26th South Symposium on Microelectronics

April 25th to 27th, 2011
Novo Hamburgo – RS – Brazil

Proceedings

Edited by
Leomar Soares da Rosa Junior
Fernando Gehm Moraes
Ewerton Artur Cappelatti

Promoted by
Brazilian Computer Society
Brazilian Microelectronics Society
IEEE Circuits and Systems Society

Organized by
Universidade Feevale
Universidade Federal do Rio Grande do Sul
Universidade Federal de Pelotas
Universidade Federal do Pampa
Pontifícia Universidade Católica do Rio Grande do Sul

Published by
Brazilian Computer Society

Universidade Feevale
Av. Dr. Maurício Cardoso, 510
Bairro Hamburgo Velho
Novo Hamburgo - RS - Brazil

Dados de Catalogação na Publicação (CIP) Internacional
Ubirajara Buddin Cruz – CRB 10/901
Biblioteca de Ciência & Tecnologia – UFPel

S726p South Symposium on Microeletronics (26. : 2011 April 25-27 : Novo Hamburgo)
Proceedings / 26. South Symposium on Microeletronics ; edited by Leomar Soares da Rosa Junior, Fernando Gehm Moraes, Ewerton Artur Cappelatti. – Novo Hamburgo : Feevale, 2011. - 215p. : il. – Conhecido também como SIM 2011.

ISSN 2177-5176

1.Microeletrônica. 2.Design digital. 3.Design análogo. 4.Ferramentas CAD. I.Rosa Junior, Leomar Soares da. II.Moraes, Fernando Gehm. III.Cappelatti, Ewerton Artur. III.Título.

CDD: 621.3817

Cover: Leomar Soares da Rosa Junior
Cover Picture: Laura Cappelatti
Edition Production: Éric Falchi Timm
Julio Domingues Saraçol Jr.
Renato Souza de Souza
Vinicius Neves Possani

Foreword

Welcome to the 26th 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. The papers submitted to this symposium 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. This year SIM takes place at Novo Hamburgo, together with the 13th edition of the regional Microelectronics School (EMICRO).

We would like to thank all individuals and organizations that helped to make this event possible. SIM 2011 was co-organized among FEEVALE, UFRGS, UFPEL, UNIPAMPA and PUCRS, promoted by the Brazilian Computer Society (SBC), the Brazilian Microelectronics Society (SBMicro) and IEEE CAS Region 9, receiving financial support from CAPES and CNPq 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.

Novo Hamburgo, April 25, 2011

Leomar Soares da Rosa Junior
Fernando Gehm Moraes
Ewerton Artur Cappelatti

SIM 2011 - 26th South Symposium on Microelectronics

April 25th to 27th, 2011
Novo Hamburgo – RS – Brazil

General Chair

Prof. Ewerton Artur Cappelatti (FEEVALE)

SIM Program Chairs

Prof. Leomar Soares da Rosa Junior (UFPEL)
Prof. Fernando Gehm Moraes (PUCRS)

EMICRO Program Chair

Prof. Ricardo Augusto da Luz Reis (UFRGS)
Prof. Alessandro Gonçalves Girardi (UNIPAMPA)

IEEE CAS Liaison

Prof. Ricardo Augusto da Luz Reis (UFRGS)

List of Reviewers

Adriel Mota Ziesemer Jr (UFRGS)
Alexandre G. Girardi (UNIPAMPA)
Alexandre de Morais Amory (PUCRS)
Antonio Carlos S. Beck Filho (UFSM)
Bruno Silveira Neves (UNIPAMPA)
Bruno Zatt (UFRGS)
Caio G. Alegretti (UFRGS)
Carlos Eduardo Klock (UFRGS)
Cláudio Machado Diniz (UFRGS)
Cristina Meinhardt (FURG)
Daniel Munari Palomino (UFRGS)
Denis Teixeira Franco (FURG)
Diego Vrague Noble (UFRGS)
Edgard de Faria Correa (UFRN)
Edson Ifarraguirre Moreno (PUCRS)
Eduarda Monteiro (UFRGS)
Eduardo da Costa (UCPEL)
Eduardo Flores (UFRGS)
Felipe de Souza Marques (UFPEL)
Felipe Marranghello (UFRGS)
Felipe Sampaio (UFRGS)
Guilherme Corrêa (UFRGS)
Guilherme Flach (UFRGS)
Gustavo Girão (UFRGS)
Juliano L. Gonçalves (UFPEL)
Julio C. B. Mattos (UFPEL)
Leomar S. da Rosa Jr (UFPEL)
Lisane B. de Brisolara (UFPEL)
Luciano V. Agostini (UFPEL)
Marcello R. Macarthy (UFPEL)
Marcelo Porto (UFPEL)
Marcio Kreutz (UFRN)
Marcio Oyamada (UNIOESTE)
Mateus Rutzig (UFRGS)
Mauricio Lima Pilla (UFPEL)
Mayler Martins (UFRGS)
Osvaldo Martinello Jr (UFRGS)
Paulo F. Butzen (UFRGS)
Rafael Soares (UFPEL)
Reginaldo Tavares (UNIPAMPA)
Renato Hentschke (INTEL)
Roger Porto (UFPEL)
Sandro Sawicki (UNIJUÍ)
Sandro Silva (UFRGS)
Sidinei Ghissoni (UNIPAMPA)
Thaísa Silva (UFRGS)
Tiago H. Trojahn (USP)
Vinicius Callegaro (UFRGS)
Vinícius Dal Bem (UFRGS)

