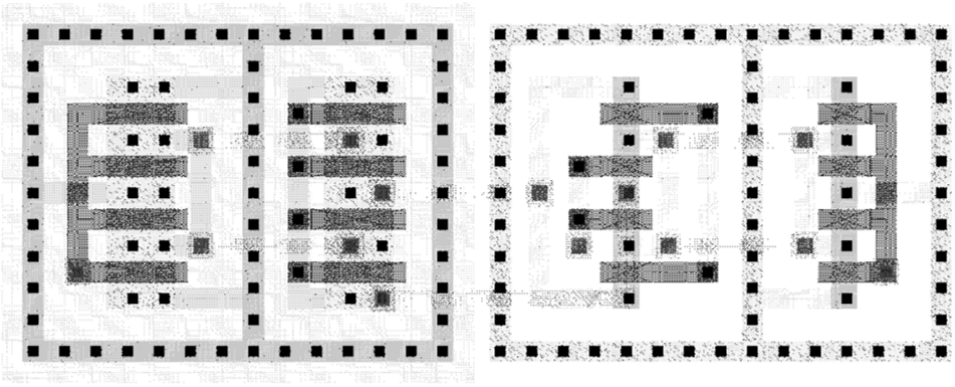


Fig. 6 – Sense Amplifier Layout.

6. Results

An electric simulation was realized to ensure correct operation of the memory, using Spectre simulator. Fig. 7 presents a writing and a reading operation of a 0 bit. The first two signals show the voltage of both bit-lines (BLneg and BL). The third signal is the sense amplifier output. The fourth and fifth signals are the pre-charge input (active low) and word-line selection, respectively. The last two signals show the memory cell internal signals. Reading and writing operation could be performed in 10 ns, so this memory can work at 100 MHz. The complete memory layout occupies an area of 2.7 mm^2.

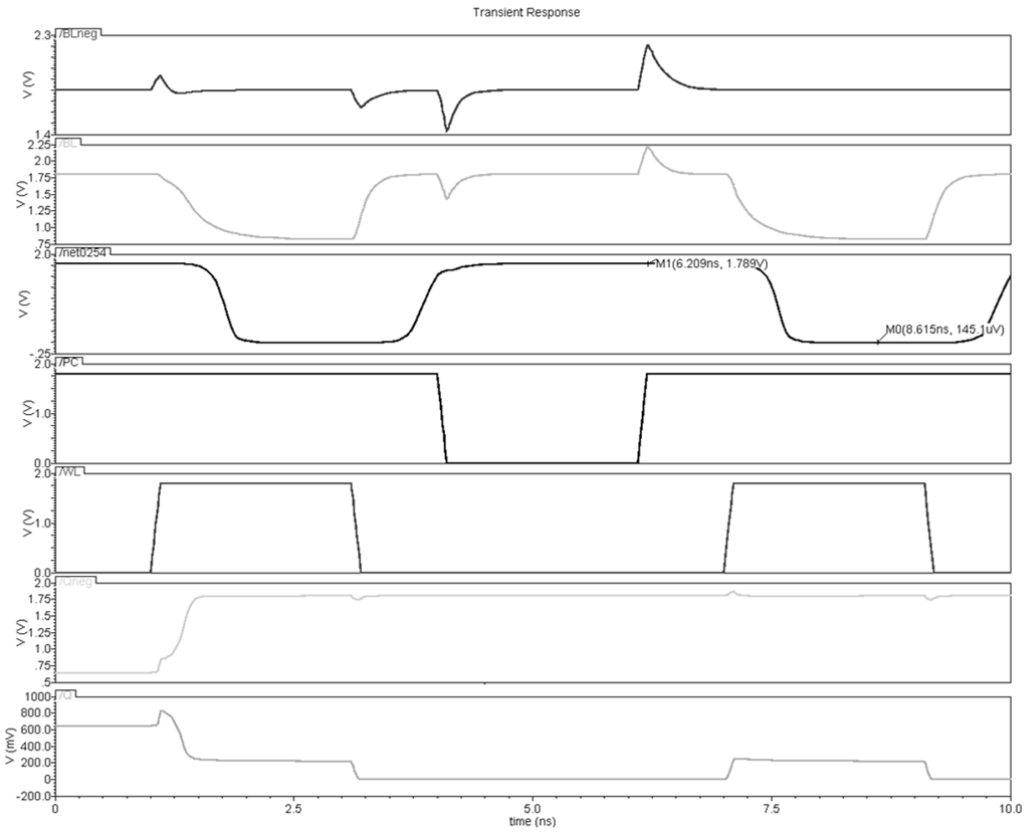


Fig. 7 – Memory Electric Simulation.

7. Conclusions

This work proposed a full-custom SRAM memory design for the motion compensation module in H.264/AVC decoder. The complete memory was described in full-custom layout for TSMC 0.18 μm technology and its final layout occupies an area of 2.7 mm^2. Simulation results showed good performance at 100 MHz, with a reading and writing operation in only one cycle.

8. References

[AZE 07] AZEVEDO, A.; ZATT, B.; AGOSTINI, L.; BAMPI, S. MoCHA: a Bi-Predictive Motion Compensation Hardware for H.264/AVC Decoder Targeting HDTV. In: IEEE International Symposium on Circuits and Systems (Mar. 27-30 : New Orleans, LA). Proceedings. IEEE, 2001.

[CON 05] CONRAD JR, E. Memória Cache – da Estimativa Cacti a Implementação Física. Projeto de Dilomação – Departamento de Engenharia Elétrica, UFRGS, Porto Alegre, 2005.

[GIR 03] GIRARDI, A. G. Uma Ferramenta para Automação da Geração do Leiaute de Circuitos Analógicos sobre uma Matriz de Transistores MOS Pré-Difundidos. Dissertação de Mestrado – PPGC, UFRGS, Porto Alegre, 2005.

[HAS 01] HASTINGS, Alan. The Art of Analog Layout. New Jersey: Prentice-Hall, 2001.

[MOS 07] MOSIS. Transistor models from the last fabrication. Available at www.mosis.org.

[SED 98] SEDRA, A. S.; SMITH, K. C. Microelectronic Circuits. 4th Ed. Oxford University Press, 1998.

[WES 05] WESTE, Neil H. E.; HARRIS, D. F. CMOS VLSI Design: A Circuits and Systems Perspective. Boston: Pearson/Addison Wesley, 2005.

Motion Estimation Algorithms Evaluation Targeting Hardware Design of Video Coders

[2]Leandro Rosa, [1]Fabiane Rediess, [1]Rafael Petry,
[1]Luciano Agostini, [2]Sergio Bampi
{frediess_ifm, rpetry_ifm, agostini}@ufpel.edu.br,
{leandrozanetti.rosa, bampi}@inf.ufrgs.br

[1]Grupo de Arquiteturas e Circuitos Integrados (GACI) – DINFO – UFPel
[2]Grupo de Microeletrônica (GME) – II – UFRGS

Abstract

This work presents an algorithmic investigation of motion estimation (ME) in video compression. This analysis is a solid evaluation of different criteria about ME algorithms, targeting the choice of the best algorithm to be designed in hardware. This choice has a direct impact in the motion vector quality and in the motion estimator performance. The implemented algorithms were: Full Search, One at a Time Search, Diamond Search, Tree Step Search, Hexagon-Based Search and Dual Cross Search and two subsample techniques (Pel Subsampling and Block Subsampling) that were associated with these algorithm. All the algorithms were developed in C language and use SAD as distortion criterion. For each algorithm, three versions were implemented, one for each block size, 4x4, 8x8 and 16x16 pixels, and were evaluated with four different search area sizes: 46x46, 80x80, 144x144 and 208x208 pixels. The algorithms were applied to ten video sequences and the average results were used to evaluate the algorithms performance.

1. Introduction

Nowadays, the compression of digital videos is a very important task. The industry has a very high interest in digital video codecs, once digital videos are present in many current applications, such as cell-phones, digital television, DVD players, digital cameras and a lot of other applications. This important position of video coding in current technology development has boosted the creation of diverse standards for video coding. Without the use of video coding, the processing of digital videos is almost impossible, due to the very high amount of resources that are necessary to store and to transmit these videos. Many video compression standards were developed to reduce this problem. The currently most used standard is MPEG-2 [ITU 94] and the latest and more efficient standard is H.264/AVC [JVT 03]. These standards group a set of algorithms and techniques that can reduce drastically the amount of data that is necessary to represent digital videos.

A modern video coder is formed by eight main operations, as shown in fig. 1: motion estimation, motion compensation, intra-frame prediction, forward and inverse transforms (T and T^{-1}), forward and inverse quantization (Q and Q^{-1}) and entropy coding. This work focuses in the motion estimation, which is highlighted in fig. 1.

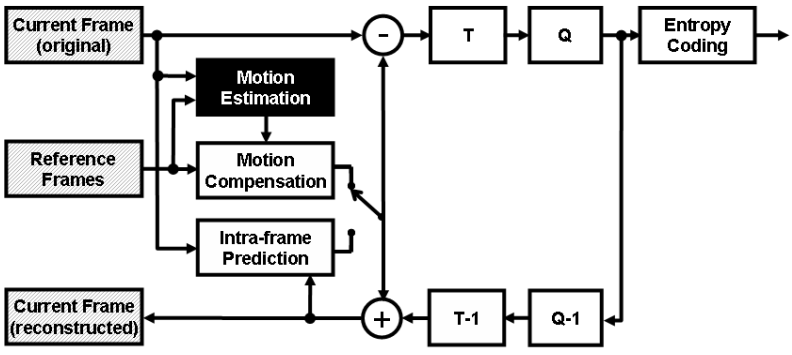


Fig. 1 - Block diagram of modern video coder

This paper presents an algorithmic investigation targeting motion estimation. These algorithms used the Sum of Absolute Differences (SAD) [KUH 99] as distortion criterion and were evaluated with four different search area sizes, 46x46, 80x80, 144x144 and 208x208 samples and three blocks size, 16x16, 8x8 and 4x4 samples.

This paper is organized as follow. Section 2 of this paper presents some basic concepts about motion estimation. Section 3 presents the search algorithms. Section 4 presents the results and section 5 presents the conclusions.

2. Motion Estimation

Motion estimation (ME) operation tries to reduce the temporal redundancy between neighboring frames [KUH 99]. One or more frames which were already processed are used as reference frames. The current frame and the reference frame are divided in blocks to allow the motion estimation. The idea is to substitute each block of the current frame by one block of the reference frame, reducing the temporal redundancy. The best similarity between each block of the current frame and the blocks of the reference frame is selected. This selection is done through a determinate search algorithm and the similarity is defined through some distortion criterion [KUH 99]. This search is restricted to a specific area in the reference frame, called search area. When the best similarity is found, then a motion vector (MV) is generated to indicate the position of this block inside the reference frame. These steps are repeated for all blocks of the current frame.

Motion compensation operation reconstructs the current frame through the reference frames and the motion vectors generated by the motion estimation. The difference between the original and the reconstructed frame (called residue) is sent to the transforms and quantization calculation.

3. Investigated ME Algorithms

The search algorithm defines how the search for the best matching will be done in the search area. The search algorithm choice has a direct impact in the motion vector quality and in the motion estimator performance.

There are a lot of algorithms to define the search method, such as: Full Search [BHA 99], One at a Time Search [RIC 02], Diamond Search [KUH 99] and Hexagon-Based Search [ZHU 02]. Just the Full Search algorithm presents the optimal results in terms of best matching. All the others are fast algorithms, which were designed to reduce the computational complexity of the motion estimation. These algorithms produce sub-optimal results of matching, since not all positions are compared.

3.1 Full Search: The Full Search (FS) algorithm searches for the best match testing every possible position within the search area. For each test match, it calculates the error for each pixel of the block. It will always find the best results.

3.2 One at a Time Search: The One at a Time Search (OT) algorithm searches for the best horizontal match and then for the best vertical match. It starts by testing the central position of the search area and then two other positions, one pixel to the left and one pixel to the right. If the central value is the lowest, the algorithm proceeds to the vertical search. Otherwise, the center is redefined as the block with the lowest SAD and the process is repeated.

3.3 Diamond Search: The Diamond Search (DS) algorithm uses two diamond-shaped patterns, one for the first steps and the other for the final refinement. At the first step it compares nine positions, and while the central position is not the one with the lowest SAD, a new center is defined and the process is repeated. After finding the center with the lowest SAD, a refinement is done by testing four new positions forming the second diamond. The position which has the lowest SAD is chosen.

3.4 Dual Cross Search: The Dual Cross Search (DCS) uses two cross-shaped search patterns and a final refinement step. First, five values are compared, being them the center and four positions next to it by a one pixel distance. While the center is not the position with the lowest SAD, a new center is defined and new comparisons are made using another cross-shaped pattern. After the best center is found, the result is refined by testing three positions between the center and the side which has the lowest value.

3.5 Tree Step Search: The Tree Step Search (TSS) algorithm uses an S variable that is set to 2^{n-1} for a search area range of +/- (2^n-1) pixels. The first step is to test the center and eight positions around it by a distance of +/-S pixels. The position with the lowest SAD is chosen as the new center. S is then divided by two and on the second step, eight new positions distant by +/-S pixels from the center are tested. A new center is then chosen and on the third step, S is divided by two again, now having value 1. Eight more positions are tested and the one with the lowest SAD is chosen as the best match.

3.6 Hexagon-Based Search: The Hexagon-Based Search (HS) algorithm is very similar to the DS algorithm, using two hexagon-shaped patterns. Is starts by comparing the center with six other positions around it, and while the center is not the one with the lowest SAD, a new center is chosen and the execution continues. When a good match is found, a final refinement is done by testing other four positions forming the second hexagon. The best match is then chosen.

3.7 Subsampling Techniques: The investigation considered also two subsampling techniques that were associated with the algorithms. The subsampling at the block level (Block Subsampling - Bck) and the subsampling at the pixel level (Pel Subsampling - Pel). The **Pel Subsampling** technique calculates the error of the block using only a few pixels of the block and simply discarding the others. In this work, five versions have been implemented for each investigated algorithm: 2:1, 4:1, 8:1 (two versions) and 16:1. In the **Block Subsampling** the blocks are subsampled and the error is calculated for a few blocks and the others are discarded. Versions using Bck 2:1 and 4:1 were implemented for every variation of FS, resulting in 10 different versions.

4. Results

The search algorithms were developed in C language [KER 99] and several analyses were developed targeting the algorithms evaluation in accordance with some different metrics. The first 100 frames of 10 different videos were used as input for this comparison. These 10 input videos are not compressed and they have a SDTV (720 x 480 pixels) resolution. The software analysis was developed considering the Sum of Absolute Differences (SAD) as distortion criterion. This defines how the differences between the regions are evaluated. Many distortion criteria were developed [KUH 99], however the most used for hardware design is SAD. Equation (1) shows the SAD criterion, where **SAD(x,y)** is the SAD value for **(x,y)** position, **R** represents the reference sample, **P** is the search area sample and **N** is the block size.

$$SAD(x,y) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|R_{i,j} - P_{i+x,j+y}\right| \qquad (1)$$

The evaluations presented in this paper were divided in five criteria: PSNR, processing time, quality of the generated vectors, number of SAD calculations and percentage of total error reduction. For fast algorithms three more criteria were evaluated: percentage of conclusion in first step, iterations average and worst case. However this paper shows only some results because of the reduced space. The full results are in [ROS 07].

Tab. 1 presents the results of error reduction when compared to the total error for some algorithms with all investigated search areas. Total error is the SAD generated without any motion estimation.

Tab. 1 - Percentage of error reduction compared to total error

	FS 16x16	FS 4x4	FS+Pel2:1 16x16	FS+Pel2:1 4x4	DS 16x16	DS 4x4	DS+Pel2:1 16x16	DS+Pel2:1 4x4	HS 16x16	HS 4x4
46x46	51.80	70.53	51.52	65.91	47.15	56.31	46.74	53.03	45.72	54.60
80x80	55.70	72.96	55.40	67.70	48.97	56.47	48.52	53.13	47.38	54.75
144x144	56.69	74.51	56.38	68.61	49.15	56.48	48.69	53.14	47.55	54.76
208x208	57.04	75.29	56.73	69.00	49.17	56.48	48.71	53.14	47.57	54.76

The FS based algorithms have the best results and it is possible to notice that the results improve when the search area is increased. On the other hand, the fast algorithms (DS and HS) do not improve the result when the search area is increased. This is because they might stop in a local minimum.

Another important information in ME estimation algorithms is the number of operations, in this case the number of SAD calculations. The number of SAD calculations is extremely important to evaluate the algorithm complexity when targeting a hardware implementation. Tab. 2 presents the results of number of SAD calculations (in billions of operations) for the same algorithms shown in the previous table. The FS based algorithms increase significantly the number of operation as the search area is increased, while the fast algorithms do not have a significant increase.

Tab. 2 - Number of SAD calculations (in billions of operations)

	FS 16x16	FS 4x4	FS+Pel2:1 16x16	FS+Pel2:1 4x4	DS 16x16	DS 4x4	DS+Pel2:1 16x16	DS+Pel2:1 4x4	HS 16x16	HS 4x4
46x46	33.21	63.90	16.60	31.95	0.82	0.74	0.41	0.37	0.60	0.57
80x80	146.02	204.91	73.01	102.46	0.86	0.74	0.43	0.37	0.63	0.57
144x144	575.11	687.09	287.55	343.55	0.87	0.74	0.43	0.37	0.63	0.57
208x208	1287.32	1452.38	643.66	726.19	0.87	0.74	0.43	0.37	0.63	0.57

Comparing tab. 1 and tab. 2, it is possible to understand that the FS based algorithms reach the best results with the search area increase and the block size decrease. However, this represents a high number of SAD operations, which makes it difficult to implement in hardware, specifically with time or area limitations.

The FS 16x16 algorithm reduced the error 4.65% more than DS 16x16, but the FS 16x16 makes 40 times more calculations that DS 16x16 for a search area with 46x46 samples. This comparison shows that the fast algorithms can accelerate the motion estimation process without a significant negative effect in the result.

Tab. 3 shows the PNSR results for the search area with 208x208 samples for the 3 block sizes and for the 11 investigated algorithms.

Tab. 3 - PSNR of algorithms (in dB) for 208x208 samples and three blocks sizes

	FS	FS + Pel2:1	FS + Pel4:1	DS	DS + Pel2:1	HS	HS + Pel2:1	DCS	DCS + Pel2:1	FS + Bck2:1 + Pel2:1	FS + Bck4:1 + Pel2:1
16x16	28.99	28.94	27.60	27.28	27.21	26.98	26.91	26.32	26.21	25.08	21.57
8x8	30.90	30.51	27.50	27.76	27.53	27.43	27.08	26.81	26.47	26.68	23.19
4x4	34.25	31.93	25.20	28.50	27.85	28.15	27.18	27.44	26.57	28.50	25.17

Again the FS followed by the FS+Pel2:1 presented the best results. Among the fast algorithms, both DS versions presented the best results. Tab. 3 also shows that block subsampling has the worst results, losing to the fast algorithms.

Tab. 4 presents the average results of number of iterations of the fast algorithms in the second pass, excepting for TSS, which has a fixed number of iterations: three.

Tab. 4 - Average number of iterations for the fast algorithms

	16x16				8x8				4x4			
	DS	**HS**	**DCS**	**OT**	**DS**	**HS**	**DCS**	**OT**	**DS**	**HS**	**DCS**	**OT**
46x46	2.83	2.67	2.65	4.83	2.58	2.43	2.50	4.34	2.15	2.04	2.18	3.43
80x80	3.18	2.99	2.90	5.57	2.66	2.50	2.57	4.55	2.17	2.05	2.19	3.47
144x144	3.23	3.04	2.94	5.76	2.67	2.51	2.57	4.59	2.17	2.05	2.20	3.47
208x208	3.90	3.04	2.94	6.22	2.67	2.51	2.57	4.59	2.17	2.05	2.20	3.47

The average iteration number is very close between the fast algorithms, and much more when it comes to 4x4 block size. The increase of the search area size did not result in a significant increase in the number of iterations. It reinforces the idea that the fast algorithms stop at local minimum with a suboptimal vector.

Comparing these data with the worst cases where the algorithms did up to 185 iterations, it is clear that the limitation in the number of iterations is very important, and this is one of our planed future works.

5. Conclusions

This paper presented an algorithmic evaluation of motion estimation. These algorithms consider three blocks sizes: 16x16, 8x8 and 4x4 samples, and four different search area sizes: 46x46, 80x80, 144x144 and 208x208 samples. These search algorithms were developed in software targeting an evaluation for a hardware implementation.

Considering the results, the most interesting algorithms to be designed in hardware were FS, FS+Pel2:1, DS and DS+Pel2:1. The FS based algorithms presents the best quality results and are easy to be hardware designed, in function of the regularity of the data consumption and because it allows a freely exploration of the parallelism level, since there are no data dependencies in this algorithm. But the FS based solutions are very intensive in hardware consumption. The DS based algorithms presented the best quality among the fast (not optimal) algorithms, with little quality losses in comparison with the FS algorithms. The DS algorithms uses a very low number of SAD calculations when compared with the FS versions, then, this implies in an important reduction in the hardware resources necessities when compared with FS for a same processing rate. In the other hand, DS algorithms present some data dependencies, causing some difficulties in the parallelism exploration.

6. References

[BHA 99] BHASKARAN, V.; KONSTANTINIDES, K. **Image and Video Compression Standards: Algorithms and Architectures.** 2. ed. Massachusetts: Kluwer Academic Publisher, 1999. 454p.

[ITU 94] INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.262 (11/94): generic coding of moving pictures and associated audio information – part 2: video**. [S.l.], 1994.

[JVT 03] Joint Video Team of ITU-T and ISO/IEC JTC 1. **Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 or ISO/IEC 14496-10 AVC)**, 2003.

[KER 99] KERNIGHAN, Brian W.; RITCHIE, Dennis M. **C: a Linguagem de Programação Padrão Ansi.** Rio de Janeiro: Campus, 1999.

[KUH 99] KUHN, Peter M.; **Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation.** Boston: Kluwer Academic Publishers, 1999.

[RIC 02] RICHARDSON, Iain E. G.; **Video Codec Design – Developing Image and Video Compression Systems.** Chichester: John Wiley and Sons, 2002.

[ROS 07] ROSA, Leandro Z. P.. **Investigação sobre Algoritmos para a Estimação de Movimento na Compressão de Vídeos Digitais de Alta Definição: Uma Análise Quantitativa**. Available in: <www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2007/mono_leandro_rosa.pdf >

[ZHU 02] ZHU, C; LIN, X; CHAU, L. **Hexagon-based Search pattern for fast Block motion estimation. In: IEEE Transactions on Circuits and Systems for Video Technology**, Volume 12, Issue 5, May 2002 Page(s):349 – 355.

Architectural Design for Forward and Inverse 4x4 Transforms of H.264/AVC Standard Focusing in the Intra Frame Coder

[1]Fabiane Rediess, [1]Felipe Sampaio, [1]Carolina Fonseca,
[2]Sergio Bampi, [2]Altamiro Susin, [1]Luciano Agostini
{frediess_ifm, fsampaio.ifm, cfonseca_ifm, agostini}@ufpel.edu.br,
bampi@inf.ufrgs.br, altamiro.susin@ufrgs.br

[1]Universidade Federal de Pelotas
Grupo de Arquiteturas e Circuitos Integrados - GACI
[2]Universidade Federal do Rio Grande do Sul
Grupo de Microeletrônica - GME

Abstract

This work presents the design and synthesis of high throughput and low latency architectures for the forward and inverse 4x4 transforms of the H.264/AVC standard. Four architectures were designed: forward and inverse DCT and forward and inverse 4x4 Hadamard. The architectures were designed focusing in the intra coding module necessities, which require a very high throughput and a low latency in the transforms and quantization modules. The architectures were described in VHDL and synthesized to some Altera FPGAs. The best result was found in the DCT mapped to Stratix II device, with 332 MHz, reaching a processing rate of 450 QHDTV frames per second.

1. Introduction

The high use of digital videos in current products (DVD players, cell phones and digital TV) is the main motivation for the high interest of industry and academy in video compression. It happened because the compression allows a dramatically minimization in the use of transmission and storage resources.