Table of Contents

Session 4: VIDEO CODING 2

Session 5: DESIGN AUTOMATION TOOLS 3

Session 6: NOCS, MPSOCS AND ANALOG DESIGN

Design Automation Tools 1

VEasy: a Functional Verification Tool Suite

Samuel Nascimento Pagliarini and Fernanda Lima Kastensmidt
{snpagliarini, fglima}@inf.ufrgs.br

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Programas de Pós-Graduacão em Microeletrônica e Computação
Porto Alegre, Brasil - 91501-970

Abstract

This paper describes a tool developed specifically for aiding the process of Functional Verification. The tool has three main built-in modules: a Verilog Linter, a Verilog Simulator and a Graphical User Interface for Testbench Automation. On top of these modules there is a methodology for collecting and analyzing functional and structural coverage. Results show that the built-in simulator enables a higher number of cycles per second while the user interface allows the creation of complex test scenarios.

1. Introduction

The primary goal of Functional Verification (FV) is to establish confidence that the design intent was captured correctly by the implementation [1]. However, the continuous increase in terms of the number of transistors per chip is resulting in a diminished validation effectiveness. Simulation is getting more expensive and providing less coverage [2]. FV strives to cope with that complexity increase trend but some of the related challenges are overwhelming. So far those challenges have been addressed with methodologies and Electronic Design Automation (EDA) tools but there is a claim for more innovation and automation improvement.

This paper describes and compares VEasy, an EDA tool suite developed by the author. VEasy aim is to be a FV solution, including a simulator and a testbench automation interface. This paper is organized as follows: Section 2 explains the tool in detail, including the possible work-flows while Subsections 2.1 and 2.2 deal with the linting and simulation built-in modules. Subsection 2.3 explains the methodology used for creating complex test scenarios. The different types of functional and structural coverage are shown in Subsection 2.4. Finally, Section 3 provides some concluding remarks

2. VEasy and the work-flows

The tool has two distinct work-flows: the assisted flow and the simulation flow. Fig. 1 illustrates the assisted mode. This flow starts when the Verilog [3] description of the Design Under Test (DUT) is parsed and analyzed. If the analysis is not successful the user must fix the errors reported by the linter before continuing. From that same input, the interfaces (i.e. Input and output signals) and special signals (i.e. clock and reset) are automatically extracted. This information is sufficient to build a template of a verification plan. The template is the input of the simulation flow.

Fig. 1 – VEasy assisted flow.

Fig. 2 illustrates the tool simulation flow. Initially, a verification plan file is loaded. This file contains all the information that is required to generate and simulate the test scenarios that the user creates through the Graphical User Interface (GUI). VEasy then is ready to create a simulation snapshot, combining the circuit description and the test generation capabilities. The use of a golden model is optional. All the generated code is ANSI-C [4], which allows it to be used in the majority of platforms and compilers. After the simulation is complete the tool automatically collects the coverage results, saves them into the verification plan and provides this info for the user analysis. If suitable a new simulation round may be started in a attempt to reach coverage holes. The verification plan file used by VEasy is actually a complete view of the verification effort. It includes the traditional lists of features and associated test cases but it also contains simulation and coverage data. This approach makes it a unified database of the current verification progress.

2.1. VEasy as a Verilog RTL Linter

Linting [5] guarantees that the input is written using strictly Register Transfer Level (RTL) Verilog constructions. A series of items are checked for RTL compliance. Passing all those checks means that the code may be converted to a C-like version.

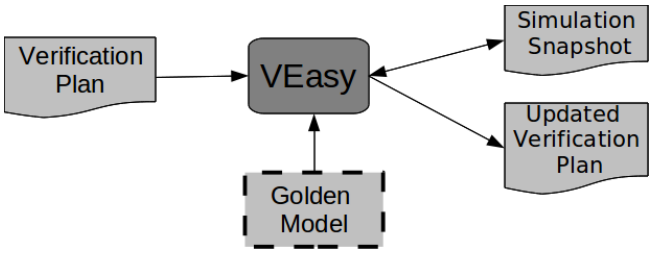


Fig. 2 – VEasy simulation flow.

2.2. VEasy as a Verilog RTL simulator

FV relies on fast and accurate simulation. In order to improve the number of cycles simulated per second, VEasy integrates the test case generation (i.e. the generation of inputs that build a certain test case) with the circuit description. The circuit description portion of the simulation snapshot is obtained from extracting three profiles from the Verilog code: the combinational logic, the reset sequential logic and the regular sequential logic. The combinational logic simulation is performed using a signature-based method. A pseudo code of the method is given in fig. 3.

Algorithm 1: Combinational logic simulation

Data: S such that S contains all combinational signals.
Result: Frozen S after a simulation cycle.

```
1  begin
2      LocalSignature ⟵ ∅
3      Signature ⟵ ∅
4      do_it_again :
5      execute update logic on S
6      foreach s ∈ S do
7          LocalSignature[i] ⟵ s
8      end
9      if LocalSignature = Signature then
10         return
11     end
12     else
13         Signature ⟵ LocalSignature
14         goto do_it_again
15     end
16 end
```

Fig. 3 – Combinational logic simulation.

The signatures used are arrays, sized according to the number of signals being updated in the combinational logic. A signal may be a primary output or a internal wire or register. If the signature is the same for two consecutive evaluations then the combinational logic is considered to be frozen since not a single signal changed from one evaluation to another, which can be observed on line 9. The update logic (line 5) is very similar to the original input since C and Verilog have similar syntaxes and operators. The main difference resides in the fact that C has no direct single bit access. This issue is resolved by using masks and logical operators (and/or) to set or clear specific bits. This method reflects the combinational logic of a circuit where the signals have a switching behavior.

The reset sequential logic simulation is trivial: all resettable signals will receive the determined reset values. On the other hand, the regular sequential logic simulation must provide the concurrency of assignments as if the clock edge were reaching all the signals at the same time. For that matter, a method that uses local copies of the signals was developed. Such method works by performing all the sequential update logic assignments at the local signal copies but using the actual signal values, i.e. no actual signal is written until the logic has evaluated. After the logic is evaluated the local signals are written on the actual signals. Therefore the order in which the signals are assigned is no longer important.

Each of the profiles is built into a C function. The simulation process will repeatedly call these functions until the desired number of simulation cycles is reached. Separating the reset behavior from the regular sequential logic behavior saves some simulation time. A set of simple circuits was defined for the purpose of comparing the speed of the simulator. Fig. 4 shows those results, where dffnrst is a d-type flip-flop with reset (negative edge), adder in an 8-bit adder with registered outputs, fsm is a simple FSM that has 8 states and performs different 16-bit data operations on each state and t6507lp [6] is a 8-bit micro-controller with 100 opcodes and 10 addressing modes.

All simulations of fig. 4 were done using 10 millions of clock cycles. The reset signal was asserted only during the first simulation cycle. The other signals were generated every cycle with a random value. Commercial I is a simulator from a major vendor in the ASIC domain. Commercial II is a simulator used mostly In the FPGA domain. Icarus Verilog [7] is a free software. The scale on the Y axis of Fig. 4 is logarithmic. When compared with the simulation times of Commercial I, VEasy performs, on average, the same simulation within less than 5% of the time Commercial I requires.

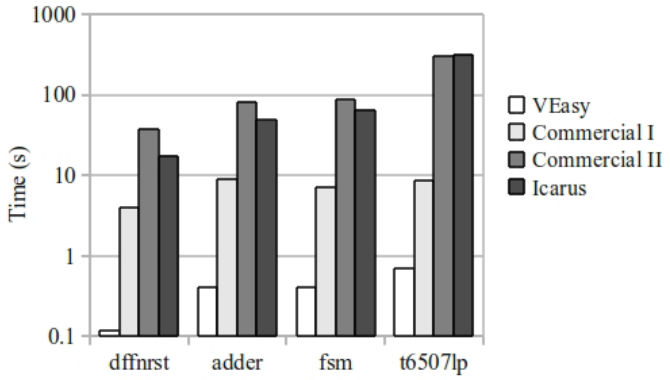


Fig. 4 – VEasy simulation results.