A several standards of video compression were defined in the past years: MPEG-2, H.263 and, the latest one, H.264/AVC (as know as MPEG-4 part 10) [JVT 03].

The H.264/AVC standard achieves significant improvements over the previous standards. It was defined intending to double the compression rates reached by the older standards. Although, to achieve this goal the codec computational complexity was increased in four times in relation to the MPEG-2 codec [RIC 03]. For this reason, codecs designed in software are not used when high resolution videos are processed and real time is needed.

In fig. 1 is presented the block diagram of the H.264/AVC encoder. Its main modules are the Inter Frame Prediction (composed by Motion Estimation and Motion Compensation), the Intra Frame Prediction, the Inverse and Forward Quantization (Q and Q^{-1}), Entropy Coding, Deblocking Filter and Forward and Inverse Transforms (T and T^{-1}). This work focused on the forward and inverse transform modules, which are highlighted in fig. 1.

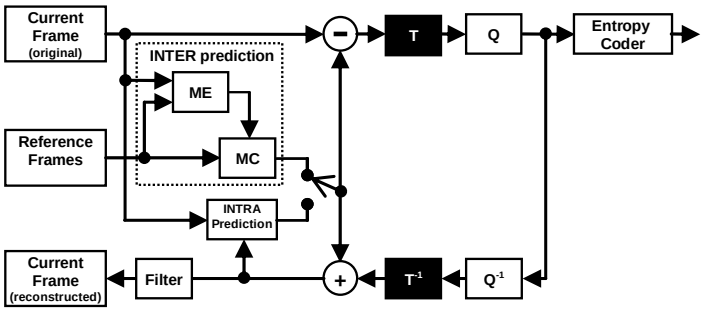


Fig. 1 – H.264/AVC coder block diagram.

This paper focuses in the forward and inverse transforms (T and T^{-1} in fig. 1) modules, purposing high throughput and low latency architectures for the 4x4 transforms defined in the H.264/AVC standard. These architectures were designed focusing in the intra prediction module.

The main importance of the high throughput and the low latency in the transforms modules is function of the intra frame prediction operation. The intra frame prediction works with macroblocks of the I type. The codification of I type macroblocks does not depend on the previously processed frames and the reference values to the intra prediction process are the previously coded macroblocks in the same frame. But these values are available only after the macroblock reconstruction and this means that the reference macroblock must firstly

pass by forward and inverse transforms and forward and inverse quantization operations (Q and Q^{-1} in fig. 1) to be ready to be used for the current macroblock processing. Then, a solution which reaches a high throughput with a low latency in Q, Q^{-1}, T and T^{-1} modules is required to allow real time in the intra frame prediction module. The 4x4 forward and inverse transforms architectures presented in this paper contributes with this goal with a full parallel solution and a pipeline designed to allow a best relation between operation frequency and latency.

2. Forward and Inverse Transform Modules

The forward (T) and inverse (T^{-1}) modules in the H.264/AVC standard are composed by three transforms: 4x4 2-D DCT, 4x4 2-D Hadamard and 2x2 2-D Hadamard. The T module uses forward DCT, forward 4x4 2-D Hadamard and 2x2 2-D Hadamard, while the T^{-1} module is formed by inverse DCT, inverse 4x4 Hadamard and 2x2 2-D Hadamard. The forward and inverse 2x2 Hadamard are identical and this transform is used in both T and T^{-1} modules.

All H.264/AVC transforms use just integer arithmetic to avoid the mismatch between forward and inverse transforms between different coders and decoders and to allow efficient hardware implementations.

In this work, were designed high throughput and low latency architectures for all the 4x4 transforms: the forward transforms (forward DCT and 4x4 forward Hadamard) and the inverse transforms (inverse DCT and 4x4 2-D inverse Hadamard).

Equation (1) shows the 2-D forward DCT defined by the H.264/AVC standard, where, **X** is the 4x4 input residual block and **Y** the 2-D forward DCT outputs. In Equation (2) it is shown the forward 4x4 Hadamard, where $\mathbf{W_D}$ is the Hadamard input and the **Y** is its output. The inverse DCT and inverse 4x4 Hadamard are similar and to the forward versions and they are not presented in this paper. These inverse transforms like the forward ones are presented in the H.264/AVC standard [JVT 03].

$$Y = C_f X C_f^T \otimes E_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} & X & \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 2 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix} \quad (1)$$

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} & W_D & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2 \quad (2)$$

3. Designed Architectures

This work presents the design of fully parallel architectures for the forward and inverse DCT and for the forward and inverse 4x4 Hadamard transform. The architectures were designed intending to reach a good balancing between a very high processing rate and a low latency, since these characteristics are very important to allow the intra frame coder to reach real time when high definition videos are being processed. As cited before, the intra frame prediction module uses as reference to predict the current block, the block that was just previously predicted. However, the block that was previously predicted by intra prediction must first to be processed by forward and inverse transforms and forward and inverse quantization (as shown in fig. 1) [JVT 03] and this block is called reconstructed block. Then, it is very important to generate a solution which is able to deliver this reconstructed block quickly to the intra prediction module and here two questions are essentials: (1) the throughput that must be as higher as possible and (2) the latency that must be as lower as possible.

The architectures presented in this paper were designed based on algorithms developed from the transforms mathematical definition, like that presented in equations (1) and (2). In tab. 1 it is presented the forward DCT algorithm, where $\mathbf{a_i}$ are internal variables, **Xi** are the input data and the **Si** are the output. The other algorithms are similar and are not presented here.

The algorithm presented in tab. 1 was defined considering its hardware implementation. Then, it was divided in two independent steps to allow hardware design with a high throughput and a low latency. The architecture was designed in a pipeline with two stages and with thirty two operators per stage, which implies in the maximum allowed parallelism. This option was the best relation between throughput and latency, because: (1) it allows a high operation frequency, since the critical path is formed by only two adders serially connected; (2) it considers a consumption and production rates of sixteen samples (or one 4x4 block) per clock cycle and (3) it has a latency of only two cycles, since the pipeline has two stages.

Tab. 1 – Forward DCT algorithm.

Step 1	Step 2
$a_0 = X_0 + X_{12} + X_4 + X_8$	$Y_0 = a_0 + a_3 + a_1 + a_2$
$a_1 = X_1 + X_{13} + X_5 + X_9$	$Y_1 = 2*(a_0 - a_3) + a_1 - a_2$
$a_2 = X_2 + X_{14} + X_6 + X_{10}$	$Y_2 = a_0 + a_3 - a_1 - a_2$
$a_3 = X_3 + X_{15} + X_7 + X_{11}$	$Y_3 = a_0 - a_3 - 2*(a_1 - a_2)$
$a_4 = X_0 - X_{12} + X_3 - X_{15}$	$Y_4 = 2*(a_4 + a_6) + a_5 + a_7$
$a_5 = X_4 - X_8 + X_7 - X_{11}$	$Y_5 = 2*(2*a_8 + a_9) + 2*a_{10} + a_{11}$
$a_6 = X_1 - X_{13} + X_2 - X_{14}$	$Y_6 = 2*(a_4 - a_6) + a_5 - a_7$
$a_7 = X_5 - X_9 + X_6 - X_{10}$	$Y_7 = 2*a_8 + a_9 - 2*(2*a_{10} + a_{11})$
$a_8 = X_0 - X_{12} - X_3 + X_{15}$	$Y_8 = a_{12} + a_{15} + a_{13} + a_{14}$
$a_9 = X_4 - X_8 - X_7 + X_{11}$	$Y_9 = 2*(a_{12} - a_{15}) + a_{13} - a_{14}$
$a_{10} = X_1 - X_{13} - X_2 + X_{14}$	$Y_{10} = (a_{12} + a_{15}) - a_{13} - a_{14}$
$a_{11} = X_5 - X_9 - X_6 + X_{10}$	$Y_{11} = a_{12} - a_{15} - 2*(a_{13} - a_{14})$
$a_{12} = X_0 + X_{12} - X_4 - X_8$	$Y_{12} = a_4 + a_6 - 2*(a_5 + a_7)$
$a_{13} = X_1 + X_{13} - X_5 - X_9$	$Y_{13} = 2*(a_8 - 2*a_9) + (a_{10} - 2*a_1)$
$a_{14} = X_2 + X_{14} - X_6 - X_{10}$	$Y_{14} = a_4 - a_6 - 2*(a_5 - a_7)$
$a_{15} = X_3 + X_{15} - X_7 - X_{11}$	$Y_{15} = a_8 - 2*a_9 - 2*(a_{10} - 2*a_{11})$

The generic block diagram of all designed architectures is shown in fig. 2.

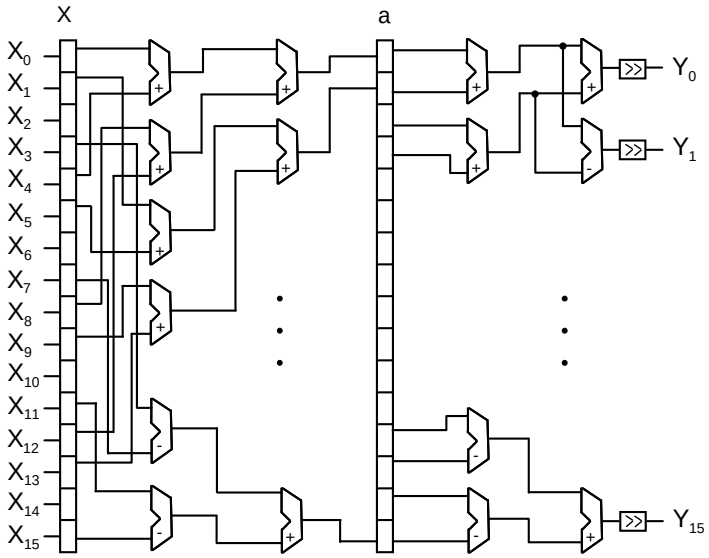


Fig. 2 - Parallel transform architecture using pipeline with 2 stages.

In the design presented in this paper, **Xi** inputs have an 8 bits bit-width and outputs **Yi** have a 14 bits bit-width in the case of forward DCT. The increase in the dynamic range was needed to avoid mistakes that should be caused by overflow in the sum operations.

To allow the pipeline implementation, time barriers (registers) were placed between the two stages. The operators used are dedicated to only one type of operation (additions or subtractions), reducing the complexity of each operator and the complexity of the control unit. The operators were described as macro function adders. This type of description allows the synthesis tools to map the adders to dedicated logic elements for additions which are present in the target FPGA, improving the adders performance.

The parallel processing solution causes an increase in the number of operators used in this design, increasing the device resources utilization. But this solution allows a control block with a much reduced complexity, since the function of control is only to verify if the data available in the input are valid and to indicate that the outputs contains valid values. This operation can be easily designed.

The architecture was validated through simulation using the tools provided by Altera.

4. Systhesis Results

The architectures were described in VHDL. The synthesis targeted some Altera FPGAs [ALT 07]. Quartus II tool was used to generate the results.

Tab. 2 presents the synthesis results for the forward and inverse DCT targeting Cyclone II EP2C35F872C6, Stratix EP1S10F780C5 and Stratix II EP2S15F672C3 devices. In tab. 3 are presented the synthesis results for the forward and inverse 4x4 Hadamard transforms. The targeting devices are the same used in tab. 2.

Tab. 2. FDCT and IDCT synthesis results.

	Forward 4x4 DCT			Inverse 4x4 DCT		
	Frequency (MHz)	Regs	LEs/LUTs	Frequency (MHz)	Regs	LEs/LUTs
Cyclone II	231.4	280	683 (2%)	223.8	280	688 (2%)
Stratix	220.3	-	811 (8%)	222.0	-	808 (8%)
Stratix II	332.0	280	683 (5%)	324.9	280	680 (5%)

Tab. 3. Forward and Inverse 4x4 Hadamard transforms synthesis results.

	Forward 4x4 Hadamard			Inverse 4x4 Hadamard		
	Frequency (MHz)	Regs	LEs/LUTs	Frequency (MHz)	Regs	LEs/LUTs
Cyclone II	229.5	240	594 (2%)	228.0	276	680 (2%)
Stratix	209.9	-	722 (7%)	200.0	-	808 (8%)
Stratix II	330.8	240	594 (5%)	323.7	276	680 (5%)

One expected result is that the architectures mapped to the Stratix device always present a worse result in frequency and in LEs/LUTs consumption. The worst frequency results are function of the technology used in this device, which is the oldest one. The worst result in terms of LE/LUTs consumption was caused because this device does not have dedicated registers slices and then, the registers must be constructed using LEs/LUTS.

The best frequency results for all architectures were reached by the Stratix II device that surpasses all other operation frequencies.

Since the architecture process sixteen samples per clock cycle, and considering the frequencies presented in tab. 2 and in tab. 3, were possible to calculate the throughput of this architecture, considering different target devices. As an example, the processing rates for the forward DCT were calculated and they are presented in tab. 4. The forward DCT was chosen because it presents the best frequency results in average. First column presents the number of samples processed per second (in millions). The second column presents the number of QHDTV (3840x2048 pixels) frames that the architecture is able to process per second. This calculation considered QHDTV videos in a 4:2:0 format, which is defined in the baseline, extended and main profiles of the H.264/AVC standard [JVT 03]. The third column shows the minimal frequency required to allow real time (30 frames per second) when QHDTV frames are being processed.

Tab. 4. Processing rates for FDCT.

Device	Maximum Throughput (Msamples/s)	QHDTV Frames/s (@ Max. Freq)	Min. Freq. for QHDTV (MHz)
Cyclone II	3,702.9	313.9	22.12
Stratix	3,525.0	298.8	22.12
Stratix II	3,657.1	450.3	22.12

Results presented in tab. 4 shows that this architecture can easily reach real time for all target devices even when processing very high resolution videos, like QHDTV. This very high throughput together with the low latency are useful for the intra frame coder design.

5. Conclusions

This paper presented the design of fully parallel architectures for the forward and inverse DCT and for the forward and inverse 4x4 Hadamard transforms. The architectures were defined in a pipeline with two stages and they were described in VHDL and synthesized to Altera FPGAs devices. The architectures were designed targeting the intra frame coder module, then the number of pipeline stages was fixed as two, keeping the high throughput due to the high level of parallelism and allowing a low latency of two cycles.

The best result was found in the DCT mapped to a Stratix II device. With a frequency operation of 332MHz, this module is able to process more than 450 QHDTV frames per second.

As future works are planed the design of T and T^{-1} modules, joining the transforms designed in this paper with the 2x2 2-D Hadamard designed in previous works. Also is planed the design of a Q and Q^{-1} modules.

6. References

[ALT 07] Altera Corporation. **Altera: The Programmable Solutions Company**. www.altera.com. 2007.

[JVT 03] JVT Editors (T. Wiegand, G. Sullivan, A. Luthra), Draft ITU-T Recommendation and final draft international standard of joint video specification (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC), 2003.

[RIC 03] RICHARDSON, I. **H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia**. Chichester: John Wiley and Sons, 2003.

IMAGE & VIDEO PROCESSING II

Reference C Software H.264/AVC Decoder for Hardware Debug and Validation

[1]Márlon A. Lorencetti, [2]Wagston T. Staehler, [1]Altamiro A. Susin
marlon.lorencetti@ufrgs.br, wagston.staehler@ulbra.br, altamiro.susin@ufrgs.br

[1]Universidade Federal do Rio Grande do Sul
[2]Universidade Luterana do Brasil

Abstract

This paper presents the design of a H.264/AVC software decoder with the following blocks: intra prediction, inverse transforms and rescaling, CAVLC entropy decoder, and deblocking filter. There are two developed versions of this software, each one providing different benefits: one structured C version and a C++ version with a GUI (graphical user interface) included. Both versions of the software were validated through output comparison with the JM Reference Decoder. At first, this work brings the advantages of a modeling task, which provides accurate knowledge of the H.264 decoding algorithm, and offers sample files to hardware design group. For some performance non-critical decoder modules, it is also possible to use the structured C version running on an embedded processor instead of its VHDL version. At second, the C++ version presents the possibility to see the entire system on activity, demonstrating the expected video reproduction behavior, and it represents how the hardware must operate. A PC was used to develop and run the codes. It was also used a prototyping board Digilent XUPV2P provided with a Xilinx Virtex-II Pro XC2VP30 FPGA, with two embedded PowerPC processors, in order to execute the structured C software.

1. Introduction

The SBTVD (Sistema Brasileiro de Televisão Digital) uses H.264/AVC (MPEG-4 part 10) [VCE 05] video encoding. A group of electrical and computer engineering students of UFRGS is developing a video decoder that is compliant with this standard. Because software based versions of this decoder rarely can get enough performance to work with high resolutions at real time, the group is developing a hardware decoder [AGO 07], implemented in FPGA.

The difficulty to test the VHDL modules motivated the implementation of a reference software decoder with particular specifications, such as a block separated code. This software can use Xilinx functions to access memory, for example, without many changes in the code, and can be compiled to run in the PowerPC of the FPGA. A modular coding would give the possibility to replace a specific software block by the same block implemented in hardware, mapped into the FPGA. This strategy allows the comparison of the responses between the software and hardware blocks, in order to debug and validate the hardware project.

The software was planned to be separable in small blocks, each one performing a specific function in the decoding process. An incremental approach was chosen to implement this software. Currently, the software decoder has only a few functionalities, but it is able to decode video streams with Intra-frame slices. Although the software is not optimized, it is not really a problem. The main goal on this implementation is its accuracy. All the processing and controlling functions were written in C language, and there is a GUI (Graphical User Interface) written in C++ that runs in the PC to facilitate the debugging of the software, exhibiting the output images for the user.

The creation of a reference software makes the hardware development easier, because there is the possibility to verify how the system works, what its main processes are, and what interface signals must be used.

2. Main blocks description

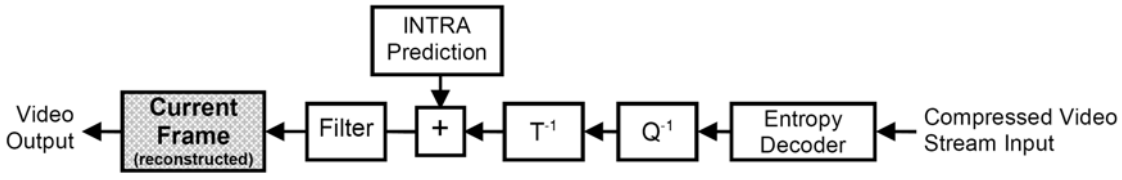


Fig. 1 – Block Diagram of the software.

2.1 Parser module

There is a set of control variables sent in the bit stream. These variables contain all parameters to decode the stream, like the size of the frame, the quantization factor, the maximum number of reference frames, the use of interlaced or progressive frames, the entropy coding to be used, and the number of slices in a frame. These parameters, which are decoded in the parser module, come in specific structures: SPS (Sequence Parameter Set), PPS (Picture Parameter Set) and Slice Header.

2.2 Intra prediction module

The intra-frame coding uses the spatial correlation of the intensity of pixel components in a small portion of the image. Using pre-decoded neighbor values, a prediction of the current block is performed, and the residual data is summed to make the actual block. This prediction can be done in a luma macro block with nine different directions for 4x4 blocks, and four directions for 16x16 blocks. In the case of a chrome macro block there are four different ways to predict the values.

2.3 CAVLC Entropy Decoder module

In the Baseline profile of H.264/AVC decoder, the entropy coding used is the CAVLC, which is based on four tables, chosen based on the previously decoded residual blocks. The main variables on this scheme are the number of non-zero coefficients in the block, the number of 1 or -1 coefficients in the beginning of the zigzag scanned block, and the number of trailing zeros in the end of the block. The adaptive aspect of the entropy coding used in the H.264/AVC standard is an innovation that brought a lot of data compression, compared to the traditional entropy coding styles.

2.4 Inverse Transform and Rescaling modules

H.264/AVC standard uses smaller transform block (4x4 instead of 8x8 like in former standards), and provides three transformations, used depending on the kind of data. The transform used in normal blocks is an integer DCT-based (Discrete Cosine Transform) one. The ordinary DCT was manipulated to avoid the use of floating point operations, using only sums and bit-shifts. If the residuals are from a luma macro block coded with 4x4 intra prediction, the Hadamard transform is used in the DC components of all 4x4 transform blocks within that macro block. If the residuals come from a chrome macro block, the DC coefficients are transformed with a 2x2 Hadamard. This is used to reduce the correlation between DC components, which represent the medium intensity of the block. In this case, the 4x4 Luma/Chrome DC coefficients are first inverse transformed, then rescaled, and finally replaced in the normal transform blocks.

In a macro block, all the transform blocks are rescaled and inverse transformed in a double zigzag order. Finally, the residual data is summed into the previously predicted blocks to compose the actual decoded macro block.

2.5 Deblocking Filter module

One of the new features present in H.264/AVC is the in-loop deblocking filter. Its differential resides on the fact that it is applied before the current frame to become a reference frame for a future inter-prediction. This avoids motion estimation errors due to artifacts present in the image [RIC 03].

The process of filtering acts in the transform block boarders to remove the artifacts that appear with a high quantization factor. The filter strength is adaptive, so it can distinguish between an artifact and a real image boarder using some thresholds to filter or not filter the boarder based on the gradients of intensity near it. In order to do it, the filter needs to access the values of four pixels in each side the boarder, decide the most appropriate strength and change the values of three pixels in each side the boarder. All the vertical and horizontal transform block boarders are filtered.

3. Experiments

3.1 Software Implementation

The need to test the hardware implemented modules of the H.264/AVC decoder suggested the use of a software model. But JM Reference Decoder [HRS 08] has an intricate code, which makes its understanding a real challenge. To solve this problem, a new implementation of a reference software would bring to the development group a better usability and understanding of the algorithms.

The prototyping board used by the hardware group of the project is a Digilent XUP V2P board, which offers a great deal of input and output interfaces, including a VGA output. It contains a Xilinx Virtex-II Pro XC2VP30 FPGA used to implement the VHDL codes written by the group. These codes are planned to be as portable as possible, permitting a future ASIC integration. This FPGA provides two embedded PowerPC processors.

In order to facilitate the posterior inclusion of the reference software in a PowerPC to debug the hardware project, it was divided in blocks. Simulating a real decoding situation, the block that is being implemented in hardware is mapped into the FPGA, and the PowerPC acts like the adjacent modules, controlling data inputs, flags and parameters. A computer sends input data through a serial connection to the PowerPC, which stores this data in a DDR memory.

The implemented software is divided in some source and header units, as shown in the tab.1. Some blocks were written based on a minimal H.264/AVC Baseline decoder implementation [FIE 04], which does not implement in-loop deblocking filter and long-term prediction.

The decoding algorithm has a high complexity, so the performance of the software is not important, because the required decoding rate will only be reached with the hardware implementation. Even though, after a communication interface between the processor and the FPGA is created, some of the modules can be used in the PowerPC since they are not critical performance ones, like the parser or the CAVLC.

Because of the incremental approach used to implement this software it is not able to decode video streams with CABAC, interlaced frames, inter predicted frames, or more than one slice per frame. Multiple parameter sets are ignored.

Tab.1 – Functionality of the implemented units

Unit	Functionality
cavlc.c	Provides structures to organize and create CAVLC code tables and functions to extract code based on bits read from the bit stream. This unit also reads Exp Golomb coded values used in the control variables.
residual.c	Implement the entropy decoding of a macro block residual data using the functions defined in cavlc.c.
common.c	Contains the function to allocate and free the buffers used to store a frame.
in_file.c	Provides functions to open, close, and read data from the input stream file.
input.c	Defines the buffers to store input data and NAL data. It also implements all the functions needed to look or read a defined amount of bits from the input buffer.
intraPrediction.c	Defines the functions to intra prediction.
mbmodes.c	Defines structures to store parameters about a macro block.
mode_pred.c	Provides functions and structures to handle information about prediction and entropy coding in a macro block.
MTransform.c	Implements all the inverse transforms and rescaling processes used to decode a stream, a structure to manipulate the blocks, and a function to map the zigzag scanned coefficients into this structure.
nal.c	Extracts a NAL (Network Access Layer) unit and copy it into the buffer defined in input.c.
params.c	Creates structures to organize all the parameters from PPS and SPS. Implement functions to extract those parameters from the bit stream.
slice.c	Decodes an entire slice, processing macro blocks one by one. This block contains a large portion of the decoder algorithm, as it is responsible for many function calls to reconstruct the macro block.
slicehdr.c	Defines a structure to store the parameters decoded from a Slice Header, and provides the function to decode it.
YUV_BMPfunctions.c	Appends the output frame into a YUV file. Provides a function to store particular frames into a BMP image file. In the case of the version with GUI, shows the current frame on a window.