2.3. VEasy and the Testbench Automation

The main feature of the tool is the testbench automation. Using a strong GUI the user is able to drag-and-drop sequence to build larger sequences. One example of that operation is shown in Fig. 5, where the user is dropping a *layer2* sequence item (**press_and_releaseY**) into a *layer3* (**correct_password**) list of sequence items. The possibility of combining more and more layers allows the construction of sophisticated test cases that are of particular interest for the functional verification of a design. There are only a few rules that must be observed:

1) Sequence items of *layer0* are the only ones capable of interfacing with the design.

2) Sequence items of *layer0* will always generate values for all the inputs of the design, whether they are constrained or not.

3) If a member is not under any constraint then it is assigned a random value within its possible values.

4) Sequence items of layer0 are the only ones that can make the simulation advance in time.

5) The only communication channel between two layers is through logical members. All data exchange relies on this approach.

6) All members must be uniquely identified to allow any layer to use them unambiguously.

7) Each sequence must have at least one sequence item, except for *layer0* sequences which do not have a list at all.

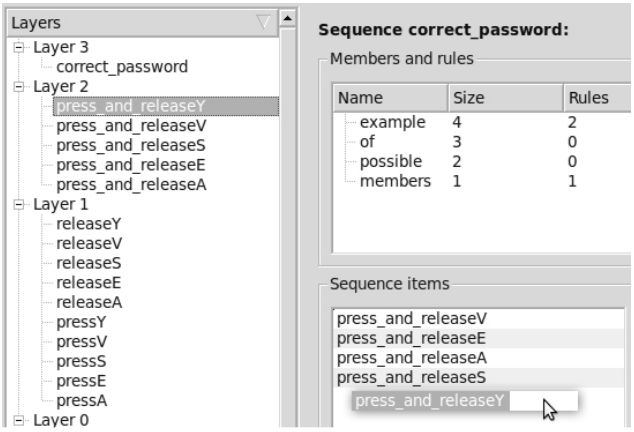


Fig. 5 – VEasy simulation flow.

The process starts by defining at least one *layer0* sequence that contains all of the physical inputs of the DUT. To build a *layerN* sequence it is only required that the user adds at least one *layerN-1* sequence item. It is also possible to add logical members into sequences of all layers. Fig. 6 contains an example of a possible layering. The example shows a sequence called **top** of a *layerN*. This sequence has two logical members (memberA and memberB) and each member is constrained with a set of rules. This same sequence has four items in its list of sequence items. One of these items is referred as **main**, which is a sequence from *layer0*. This sequence item in particular has no list of items since it is in the bottom of the hierarchy. Yet, since this is a

sequence from *layer0*, it contains a physical member referred as phy_member. As mentioned, each member of a layer, either physical or logical, might be constrained using rules. Currently the tool supports 7 types of rules:

- Keep value less than (<) or greater than (>)
- Keep value less or equal than (<=), or greater or equal than (>=)
- Keep value equal to (==)
- Keep value ranged between ([a:b]) or in a list of possible values ([a,b,c])

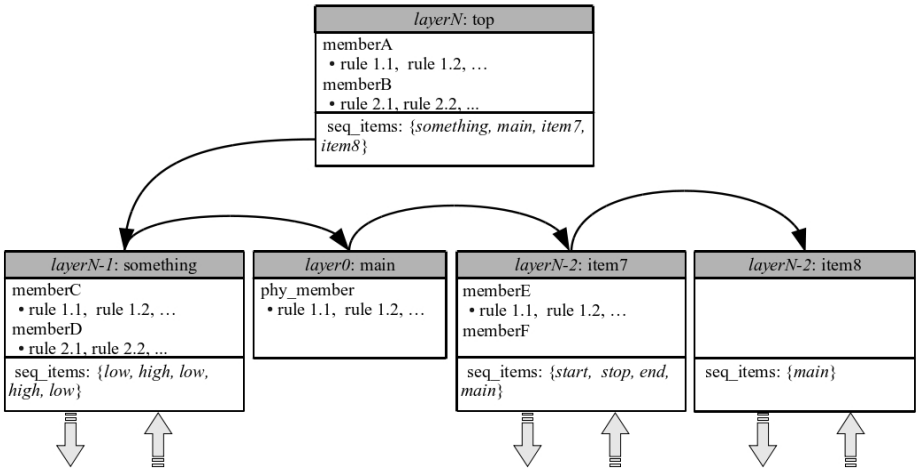


Fig. 6 – Layering example.

2.4. Coverage Methodology

The quality of the verification relies on coverage metrics, either functional or structural [8]. VEasy has integrated three different metrics that are based on structural coverage: block coverage, expression coverage and toggle coverage.