3.2 Results

The equipment used to test the software was a PC with a Core2Quad Q6600 processor and 4GB DDR2). A first test was done running the software compiled with Borland C++ Builder 6.0 on Windows. The results for some video sequences are shown in tab.2. In order to get more detailed results about how much time each function processes, an additional test was performed. The software was now compiled in Linux, and the gprof tool was used to get the statistical results shown in tab. 3. The sequence used in this test was Coastguard (352x288 pixels) with 300 frames. All the video sequences were generated with the JM Reference Software containing only I-slices and CAVLC entropy coding.

Tab.2 – Results for some standard sequences

Sequence (number of frames)	Resolution (pixels)	Quantization	Frame Rate with GUI (fps)	Frame Rate without GUI (fps)
Coastguard (300)	352x288	36	26.12	69.06
Coastguard (300)	352x288	28	25.43	64.21
Parkrun (20)	1280x720	36	3.27	6.84
Parkrun (20)	1280x720	28	3.01	6.63

Tab.3 – Detailed performance results

Function purposes	Function names	Calls	% time
Filter the block edges in the current frame	filterBlockEdges	22425600	42.96
	deblock_filter	300	2.47
Decodes the residual blocks	residual_block	2070901	7.92
	get_code	6633932	7.48
	coeff_scan	2248160	0.53
Make all the inverse transforms and rescaling processes used to decode the stream	inverse_quantize	2248160	6.34
	inverse_core_transform_fast	2248160	3.08
	transform_luma_dc	5537	0.00*
	transform_chroma_dc	86840	0.00*
Controls the decoding process of a slice	decode_slice_data	300	8.45
Make all the predictions in the decoding stream	PredizLuma4x4	1812208	4.40
	PredizCroma8x8	237600	2.47
	PredizLuma16x16	5537	0.00*

(*) insignificant processing time

4. Conclusions and Future Works

Since the implementation of H.264 decoder is very complex, several designers work in independent modules. Tests are necessary to validate the correct operation of each module. This software implementation reaches its specification to decode H.264 videos using Intra-frame slices. Some tests were performed by the hardware development group using these C codes and test files. Even though performance is not required, results show a satisfactory processing time to be used as validation system.

As future works, the decoder can get new features, like inter prediction, CABAC entropy coding, interlaced frames, and others. Also, an interface between FPGA and PowerPC must be done for a real integration of software and hardware.

The H.264 encoder uses many decoder blocks in-loop to provide a better image reconstruction. A possible next step in the development of reference software for hardware validation is the implementation of the H.264 encoder, which will use these already implemented blocks.

5. References

[AGO 93] AGOSTINI, Luciano V. et al. **Design and FPGA Prototyping of a H.264/AVC Main Profile Decoder for HDTV.** In: IEEE International Workshop on Rapid System Prototyping (2007 174-180 : Porto Alegre). Proceedings, 2007.

[FIE 04] FIEDLER, Martin. **Implementation of a basis H.264/AVC Decoder.** Seminar Paper, 2004.

[FIG 05] FIGUEIRÓ, Thiago et al. **H.264 Implementation Test Using the Reference Software.** In: XVIII Brazilian Symposium on Computer Graphics and Image Processing. Proceedings, 2005.

[HRS 08] H.264 Reference Software. http://iphome.hhi.de/suehring/tml/, Mar. 2008.

[RIC 03] RICHARDSON, Iain E. G. **H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia.** John Wiley and Sons, 2003.

[VCE 05] VIDEO CODING EXPERTS GROUP. **ITU-T Recommendation H.264 (03/05): Advanced vídeo coding for generic audiovisual services.** International Telecommunication Union, 2005.

Architecture Design and Prototyping of a Fully Parallel H.264/AVC 2x2 Hadamard Transform Targeting Real Time in Very High Resolution Videos

[1]Felipe Sampaio, [1]Carolina Fonseca, [1]Fabiane Rediess,
[2]Sergio Bampi, [2]Altamiro Susin, [1]Luciano Agostini
{fsampaio.ifm, cfonseca_ifm, frediess_ifm, agostini}@ufpel.edu.br,
bampi@inf.ufrgs.br, altamiro.susin@ufrgs.br

[1]Universidade Federal de Pelotas
Grupo de Arquiteturas e Circuitos Integrados - GACI
[2]Universidade Federal do Rio Grande do Sul
Grupo de Microeletrônica - GME

Abstract

This paper presents the design, synthesis, simulation and prototyping of a 2-D 2x2 Hadamard Transform architecture focusing in H.264/AVC video compression standard. This architectural design targets real time when processing high resolution videos (QHDTV @ 30fps). The architecture was described in VHDL and synthesized to Cyclone II, Stratix and Stratix II FPGAs from Altera and to Spartan 2E, Spartan3E, Virtex 2P, Virtex 4 and Virtex 5 FPGAs from Xilinx. Stratix II was the Altera device which reaches the highest operation frequency (500 MHz) and it can process 2 billions of samples per second. Among the Xilinx devices, the highest operation frequency was reached by Virtex 5 device (655.9 MHz) and it can reach a processing rate of 2.6 billions of samples per second. The architecture was prototyped in a Digilent XUP V2P board with a Virtex II Pro FPGA.

1. Introduction

The H.264/AVC is the latest video compression standard [ITU 03]. It was developed intending to double the compression rates reached by the previous standards, like MPEG-2. This coding efficiency causes a dramatic increase of the computational complexity of compression/decompression operations, which is around four times higher than the MPEG-2 complexity [RIC 03]. This computational complexity hinders the use of H.264/AVC codecs designed in software when video resolutions are high and when real time processing is desired. In these cases, a hardware design is necessary.

The main modules of H.264/AVC coder are the Inter Frame Prediction (composed by Motion Estimation and Motion Compensation), the Intra Frame Prediction, the Inverse and Forward Quantization (Q and Q-1), Entropy Coding, Deblocking Filter and Forward and Inverse Transforms (T and T-1).

The modules of Forward (T) and Inverse (T^{-1}) Transforms of the H.2.64/AVC standard are formed by three transforms: 4x4 2-D DCT, 4x4 2-D Hadamard and 2x2 2-D Hadamard. T module uses forward DCT and forward 4x4 Hadamard transforms and T^{-1} module uses inverse DCT and inverse 4x4 Hadamard transforms. But both T and T^{-1} modules use the same 2-D 2x2 Hadamard transform, since the forward and inverse 2x2 Hadamard are identical.

The use of 2-D Hadamard transforms is an innovation of the H.264/AVC standard. These transforms were inserted in order to explore a residual correlation presented in homogeneous areas of videos, reaching higher compression rates.

The inputs of Forward Transforms module are the 4x4 residual sample blocks generated by Inter frame or Intra frame coding. The Forward 2-D DCT is applied to all luma or chroma input data. The DC coefficients of chroma block, which were generated by the 2-D DCT, are processed by this 2x2 transform.

This paper focuses in the Forward (T) and Inverse (T^{-1}) Transforms modules, more specifically, in the 2-D 2x2 Hadamard, one of the H.264/AVC innovations. This work proposes a fully parallel architecture to get very high throughputs in its operations.

2. Designed Architecture

The architecture designed for the 2-D 2x2 Hadamard Transform is fully parallel and it was designed with the goal of reach a very high processing rate. The high processing rate of this module is very important to allow the encoder or decoder to reach real time when high definition videos are being processed. This high throughput is more important in the Intra Frame prediction module, since it uses as reference to predict the current block, the block that was just previously predicted. However, the block that is currently being processed by intra prediction must be first processed by Forward and Inverse Transforms and Forward and Inverse Quantization [ITU 03]. So it is very important to reach high processing rates and low latencies in these modules, to reduce

the wasted time in the intra prediction module, since it must wait for new and valid references. Then, the architecture proposed in this paper contributes to this goal, since it can process a complete block with 2x2 samples at each clock cycle.

The 2-D 2x2 Hadamard defined in the H.264/AVC standard is presented in Equation (1), where $\mathbf{W_D}$ represents the four DC coefficients of the 2-D FDCT resulting matrices. This calculation is applied to both forward and inverse 2x2 Hadamard and then, the same designed architecture is able to be used in the forward and inverse transform modules.

The architecture was designed based on an algorithm obtained from Equation (1). This algorithm is presented in tab. 1, where $\mathbf{a_0}$ to $\mathbf{a_3}$ are internal variables that are used in the implementation of the parallel architecture, $\mathbf{W_0}$ to $\mathbf{W_3}$ are the input that will be processed by the transform and **S0** to **S3** are the 2x2 Hadamard transformed coefficients.

$$W_{QD} = \left(\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} W_D \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \right) \qquad (1)$$

Tab.1 - 2-D 2x2 Hadamard algorithm.

$a_0 = W_0 + W_2$	$S_0 = a_0 + a_2$
$a_1 = W_0 - W_2$	$S_1 = a_0 - a2$
$a_2 = W_3 + W_1$	$S_2 = a_1 - a_3$
$a_3 = W_3 - W_1$	$S_3 = a_1 + a3$

The algorithm presented in tab. 1 was defined considering the future hardware implementation of this algorithm. Then, it was divided in two independent steps to allow the pipeline design with a consumption and production rates of four samples per clock cycle. This option allows a parallel processing and a high operation frequency, in function of the simplicity of the operations presented at each pipeline stage.

The block diagram of the designed architecture is shown in fig. 1. This architecture was designed in a two stages pipeline, with four operators per stage. Then, the architecture has a latency of two clock cycles.

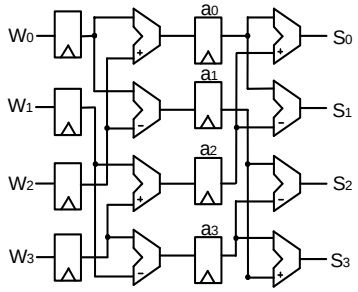


Fig. 1 – 2-D 2x2 Hadamard parallel architecture.

In fig. 1, **Wi** inputs have a bit-width of 8 bits and **Si** outputs have a bit-width of 10 bits. The increase in the dynamic range was needed to avoid mistakes that could be caused by overflow in the sum operations.

To allow the pipeline implementation, time barriers (registers) were placed between the two stages. The operators used are dedicated to only one type of operation (additions or subtractions), reducing the complexity of each operator and of the control unit. On the other hand, the parallelism causes an increase in the number of components used in this design. The operators were described as macro function adders. This type of description allows the synthesis tool to map the adders carry propagation path to dedicated hardware presented inside the target FPGA, improving the adders performance.

The parallel processing solution for 2-D 2x2 Hadamard reduces dramatically the control block complexity, because the function of control, in this case, is only to verify if the data available in the input are valid and to indicate that the outputs contains valid values. This operation can be easily designed.

The architecture was validated through post place-and-route simulation using the tools provided by Altera and Xilinx. After that, the architecture was also prototyped in a Virtex II Pro Xilinx FPGA. The validation and prototyping process will be presented in section 4.

3. Synthesys Results

The 2-D 2x2 Hadamard transform architecture was described in VHDL. The synthesis targeted several Altera and Xilinx FPGAs. Quartus II tools, provided by Altera, and ISE tool, provided by Xilinx, were used to generate the results.

Tab. 2 presents some synthesis results of this architecture when mapped to Altera FPGAs. The selected FPGAs were: Cyclone II EP2C35F672C6, Stratix EP1S110F672C6 and Stratix II EP2515F672C3. Tab. 3 shows the synthesis results of the Hadamard 2-D parallel architecture for Xilinx FPGAs devices. The target FPGAs were: Virtex 2P XC2VP2, Virtex 4 XC4XLX15, Virtex 5 XC5VLX30, Spartan 2E XC2S50E e Spartan 3E XC3S100E.

Tab. 2 - Synthesis results for Altera FPGAs

Device	Freq. (MHz)	LUTs	Registers
Cyclone II	420.17	106 (<1%)	66
Stratix	421.94	108 (1%)	-
Stratix II	500	76 (<1%)	66

Tab. 3 - Synthesis results for Xilinx FPGAs

Device	Freq. (MHz)	Slices / bit slices	Slices Flip-Flops / Registers	LUTs / 4-input LUTs
Spartan 2E	225.07	58 (7%)	70 (4%)	70 (4%)
Spartan 3E	290.61	58 (6%)	70 (3%)	70 (3%)
Virtex 2P	324.04	58 (4%)	70 (2%)	70 (2%)
Virtex 4	575.54	58 (0%)	70 (0%)	70 (0%)
Virtex 5	655.86	32(29%)	70 (0%)	70 (0%)

Considering the selected Altera FPGAs, Cyclone II presented the lowest operation frequency, with 420.17 MHz as maximum frequency. Stratix II reached the highest operation frequency (500 MHz). In this case, the architecture reaches the maximum operation frequency allowed to this device, showing the highly efficient designed architecture.

Among the analyzed Xilinx FPGAs, Spartan 3E presented the lowest operation frequency, with 225.1 MHz. The highest operation frequency was reached with the Virtex 5, reaching 655.9 MHz.

Since the architecture process four samples per clock cycle, and considering the frequencies presented in tab. 2 and in tab. 3, it was possible to calculate the throughput of this architecture, considering different target devices. The results presented in tab. 4 shows that this architecture can easily reach real time (30 frames per second) for all target devices even when processing very high resolution videos, like QHDTV (3840x2048 pixels) at a 4:2:0 color relation, as defined in H.264/AVC main profile. These results show that this architecture successfully reaches the high throughput goal, allowing its use in a hardware design that considers the Intra Frames prediction performance restrictions. In the third column is showed the minimal frequency required to allow real time when QHDTV frames are being processed.

Tab. 4 – Processing rates for Altera and Xilinx FPGAs

Device	Maximum Throughput (Msamples/s)	QHDTV Frames/s (@ Max. Freq)	Min. Freq. for QHDTV (MHz)
Cyclone II	1,680.7	6,838.79	1,84
Stratix	1,687.8	6.867.68	1,84
Stratix II	2,000.0	8,138.02	1,84
Spartan 2E	900.3	3,663.33	1,84
Spartan 3E	1,162.5	4,730.22	1,84
Virtex 2P	1,296.2	5,274.25	1,84
Virtex 4	2,342.2	9,530.44	1,84
Virtex 5	2,623.5	10,675.05	1,84

It is important to emphasize that the performance of this architecture depends on the continuous availability of four input samples per clock cycle. If these data are not always available, so the architecture throughput will be reduced.

4. Architecture Prototyping

The first validation step was a C implementation of the 2-D 2x2 Hadamard Transform which was used to generate the input stimuli that were used in the next validation step, allowing a comparison between the software and hardware results. After that, the designed architecture was prototyped in a Digilent XUP-V2P board which has one Virtex II Pro XC2VP30 Xilinx FPGA. Some embedded memory blocks of the target FPGA was used in this prototyping process. One FPGA embedded block memory was used to store the input stimuli, other to store the architecture results and a third embedded memory was used to store the results generated by the software implementation. This third memory was used only for validation purposes. An auxiliary architecture was also described in VHDL to realize de comparison between the results generated by software and by the designed.

One switch of the used prototyping board was used to implement the architecture reset. When the reset switch is changed to zero, the process starts. One LED of the board is turned on when the process finishes. If the results generated by the architecture are equal to the results generated by software, then another LED of the board is turned on to indicate that the architecture generated the expected outputs.

Fig. 2 shown the prototype of the 2x2 Hadamard transform architecture. The two LEDs are turned and are highlighted in fig. 2. This means that the architecture finishes to process the input stimuli and that the comparison with the software results was successfully. Therefore the prototype of this architecture was considered validated.

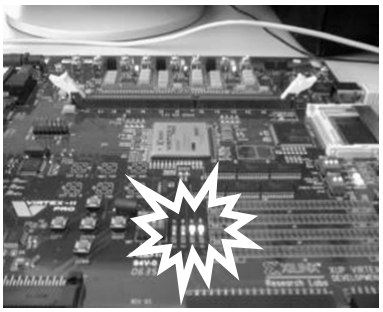


Fig. 2 – Architecture prototype.

5. Comparisons

This section presents a comparison with a serial design for 2x2 Hadamard, with a data consumption and production of one sample per clock cycle. This serial version was previously designed in our research group [AGO 01]. This architecture was also designed in a pipeline with two stages, but only one operator is used in each stage. Tab. 5 presents a comparison between these two architectures. In this comparison, the two architectures were mapped to a Xilinx Virtex 2P FPGA.

Tab. 5 Comparison between serial and parallel architectures in Xilinx Virtex 2P FPGAs.

Implementation	Freq. (MHz)	4-input LUTs	Max. Throughput (Msamples/s)
Parallel	324,04	70	1.296,2
Serial	201,7	98	201,7

The first analysis about the results presented in tab. 5 is related to the hardware resources consumption. The serial version of the architecture consumes more resources than the parallel version. Even using four operators per pipeline stage instead of one operator that is used in the serial version, the parallel architecture consumes less hardware than the serial architecture. This is mainly caused by the drastic simplification in the control and in the input and output management.

Other important information is that the parallel version, even using less hardware resources, presents a throughput 6.4 times higher than the serial architecture.

6. Conclusions

This paper presented the hardware design of a parallel architecture for the 2-D 2x2 Hadamard transform. The architecture was defined in a pipeline with two stages and it was described in VHDL and synthesized to several Altera and Xilinx FPGAs devices.

Considering the Altera FPGAs, the device which reaches the best performance was the Stratix II, with 500 MHz as maximum operation frequency, processing about 2 billions of samples per second.

Among the Xilinx FPGAs, Virtex 5 got the higher operation frequency, with 655.86 MHz, reaching a throughput of 2.63 billions of samples per second.

The designed architecture was prototyped in a Digilent XUP V2P board and validated.

The designed architecture was compared with a previous work of our research group and the architecture presented in this paper reaches a lesser hardware consumption and a very higher throughput than that previous solution.

The designed architecture, when mapped to all considered FPGAs, is able to reach real time even when processing very high resolution videos like (QHDTV). Then, the goal of this paper was reached.

7. References

[AGO 01] AGOSTINI, L. V. et al. **High Throughput Architecture for H.264/AVC Forward Transforms Block.** In: ACM Grate Lakes Symposium on VLSI, GLSVLSI, 2006. Proceedings… New York: ACM, 2006b. p. 320-323

[RIC 03] RICHARDSON, I. **H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia.** Chichester: John Wiley and Sons, 2003.

[ITU 03] **ITU-T Recommendation and final draft international standard of joint video specification (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC), 2003.**

A High Throughput Diamond Search Architecture for HDTV Motion Estimation

[1]Marcelo Porto, [2]Luciano Agostini, [1]Sergio Bampi, [1]Altamiro Susin
{msporto, bampi}@inf.ufrgs.br, agostini@ufpel.edu.br, altamiro.susin@ufrgs.br

[1]Grupo de Microeletrônica (GME) – II – UFRGS
[2]Grupo de Arquiteturas e Circuitos Integrados (GACI) – DINFO – UFPel

Abstract

This paper presents a high throughput and low hardware cost architecture for motion estimation using a Quarter Sub-sampled Diamond Search algorithm (QSDS). The Diamond Search algorithm (DS) was presented, and also the sub-sampling technique. The designed hardware architecture considered a search area of 100x100 samples, with blocks of 16x16 pixels. The architecture was described in VHDL and mapped to a Xilinx Virtex-4 FPGA. Synthesis results indicate that QSDS architecture is able to run at 212.5 MHz, using only 3541 LUTs. This architecture can reach real time at 30 fps for HDTV (1920x1080 pixels) in the worst case, and it can process 187 HDTV frames per second in the average case.

1. Introduction

Motion estimation (ME) is the most important task in the current standards of video compression. Full Search (FS) is the most used algorithm for hardware implementation of ME, due to its regularity, easy parallelization and good performance. However, hardware architectures for FS algorithm usually require lots of hardware resources to achieve real time, mainly for high resolution videos.

Diamond Search (DS) [KUH 99] algorithm can drastically reduce the number of SAD calculations when compared to FS algorithm. Using DS algorithm it is possible a significantly increase in the search area, generating quality results close to the FS results. This reduction in SAD calculations can significantly reduce the necessary hardware resources to achieve real time for high resolution videos. Another strategy to reduce the use of hardware resources and to increase the throughput is the Pel Subsampling (PS) technique [KUH 99]. Using PS it is possible to reduce significantly the number of SAD calculations and to speed up the architecture, with a small degradation in the signal quality.

This paper presents a high throughput architecture for ME using DS algorithm and PS 4:1, named Quarter Sub-sampled Diamond Search (QSDS). This architecture was described in VHDL and mapped to a Xilinx Virtex-4 FPGA. This architecture can reach real time for HDTV videos with a low hardware cost.

2. Used Search Algorithm

DS algorithm has two diamond patterns, the Large Diamond Search Pattern (LDSP) and the Small Diamond Search Pattern (SDSP) [KUH 99]. The LDSP consists of nine comparisons around de center of the search area. The SDSP is used in the end of the search, when four candidates block are evaluated around the center of the LDSP to refine the final result. Fig. 1 (a) shows the LDSP and the SDSP.

The search ends when the lower SAD is found at the center of the LDSP. So the SDSP is applied and the search is over. When the best match is found in a vertex, more five blocks are evaluated to form a new LDSP, as showed in fig. 1(b). If the best match was obtained in an edge, more three blocks are evaluated, as can be seen in fig. 1 (c).

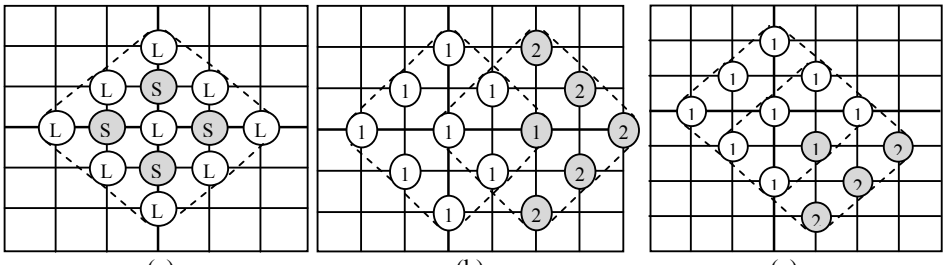


Fig. 1 - (a) LDSP and SDSP, (b) Search for a vertex, and (c)Search for an edge.

PS 4:1 is applied to each evaluated candidate block. Thus each line of the block has eight samples, instead of 16 in a non sub-sampled block. The number of lines is also sub-sampled, only eight lines are considered. Thus each candidate block with 16x16 samples, becomes a sub-sampled block with 8x8 samples.

3. Designed Architecture

The designed architecture uses QSDS algorithm with sub-sampled blocks of 8x8 pixels. SAD [KUH 99] was used as distortion criteria. The architectural block diagram is showed in fig. 2. The architecture has nine processing unities (PU). The PU can process eight samples in parallel and this means that one line of the sub-sampled 16x16 block is processed in parallel. So, eight accumulations must be used to generate the final SAD result of each block, one accumulation per line.

The nine candidate blocks of the LDSP are calculated in parallel and the results are sent to the comparator. Each candidate block receives an index, to identify the position of this block in the LDSP. The comparator finds the lowest SAD and sends the index of the chosen candidate block to the control unit. The control unit analyses the block index and decides the next step of the algorithm. If the chosen block has index 4, the lowest SAD was found in the center of the LDSP. So, the control unit starts the final refinement with the SDSP. In the SDSP calculation, four more candidate blocks are evaluated. The lowest SAD is identified and the control unit generates the corresponding motion vector to this block.