On the functional coverage side, VEasy allows coverage of inputs and outputs. The output coverage is performed directly on the primary outputs of the design. The input coverage, on the other hand, may be performed on the primary inputs or using specific logical members of the layers. The user must choose such members manually. In another words, this allows the user to define the functional coverage metrics of interest.

3. Conclusion

Design verification has been accomplished following two principal techniques, known as formal and functional verification [9]. FV is mainly simulation based. Although new methodologies that combine formal, semi-formal and functional solutions have been proposed [10] and adopted by the industry, these methodologies are still limited. In that context, this paper described a tool that enhances the FV traditional flow. Therefore, results comparing different simulators were shown on Fig. 4. Later the layering scheme was detailed and a example was provided as well. Combining the simulation speed of the integrated simulator with the GUI that allows the creation of complex test scenarios, it is possible to perform the FV of designs without writing a single line of code, lowering the verification effort considerably.

4. References

[1] Specification for VC/SoC Functional Verification, VSI Alliance, 2004.
[2] C. Yan and K. Jones, “Efficient simulation based verification by reordering,” presented at the Design and Verification Conference, 2010.
[3] Standard for the Verilog Hardware Description Language, IEEE Std. 1364, 2001.
[4] The C Programming Language Standard, ANSI Std. X3.159, 1989.
[5] L. Bening and H. Foster, Principles of verifiable RTL design: a functional coding style supporting verification processes in Verilog. Springer.
[6] S. Pagliarini and G. Zardo. (2009) t6507lp ip core. [Online]. Available: http://opencores.org/project,t6507lp
[7] S. Williams. (1999) Icarus verilog. [Online]. Available: http://www.icarus.com/eda/verilog/
[8] R. Grinwald et al., “User defined coverage - a tool supported methodology for design verication,” in Proc. 35th annual Design Automation Conference, San Francisco, United States, June 15–19, 1998, pp. 158–163.
[9] J. Bergeron, Writing Testbenches: Functional Verification of HDL Models, 2nd ed. Boston: Kluwer Academic, 2003.
[10] O. Cohen et al., “Designers work less with quality formal equivalence checking,” presented at the Design and Verification Conference, 2010.

Gate Sizing Minimizing Delay and Power/Area

Gracieli Posser, Guilherme Flach, Gustavo Wilke, Ricardo Reis
{gposser,gaflach,wilke,reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul (UFRGS)
Instituto de Informática – PPGC/PGMicro
Av. Bento Gonçalves 9500 Porto Alegre, RS - Brazil

Abstract

In this work we present a gate sizing tool based on Geometric Programming. The optimization can be done targeting both, delay and power/area minimization. In order to qualify our approach, the ISCAS'85 benchmark circuits are mapped for 350nm and 45nm technologies using typical standard cell libraries. Next, the mapped circuit is sized using our tool and the result is compared to the original mapped circuit. The speed is increased by 21% and 4.5%, on average, for 45nm and 350nm technology, respectively, keeping the same area and power values of the sizing provided by standard-cells library. For power/area optimization, where the delay was restricted to the delay value found at delay minimization, the reduction was 28.2% in area and 27.3% in power consumption, on average, considering 45nm technology and 29.9% in area and 28.5% in power, on average, considering 350nm technology.

1. Introduction

One can make a circuit faster or consume less power by sizing its logic gates properly. Sizing a gate means that the transistors, which compose the gate, are made shorter or larger according to a scale factor.

There is an optimal scale factor for each gate, considering that increasing the size of the gate and, consequently, of the transistors, increases their ability to carry a load, reducing the time required for the gate switching its signal. However, by increasing the port size reduces its output resistance, and increases its input capacitance, giving its driver higher capacitive load.

The gate sizing tool developed in this work is able to handle CMOS circuits and can be configurable to several CMOS manufacturing technologies. Moreover, the optimization can be done in two ways: (1) delay minimization subject to an area constraint and (2) area minimization subject to a delay constraint. Elmore Delay [1] is used in order to model the gate sizing problem as a Geometric Program (GP) [2]. GP is a mathematical optimization problem that can be efficiently solved in polynomial time guaranteeing that the optimal solution is found if one exists.

The contributions of this work are:

- a gate sizing tool based on GP that can be used along with an automatic cell layout generation tool taking advantage of "continuous" gate sizing avoiding the rounding-off problems;
- a gate sizing configurable to several manufacturing technology by changing the technology parameters;
- a gate sizing tool targeting both, delay and area;
- a more accurate gate sizing method combining [3] and [2] works.

The paper is organized as follows. On section II, some related works are shown. We show the problem formulation on section III. The section IV presents the gate sizing development using GP. On section V, we show the results and, finally, on section VI, we present our conclusions.