When the chosen block in the first step has index 0, 3, 5 or 8, the control unit starts the second step of the algorithm with a search for vertex. In this case, other five candidate blocks are calculated and compared to the lowest SAD of the previous step. If the chosen block from the first step has index 1, 2, 6 or 7, the control unit starts the second step of the algorithm with a search for edge, and three more candidate blocks are calculated. The second step can be repeated *n* times, till the lowest SAD is found in the center of the LDSP. Then the SDSP is applied.

The number of iterations in the second step is restricted to 20 in QSDS architecture. This restriction is done to define a search area and to allow an easier synchronism of this module with other encoder modules. Thus, the maximum search area of QSDS is 100x100 samples.

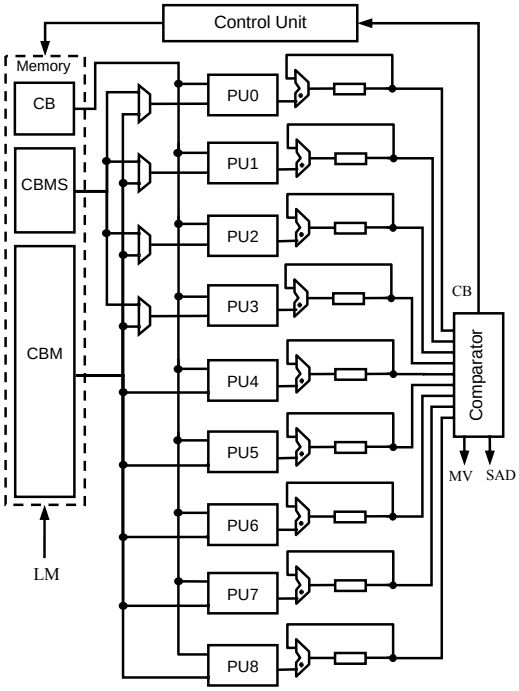


Fig. 2 – QSDS block diagram architecture.

3.1 Memory organization

The internal memory is organized in 15 different local memories, as presented in fig. 3. The local memory (LM) stores the region with the nine candidate blocks of the first LDSP and all the possible blocks for the next step. Thus, when the control unit decides which blocks must be evaluated in the next step, the LM already has this data. LM memory is composed of 16 words with 128 bits. Another 13 small memories are used to store the candidates block (CBM) and one for the current block (CB). These 14 memories are composed of 8 words with 64 bits (8 samples of 8 bits) and they store one sub-sampled block with 8x8 samples.

LM memory is read line by line and the data is stored in the CBM memories. Nine CBMs are used to store the candidate blocks from the LDSP. Four CBMs are used to store the blocks of the SDSP. These blocks are always stored, and they are ready to be calculated if the control decides to start the SDSP. This solution speeds up the architecture and reduces the memory access latency. When the SDSP mode is active, the control unit selects the multiplex in fig. 2 and the PUs receive the data from these memories.

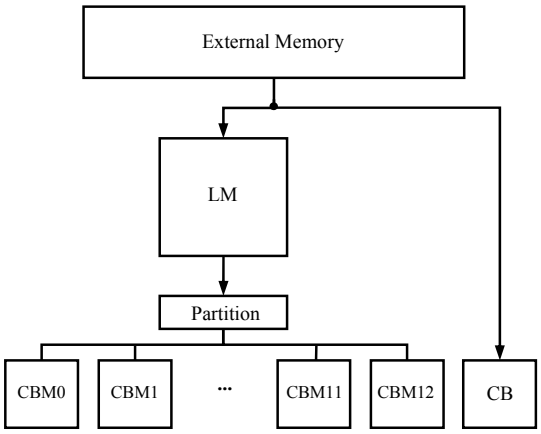


Fig. 3 - Memory organization.

Each line from the LM has 128 bits. However, a partition unity (Partition in fig. 3) cuts the line into 64 bits. This unit selects the right part of the 128 bits which corresponds to the candidate block. When the first line of the LM is read, the PU selects the right part of the word which corresponds to the candidate block 0, to be stored in the CBM0. When the second line is read, this module selects the right part of the second line of the candidate block 0, to be stored in the CBM0, and the correct part for the first line of the candidate block 1, to be stored in the CBM1. This process finishes when all LM is read and all CBMs are full.

A local control was developed to control the memory access. The control unit in fig. 2 is not responsible for controlling the memories read/write process. When the datapath finishes the SAD calculation, the control unit informs the memory if the search should continue or if the SDSP should be applied. If the search continues, by an edge or a vertex, the CBMs are stored with the new candidate blocks and the LM memory is reloaded. In this case, there are 10 cycles of latency to write the first line in the nine CBMs. If SDSP is applied, the four candidate blocks are already stored and the datapath starts with no memory latency. Even using 15 different memories, the total memory consumption is small. The search area is loaded accordingly to the algorithms necessity, so, no irrelevant data is stored. A total of 9.2Kbits is used for all motion estimator memories.

3.2. Performance Evaluation

QSDS algorithm stops the search when the best match is found at the center of the LDSP. This condition can occur in the first application of the LDSP. Thus, no searches for edge or vertex are done. It is the best case, when only thirteen candidate blocks are evaluated: nine from the first LDSP and four from the SDSP.

This architecture uses 26 clock cycles to fill all the memories and to start the SAD calculation. The PUs have a latency of four cycles and it needs seven more cycles to calculate the SAD of one block. The comparator uses five cycles to choose the lowest SAD. Thus, 42 clock cycles are necessary to process the first LDSP. The SDSP needs only 20 cycles to calculate the SAD for the four candidates block. So, in the best case, this architecture can generate a motion vector in 62 cycles.

In the cases where the LDSP is repeated, for an edge or vertex search, 10 cycles are used to write in the CBMs. The same 16 cycles are used by the PUs and the comparator. Both edge and vertex search use the same 26 cycles. QSDS architecture can do 20 LDSP repetitions, and for each one, more 26 cycles are used.

4. Synthesis results

The proposed architecture was described in VHDL. ISE 8.1i tool was used to synthesize the architecture to the Virtex-4 XC4VLX15 device and ModelSim 6.1 tool was used to simulate and to validate the architecture design. Tab. 1 presents the synthesis results.

The device hardware utilization is small, since only 3,5k LUTs are used. The synthesis results show the high frequency achieved by the QSDS architecture.

Tab. 1- Synthesis Results for QSDS architecture

Parameter	QSDS Results
BRAMs	32
Slices	1964
Slice FF	1980
LUTs	3541
Frequency (MHz)	212.5
HDTV fps (worst case)	45
HDTV fps (average case)	187

Tab. 1 shows also the architecture performance considering HDTV (1920x1080 pixels) videos. The QSDS architecture can use, in the worst case, 562 clock cycles to generate a motion vector. This number of clock cycles is enough to execute the 20 iterations predefined for the second step of the algorithm. After that, the control unit stops the search and starts the SDSP. The QSDS architecture can achieve real time for HDTV video in the worst case, because it can process till 45 HDTV frames per second.

The average case was obtained through the software implementation of the algorithm. For a search area of 100x100 samples, the algorithm uses an average of three iterations in the second step of the algorithm. Then, the QSDS architecture needs 140 clock cycles to generate a motion vector in this average case. This implies a processing rate of 187 HDTV frames per second.

QSDS FPGA results were compared to FS [LAR 06], FTS (Flexible Triangle Search) [MOH 05] and FS+PS4:1 and early termination [REH 06] architectures, as presented in tab. 2. Solutions [LAR 06] and [MOH 05] have a constant throughput and [REH 06] have a variable throughput due to the early termination. The search area used by [REH 06] is of 32x32 samples, [LAR 06] and [MOH 05] do not present this information. Tab. 2 shows the synthesis results comparisons of founded related works and QSDS architecture. The processing rate of HDTV frames (1920x1080) is also presented. The number of CLBs used by [REH 06] is not presented.

Tab. 2 - Comparative results with related works

Solution	FPGA	CLBs	Frequency (MHz)	Search Area	HDTV (fps)
[LAR 06]	SPARTAN 2	939	109.8	-	1.66
[MOH 05]	SPARTAN 3	6142	74.0	-	45
[REH 06]	VIRTEX 2	-	120.0	32X32	*24
QSDS	VIRTEX 4	492	212.5	100X100	*187

*Average throughput

The unit of hardware resources measurement used for the comparison is the number of Configurable Logic Blocks (CLB) used by each compared solution. However, the ISE 8.1i tool does not present this information clearly, so it is possible to estimate this information through the number of slices used. Each slice has four CLBs, thus the number of CLBs used for the QSDS architecture was calculated as the forth part of the number of used slices.

QSDS architecture is the fastest one among the compared designs. Due to the efficiency of the QSDS algorithm, the designed architecture uses less hardware resources than other solutions, reaching the highest throughput, even using a big search area.

5. Conclusions

This paper presented hardware architecture for a Quarter Sub-sampled Diamond Search algorithm named QSDS. The synthesis results showed that the designed architecture can achieve high throughput with a low hardware cost.

The presented architecture is able to run at 212.5 MHz in a Xilinx Virtex 4 FPGA. Only 3541 LUTs were used to implement the architecture. The QSDS architecture can work in real time for HDTV (1920x1080 pixels) videos. In the worst case, the QSDS architecture can process 45 HDTV frames per second. In the average case, the architecture is able to process 187 HDTV frames per second.

QSDS architecture reaches the best results for maximum throughput, and hardware resources utilization in comparison with presented related works.

6. References

[KUH 99] KUHN, P. M. **Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation.** Boston: Kluwer Academic Publishers, 1999.

[LAR 06] LARKIN D. et al. A **Low Complexity Hardware Architecture for Motion Estimation.** In**:** IEEE International Symposium on Circuits and Systems, Island of Kos, Greece, 2006, pp. 2677-2688.

[MOH 05] MOHAMMADZADEH, M. et al. **An Optimized Systolic Array Architecture for Full-Search Block Matching Algorithm and its-Implementation on FPGA chips.** In: The 3rd International IEEE-NEWCAS Conference, 2005, pp.174-177.

[REH 06] REHAN, M. et al. **An FPGA Implementation of the Flexible TriangleSearch Algorithm for Block Based Motion Estimation.** In: IEEE International Symposium on Circuits and Systems, Island of Kos, Greece, 2006. pp. 521-523.

Level Decoder Architecture for CAVLD of H.264 Video Compression Standard

[1]João Vortmann, [2]Thaísa Silva, [1]Luciano Agostini, [2]Altamiro Susin, [2]Sergio Bampi
{jvortmann_ifm, agostini}@ufpel.edu.br,
{tlsilva, bampi}@inf.ufrgs.br, Altamiro.Susin@ufrgs.br

[1]Grupo de Arquiteturas e Circuitos Integrados (GACI) – DINFO – UFPel
[2] Grupo de Microeletrônica (GME) – Instituto de Informática – UFRGS

Abstract

This paper proposes an architecture for level decoding presented in the Entropy Decoder of H.264/AVC video compression standard. Initially, a version in software was developed using C programming language. The architecture designed was described in VHDL and synthesized to an Altera Stratix-III FPGA. From the synthesis results it was possible to verify that the architecture reached a medium throughput of 61.83 million of samples per second. The software version was also used for data generation, required in the validation process. The results obtained indicates that this solution attends the requirements to process SDTV (720x480 pixels) frames in real time.

1. Introduction

The wide diffusion of digital video through the several communications technologies as the internet, mobile phone and DVD, only is possible due to research in video compression.

H.264/AVC [JOI 03] is the latest video compression standard and it was developed by JVT [INT 07]. The first version of the H.264/AVC was approved in 2003 [INT 05]. This standard doubled the compression rates reached by previous standards but this coding efficiency caused an important increase in the computational complexity of the codec operations. This complexity hinders, in the current technology at least, the use of H.264/AVC codecs designed in software when the video resolutions are high and/or when real time is desired. In these cases, a hardware design of H.264/AVC codec is necessary.

The H.264/AVC decoder is formed by the following modules: inter-frame predictor, intra-frame predictor, inverse transforms (T^{-1}), inverse quantization (Q^{-1}) and entropy decoder, as presented in the Fig.1.

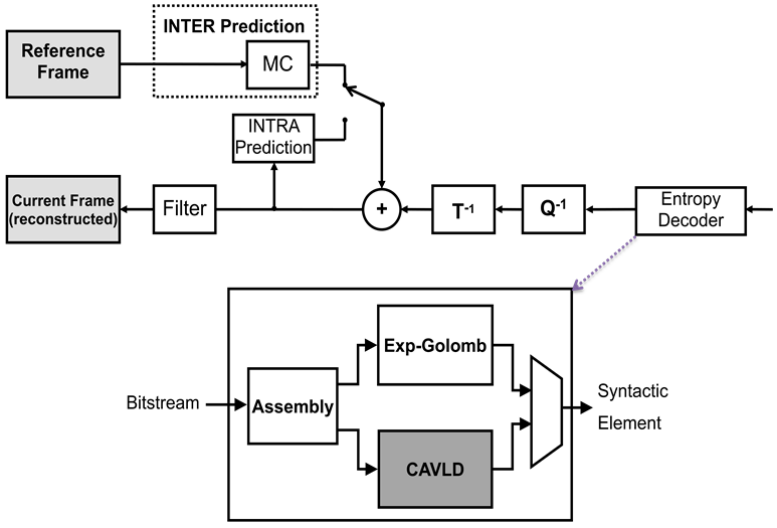


Fig. 1 – H.264/AVC Decoder Block Diagram with emphasis in the CAVLD from the Entropy Decoder

The Entropy Decoder, in the baseline profile, uses two main tools to allow a high data compression: the Exp-Golomb decoding and Context Adaptive Variable Length Decoding (CAVLD) [RIC 03]. The focus of this work is the Decoder Level, which is one of the critical stages of the CAVLD, highlighted in Fig.1. The Level Decoder architecture was described in VHDL, mapped for Altera FPGAs [ALT 08] and partially validated.

This paper is organized as follows. The second section presents briefly the Context Adaptive Variable Length Decoder. Third section presents the designed Level Decoder architecture. Section four presents the synthesis results and the section five presents the conclusions and future works.

2. Context Adaptive Variable Length Decoder

The decoder receives a compressed bitstream from the NAL inputs [RIC 03]. The data elements are entropy decoded and reordered to produce a set of quantized coefficients, which are sent to the inverse quantization block. In Baseline profile, the residual block data is decoded using CAVLD scheme, and other variable-length coded units are decoded using Exp-Golomb codes [SAL 00][INT 03][RIC 03].

The CAVLD process can be divided in the following main operations:

- Decoding the number of coefficients and trailing ones:
  The first step in CAVLD is to decode the total number of non-zero coefficients (TotalCoeffs) and the number of trailing +/-1 values (T1). The TotalCoeffs is ranged from 0 to 16, while T1 is ranged from 0 to 3. There are five choices of look-up tables in this part: Num-VLC0 (nC= -1), Num-VLC1 (0<=nC<2), Num-VLC2 (2<=nC<4), Num-VLC3 (4<=nC<8), and FLC (8<=nC).
- Decoding the sign of each T1:
  From the above coeff_token process, we can know the value of T1. Thus, a single bit is used to decode the sign (0=+1, 1=-1) of each T1 in reverse order.
- Decoding the levels of the remaining non-zero coefficients:
  The level of each non-zero coefficient is decoded in reverse order. The choice of VLC tables to decode each level is decided accordingly to the magnitude of each successive coded level, i.e. the context adaptive feature in CAVLD.
- Decoding the total number of zeros before the last coefficient:
  The total_zeros tables are applied for decoding AC 4x4 blocks or DC 2x2 blocks. The VLC tables to decode the total zeros is decided accordingly to the total number of the non-zero coefficients in the current AC 4x4 or DC 2x2 blocks.
- Decoding each run of zeros:
  The number of zeros preceding each non-zero coefficient is decoded in reverse order. The VLC table for each run of zeros is chosen depending on the previous number of zeros (zeroleft) which were not decoded yet.

3. Level Decoder Architecture

The designed architecture is formed by a simple datapath, a finite state machine to control it, and a buffer.

The process to decode a level has a regular structure and it does not need tables, as cited above. The process is carried out through the reading of bits from the input and a level value is generated respecting a set of steps.

A level code consists of a prefix and a suffix. The prefix is defined as a sequence of zeros, that has its size determined by tam_prefix, until de first bit one. The suffix has a variable size, from 0 to 6 bits. The size of suffix is adapted as the level decoding process is performed, relying on previous levels magnitudes. After read prefix and suffix codes from input, the CodeNo is computed, as shown in (1) below. In (1), table indicates the size of suffix.

$$CodeNo = ((tam\_prefix \times (2^{table})) + suffix) \quad (1)$$

From CodeNo, it is possible to reach the level value. If the CodeNo is even, the level is calculated as (2) or (3).

$$Level = ((CodeNo + 2) /2) \quad (2)$$
$$Level = - ((CodeNo + 1) /2) \quad (3)$$

The datapath of the designed architecture is shown in Fig. 2. It is formed by 2 adders, 1 multiplier, 1 decoder, 4 registers, 2 multiplexers, the module responsible to adapt the decoding table, the module that arrange the buffer output in the suffix register input, a controlled NOT and a shifter.

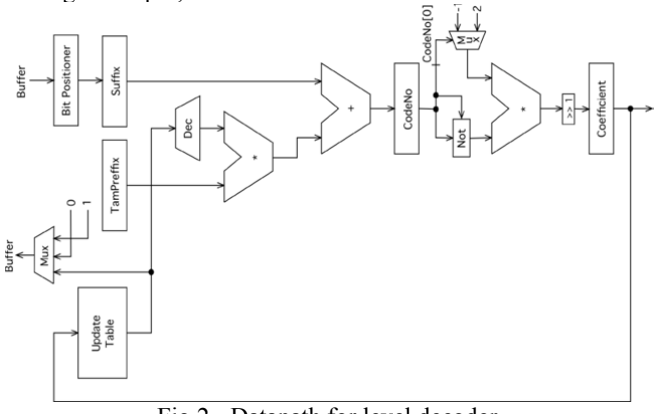


Fig.2 - Datapath for level decoder

The finite states machine is shown in Fig. 3. It has 10 states that work as follows.

The state 0 reset the architecture. The state 1 start the decoding process for a set of levels. The 2 and 3 states are responsible to read the prefix and suffix from the input, respectively. The state 3 takes just 1 cycle because the suffix bits are read in parallel from the buffers content. Thus, the CodeNo can be generated as described above. This operation is done in state 4.

In state 5 the level value is generated from CodeNo. After that, some states were inserted to solve the context adaptability and special cases. The state 6 adapt the suffix size based on the last decoded level. The state 7 and 8 are used to increment or decrement the level magnitude, a special case specified in H.264/AVC standard. Finally, the state 9 indicates that a level code is ready and then the FSM returns to state 1, where a new cycle, for the next level, begins.

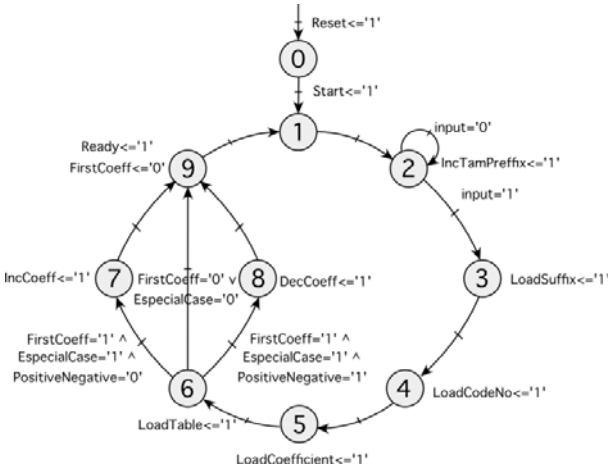


Fig.3 - Finite states machine for level control

4. Synthesis Results

The synthesis of the Level Decoder architecture was targeted to the Altera Stratix-III EP2S15F484C2 FPGA [ALT 08]. These results are presented in Table 1.

Tab.1 - Synthesis Results

Architecture	**Logic Cells**	***Frequency (MHz)***	***Throughput (Msamples/s)***
Level Decoder	251	171.44	61.83

Device EP3SE50F484C2

Through the obtained results it was possible to verify that the level decoder used 251 logic cells of the target FPGA. In performance terms, the critical path estimated by the synthesis tool was of 5.83 ns, allowing a maximum operation frequency of 171.44 MHz.

Considering the results from the software evaluation, it was possible to estimate a mean quantity of 14 cycles to decode of a level. Thus, with an operation frequency of 171.44 MHz this architecture is able to generate 12.25 million of levels per second. Moreover, from the software valuation, it was possible to determine that a level is generated in average at each 5.1 samples of video. Then, the throughput of this architecture was estimated in 61.83 million of samples per second.

Although the designed architecture has a simple datapath, it presented a high performance, reaching the requirements necessary to process SDTV (720x480 pixels) frames in real time.

5. Conclusions and Future Works

This work designed an architecture to level decoding, presented in Entropy Decoder of H.264/AVC video compression standard.

The designed solution presented good results. The average processing rate presented is able to process SDTV (720x480 pixels) frames in real time, fulfilling the requirements of H.264/AVC standard.

As future works, is intended to use a first one detector, as seen in [ALL 06], aiming to increase prefix reading speed and so maximizing the architecture capabilities. Moreover, the integration of level decoder with the other building modules of CAVLD and Exp-Golomb Decoder, will allow a complete solution for Entropy Decoder.

6. References

[ALL 06] M. Alle, J. Biswas, S. K. Nandy, **High Performance VLSI Architecture Design for H.264 CAVLC Decoder**. IEEE Application-specific Systems, Architectures and Processors, 2006.

[ALT 08] ALTERA Corporation. Available in: http://www.altera.com

[INT 07] INTERNATIONAL TELECOMMUNICATION UNION. Joint Video Team (JVT), Available in: <www.itu.int/ITUT/studygroups/com16/jvt/>.

[INT 03] INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Recommendation H.264 (05/03): advanced video coding for generic audiovisual services. 2003.

[INT 05] INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Recommendation H.264 (03/05): Advanced Video Coding for Generic Audiovisual Services. 2005.

[JOI 03] Joint Video Team, **Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification***,* ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, Maio 2003.

[RIC 03] I. Richardson, **H.264 and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia**. Chichester: John Wiley and Sons, 2003.

[SAL 00] D. Salomon, **Data Compression: The Complete Reference**. New York: Springer, 2000.

High Efficiency Hardware Design for Binary Arithmetic Decoder Engines of CABAD Based on Bitstream Flow Analysis

Dieison Antonello Deprá, Claudio Diniz, Bruno Zatt, Sergio Bampi
{dadepra, cmdiniz, bzatt, bampi}@inf.ufrgs.br

Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)

Abstract

This paper presents the design and implementation of a hardware architecture dedicated to binary arithmetic decoder (BAD) engines of CABAD, as defined in the H.264/AVC video compression standard. The BAD is the most important CABAD process, which is the main entropy encoding method defined by the H.264/AVC standard. The BAD is composed by three different engines: Regular, Bypass and Final. We conducted a large set of software experiments to study the utilization of each kind of engine. Based on bitstream flow analysis we proposed a high efficiency dedicated hardware architecture, which maximizes the trade-off between hardware cost and throughput of BAD engines. The proposed solution was described in VHDL and synthesized to Xilinx Virtex2-Pro FPGA. The results shown that developed architecture reaches the frequency of 103 MHz and can deliver up to 4 bin/cycle in bypass engines.

1. Introduction

Today we can observe a growing popularity of high definition digital videos, mainly for applications that require real-time decoding. This creates the need for higher coding efficiency to save storage space and transmission bandwidth. Techniques for compressing video are used to meet these needs [WEI 03b]. The most advanced coding standard, which is considered state-of-the-art, is the H.264/AVC, defined by the ITU-T/ISO/IEC [ITU 05]. This standard defines a set of tools, which act in different domains of image representation to achieve higher compression rates, reaching gain in compression rate of 50% in relation to MPEG-2 standard. The H.264/AVC standard introduces many innovative techniques to explore the elimination of redundancies, present in digital video sequences. One of these main techniques is related to the entropy coding method.

The H.264/AVC standard specifies two entropy coding methods: Context-Adaptive Variable-Length Coding (CAVLC) and Context-Adaptive Binary Arithmetic Coding (CABAC). Both are based on the fact that the digital video sequences present not stationary but statistical behavior [MAR 03]. Moreover, this statistic behavior is highly dependent on the type of content that is being processed and on the capture techniques [WEI 03a] [WEI 03b]. To deal with this behavior, the H.264/AVC standard adopts an innovator mechanism that provides a dynamic adaptive context, which is introduced to CAVLC and CABAC [MAR 03].

The CABAC is the most important entropy encoding method defined by the H.264/AVC standard, allowing the H.264/AVC to reach 15% coding gain over CAVLC. However, to obtain this gain it is necessary to pay the cost of increasing computer complexity. The CABAC is considered the main bottleneck in the decoding process because its algorithm is essentially sequential. Each iteration step produces only one bin and the next step depends on the values produced in the previous iteration [WEI 03b].

Techniques to reduce the latency and data dependency of CABAD (Context-Adaptive Binary Arithmetic Decoding) have been widely discussed in the literature. Basically, five approaches are proposed: 1) pipeline; 2) contexts pre-fetching and cache; 3) elimination of renormalization loop; 4) parallel decoding engines, and 5) memory organization. The pipeline strategy is employed by [YAN 06] to increase the rate of bins/cycle. [EEC 06] presents an alternative to solve the latency of renormalization process. The speculative processing through the use of parallel decoding engines is explored in [YU 05], [BIN 07] and [KIM 06]. High efficiency decoding process by employing pre-fetching and cache contexts is discussed in [YU 05] and [ZHA 07], respectively. The memory optimization and reorganization are addressed in [YAN 06].

This work presents the development of hardware architecture dedicated to binary arithmetic decoder engines of CABAD. The design aims to high efficiency implementation, based on software experiments of bitstream flow analysis. In the next section are presented details for all BAD engine kinds of CABAD as defined by H.264/AVC standard. Section 3 presents the bitstream flow analysis. The architecture proposal and results achieved for the proposed architecture are presented in Section 4. Finally, some conclusions are made in Section 5.

2. Context Adaptive Binary Arithmetic Decoder

The CABAC is a method of entropy encoding which transforms the value of a symbol in a word of code, it generates variable length codes near to the theoretical limit of entropy. The CABAC works with recursive

subdivision of ranges combined with context models that allow reaching better coding efficiency. However, modeling probabilities of occurrence of each symbol brings great computational cost. To decrease computational complexity, CABAC adopts a binary alphabet. Thus, it is necessary to do an additional step that converts the symbol values in binary sequences reducing to two the set of possible values, zero or one. For more details see [WEI 03a].

The H.264/AVC standard defines CABAC as one of the entropy encoding methods available in the Main profile. Here the CABAC is employed, from the macroblock layer, for encoding data generated by tools that act on the transformation of spatial, temporal and psycho-visual redundancies [ITU 05].

CABAD is the inverse process of CABAC and it works in a very similar way. CABAD receives as input an encoded bitstream, which contains information about each tool and its parameters used by the encoder to make its decisions. Inside CABAD these information are named syntax elements (SE) [MAR 03]. To decode each SE, the CABAD can perform N iterations and for each step one bit is generated consuming zero, one or more input bits. Each generated bit is named “bin” and the “bin” produced on each iteration is concatenated with the previous forming a sequence named "binstring".

The flow for decoding each SE involves five procedures that can be organized into four basic stages: I) Context address generating II) Context memory access; III) Binary arithmetic decoding engines; IV.a) Binstring matching; and IV.b) Context updating. In the first stage, the context address is calculated according to SE type that is being processed. Following, the context memory is read returning one state index and one bit with the value of the most probable symbol (MPS). From this index, the estimated probability range of the least probable symbol (LPS) is obtained. After, BAD engine receives the context model parameters and produces a new bin value. In the fourth stage, generated bins are compared with a pattern of expected bins and, in parallel, the context model should be updated in memory to guarantee that the next execution will receive the updated probability of occurrence of each SE.

The CABAD become the bottleneck of decoding process because its nature is essentially sequential due to the large number of data dependencies. To decode each "bin" it is necessary to obtain information from previous decoded "bins" [KIM 06]. For hardware architectures implementation this fact causes a huge impact because four cycles are needed for decoding one bin and one SE can represented by several bins.

As mentioned in the introduction section, the main proposal founded in the literature to minimize the effects of the data dependency is the exploitation of speculative processing through the use of multi-decoding engines in parallel. [YU 05] proposes a decoding architecture with parallel engines arranged in the shape of a tree with two branches, one for regular engines and another for bypass engines. That proposal served as the basis for other works such as: [KIM 06], [BIN 07] and [YAN 07]. However, this organization results in great complexity to BAD, which is in the CABAD critical path. Figure 1 shows hierarchical decoder engines as proposed by [YU 05]. In next section is discussed the gain achieved increasing the parallelism of this architecture.

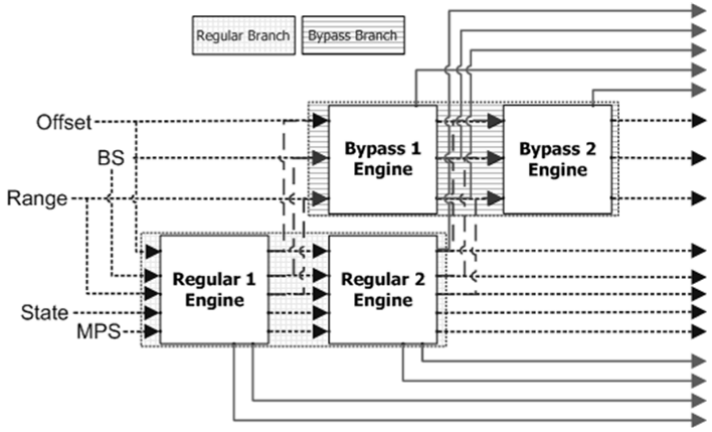


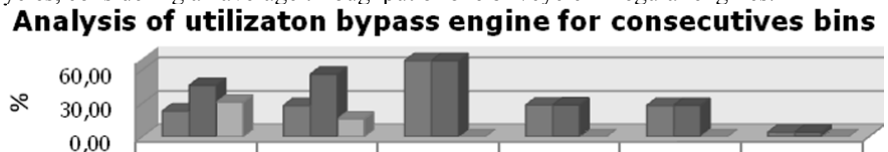
Fig. 1 – Tree of BAD engines in hierarchical arrangement with two branches.

3. Bitstream Flow Analysis

To explore the appropriate level of parallelism for decoder engines, a series of evaluations over bitstream generated by the reference software (JM) of H.264/AVC standard [SUR 08] was performed. To conduct these experiments were used the following digital video sequences: rush-hour, pedestrian_area, blue_sky, riverbed, sunflower and tractor. All these digital video sequences have resolution of HD1080p. The tool used for encoding was JM, version 10.2, with the following encoding parameters: Profile IDC = 77, Level IDC = 40, SymbolMode = CABAC. For all the video sequences, seven encoding configurations were performed varying the quantization parameters of QPISlice and QPPSlice, the following value pairs were used: 0:0, 6:0, 12:6, 18:12, 24:18, 28:20 e 36:26, resulting in 42 different encoded bitstreams.

To find the average of bypass decoder engine utilization for two consecutive decoding bins changes were implemented in the JM10.2 decoder to collect statistical information about usage of bypass engines for every SE kind. The implemented modification collects information on how much times each kind of engine is triggered to each SE type, how often bypass the engines is used in consecutive order for the same SE and how many times more than four consecutive bins are processed via bypass engine.

Analyzing the results, is possible to note that only two types of SEs present significant use of bypass coding engine. So SE are fetched in only three classes, which are: 1) coefficients transformed (COEF); 2) Component of the motion vector differential (MVD); and 3) Other SEs. The results were summarized in Figure 2, where averages for all digital video sequences are shown. On average, SE COEF and MVD match for 69.06% of all occurrences of SEs in the bitstream, but these two SE kinds produce approximately 84.06% of all the bins. Moreover, COEF and MVD together use the bypass engine to produce, on average, 68.20% of their bins. The rate of bypass bins processed in consecutive way corresponds approximately to 28.39% of the total. However, we observed that on average 28% of times that the bypass engine is performed, it produces 4 or more consecutive bins. Thus, the use of four bypass engines in parallel can provide a reduction to 3.98% in the total number of cycles, considering an average throughput of one bin/cycle in regular engines.



	SE occurs %	Bins %	Bins in Bypass %	Consective Bypass %	Bypass consective >= 4 %	Total gain with 4 bypass engines %
COEF	23,00	27,98	68,14	28,34	27,97	3,97
MVD	46,06	56,08	68,24	28,43	28,09	3,99
Others	30,94	15,94	0,00	0,00	0,00	0,00

Fig. 2 – Analysis of bypass engines utilization and gain with 4 bypass engines over two.

4. Designed Architecture for BAD Engines and Results

This work proposal combines the architectural model presented in [YU 05], as showed Figure 1, with an extension of decoding tree in bypass branch, from 2 to 4 engines in parallel. In this way, the proposed solution can process up to 4 bypass bins, or 2 regular bins and 1 bypass bin, or 2 regular bins, or 1 regular and 2 bypass bins without increasing the critical path. The main datapath is composed by five modules organized into four pipeline stages. The Figure 3 illustrates the block diagram of the proposed architecture. In BAD engines, the proposed design is very similar to that shown in Figure 1. The main difference resides in branch bypass, because four parallel bypass engines are employed. The architecture changes related to Figure 1 are shown in Figure 4.

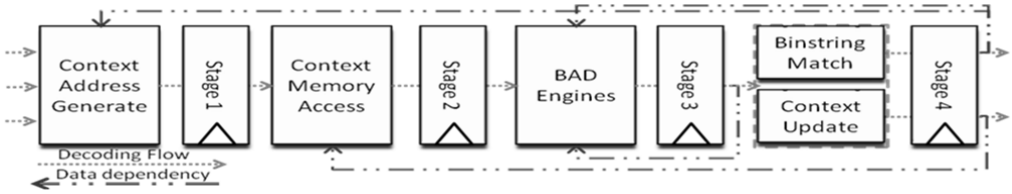


Fig. 3 – CABAD main block diagram showing decode flow and data dependencies.

The developed architecture was described in VHDL and synthesized to a Xilinx Visrtex2-PRO FPGA. The Xilinx XUPX2CPVP30 device was used for the synthesis. The Xilinx ISE 8.1 was used for the architecture synthesis and validation. The maximum achieved frequency was 103.4 MHz. The comparison between the synthesis results obtained to BAD architectures with 2 and 4 bypass engines are shown in Table 1.

Tab.1 – Synthesis results for BAD with 2 and 4 bypass engines.

Resource	BAD with 2 bypass engines	BAD with 4 bypass engines
Slices	788	824
LUTs	1446	1503
Frequency	103.5	103.4

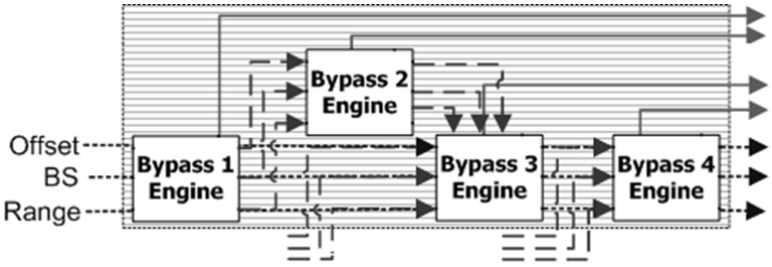


Fig. 4 – BAD block diagram with architecture changes inside bypass branch.

5. Conclusions

This work presents the challenges of the high efficiency implementation for BAD engines and proposes a new architecture based on an analysis of bitstream flow and common approaches founded in literature. The new architecture is more efficient in relation to [YU 05] because may deliver up to 4 bins by cycle. The developed work demonstrates that is possible to discover new alternatives to CABAD constrain thought analysis of bitstream flow. As future work, we plan to extend these experiments to evaluate the efficiency of regular engines branch to obtain greater throughput to complete system.

6. References

[BIN 07] BINGBO, Li; DING, Zhang; JIAN, Fang; LIANGHAO, Wang; MING, Zhang. **A high-performance VLSI architecture for CABAC decoding in H.264/AVC**. In: ASICON '07. 7th International Conference on, pp. 790-793, October 2007.

[EEC 06] EECKHAUT, Hendrik; et al. **Optimizing the critical loop in the H.264/AVC CABAC decoder**. In: Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on. December 2006.

[ITU 05] ITU, INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.264 Advanced video coding for generic audiovisual services**. 2005.

[HA 05] HA, Victor H. S.; et al. **Real-time MPEG-4 AVC/H.264 CABAC Entropy Coder**. In: IEEE International Conference in Consumer Electronics, Suwon, South Korea. January, 2005.

[KIM 06] KIM, Chung-Hyo; PARK, In-Cheol. **High speed decoding of context-based adaptive binary arithmetic codes using most probable symbol prediction**. In: Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on. May 2006.

[LE 06] LE, Thinh M.; et al. **System-on-Chip Design Methodology for Statistical Coder**. In: IEEE 17th International Workshop on Rapid System Prototyping, Singapore. June, 2006.

[MAR 03] MARPE, Devlet. **Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard**. In: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, Nº 7, July 2003.

[OSO 05] OSORIO, Roberto; BRUGUERA, Javier D. **A New Architecture for fast Arithmetic Coding in H.264 Advanced Video Codec**. In: IEEE 8th Euromicro Conference on Digital System Design, Porto, Portugal. August, 2005.

[OSO 06] OSORIO, Roberto; BRUGUERA, Javier D. **High-Throughput Architecture for H.264/AVC CABAC Compression System**. In: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 16, Nº 11, November 2006.

[SHO 05] SHOJANIA, Hassan; SUDHARSANAN, Subramania. **A VLSI Architecture for High Performance CABAC Encoding**. In: IEEE Visual Communications and Image Processing, pp. 1444-1454, Beijing, China. July, 2005.

[SUR 08] SUHRING, Karsten. **H.264/AVC Reference Software**. In: Fraunhofer Heinrich-Hertz-Institute. Available in: http://iphome.hhi.de/suehring/tml/download/. [Accessed: March 2008].

[YAN 06] YANG, Yao-Chang; LIN, Chien-Chang; CHANG, Hsui-Cheng; SU, Ching-Lung; and GUO, Jiun-In. **A High Throughput VLSI Architecture Design for H.264 Context-Based Adaptive Binary Arithmetic Decoding With Look Ahead Parsing**. In: (ICME) Multimedia and Expo, 2006 IEEE International Conference on, pp. 357-360, July 2006.

[YU 05] YU, Wei; HE, Yun. **A High Performance CABAC Decoding Architecture**. In: IEEE Transactions on Consumer Electronics, Vol. 51, pp. 1352-1359, No. 4, November 2005.

[WEI 03a] WEIGAND, Thomas; et al. **Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14 496-10 AVC**. In: Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-TSG16/Q.6, March 2003.

[WEI 03b] WEIGAND, Thomas; et al. **Overview of the H.264/AVC Video Coding Standard**. In: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, pp. 560-576, Nº 7, July 2003.

CAD TOOLS II

Cellgen - An Automatic Cell Synthesis Tool

Adriel Ziesemer Jr., Ricardo Reis
{amziesemerj,reis}@inf.ufrgs.br

Grupo de Microeletrônica (GME)
Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15064 – CEP 91501-970 – Porto Alegre – RS – Brasil

Abstract

This paper presents a cell synthesis tool that makes it possible to automatically produce cell layouts from its transistor level netlist description in SPICE format. The tool generates the cells under a linear matrix (1D) similar layout style and is able to support unrestricted circuit structures, continuous transistor sizing and folding. It features a transistor placement algorithm for width minimization that aims the reduction of the number of diffusion gaps and the total wirelenght of the internal connections. The circuit nets are routed using a negotiation-based algorithm, and an Integer Linear Programming (ILP) solver is used to compaction. The layouts were compared to equivalent cells from a standard cell library and the results shown that our realizations presented only about 14% of area overhead.

1. Introduction

With the constant increase in complexity of the VLSI circuits, layout design is becoming more complex, error-prone and time-consuming. Current circuits that require an efficient area and performance realization, often employ cell-based methodologies that need the generation of large cell libraries (with different logics and drive strengths) whose layouts must be optimized. To address this problem, cells synthesis tools are used to quickly generate physical layouts for a given transistor network and accordingly to the design rules and some constrains specified by the designer.

Uehara [UEH 81] was the first to propose a method and a general layout style (called linear matrix) for width minimization of static dual series-parallel cells. Methods based on the linear matrix style are also known as one-dimensional (1D) since transistors are placed exclusively in one direction. Since then, several transistor placement algorithms for the 1D style were proposed like [HSI 90] and [GUP 96]. Most the existing algorithms addresses exclusively the problem of generate dual static CMOS cells and just a few can handle efficiently cells with arbitrary networks and unequal number of P and N transistors [IIZ 04] as the case of the pass transistor logic (PTL) family. Moreover, almost none of the previously methods is aware about the routing complexity inside the cell producing sub-optimal results.

To solve this problem, we developed a new cell synthesis tool called Cellgen that is able to generate instant layouts of CMOS cells from its transistor-level netlist description. Details are presented in the next sections.

2. Cellgen Flow

The tool receive as input a file containing a SPICE netlist of the cells (with their respective and individually sized transistors and interconnections), a configuration file (which defines the layout topology and control parameters to the generator), and a technology file which contains a description of the design rules.

The design flow objective is, for a given transistor network, to place and route the transistors using the proposed layout style in such a way that the cell width and interconnections length are minimized. At the end, the circuit is compacted to produce a design error-free layout in CIF, GDSII and LEF formats.

Layout Style

While the 1D style is well defined in [UEH 81], our layout style have some modifications to better support recent fabrication technologies and adequate itself to produce layouts in the standard cell format. An illustration of our layout style is shown in Figure 1. From this model, the following assumptions were defined:

- Support to unrestricted transistor structure and individually sized transistors;
- Transistors placed into two parallel horizontal rows, for the PMOS and NMOS transistors;
- Intra-cell routing made exclusively with: polysilicon , metal 1 and diffusion;
- Internal tracks with adjustable size between the P/N regions for polysilicon and metal 1 routing;
- Additional tracks over the P/N diffusion regions for metal routing over the active area;
- Supply rows in the top and bottom limits of the cell using metal 1 for VDD and GND connections;
- Ties placed under the supply rows in the cell boundary and aligned to the routing grid;
- Standard cell library standard with input/output ports and cell boundaries aligned to the routing grid;
- Cell height and size/position of the internal track provided by the configuration file;
- No re-ordering in the transistor structure;

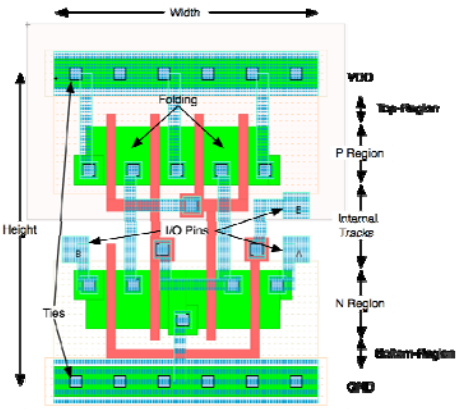


Fig. 1 - Layout style

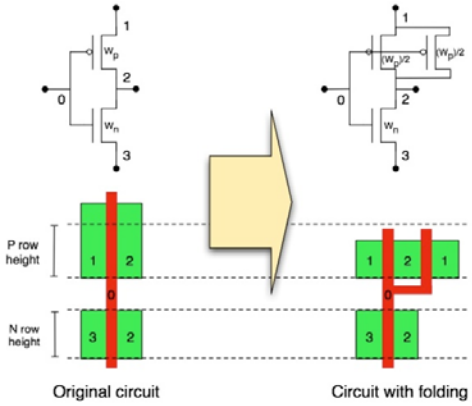


Fig. 2 - Execution of the folding algorithm

As the height of the cell is fixed, the maximum size of the diffusion rows is given by the space available between the internal tracks and the ties/supply row. Polysilicon routing in the top/bottom regions is just allowed in case of there is enough space available.

Folding

Transistor sizing is essential to produce high performance circuits. Layouts produced in the 1D layout style with different sized transistors tend to waste area since the height of each diffusion row is adjusted accordingly to its tallest transistor. To solve this problem, one of the most used methods is the transistor folding. It consists of break bigger transistors into smaller ones connected in parallel to keep short the cell height, at the expense of a little increase in the cell width.

In our approach we addressed the dynamic placement with static folding problem. Given the diffusion rows limits, we fold the transistors by directly modifying the cell netlist, creating new transistors in parallel, before the execution of the placement step.

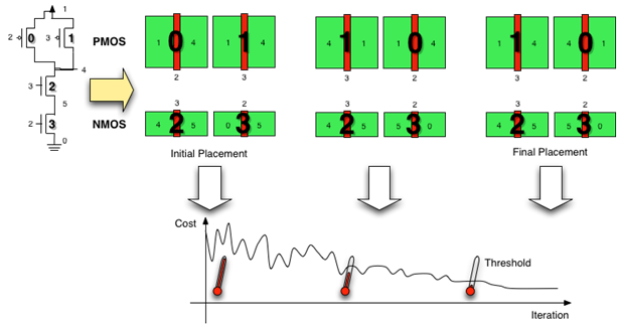
Placement

The aim of the placement step is to find out a transistor ordering that minimizes some desired function. In our approach, we implemented transistor placement by using the Threshold Accept (TA) algorithm. We were able to find the optimal or very nearly optimal result to practically all our test cases in a feasible time. An illustration of a possible execution of the TA algorithm is shown in Figure 3a.

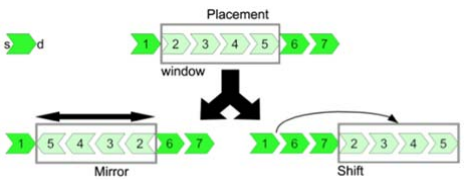
Given a folded transistor-level netlist, we create two different lists of P and N transistors and make P/N pairs accordingly to its absolute position in both lists. Eventually unpaired transistors are associated with new blank elements, which are created in the same position in the opposite list.

The minimization algorithm is implemented by creating a perturbation and a cost procedure over this structure. The perturbation function performs transistor movements in both lists to explore the solution space while the cost function returns a score for each partial solution. Smaller scores lead to a better solution.

As first step in our approach, an initial random placement without regard to the optimal placement solution is performed. Subsequently, a perturbation function incrementally modifies the initial placement by moving a set of contiguous transistors over the P/N lists at each iteration as shown in Figure 3b. Prior to performing the move operation, a window and a move type are selected. The first type shifts the selected window inside the list. The other inverts the transistor ordering inside the window as well as its orientation. All parameters are randomly selected: window position/size, move type and list that will experience the perturbations.



(a) Threshold Accept



(b) Perturbation function

Fig. 3 - Transistor placement algorithm

The cost function determines the quality of the solutions along the iterations. A good move provided by the perturbation function gives a better score while a bad move relaxes this constraint. To measure the cost, several quality factors were implemented.

The first one is the gate mismatch (W_{gm}). It counts the number of transistors with different gate signals located in each P/N pair. These transistors, when placed together, tend to minimize the routing complexity inside the cell and have better electrical characteristics than when placed far one from each other.

The second one is the total number of diffusion gaps (W_g) in both lists. Gap is the space between two non-continuous diffusion regions. It inserts extra space in the cell width producing a shorter transistor density and for these reason its use is avoided.

The third measure is the number of elements (W_c) needed to transpose the given placement to the abstract cell representation. It counts as element each gate, diffusion and spaces inserted in the cell according to the proposed transistor ordering. It can also be referred as a width measure of the cell since the width is directly related to this value. This way, diffusion gaps inserted in the same place in both diffusions doesn't count twice and have an overall better score than when placed separately.

The latest quality factor is the wirelength cost (W_{wl}), which contributes to reduce the routing complexity and can improve the electrical characteristics of the cell. To calculate the cell wirelength, we measure the horizontal distance, in number of elements, from the beginning of each net until the end. The sum of the individual values obtained for each net of the cell, excluding VDD and GND, is the score.