2. Related Works

Gate-sizing problem has been studied in many papers using different ways to solve it. The most widely known is the logical effort method [4], which provides fast heuristics or design guidelines for solving the gate-sizing problem approximately. Linear Programming is used in [5] and [6]. In [7] and [8] is used Non-Linear Programming. [9] are concerned about scalability of the circuit, i.e., the execution time needed to size large circuits. The traditional gate sizing methodologies [10], [3], [2] use Elmore delay to model delay as posynomial functions allowing the gate sizing to be formulated as a Geometric Program.

This work combines the ideas of the works [3] and [2]. [3] presents a method for transistor sizing, where the sizing problem is modeled and solved by GP. The gates are modeled using the Switch-Level RC Gate Model. In this model, a gate is viewed as a set of RC trees one for each possible input vector and the gate delay is the maximum delay generated by its compound RC trees. The RC tree is built by replacing turned-on transistor by an equivalent resistance. Node capacitances is composed by source-to-bulk and drain-to-bulk capacitances. A load capacitance is connected to the output node of the gate.

Work [2] shows a gate sizing method, where a variable X_i is associated for each logical gate. The scale factors are the optimization variables of the GP. The gate delay is estimated by a linear function on the scale

factors. Circuit area is the sum of the area of each port that make up the circuit. The path delays are given by the RC product and circuit delay is the maximum delay among all paths of the circuit.

3. Problem Formulation

This work supports two different formulations, delay minimization subject to an area constraint and area minimization subject to a delay constraint. The two formulations are shown below.

3.1. Minimizing Delay Subject to Maximum Area

When the objective is minimize delay, the optimization problem is formulated as following:

$$\begin{aligned} &\text{minimize} && D = max(D_1 ... D_n) \\ &\text{subject to} && X_{\min} \le X_i \le X_{\max} \\ & && C_{in} \le C_{in}^{max} \qquad A \le A_{max} \end{aligned} \tag{1}$$

where D values are the delay of the circuit paths. The X_{min} and X_{max} are the minimum and maximum size of the gate. C_{in}^{max} is the maximum input capacitance acceptable to the circuit and A^{max} is the maximum circuit area.

3.2. Minimizing Area Subject to Maximum Delay

For this optimization, we make a change, the objective function becomes the area and delay becomes a constraint. The formulation is the following:

$$\begin{aligned} &\text{minimize} && Area \\ &\text{subject to} && X_{\min} \le X_i \le X_{\max} \\ & && C_{in} \le C_{in}^{max} \qquad D \le D^{\max} \end{aligned} \tag{2}$$

4. Gate Sizing Development

It can be said that the gate sizing problem is the problem of choosing the scale factors in order to find the minimum delay (area) subject to limits on the total area (delay) and others constraints.

Our gate sizing tool was developed as follow:

1) The logic gates are modeled using the Switch-Level RC Gate Model [3].
2) For each port is set a scale factor that multiplies the transistor widths.
3) Capacitance and resistance values used to calculate delay and power are obtained throughout SPICE simulations for PMOS and NMOS transistors. The capacitances that compose a gate are proportional to the scale factor and the driving resistance is approximately inversely proportional.
4) The delay is calculated using the Elmore delay model, which produces posynomial functions, enabling the problem solution by Geometric Programming.
5) Circuit delay is the maximum delay among all circuit paths.
6) The area of a scaled gate *i* is proportional to the gate scale factor X_i, where, n is the number of gates of the circuit and $A_{base}(Y_i)$ is the area base of the gate *i*, as shown below:

$$A_{total} = \sum_{i=1}^{n} X_i * A_{base}(Y_i) \tag{3}$$

7) The power is calculated considering only switching. It is made using the equation (4), where, C_{load} is the load capacitance of the circuit. C_{in} is the input capacitance of each gate of the circuit. *Vdd* is 1.1V for 45nm technology and 3.3V considering 350nm technology. α is the probability of switching, we considerate that the circuit switching 20% of the time and *f* is the clock frequency and it was set 500MHz for our test cases.

$$P = (C_{load} + \sum_{i=1}^{n} Cin_i) * Vdd^2 * \alpha * f \tag{4}$$

5. Results

[11] highlighted that once the minimum delay is achieved, a new optimization program may be performed targeting area minimization using the minimum delay as constraint. This allows the area to be further minimized since the former problem are not concerned in area minimization. This approach is used in this work. First, the delay is minimized. Then, the area is further reduced with no delay penalty.