Some of these metrics contribute more than others to the overall cell quality. The weighting is necessary to balance the contribution of each of these factors in the final cost. The weights used in our cost function was experimentally determined and the score is given by the following equation:

$$W = 4W_{gm} + 2W_g + 3W_c + W_{wl}$$

Routing and Ports Placement

In our approach, both cell routing and ports placement are solved together using the same algorithm.

As happen with standard cells, we reserved two layers for intra-cell routing: polysilicon and metal 1. We adopt a graph-based methodology because its versatility which allows routing in other cell regions like over the active areas.

Prior to the execution of the detailed routing algorithm, we create a structure with the representation of the cell components according to the 1D placement result. A list of the vertical cell components (called "elements") is created. Each element holds a P and N transistor part (source/drain, gate, or a blank indication - if there is no transistor associated), a diffusion gap indicative flag and a structure to store the nodes of the graph (in the points where can cross connections according to our layout style). At this point, the adjacent transistors that share some connection using diffusion are already represented. Two additional blank elements are inserted in the beginning and in the end of this list to allow routing in the transistor boundaries.

Next, a graph is created over this structure, as illustrated in Figure 3, and solved using a negotiation based router and a optimization step to find steiner points and reduce the wirelenght. To achieve a better performance, we set different weights to the graph edges according to its layer and position.

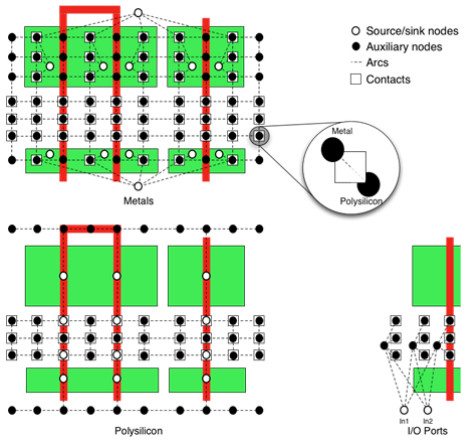
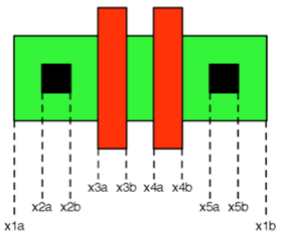


Fig. 3 - Routing graph




```
01: x2a - x1a >= EDFCT;
02: x2b - x2a =  WCT;
03: x3a - x2b >= SCTP1;
04: x3b - x3a =  WP1;
05: x4a - x3b >= SP1P1
06: x4b - x4a =  WP1;
07: x5a - x4b >= SCTP1;
08: x5b - x5a =  WCT;
09: x1b - x5b >= EDFCT;
* Width
10: x1b = Width;
* Diffusion area
11: x5a – x2b = Diff;
*Minimization function
12: min: Width + Diff;
```

Fig. 4 - Layout compaction using ILP

Compaction

Compaction is used to produce the final layout, according to the result provided by the placement/routing steps and regarding the technology design rules. We used an ILP (Integer Linear Programming) solver to

perform compaction in the X and Y direction. A set of equations representing the layout geometries is created, as illustrated in Figure 4, and the solver is called to find a feasible solution that minimizes some constraints like cell width and diffusion area.

To achieve a better performance, we introduced other minimization functions with parametrized weights. In our tests, we put the cell width constraint with the biggest weight followed by the width of the P/N diffusion islands, size of the polysilicon doglegs and the size of the metal doglegs.

Most standard-cell place and route tools use grid routers. For this reason, restrictions to place the I/O ports and the cell boundaries over the routing and placement grid are inserted during this step.

3. Results

The results were produced using a PowerPC G5 workstation with 1Gb of RAM. The cells netlist used to produce the layouts in our tests were extracted from the original standard cells in a 0.35um technology. Table 1 show the results obtained.

Tab. 1 - Cell area comparison between Cellgen and Standard Cell

	Std-cell (Height=13um)			**Cellgen (Height=12.6um)**				**Gain**
Cell	**# Tr.**	**# GAPs**	**Width**	**# Tr. ***	**# GAPs**	**Runtime**	**Width**	**Area (%)**
INV0	2	0	2.8um	2	0	0.1s	2.8um	3.08
NAND21	4	0	4.2um	4	0	0.3s	4.2um	3.08
AOI210	6	0	5.6um	6	0	1.5s	7um	-21.15
AOI312	8	0	8.4um	15	1	10.7s	12.6um	-45.38
NAND40	8	0	7um	8	0	2.3s	7um	3.08
OAI222	8	0	7um	13	1	7.1s	12.6um	-74.46
NOR33	9	0	9.8um	15	2	7.6s	14um	-38.46
OAI312	10	0	8.4um	15	1	29.8s	12.6um	-45.38
XOR20	10	1	9.8um	10	1	4.5s	9.8um	3.08
MUX21	12	0	8.4um	12	0	7.3s	9.8um	-13.08
ADD22	14	2	11.2um	14	3	11.9s	14um	-21.15
XNR30	20	4	15.4um	20	3	13.8s	16.8um	-5.73
DF1	26	7	21um	26	4	103.4s	23.8um	-9.85
MUX41	26	4	18.2um	26	4	55.3s	21um	-11.83
ADD31	28	4	21um	28	2	132.9s	21um	3.08
ADD32	28	4	21um	28	2	241.8s	21um	3.08
XOR41	30	9	21um	30	4	52.7s	25.2um	-16.31
JK1	34	11	26.6um	34	6	229.2s	30.8um	-12.23
TOTAL	**257**	**39**	**226.8um**	**280**	**30**	**808.8s**	**266um**	**-13.68**

* after folding

4. Conclusion

We presented a new cell synthesis tool that is capable of support circuits with unrestricted transistor structure by using Threshold Accept to place the transistors in a one-dimensional layout style. A negotiation based router and ILP were also successfully applied to produce the full layout of the cells. Our results demonstrate that our methodology is very efficient in area producing layouts only about 14% bigger than the equivalent standard cell and that it is computationally feasible for a wide range of cells.

Greater results should be obtained comparing complex cells produced by our tool with the equivalent standard cell solution that frequently need two or more cells to implement the same function.

5. References

[DUE 90] DUECK, G. et al., **Threshold accepting: A general purpose optimization algorithm appear superior to simulated annealing.** Journal of Computational Physics, 1990.

[GUP 96] GUPTA, A.; THE, S. C.; HAYES, J. P., **XPRESS: A Cell Layout Generator with Integrated Transistor Folding.** Proceedings of the 1996 European Design and Test Conference, Washington, DC, USA. Anais. IEEE Computer Society, 1996. p. 393.

[HSI 90] HSIEH, Y. C.; HWANG, C. Y.; LIN, Y. L.; HSU, Y. C., **LiB: a cell layout generator.** In: DAC '90: PROCEEDINGS OF THE 27TH ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, 1990, New York, NY, USA. Anais. ACM Press, 1990. p. 474–479.

[IIZ 04] IIZUCA, T.; IKEDA, M.; ASADA, K., **High speed layout synthesis for minimum-width CMOS logic cells via Boolean satisfiability.** ASP-DAC '04: Proceedings of the 2004 conference on Asia South Pacific design automation, p. 149-54, 2004.

[MCM 95] MCMURCHIE, L.; Ebeling C., **PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs.** Field-Programmable Gate Arrays, 1995. FPGA '95. Proceedings of the Third International ACM Symposium, p. 111- 117, 1995.

[UEH 81] UEHARA, T.; Cleemput, W., **Optimal Layout of CMOS Functional Arrays.** In: IEEE Transactions on Computers, Vol. C-30, No. 5, May 1981, p. 305-312.

Partitioning in the PARROT Flow for Physical Synthesis

Samuel Nascimento Pagliarini, Glauco Borges Valim dos Santos, Ricardo Reis
{snpagliarini,gbvsantos,reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul

Abstract

This paper presents an ongoing work which comprises the insertion of a Partitioning step in the PARROT Flow for Physical Synthesis. A partitioning step was inserted to enclose a simulated annealing based-placement step of the flow. Runtime and wirelength data were collected to compare the results within and without the partitioning step on a group of circuits. Also, using these same circuits, we have collected area and wirelength to analyze the cost of the partitioning step versus the benefits in runtime and routability. Besides that, the different ways to create the partitions and manage the connections between them will be discussed.

1. Introduction

The reason we started to implement this partitioning tool was because of the existence of a complete circuit design flow, including placement, layout generation and routing, developed at our university. This flow is called PARROT Flow [LAZ 2003a] [LAZ 2003b]. However, currently this flow is extremely time demanding because of the simulated annealing (S.A.) approach in the placement tool, called MangoParrot [HEN 2002]. Because of its characteristics, the placement tool requires some tuning in order to perform correctly. The most important parameter to set is the *num_reps_per_cell*, which controls the number of iterations in the inner loop of the S.A. technique. In a previous study it was determined that this parameter should be set as the number of cells to the second power, so the circuit could have an acceptable wirelength/runtime tradeoff.

When the number of cells in the circuits starts to increase, the demanding time to place them by a S.A. approach becomes way too large. Altough an analytical placement tool may be used to reduce the runtime, simulated annealing based techniques are probably the best ones when it comes to wirelength reduction[GER 99]. This is why a partitioning tool was implemented: by splitting the circuit into multiple parts, it is possible to keep the number of iterations under control so that the placement tool runtime is considerably decreased, therefore, diminishing the placement runtime problem.

Some problems arise when we must decide how to split the circuit and, after placing each partition, we must bind then together again in a certain way that the connectivity of the circuit is optimized. This problems will be presented in the next sections when the virtual pin representation will be introduced.

From the placement point of view it really does not matter how the partitioning is done, since the goal is only run time improvement. However, that is not true for the routing since it is possible to mess with all the connectivity of the circuit by making wrong decisions on which cell belong to which partition. The main idea is that a group of cells that are strongly connected should be put in the same partition, in a certain way that the connections between partitions should be minimized, so the circuit characteristics are kept close to the ones in the original circuit.

In graph theory, a cut is a partition of the vertices of a graph into two sets. The size of a cut is the total number of edges connecting two partitions, and a cut is minimal if the size of the cut is not larger than the size of any other cut. The idea of finding a minimal cut in a graph comprises the approach of several partitioning tools available.

Section two presents the current state of this work. In section three some preliminary results are shown and discussed. And finally, the conclusions and future work are hold in section four.

2. Current state of the implementation

Once the problem is identified we can make use of a third party tool to create the partitions. The chosen tool is called hmetis [KAR 2000a] and it can split very large graphs very fast, by using algorithms based on multilevel hypergraph partitioning schemes[KAR 2000b]. In the circuits that we performed experiments and collected our data this time is always at most 0,00002% of the total placement time without partitioning, which helps minimizing the placement runtime.

We must provide a file containing a weighted graph description that represents the input circuit for the hmetis partitioner. The developed tool, called **weezer**, is responsible for the generation of this file and the partitioning/placement process management. For later comparison with other tools and simplicity at the moment, it is desired that each partition have the same circuit area or something close to that. To reach this requirement we provide hmetis with a parameter defining the maximum acceptable difference between partitions areas. This parameter, called the unbalance factor, is currently set as 1%, the lowest possible value. Since weezer itself can not estimate each cell area, it uses the results from another external tool called CellSE [ZIE 2006] to estimate the areas and later put these values in the hmetis graph description file as weights.

Using a last parameter which specifies the desired number of partitions we are ready to start the partitioning and weezer immediately makes a system call and runs hmetis. After the partitioning is done it writes a file where each cell from the original circuit is given a partition number in which it belongs. Weezer reads that file and creates a new circuit description file (CDF) for each partition, corresponding to different placement inputs. These new files contain cells, nets, regular pins and virtual pins.

The nets that connect two or more cells from the same partition are written to the corresponding partition CDF. The regular pins that are used by at least one cell from the partition are also written. The virtual pins are created when a net that connects two or more cells from different partitions is found. The virtual pin representation is used to maintain and evaluate the partitions connectivity.

It is important to notice that the number of virtual pins is equal to the minimal cut (or double if you count it once in each partition) only if we split the circuit in two partitions. When we partitionate a circuit using more than two partitions the sum of the virtual pins is equal to the hyperedge cut(sum of minimal cuts from all internal bisections that hmetis made).

Before we do the placements we must estimate the circuit area and since all placements are row based, this means we must estimate how many rows each partition will occupy. The number of rows is the way we control how the circuit aspect ratio will be. Currently we are aiming to keep the circuits with the width the same as the height, in a squared shape. If necessary or desired this shape can be easily changed. This feature will also be useful if a future floorplanning strategy provides a desired area for the weezer tool. To estimate the number of rows we roughly add up the areas from all the cells and divide it by the strip height, which will give us an average slightly bigger width, in comparison with the height.

At this stage weezer is ready to perform the placement of all individual partitions which is done by calling the MangoParrot placer for each one.

Since MangoParrot is a wirelength driven placer all the circuits placed with it looks like the left side of fig. 1, where the center is heavy populated. When weezer is used the circuit will look like the right side of fig. 1, where the congestion is more spread among the hole circuit area. The image shows that the rounded shapes may create some areas without cells, which could mess up the entire placement/partitioning process since our flow deals with relative positioning. So, when weezer is binding the partitions it detects these blank spaces and inserts floating inverters to mantain the circuit shape. These floating inverters act as filler cells and help the circuit to be more routable.

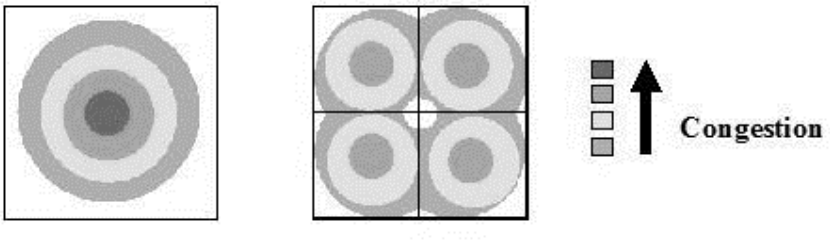


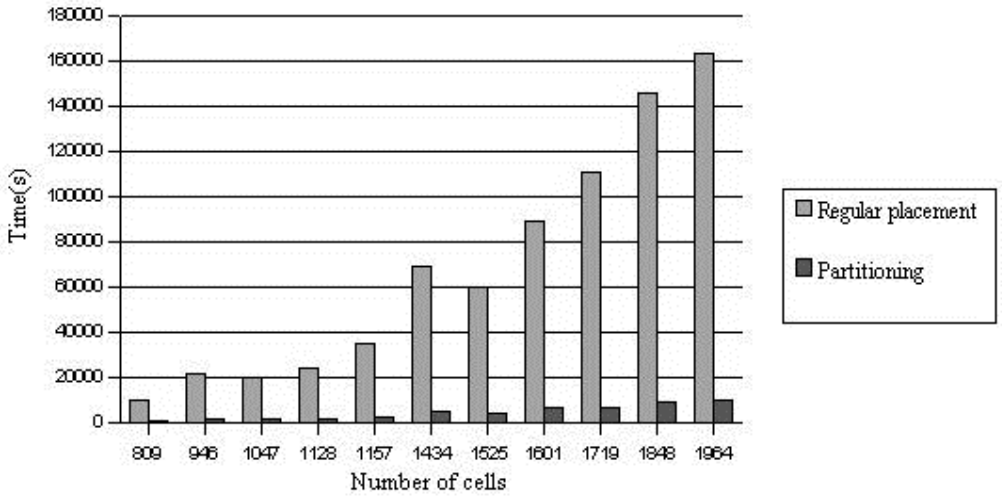
Fig. 1 – Congestion analisys

3. Preliminary Results

In order to observe the impact of the partitioning step in the placement runtime reduction we performed several placements using a variety of circuits, mostly obtained from ISCAS98 benchmarks. The results are shown in fig. 2. The weezer results were obtained using a 2x2 partitioning grid. In the *X* axis we have the number of cells and, since that number is obtained from real existing circuits, the axis is not in scale. The *Y* axis represents the time in seconds and it is in scale.

Since the *X* axis is not in scale, the behavior of the regular placement curve is a bit twisted but it is close to quadratic. The same way is the weezer curve, in which the actual behavior is close to linear. If we aim to change the curve behavior into really linear we can define a maximum number of cells allowed per partition. Let us say that the chosen number is 200 cells and it takes 250 seconds to place this number of cells. So then all placements maximum runtime can be calculated by ((*N* div 200)+1)*250 seconds, where *N* is the total number of cells for a particular circuit.

One important result that is not shown in fig. 2 is the weezer runtime, that is the sum of CellSE, hmetis and weezer itself runtimes, which for the circuits placed in fig. 2 ranges from 0,99s to 1,92s, representing less than 0,001% of the sum of the placement time for all four partitions.



. Fig. 2 – Mango versus Weezer

Without the partitioning the current flow was not able to benefit from the use of parallelization. Now several partitions placements can be made in parallel which can minimize the runtime even more, since each partition placement is independent from each other. However, all the results shown in this paper were made in the same single core machine, an ATHLON XP 2000+.

4. Conclusions and Future work

The main conclusion at the moment is that partitioning allows simulated annealing based placements to run in reasonable time. Since simulated annealing strategies may achieve a wirelength very close to optimal, we must now find out if partitioning does not affect that. However, even a wirelength increase, due to congestion distribution, can improve the flow convergence by improving routability.

It is interesting to observe that the placement itself does not require the creation of virtual pins but its final result may be improved if they are properly placed in each block boundary, especially for wirelength reduction.

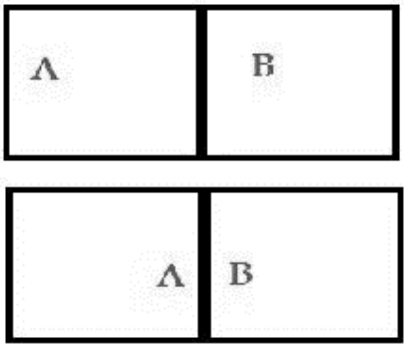


Fig. 3 – Virtual pins impact on the placement of multiple partions

In the top of fig. 3 we show an example of how two connected cells, A and B, may be placed without virtual pins. In the bottom of fig.3 we illustrate how the cells can get much closer to each other if a virtual pin is properly placed in the boundary between the two regions, which helps the wirelength reduction. The net that connects A and B is likely to be much smaller since the sub-nets connecting both A and B to their virtual pins is also driven for smaller wirelengths along the iterations of the traditional S.A. looping. In another words, using virtual pins is an affordable way to decrease the additional wirelength without adding more complexity to the simulated annealing approach.

When it comes to routability, all our study is based on wirelength and area analisys, and how routability decreases may be diminished if the circuit congestion is spread due to the partitioning strategy. However, in the current stage of research we have only focused on strategies to place the pins and have kept the partitioning as simple as possible.

The first idea that we implemented was very simple: all the virtual pins were equally spread in all four directions. This idea, as expected, proved to be very negligent with wirelength since the cells could be attracted

to any direction. Since the pins were equally spread so were the cells, and that could sometimes keep the routability acceptable. Nevertheless, not enough tests were performed in order to generate sufficient data or conclusive results.

The consecutive attempt was to place the virtual pins with some kind of order: we applied a very simple rule in which a virtual pin position will be the closest possible to any other partition that contains the same net. This is clearly not optimal, but still considerably reasonable and fast to implement and run. It is reasonable because once close to one of the other possible partitions, in the worst case it will be just shifted inside the bounding box formed by this chosen partition and the possible other ones. Since we are working with regular-side partitioning grids (2x2, 3x3, etc.) sometimes the closest partition can be in a direction that we can not set, like northeast. In this case a direction is randomly chosen between north and east. We are currently performing experiments to evaluate this policy.

After these tests are done, and if the results are satisfactory, that algorithm can be refined to match the optimal direction of the virtual pins, in the cases when nets have terminals in more than two partitions.

These ideas are either under development or testing, so the results at the moment are not very conclusive. This paper is mostly based on ongoing research altough we believe that all these ideas deserve a closer look since all of then look very promising.

5. References

[GER 99] GEREZ, S.H. **Algorithms for VLSI Design Automation**. Chichester: John Wiley, 1999.

[HEN 2002] HENTSCHKE, Renato Fernandes. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. Master Dissertation, Universidade Federal do Rio grande do Sul, 2002.

[HEN 2003] HENTSCHKE, Renato Fernandes; REIS, Ricardo. **Improving Simulated Annealing Placement by Applying Random and Greedy Mixed Perturbations**. SBCCI, São Paulo, 2003.

[KAR 2000a] KARYPIS, George; KUMAR, Vipin. **Multilevel k-way Hypergraph Partitioning**. VLSI Design, Vol. 11, No. 3, pp. 285 - 300, 2000.

[KAR 2000b] KARYPIS, George; KUMAR, Vipin. **METIS - Family of Multilevel Partitioning Algorithms.** http://glaros.dtc.umn.edu/gkhome/views/metis [viewed 24/03/2008].

[LAZ 2003a] LAZZARI, C.; DOMINGUES, C.; GÜNTZEL, J. and REIS, R.. **A New Macrocell Generation Strategy for Three Metal Layer CMOS Technologies**. VLSI-SoC, Germany, Dec, 2003.

[LAZ 2003b] LAZZARI, Cristiano. **Parrot Punch**. http://www.inf.ufrgs.br/~clazz/parrotpunch [viewed 24/03/2008].

[ZIE 2006] ZIESEMER, Adriel Mota; SANTOS, Glauco Borges Valim dos; HENTSCHKE, Renato Fernandes; REIS, Ricardo Augusto da Luz. **Cell Size Estimation in an Automatic Layout Generation Flow**. 21th South Simposium on Microelectronics, Porto Alegre, 2006.

Design Methodology for Cell Library Testbench Circuit

S. Bavaresco[1], M. Lubaszewski[1], A. I. Reis[2], R. P. Ribas[1]

[1]Instituto de Informática – UFRGS, Porto Alegre, RS, Brazil
[2]Nangate Inc., Menlo Park, CA, USA
simoneb@inf.ufrgs.br, luba@ece.ufrgs.br, are@nangate.com, rpribas@inf.ufrgs.br,

Abstract

The use of automatically generated CMOS logic gates in standard cell IC design flow represents an attractive perspective for ASIC design quality improvement. Automatic cell generators can be considered as soft IPs and represent the key elements for the library-free technology mapping approach, for instance, already proposed in literature and now being adopted by the industry. This methodology leads to an IC design flow based on logic cells which are created on-the-fly by software. This work proposes a validation methodology to silicon prove the set of automatically designed logic cells. The validation covers the full functionality of the cells and provides timing and power consumption data useful to validate and fine tune cell data models derived from electrical characterization. This methodology allows having reliable data to be used in the performance estimation during the circuit design.

1. Introduction

Cell-based design is definitely the most applied approach in the ASIC market today. This design approach implies in re-using library cells to build more complex digital circuits. A typical standard cell design environment includes timing and power analysis, as well as automatic assembling of circuit layout through place and route tools. In standard cell libraries, three groups of cells co-exist: (1) inverters/buffers; (2) combinational cells and (3) sequential ones. Mainly due to the large number of different logic functions and driving strength options needed in typical designs, the largest of the three aforementioned groups is the set of combinational logic gates. The handcraft creation of standard cell libraries demands skilled designers and long development times, even when simply dealing with technology migration for the same set of cells. Each cell must be carefully designed and characterized for different input slopes, output loads and design corners. In practice, the high engineering costs of these tasks imposes a limitation on the number of available combinational cells in libraries.

As the technology mapping step in standard IC design flow is based on pre-characterized data of pre-designed cells, the ASIC design space and efficiency turns to be bounded by the library in use. The more cells and drive strength options are available, the larger are the possibilities to improve the circuit design. The enrichment of a library can be done by adding only new drive strengths [1], or through the addition of new functions with standard series/parallel implementation [2], or even with special transistor topologies [3-5]. New topologies can also be considered for in-place optimization (IPO), including in-context cell sizing [6]. The use of extended libraries leads to an optimized fit for particular applications. Due to this added flexibility to the design space, there is an increasing commercial interest for approaches that consider on-the-fly generation of cells, like [1,6-9]. These approaches are sometimes referred as library-free or as soft-cell or liquid library based.

The main drawback of library-free technology mapping technique is the use of such soft IPs or non-silicon-validated set of cells in the ASIC design. This fact makes conservative customers reluctant in adopting this design technique.

This work proposes a validation methodology to silicon-prove the set of automatically designed logic cells. This validation methodology covers the full functionality of the cells and provides timing and power consumption data useful to validate and fine tune cell data models derived from electrical characterization. Notice that, the circuit speed and consumption estimation tasks, during the design flow, are performed based on cell data from electrical characterization, available in a Liberty™ file, for instance. Indeed, the on-silicon testbench for soft libraries, proposed herein, allows checking whether data used in the performance estimation during the circuit design flow produces good prediction of silicon behavior.

In Section 2, the motivations for this work are further discussed. The basic combinational block of the testbench is described in Section 3. In Section 4, the circuit architecture is then presented and its operating modes are shown pointing out their goals. The conclusions are outlined in Section 6.

2. Problem Statement and Goal of Work

ASIC designs are usually bounded by the standard cell library in use. For this reason, there are a number of commercial and academic efforts that take advantage of on-the-fly creation of cells. The method proposed by DeDood *et al.* [1] creates new drive strengths for existing cells (from a starting cell library), in order to save

power and reduce delay in a target ASIC. A method that creates dedicated complex cells to reduce delay in applications needing high-speed ASICs is introduced in [6, 8]. Jones *et al.* [9] proposed a method to optimize a design by changing transistor sizes in the available cells and then redesigns the original library to accommodate these new cells. For advanced technology nodes, research initiatives by leading semiconductor companies are considering the importance of the target library as part of the design space [10]. The use of more complex gates can reduce the overall number of transistors and provide layouts that are less dense in transistors/mm^2 but denser in terms of logic/mm^2. This idea has been pointed as part of a regular layout solution for process variability [11]. The use of such complex gates can also bring advantage to the regular ASIC world. Indeed, several methods to generate efficient transistor networks [3,5,7,11] and to perform technology mapping targeting complex gates [2,4,6,8] have been recently proposed.

Conservative customers feel uncomfortable with the idea of using a 'non silicon-proven' library. The silicon proof should address two main issues. First, it has to guarantee that the models used during the synthesis flow represent adequately the final silicon performance. Second, it has to prove that the cells are reliable under the expected working conditions. This aspect guarantees that the cells present a sufficient number of contacts, do not have latch-up problems, and so on. Notice that this proof has to be done properly by library providers every time a new technology node is available.

An efficient approach to generate a testbench for testing a set of new cells, possibly created on-the-fly, should cover the following aspects:

(1) to ensure complete functionality test for the instantiated cells;
(2) to ensure coverage (instantiation) of all the cells to be tested;
(3) to allow the verification of the accuracy of the models used in the design process;
(4) to provide means to perform long and medium term reliability tests (needed for electromigration, sufficiency of contacts, NBTI degradation, etc) without additional equipment;
(5) to have a feasible number of cell instances compared to the set of cells to be tested.

In this work, a straightforward and efficient testbench methodology is proposed aiming the validation of an entire set of soft-cells in terms of logic and electrical behavior. The presented solution merges well-established design and test concepts to cope with the five aspects mentioned above. A specific combinational block is built to guarantee the logic coverage (aspect 1) of a sub-set of the cells to be validated, and to provide at the output the same bit vector received at its inputs, allowing thus to cascade long chains with these blocks. The use of several blocks allows to instantiate all the cells (aspect 2). The circuit architecture is then composed of such combinational chain in a ring configuration, synchronized by a register barrier. Both synchronous and asynchronous operating modes provide different features for the proposed goals. The ring configuration allows verifying the accuracy of the models, by comparing with the predicted circuit behavior (aspect 3). The oscillation BIST technique is also included in the circuit operation for a wide range of different paths [12], and allows medium and long term tests (aspect 4). In case of an eventual error, the circuit diagnostic is facilitated through an arrangement of multiplexers. Finally, as the ring oscillator is composed of a variety of cells, the number of instances is not very expensive compared to the initial set of cells (aspect 5), as shown in the overhead section.

From a business model point-of-view, the methodology presented herein is useful for the soft-library vendor and to the ASIC designer client. For the vendor, it is quite important to dispose of a physical testbench in order to guarantee the correctness of its EDA environment, as well as to verify the quality of the generated cells in terms of performance and reliability, including design-for-manufacturability issues [13]. This is essential for the continuous improvement of the library generation CAD tool. For the ASIC designer, a circuit that validates all distinct cells created on-the-fly to be used in a specific circuit provides means to exclude those errors on silicon due to the cell generators. If this test circuit is fabricated in the same die of the ASIC, it can act as a kind of 'certification circuit' for the soft-library, in different design corners and operating conditions. In this case, the overhead in terms of area and I/O pins is a compromise in fabricating together the test circuit and the ASIC, and the low cost approach presented here is very attractive.

3. Testbench: the Combinational Blocks

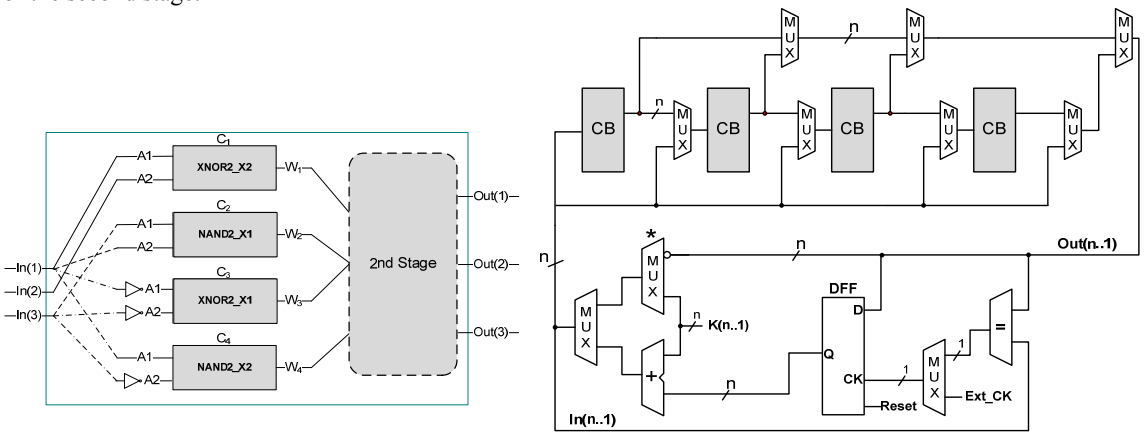
The combinational blocks are built in a way to guarantee the complete and correct logic behavior of all cells included in the soft-library under test. To attain this, each combinational block is composed by two sub-blocks or stages. The first one is built by cells in a single logic depth level, all of them excited by the (shared) primary inputs. These cells, in the first stage, are connected to the block inputs for full logic exercising. The output signals of the cells in the first stage are then used as inputs to the second stage, which recomputes the same bit vector applied at the first sub-block primary inputs. As a result, the primary inputs of the first sub-block are equal to the primary outputs of the second sub-block. The internal interface of the two sub-blocks can be viewed as an intermediate code for which primary inputs are translated and then recovered.

In the construction of the combinational blocks, the following requirements might be pointed out:

(1) all blocks present the same number of input and output nodes, which must be equal or higher than the biggest number of inputs in a single cell;
(2) input and output vectors have equal steady state logic values.
(3) every cell has to be instantiated at least once in the first stage of a combinational block;

(4) the total number of combinational blocks depend on the size of the cells set and also the quantity of cells necessary to compose the first stage of each block;
(5) the second stage of the combinational blocks is synthesized taking into account only the cells present in the soft-library to be validated.

To illustrate the construction of the first stage, consider a 3-input combinational block, as shown in Fig. 1a. The intermediate code composed of 'Wi' bits requires 2^3 distinct vectors, necessary to rebuild the 3-bits input vectors at the 3-bits block outputs. This requirement was not attained with the three first selected cells. Thus, at least one additional cell C_4 must be added to the first stage. The second stage re-creates the input vectors at the output of the block, from the intermediate codes represented at 'Wi'. The output signals of the first stage are used as inputs to the second stage. Since the length of the intermediate code can be larger than the length of I/O codes, some 'Wi' combinations will never occur. This way, *don't cares* may be used to optimize the synthesis of the second stage.



(a) (b)

Figure 1 – (a) 3-bits combinational block: example and (b) block diagram of the proposed circuit architecture.

4. Testbench: Overall Architecture and Operation Modes

The combinational blocks guarantee the functional validation of the entire set of logic cells if each cell has been instantiated at least once in the first stage of a block, and the 2^n input combinations are applied at 'n' inputs of each block. Once the block outputs reproduce the same input signals, they can then be arranged in different ways to create the circuit architecture. Long paths can be built, for instance, by cascading the combinational blocks in chain configuration. This way, the primary input values should be observed at the output of chain in the case of fault free behavior.

The global circuit architecture is presented in Fig. 1b. To provide a sequence of test vectors with minimum external intervention, the signals at the end of the chain are reconnected to the primary inputs. A register barrier, composed by D-type flip-flops, is added to the feedback path to avoid racing. An adder is available to increment the binary vector and make the circuit act as a counter to modify the feedback signals and provide the chain input variation. The adder can perform sums by 'K' allowing other than just a counting 1-by-1 operation, allowing thus different vector transitions that are important to check charging and discharging conditions at internal nodes intra- and inter-cells.

The basic architecture is slightly modified by adding a comparator and multiplexers to allow different operating modes which provide distinct forms of data evaluation. Notice that, the aim of this circuit is to validate the full functionality of the entire set of cells, as well as evaluate the accuracy of the electrical characterization values of the cells (timing and power dissipation data) through the correlation of the static timing analysis (STA) and power analysis with experimental measures.

The circuit operating modes to meet the five requirements described in Section 2 are:

a) **Synchronous mode** - In this mode, the register barrier is controlled by an external clock signal. The adder is used to increment the vector in the ring, acting as a synchronous '+K' counter. The right behavior of the counting sequence demonstrates the correct functionality of the combinational blocks and, consequently, the whole set of cells under test. The main benefit of the synchronous mode is the evaluation of the power dissipation, including its dynamic and static components. The external control of the clock signal imposes the frequency operation for switching, and the dynamic power consumption can be related to that. The static power, on the other hand, can be measured at low frequencies or even by using an external clock manually controlled. At each new input state in the chain the static consumption can be obtained since such power dissipation component depends strongly on the circuit steady state.
b) **Asynchronous mode**- In the asynchronous mode, or self-timed ring configuration, the clock signal of the flip-flops is provided by the comparator that checks whether or not the input vector In(n..1) has

already propagated to the end of the chain Out(n..1). When the same vector applied to the circuit inputs get to the end of the chain, the comparator switches from '0' to '1', clocking the register. The new data is stored in the register and passed to the adder. The adder increments the register output and applies the new vector to the chain. At this moment, since In(n..1) no longer equals Out(n..1), the comparator output is back to '0' and remains at this state until the new vector propagates through the whole chain of combinational blocks. If a cell is defective, the data at the end of the chain will not be equal to the data at the circuit inputs. This way, the comparator will not switch to '1' and the self-timed execution will stop. The self-checking property of the asynchronous mode makes it quite appropriate for functional cell verification with least external intervention.

c) **Oscillation BIST mode** - In synchronous or asynchronous mode, the same binary value of an i-index input is expected to re-appear at the corresponding i-index output of the testbench circuit. This property is ensured by construction of the combinational blocks. If the i_{th}-output is directly connected to the i_{th}-input, the i_{th}-path is kept in steady state, while closing the feedback loop. However, when the i_{th}-output is inverted before connecting to i_{th}-input, a negative polarity logic feedback occurs and the i_{th}-path oscillates. According to this principle, in oscillation mode the feedback loop is closed such that at least one of the primary inputs of the chain receives the negation of its previous value [12].

d) **Diagnosis mode** - To perform defect diagnosis, additional multiplexers are included in the circuit architecture, at no significant penalty in area, to select part of the combinational chain. Multiplexers at the inputs of each combinational block select the signal from the previous block or directly from the beginning of the chain, removing the influence of the previous blocks in the ring loop. Similarly, multiplexers at the output of the blocks send the data from the middle of the chain directly to its end. This way, the chain is easily reduced to a single block or even none, allowing in this case the verification of the counter and the register barrier without the influence of the combinational chain

5. Conclusions

The use of automatically generated CMOS logic gates in standard cell IC design flow represents an attractive perspective for ASIC design quality improvement. This methodology leads to an IC design flow based on logic cells which are created on-the-fly by software, which have not been previously validated on silicon yet, until the target ASIC is prototyped. This fact makes conservative customers reluctant in adopting this design technique. This work proposed a validation methodology to silicon prove the set of automatically designed logic cells. The validation covers the full functionality of the cells and provides means to compare cell data models derived from electrical characterization against silicon performance. Self-testing modes allow performing medium and long term reliability certification without depending on ATE, which allows to reduce costs and to increase the number of samples under test.

6. References

[1] P. de Dood, B. Lee and D. Albers. "Optimization of circuit designs using a continuous spectrum of library cells". US Patent 7107551. 2003.

[2] S. Gavrilov, A. Glebov, S. Pullela, S.C. Moore, A. Dharchoudhury, R. Panda, G. Vijayan and D. T. Blaauw, "Library-less synthesis for static CMOS combinational logic circuits", ICCAD 1997, pp.658-662.

[3] D. Kagaris and T. Haniotakis. "Methodology for transistor-efficient supergate design". IEEE Trans. VLSI, Apr. 2007, vol. 15, no. 4, pp. 488-492.

[4] F. S. Marques, L. S. Rosa Jr, R. P. Ribas, S. S. Sapatnekar and A. I. Reis. "DAG based library-free technology mapping". GLSVLSI 2007, pp. 293-298.

[5] F. R. Schneider, R. P. Ribas, S. S. Sapatnekar and A. I. Reis, "Exact lower bound for the number of switches in series to implement a combinational logic cell". ICCD 2005, pp.357-362.

[6] R. Roy, D. Bhattacharya and V. Boppana. "Transistor-level optimization of digital designs with flex cells". IEEE Computer, Feb. 2005, vol. 38, no. 2, pp. 53- 61.

[7] A. I. Reis, F. R. Schneider and R. P. Ribas. "Methods of deriving switch networks". US Patent Application #20070214439.

[8] D.Bhattacharya, V.Boppana, R.Roy and J.Roy. "Method for automated design of integrated circuits with targeted quality objectives using dynamically generated building blocks". US Patent 7,225,423. May 29, 2007.

[9] L.G.Jones, D.T.Blaauw, R.L.Maziasz and M. Guruswamy. "Method and apparatus for designing an integrated circuit". US Patent 5,666,288. Sep. 9, 1997.

[10] EEtimes. "NXP adopts Nangate's tool for 32-nm research". By: A.-F.Pele, EE Times Europe, 02/19/2008, available on: http://www.eetimes.eu/ france/206800280?cid=RSSfeed_eetimesEU_france.

[11] H. Yoshida, M. Ikeda, and K. Asada. "A structural approach for transistor circuit synthesis". IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E89-A, 12 (Dec. 2006), 3529-3537.

[12] E. Arabi , H.n Ihs I, C. Dufaza and B. Kaminska, "Digital oscillation-test method for delay and stuck-at fault testing of digital circuits", ITC 1998, pp. 91-100.

[13] J. Kibarian, "Overcoming the process variability crisis via proactive DFM". Keynote talk at ICCAD 2007.

Standard Cell Design Flow for H.264 Luma Motion Compensation Architecture

[1]Thaísa Silva, [1]Érico Sawabe, [2]Luciano Agostini, [1]Altamiro Susin, [1]Sergio Bampi
{tlsilva, eksawabe, bampi}@inf.ufrgs.br, Altamiro.Susin@ufrgs.br, agostini@ufpel.edu.br

[1]Grupo de Microeletrônica (GME) – Instituto de Informática – UFRGS
[2]Grupo de Arquiteturas e Circuitos Integrados (GACI) – DINFO – UFPel

Abstract

This paper presents the standard cell design flow for H.264 Luma Motion Compensation. The standard cell layout generation used initially the RTL Compiler tool and from the files generated by this tool, the Encounter tool was used to achieve the digital flow, both tools used are from Cadence. The technology used in this flow was IBM 0.18 um. The standard cell design flow of the Luma Interpolator was completed with success, reaching the results presented along this paper. The designed architecture used 30064 cells of the target technology and the total area of the designed chip was of 1.19 mm^2.

1. Introduction

The H.264/AVC (also known as MPEG-4 part 10) [INT 05] is the latest video coding standard which achieves significant improvements over the previous standards, in terms of compression rates [SUL 05]. H.264/AVC standard defines four profiles: baseline, main, extended and high [INT 05][RIC 03], each one covering a set of applications. Main profile is the target of this work.

The main profile differs from the baseline profile mainly because of the inclusion of bi-predictive slices, weighted prediction (WP), interlaced video support and context-based adaptive binary arithmetic coding (CABAC) [JVT 03].

H.264/AVC decoder uses a structure similar with that used in the previous standards, but each module of a H.264/AVC decoder presents many innovations when compared with previous standards as MPEG-2 (also called H.262 [VID 94]) or MPEG-4 part 2 [MOT 99].

Figure 1 shows the schematic of the decoder with its main modules. Input bit stream passes first through the entropy decoding. The next step is the inverse quantization and inverse transforms (Q^{-1} and T^{-1} modules in Figure 1) to recompose the prediction residues. INTER prediction (also called motion compensation - MC) reconstructs the macroblock (MB) from neighbor reference frames while INTRA prediction reconstructs the macroblock from the neighbor macroblocks in the same frame. INTER or INTRA prediction reconstructed macroblock is added to the residues and the results of this addition are sent to the deblocking filter. Finally, the reconstructed frame is filtered by deblocking filter and the result is sent to the frame memory.

This paper presents the standard cell design flow for H.264 luma motion compensation architecture, which supports the main profile of the H.264/AVC standard and it was designed targeting HDTV.

This paper is organized as follows. The second section presents briefly the Motion Compensation Architecture. Third section presents the standard cell flow for luma motion compensation. Section four presents the synthesis results and the section five presents the conclusions and future works.

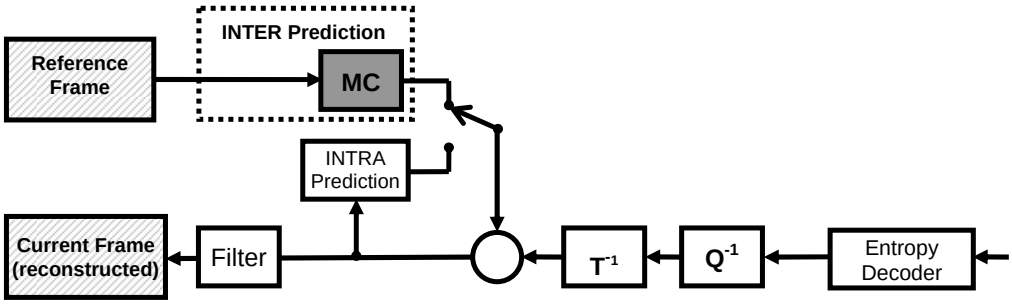


Fig. 1 – H.264/AVC Decoder Block Diagram with emphasis in the Motion Compensation

2. Motion Compensation Architecture

In order to increase coding efficiency, the H.264/AVC adopted a number of new technical developments, such as variable block-size; multiple reference pictures; quarter-sample accuracy; weighted prediction; bi-prediction and direct prediction. Most of these new developments rely on the motion compensation process [SUL 05]. Thus, this module is the most demanding component of the decoder, consuming more than half of its computation time [ZHO 03].

Motion compensation operation is, basically, to copy the predicted MB from the reference frame, adding this predicted MB to the residual MB. Thus, this operation allows the reconstruction of the MB in the current frame. Motion compensation architecture consists of the motion vector prediction, frame memory access and the sample processing. Motion vectors for neighboring partitions are often highly correlated. Then, to its advantage, in H.264/AVC each motion vector is predicted from neighbor vectors, from previously coded partitions.

The decoded frames used as reference are stored in an external memory. The area for the motion compensation is read from this external memory. The problem here is that the area, and sometimes the reference frame, is known just after the motion vectors prediction, making this process longer. The sample processing is formed by the quarter-sample interpolation, weighted prediction and clipping. The bi-predictive motion compensation is also supported by the architecture mentioned in this work, thus architecture was developed in [AZE 06]. Moreover, this architecture was designed in a hierarchical pipeline formed by: Motion Vector Predictor, Memory Access and Sample Processing. Figure 2 shows the datapath of the motion compensation module.

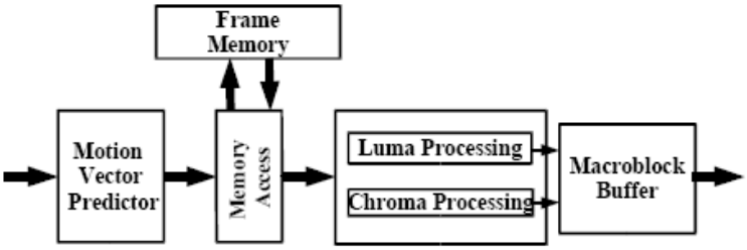


Fig.2 - Motion Compensation (MC) datapath.

The Motion Vector Predictor (MVPr) function restores the motion vectors and reference frames needed in the interpolation process. The predictor inputs are differential motion vectors and reference frames, besides the slice and macroblock information. For bi-predictive blocks, the direct prediction mode can be set, using information from spatial or temporal neighbor blocks.

Due to filter window size, some reference areas are used many times to reconstruct image blocks, and every time these areas must come from the memory to the MC interpolator. The retransmission of this redundant data uses large amount of memory bandwidth. To reduce the memory bandwidth, a memory hierarchy was designed. As presented in Figure 2, the luma and chroma samples processing works in parallel. There are one luma and one chroma datapaths for motion compensation of the blocks. A 384 positions buffer stores the results of one macroblock, which is composed of 256 luma samples and 128 chroma samples totaling 384 samples, besides this buffer synchronizes the MC module to other decoder components.

Figure 3 shows the architecture of the luma processing component, which is the focus of this work. The luma has an input buffer, a quarter-pel luma interpolator, a Weighted Predictor (WP), a 4x4 buffer, an average to bi-predictive processing, a selector for single/bi-predictive results and a clipping component. The chroma processing element has a similar datapath, except by the interpolator, which is bilinear, and the 2x2 buffer, replacing the 4x4 used in the luma processing. The luma datapath process 4x4 blocks. Chroma datapath process 2x2 blocks. These formats are the basic construction for every macroblock or macroblock (sub) partitions.

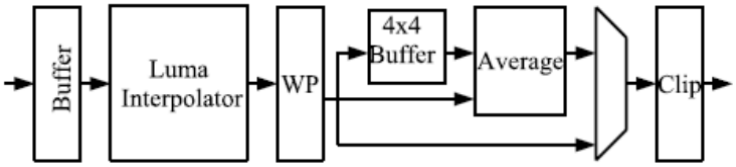


Fig.3 - Luma sample processing datapath.

The luma interpolator is presented in Figure 4 and it decomposes the 2-D FIR filter in two 1-D FIR filters (one vertical and other horizontal). Four vertical and nine horizontal 6-tap FIR filters (1, -5, 20, -5, 1) generate the half-sample interpolation while four bilinear interpolators generate the quarter-sample interpolation results.

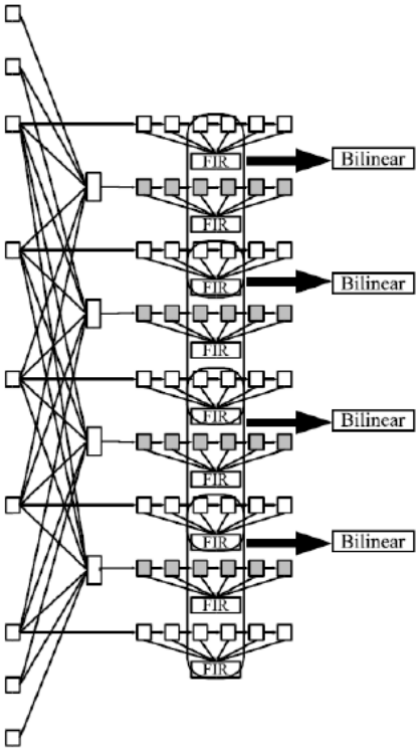


Fig.4 - Luma interpolator architecture

Luma datapath is four samples wide and each chroma datapath is two samples wide. This strategy is required to meet the throughput for HDTV decoding.