We use a set of the ISCAS'85 benchmark circuits for our tests. Initially, the circuit was mapped using RTL Compiler tool from Cadence to 45nm library. Design Compiler from Synopsys was used to 350nm library. In both, only CMOS cells were considerate. These circuits are inserted to our sizing tool where the area, timing and power values were calculated. We considerate as load capacitance a value six times greater than the input capacitance of a gate not scaled, i.e., with scale factor one.

5.1. Results for 45nm Technology

Tab. 1 shows the values and reductions (R) considering the sizes found in a standard cell library (SC) and the sizes given by our gate sizing using Geometric Programming (GP). The gate sizing in Tab. 1 is a delay minimization, where the area is restricted to the same area of the circuit using standard cells. The circuit sized using our methodology (GP) obtained a reduction, on average, of 21% in delay, keeping the same area and power values of the sizing provided by standard-cells library.

Tab. 1 – Comparison results between standard cells (SC) sizing and sizing using Geometric Programming (GP) proposed in this work to 45nm minimizing delay subject to area

	# Gates	Power (µW)			Timing (ρs)			Area (µm²)		
		SC sizing	GP sizing	R (%)	SC sizing	GP sizing	R (%)	SC sizing	GP sizing	R (%)
C432	184	22.2	22.4	-0.9	718	666	7.3	210.4	210.4	0.0
C499	403	58.3	58.4	-0.2	750	651	13.1	536.4	536.4	0.0
C1908	259	33.6	33.7	-0.3	472	425	10.0	304.3	304.3	0.0
C880	232	31.4	31.1	1.1	451	330	26.8	281.0	277.4	1.3
apex1	1728	239.8	239.5	0.1	673	504	25.2	2304	2296	0.4
apex2	4110	527.1	523.6	0.7	863	650	24.7	5180	5145	0.7
apex3	1939	254.3	251.9	0.9	687	507	26.3	2441	2413	1.2
apex5	1942	264.6	258.3	2.4	662	431	34.9	2512	2446	2.6
Avg.	1350	**178.9**	**177.3**	**0.5**	**660**	**521**	**21.0**	**1721**	**1704**	**0.8**

Tab. 2 shows the values for gate sizing minimizing area/power subject to the delay to the minimum value found by the delay minimization, shown in Tab. 1. Tab. 2 shows area and power values given by delay minimization (Min. Delay) and by area minimization (Min. area) and their reductions.

Area minimization allowed a reduction, on average, 28.2% in area and 27.3% in power consumption, considering the same delay value given by delay minimization, Tab. 1. This improvement is possible when there are multiple optimal points that minimize the delay, given an area budget. So, the area minimization problem is able to find the minimum area considering the minimal circuit delay.

Tab. 2 - Results for gate sizing minimizing area subject to delay (min. Area) compared to the values from delay minimization (min. Delay) for 45nm

	Power (µW)			Area (µm²)		
	Min. Delay	Min. Area	Reduction (%)	Min. Delay	Min. Area	Reduction (%)
C432	22.4	22.4	0.00	210.4	210.4	0.00
C499	58.4	58.4	0.00	536.4	536.4	0.00
C1908	33.7	33.7	0.00	304.3	304.3	0.00
C880	31.1	20.2	34.9	277.4	171	38.4
apex1	239.5	137.3	42.7	2296	1293.5	43.7
apex2	523.6	270.3	48.4	5145	2647.3	48.5
apex3	251.9	135.8	46.1	2413	1274.1	47.2
apex5	258.3	138.5	46.4	2446	1269	48.1
Avg.	**177.3**	**102.1**	**27.3**	**1704**	**963.3**	**28.2**

5.2. Results for 350nm Technology

Considering 350nm technology, the sized circuits using our gate sizing tool showed a reduction, on average, of 4.5% in delay, and area and power values are similar to the values presented by the circuits mapped to standard-cells library, as showed at Tab. 3.

Tab. 4 shows the values obtained for gate sizing minimizing area subject to delay found by minimizing delay, Tab. 3. Tab. 4 presents area and power values by delay minimization (Min. delay) and area minimization (Min. area) and their reduction (R) values. Area optimization allowed a reduction, on average, of 29.9% in area and 28.5% in power consumption.