3. Standard Cell Flow for Luma Motion Compensation Architecture

The standard cell design flow for luma interpolator architecture presented in this paper was achieved using the RTL Compiler and Encounter tools from Cadence. Initially, the RTL Compiler was used to generate the files necessary for utilization of the Encounter tool. The main input files for RTL Compiler were: the modules (.vhd) of the Luma Interpolator design, LEF (Layout Exchange Format) files and Technology Library (IBM 0.18um), generating the verilog files, configuration files and sdc files necessary to start the Encounter process.

The main steps for generation of the standard cell layout [CHI 05][SAN 07] of the luma interpolator architecture through the Encounter tool, are presented below:

1. **Import Design**: Import design files into Encounter environment.
2. **Floorplanning**: When calculating core size of standard cells, the core utilization must be decided first. Usually the core utilization is higher than 85%. The recommended core shape is a square, i.e. Core Aspect Ratio = 1. The floorplan also defines the rows to host the core cells and the I/O pad cells (if required), and the location of the corner cells.
3. **Power Planning**: This step generates the VDD and ground power rings around the core and optionally adds a number of vertical and/or horizontal power stripes across the core. Stripes ensure a proper power distribution in large cores.
4. **Placement** (Fig. 5): This steps places the cells of the imported verilog netlist in the rows.
5. **Pre-CTS (Clock Tree Synthesis)**: Performs timing optimization on the placed design, before the clock tree is built. By default, repairs DRVs and setup violations for all path groups. It generates detailed timing reports.
6. **CTS**: The goal of clock tree synthesis includes: Creating clock tree spec file; building a buffer distribution network; routing clock nets;
7. **Post-CTS**: Performs timing optimization after the clock tree is built. Checks setup time and hold time. Corrects setup violations, design rule violations and hold violations.
8. **Routing** (Fig 6):
   - SRoute: Connect standard cell power.
   - NanoRoute: This step generates all the wires required to connect the cells according to the imported gate-level netlist.
   - PostRoute: Performs timing optimization on designs whose routing is complete.
9. **Add Filler**: Fill all the gaps between standard cell instances. Ensure the continuity of power/ground rails and N+/P+ wells in the row.

10. **Design checks**: This step checks that the design has been properly placed and routed.

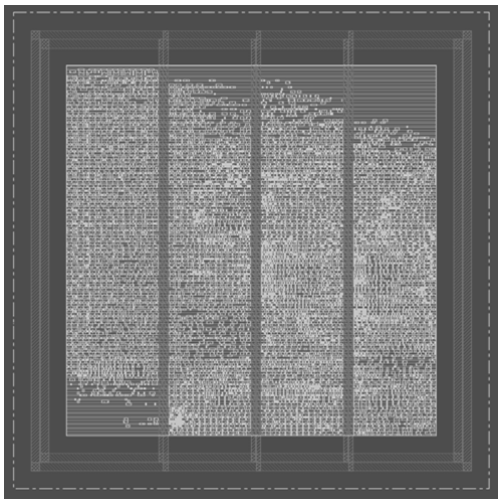


Fig.5 – Placement

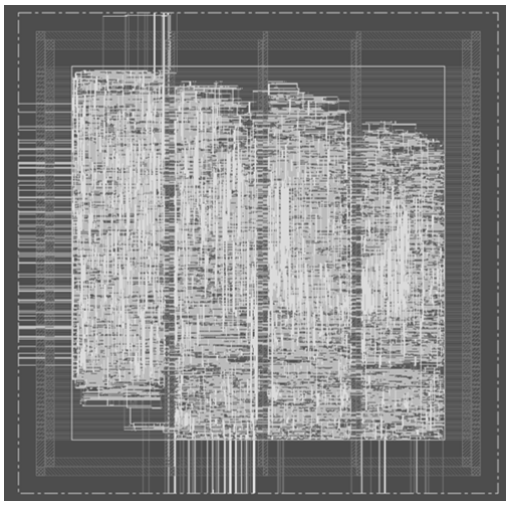


Fig.6 – Routing

4. Synthesis Results

The synthesis results of this digital flow are presented in Table 1.

Tab.1 - Synthesis Results

Total Standard Cell Number (cells)	30,064
Total Standard Cell Area (µm^2)	719,305.57
Total Area of Core (µm^2)	722,500.00
Total Area of Chip (µm^2)	1,188.972.16

5. Conclusions and Future Works

This work presented a standard cell design flow for luma motion compensation architecture, presented in Motion Compensation of H.264/AVC video compression standard.

The layout generated from the utilization of the Encounter tool reaches a good result.

As future work, it is planned to achieve the standard cell design flow of the others modules from Motion Compensation architecture.

6. References

[AZE 06] A. Azevedo. **MoCHA: Arquitetura Dedicada para a Compensação de Movimento em Decodificadores de Vídeo de Alta Definição, Seguindo o Padrão H.264.** 2006. 120 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2006.

[CHI 05] CHIH-LUNG, C. Cell-based APR Design Flow. System Integration & Silicon Implementation. National Chiao Tung University - Dept. of Electronics Engineering, NCTU-EE ICLAB II, 2005.

[INT 05] INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Recommendation H.264 (03/05): Advanced Video Coding for Generic Audiovisual Services. 2005.

[JVT 03] JVT Editors (T. Wiegand, G. Sullivan, A. Luthra). **Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification**, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[MOT 99] Motion Picture Experts Group. **MPEG-4 Part 2: Coding of audio visual objects – Part 2: Visual. ISO/IEC Recommendation 14496-2**, International Organization for Standardization, 1999.

[RIC 03] I. Richardson, **H.264 and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia**. Chichester: John Wiley and Sons, 2003.

[SAN 07] SANTOS, M.; GOMES, S. Standard Cell Placement and Routing Tutorial, AustriaMicroSystems C35B3 (HIT-Kit 3.70). Instituto superior técnico - Universidade Técnica de Lisboa, 2007.

[SUL 05] G. Sullivan, T. Wiegand. **Video compression – from concepts to the H.264/AVC standard**. Proceeding of the IEEE, 93(1):18-31, 2005.

[VID 94] Video Coding Experts Group. **Generic coding of moving pictures and associated audio information – Part 2: Video. ITU-T Recommendation H.262**, International Telecommunication Union, 1994.

[ZHO 03] X. Zhou, E. Li, Y. Chen. **Implementation of H.264 decoder on general-purpose processors with media instructions**. In Proceedings of SPIE Conference on Image and Video Communications and Processing, Santa Clara, pages 224-235, 2003.

Recent Advances on Logic Synthesis

William Lautenschläger, Ricardo Reis
{wrlautenschlager,reis}@inf.ufrgs.br

Grupo de Microeletrônica
Universidade Federal do Rio Grande do Sul

Abstract

The logic synthesis is one of the stages of the integrated circuits design flow. It involves large descriptions and problems of great complexity. This fact means that there is always demand for new solutions in this area. This paper presents the latest techniques in logic synthesis. These works were chosen according to their importance in most notable vehicles dealing on the matter. The conclusion is that there is a tendency for the resurgence of traditional approaches, despite the emergence of new ways to express logical representations and equivalences.

1. Introduction

The transistors size reduction due to the advances in integrated circuits manufacture technologies allows the growth of circuit's complexity. At same time, some related costs, like global delay and power consumption have been increased due to the possibility of put more switches into a same die area.

On the other hand, the circuits must be described to maximize the benefit of newest technologies, reaching in this manner the market requisites, like high performance and low power consumption. Moreover, optimized descriptions allow the circuits to acquire a still bigger complexity in the functions that it performs.

The logic level is an appropriate target to implement changes aimed at optimizes the representation of a circuit. This level is described mainly in terms of Boolean functions, which refer to a wide theoretical field that aims its minimization. Moreover, the equations representing logic functions are a common property between the logic level and the cells that compose the physical level of the circuit. In this way, optimizations performed at the logic level are reflected directly in the implementation of the final circuit.

These aspects delimit the two main occupations of the logic synthesis problems. The *logic minimization* seeks to reduce the representation of the circuit using the typical properties of this level. These properties generally stem from Boolean algebra. The other occupation of logic synthesis is the *technology mapping*, whose goal is to conduct a representation in the logic level for the physical level, according to its resources. Therefore, the technology mapping is subject to the way that the circuit layout is implemented. For a mechanism for automatic cells generation as presented in [LAZ06], the mapper do not needs take into account the limited functions implemented by a cell library.

In recent years, efforts in logic synthesis have been distributed in various fields of knowledge. This paper combines representative works on the various trends for the solution of logic synthesis problems. We consulted the main vehicles of this scientific field. Section 2 introduces some basic concepts. In Section 3 the manipulating AIGs techniques are discussed. Section 4 presents representations introduced recently in the literature. In Section 5 the focus is the new techniques to the Boolean matching problem. The section 6 deals with manipulating the logic level driven to other synthesis step. Finally, some final remarks appear in Section 7.

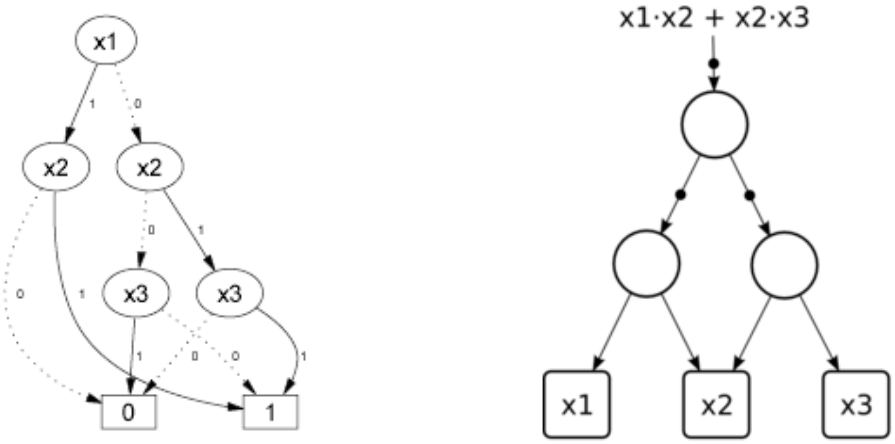
2. Basic Definitions

Many concepts utilized along this work are recurrent. A brief explanation about the most important is done below.

According [BRA08], a *Binary Decision Diagram* (*BDD*) is a rooted directed acyclic graph. The nodes, including the root, are assigned to one variable. Each node has two child nodes, which represents an assignment of the variable to 0 or 1. It is said *reduced* if any node with two identical children is removed and two nodes with isomorphic BDDs are merged. If different variables appear in the same order on all paths from the root, the BDD is said *ordered*. A BDD can be viewed in fig. 1(a).

An *AND-Inverter Graph* (*AIG*) is a directed acyclic graph that represents a structural implementation of the logical functionality of a circuit or network. An AIG ([MIS06a]) consists of two-input nodes representing logical AND, terminal nodes labeled with variable names, and edges optionally containing markers indicating logical negation. The fig. 1(b) shows an AIG.

The *Boolean matching* problem consists in the verification of equivalence between two logic functions under certain mapping rules. Thus, two functions may be equivalents either by negating or permuting inputs, or by negating outputs. Depending on the applied mapping is said that two equivalent functions pertain to a certain equivalence class. For example, equivalent functions by permuting inputs pertain to the P equivalence class. The NPN class groups functions that are equivalents by performing the three transformations.



(a) (b)

Fig. 1 – Examples of (a) a BDD (b) an AIG

Satisfiability (SAT) is the problem of determining if the variables of a given Boolean equation can be assigned in such a way as to make the formula evaluate to true. This problem is very common in many computational areas, and is classified as NP-complete. The logic representation of a combinational circuit can be immediately generating logic equations, what makes the SAT problem fundamental to logic synthesis. A SAT instance can be used, for example, to verify if a function is implementable in a programmable device ([HU07]).

3. The AIG Approach

The AIGs have been used in studies published in conferences in recent years. Many problems of optimization have been modeled on AIGs, creating a comprehensive and cohesive set of theories to handle these graphs.

The [MIS06a] work manipulates AIGs to obtain a technology-independent optimization using a pre-processed library. This approach utilizes NPN-equivalence classes. Thus, a set of subgraphs that can substitute a cut in the AIG is established. A cut is an AIG subgraph that removes any path between the primary inputs and a certain output, when this subgraph is extracted. The best subgraph inside an equivalence class that contains the cut is chosen to substitute it. The original circuit logic is preserved.

In [MIS06b] is performed a study about the application of modern computational techniques over AIGs. Assigning metric to the cuts, the work shows how typical problems of logic optimization can be approached. Among these problems, the detection and transformation of reconvergences, which occurs when paths that starts in a node output returns to this node before reach the primary outputs. The work addresses also the resubstitution, which consists in reuse graphs yet existing in the circuit to implement other functionally equivalent subgraph. Other problem approached by this technique is the elimination of logic redundancies, which occurs when some nodes do not influence in primary outputs. In [MIS07], all these problems are addressed by an alternative way, transforming the original algorithms to reduce them to the SAT problem.

The [BRA07] technique extends the [MIS06a] work with AIGs for sequential circuits. The AIG representation is allowed to contain latches. A rules set relative to the latches placement in the circuit are applied on the graph until lead it to a canonical form, where is applied the combinational transformations. In this work, the concept of nodes equivalence is extended, allowing the optimization using choice nodes. A graph containing the transformations history is utilized and can group combinational and sequential structures in the same equivalence class. Thus, more structures can be utilized, improving the circuit representation.

A set of rules for application in AIGs is utilized in [BRU06], trying to avoid the worsening of the global structure, a common problem in local optimization techniques. Rules that do not add nodes in the circuit are defined, as well some specific conditions to apply these rules. The rules are grouped into four levels according to the complexity of the replacements. These levels keep a degree of priority among themselves. The correct implementation of these rules ensures that the circuit will be optimized without negatively affecting its overall structure.

4. Innovative Representations

New forms of equivalence have also been presented. They increase the range of simplifications that can be performed in a circuit.

A work aimed at FPGAs is presented in [SOS07], introducing the concept of the sum-of-generalized products. By this definition, a circuit is represented by an OR function of several look-up tables (LUTs), which may represent multiple functions beyond the logical AND functions of the conventional sum of products. A formal framework for minimization of circuits in this form is developed, including definitions of compatibility of functions and prime generalized implicants. These definitions allow minimize a logic function on a more compact representation, occupying fewer resources for programmable devices.

The strategy utilized in [GOL07] is optimizing circuits through a technique called toggle equivalence. In a simple manner, when a pair of inputs is applied to a function *X* resulting in different outputs, while the same pair is applied to another function *Y* and also resulting in two distinct outputs, it said that *X* and *Y* are toggle-equivalent. Using Boolean satisfability relations, there is a mapping between these two variables functions, allowing substitute *X* for *Y*.

The work developed in [ROS07] tries to minimize the number of transistor in series in a CMOS logic cell. Using a topology where the PMOS network cell is not necessarily complementary to the network NMOS, the transistors in series count in critical path among the supply terminals of the cell decreases. Taking up the concept of disjunction between the networks P and N, the work still proposes a topology where pull-up and pull-down portions of the cell are generated separately through the mapping of BDDs to networks of transistors. These methods were compared to standard implementations and other related works, with gains in transistors count and logic effort.

5. Boolean Matching Advances

Many studies have addressed efficient techniques to solve the problem of Boolean matching, which is of fundamental importance to the technology mapping. In [WEI06] is shown as a data structure called binary permutation matrix can drastically decrease the computing time for the verification of equivalence between a function fully specified and another whit outputs not defined for all inputs. In [AGO07] is developed a formalism to obtain equivalent functions by permutation of inputs (P equivalence). This work defines a canonical form to represent all possible permutations, as well as a function of cost to be achieved. The functions are then transformed to this form through linear transformations and manipulation of Reduced Ordered BDDs (ROBDDs). Once obtained a canonical form, the P-equivalent functions can be extracted quickly.

Symmetry detection in Boolean functions is utilized in [HU07], reducing the search space to apply a SAT solver aiming performs Boolean matching in FPGAs. The algorithms for solving Boolean satisfability problem are increasingly more efficient. Therefore, the SAT modeling has been each time more used to solving problems in logic synthesis. Besides [HU07], this fact can be observed in [MIS07] and [GOL07].

6. Logic Level Manipulation Driven to other Synthesis Steps

Many features of the circuit can be modeled already on the logical level. Among these features are the slack of a gate or the number of beams utilized in the circuit masking. Works with this propose are common, so that the applications of logic synthesis are always increasing.

In [SUG06] is proposed a mapping directed to a lithography technique called character projection. In this technique is used a mask containing several patterns called characters. Through the deflection of electron beams (EBs), each character is projected on the wafer. In the proposed algorithm, in addition to area minimization with time restrictions, a second equation is taken into account. This equation aims to decrease the number of projections of EB necessary for the construction of the circuit, thus increasing the throughput of the process.

The work developed by [YE07] performs manipulations in the logic level to reduce the leakage of a circuit, without violating its timing restrictions. The circuit is partitioned according to the slack of each port, and implements the changes inside each group. These transformations are performed focusing on energy consumption, by modifying the dimensions of the gate and its threshold voltage. The changes are applied if the probability of violating timing restrictions for the partition containing the gate does not exceed a certain level.

In [BAL07] is developed a study about various representations used in logic synthesis, based on functions that are used to compose the circuit. The AIGs, composed only by AND nodes and inverters, are among the representations analyzed, as well representations that use only logical OR and inverters, or logical AND and OR. Using a set of combinational circuits with one output and no more than 25 inputs, the work shows that a representation containing only inverters, and NAND/NOR nodes with two inputs, can be directly mapped to a circuit with gain in area, delay and consumption, compared with circuits represented in other ways.

7. Conclusion

This work made a compilation of recent advances in the field of logic synthesis. In recent years the vehicles and conferences in this field has also approached the logic synthesis more extensively, including studies on testability and synthesis of reliable circuits. However, this work focused on processes of logic minimization and technology mapping, considered the two essential steps of the flow of logic synthesis. In addition, potential applications for handling the level logical were shown.

A summary table bringing together the works discussed in this paper is presented in tab. 1. It allows observing that there is a trend towards the use of traditional solutions, especially Boolean matching and Boolean Satisfability, revisited by newer computational techniques. However, this does not inhibit the creation of new forms of representations and equivalence. Many of these works seem to have potential extension. In fact, some of them already have been the subject of more detailed studies.

Tab. 1 – Summary of Presented Works

	Work	Goal	Utilized Approach
AIG Approach	[MIS06a]	Logic optimization	Equivalence Classes on AIGs
	[MIS06b]	Logic optimization	Metrics Assigned to AIG cuts
	[BRU06]	Logic optimization	Rules Set on AIGs
	[MIS07]	Logic optimization	SAT-based solution for typical problems
	[BRA07]	Logic optimization	Sequential optimization on AIGs
New Representations	[GOL07]	Logic optimization	Toggle Equivalence
	[SOS07]	Logic optimization	Sum-of-Generalized Products
	[ROS07]	Technology Mapping	Non-complementary topologies
Boolean Matching	[AGO07]	Technology Mapping	Formal Framework for P-Equivalence
	[HU07]	Technology Mapping	Symmetry, Boolean Matching, SAT
Driven Logic Level Manipulation	[BAL07]	Technology Mapping	Modify Gates Used in Logic Representation
	[SUG06]	Technology Mapping	Electron Beams Shots Modeling
	[YE07]	Leakage Reducing	Circuit Partitioning According Slack

8. References

[AGO07] AGOSTA, G. et al. **A Unified Approach to Canonical Form-based Boolean Matching**. In: Design Automation Conference (44th DAC, June 4-8, 2007: San Diego, USA). Proceedings... ACM Press, 2007.

[BAL07] BALASUBRAMANIAN, P. ANANTHA, K. **Power and Delay Optimized Graph Representation for Combinational Logic Circuits.** International Journal of Computer Science v.2 n.1. pp. 47-53. WASET Prees, 2007.

[BRA07] BRAYTON, R. MISCHENKO, A. **Sequential Rewriting and Synthesis**. In: International Workshop on Logic Synthesis (16th IWLS, May 30 – June 1, 2007, San Diego, USA). Proceedings… ACM Press, 2007.

[BRA08] BRAYTON, R. KUEHLMANN, A. **Logic Synthesis – Binary Decision Diagrams**. Available online at `users.ece.utexas.edu/~adnan/syn-07/003-BDDs.ppt`. 17/03/2008.

[BRU06] BRUMMAYER, R. BIERE, A. **Local Two-Level And-Inverter Graph Minimization without Blowup**. Technical Report. Johanes Kepler University Linz, Austria, 2006.

[GOL07] GOLDBERG, E, GULATI, K. SUNIL, K.. **Toggle Equivalence Preserving (TEP) Logic Optimization**. In: International Workshop on Logic Synthesis (16th IWLS, May 30 – June 1, 2007, San Diego, USA). Proceedings… ACM Press, 2007.

[HU07] HU, Y. et al. **Exploiting Symmetry in SAT-Based Boolean Matching for Heterogeneous FPGA Technology Mapping**. In: International Workshop on Logic Synthesis (16th IWLS, May 30 – June 1, 2007, San Diego, USA). Proceedings… ACM Press, 2007.

[LAZ06] LAZZARI, C. DOMINGUES, C. GÜNTZEL, J. REIS, R. **A Novel Full Automatic Layout Generation Strategy for Static CMOS Circuits**. in IFIP - International Federation for Information Processing, Volume 200, VLSI-SOC: From Systems to Chips, eds. Glesner, M., Reis, R., Indmsiak, L., Mooney, V., Eveking, H., (Boston: Springer), pp. 197-211.

[MIS06a] MISCHENKO, A. CHATTERJEE, BRAYTON, R. **DAG-Aware AIG Rewriting**. In: Design Automation Conference (43rd DAC, July 24-28, 2006: San Francisco, USA). Proceedings... ACM Press, 2006.

[MIS06b] MISCHENKO, A. BRAYTON, R. **Scalable Logic Synthesis using a Simple Circuit Structure**. In: International Workshop on Logic Synthesis (15th IWLS, June 7-9, 2006: Vail, USA). Proceedings… ACM Press, 2006.

[MIS07] MISCHENKO, A. JIANG, J. JANG, S. **SAT-Based Logic Optimization and Resynthesis**. In: International Workshop on Logic Synthesis (16th IWLS, May 30 – June 1, 2007, San Diego, USA). Proceedings… ACM Press, 2007.

[ROS07] ROSA, L. et al. **A Comparative Study of CMOS Gates with Minimum Transistor Stacks**. In: Symposium .on Integrated Circuits and System Design (20th SBCCI, September 3-6, 2007: Rio de Janeiro, Brazil). Proceedings… ACM Press, 2007.

[SAS07] SASAO, T. **Sum-of-Generalized Products Expressions Application and Minimization**. In: International Workshop on Logic Synthesis (16th IWLS, May 30 – June 1, 2007, San Diego, USA). Proceedings… ACM Press, 2007.

[SUG06] SUGIHARA, M et al. **Technology Mapping Technique for Throughput Enhancement of Character Projection Equipment**. Technical Report. 2006.

[WEI06] WEI, Z. et al. **Fast Boolean Matching with Don't Cares**. In: International Symposium on Quality Electronic Design (7th ISQED, March 27-29 2006: San Jose, USA). Proceedings. IEEE Computer Society Press, 2006.

[YE07] YE, X. ZHAN, Y. LI, P. **Statistical Leakage Power Minimization Using Fast EquiSlack Shell Based Optimization**. In: Design Automation Conference (44th DAC, June 4-8, 2007: San Diego, USA). Proceedings... ACM Press, 2007.

Author Index