Tab. 3 – Comparison results between standard cells (SC) sizing and sizing using Geometric Programming (GP) proposed in this work to 350nm minimizing delay subject to area

	# Gates	Power (µW)			Timing (ρs)			Area (µm²)		
		SC sizing	GP sizing	R (%)	SC sizing	GP sizing	R (%)	SC sizing	GP sizing	R (%)
C432	15	0.56	0.56	0.0	1.58	1.52	4.2	444	444	0.00
C499	388	7.99	8.07	-1.0	7.29	6.68	8.4	8015	8015	0.00
C1908	79	3.00	3.05	-1.5	3.26	3.09	5.2	2407.4	2407.4	0.00
C880	177	6.03	5.59	7.3	3.8	3.68	3.2	5609	5303	5.5
apex1	1455	27.9	27.9	0.0	7.28	7.11	2.4	27803	27783	0.1
apex2	778	15.2	15.2	0.1	6.59	6.25	5.2	15786	15765	0.1
apex3	1715	34.5	34.3	0.4	6.03	5.89	2.3	34965	34940	0.1
apex5	958	19.6	19.6	0.0	4.73	4.49	5.0	18699	18685	0.1
Avg.	659.6	**14.4**	**14.3**	**0.6**	**5.07**	**4.74**	**4.5**	**14216**	**14168**	**0.7**

Tab. 4 - Results for gate sizing minimizing area subject to delay (min. Area) compared to the values from delay minimization (min. Delay) for 350nm

	Power (µW)			Area (µm²)		
	Min. Delay	Min. Area	Reduction (%)	Min. Delay	Min. Area	Reduction (%)
C432	0.56	0.56	0.00	444	444	0.00
C499	8.07	8.07	0.00	8015	8015	0.00
C1908	3.05	3.05	0.00	2407.4	2407.4	0.00
C880	5.59	3.11	44.4	5303	2710.2	48.9
apex1	27.9	14.79	47.0	27783	14282.4	48.6
apex2	15.2	8.208	46.1	15765	8419.4	46.6
apex3	34.3	16.89	50.8	34940	16659.3	52.3
apex5	19.6	11.82	39.6	18685	10748.3	42.5
Avg.	**14.3**	**8.3**	**28.5**	**14168**	**7960.7**	**29.9**

6. Conclusion

Gate sizing using GP achieves better results compared with a circuit using commercial standard cell sizes selected by RTL Compiler

In this paper, we solve a gate sizing problem using GP combining the works [3], [2]. To model the gates, we use a Switch-Level RC Gate Model. In our tool, the gate sizing problem can be formulated in two ways:

1) Aiming to minimize the circuit delay subject to a maximum area, or

2) Minimizing area subject to a delay restriction.

We showed that, for our test cases in 45nm and 350nm, the gate sizing tool produced better results compared with a circuit using commercial standard cell sizes selected by RTL Compiler. The tests show that our gate sizing tool using GP reduced the delay in 21%, on average, for 45nm and 4.5% for 350nm, considering delay minimization. For area minimization, considering 45nm technology, our gate sizing reduced the area in 28.2%, on average, and 27.3% in power consumption. For 350nm, the reduction in area was 29.9% and 28.5% in power consumption, on average.

Using an automatic cell generation tool, as ASTRAN [12], we can generate cells in the desired size and take advantage of the better results in timing, area and power, which is critical in recent technologies.

7. Acknowledgment

This work is partially supported by Brazilian National Council for Scientific and Technological Development (CNPq–Brazil) and Coordination for the Improvement of Higher Education Personnel (CAPES).

8. References

[1] W. Elmore, “The transient analysis of damped linear networks with particular regard to wideband amplifiers,” J. Applied Physics, vol. 19, 1948.

[2] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, “A tutorial on geometric programming,” Springer Science+Business Media, LLC 2007, pp. 67–127, 2007..

[3] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz, “Digital circuit optimiza- tion via geometric programming,” Operations Research, vol. 53, no. 6, pp. 899–932, Nov.-Dec. 2005.

[4] I. Sutherland, B. Sproull, and D. Harris, Logical Effort: Designing Fast CMOS Circuits. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[5] M. Berkelaar and J. Jess, “Gate sizing in mos digital circuits with linear programming,” in EDAC‘90: Conference on European Design Automation, Glasgow, Scotland, 1990, pp. 217–221.

[6] K. Bhattacharya and N. Ranganathan, “A linear programming for- mulation for security-aware gate sizing,” in 18th ACM Great Lakes symposium on VLSI, Orlando, Florida - USA, 2008, pp. 273–278.
[7] S. S. Sapatnekar and W. Chuang, “Power-delay optimization in gate sizing,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 5, no. 1, pp. 98–114, 2000.
[8] V. Mahalingam and N. Ranganathan, “A nonlinear programming based power optimization methodology for gate sizing and voltage selection,” in ISVLSI, Tampa, Florida - USA, 2005, pp. 180–185.
[9] S. Joshi and S. Boyd, “An efficient method for large-scale gate sizing,” IEEE Transactions on Circuits and Systems, vol. 55, no. 9, pp. 2760 – 2773, October 2008.
[10] S. Sapatnekar, V. Rao, P. Vaidya, and S.-M. Kang, “An exact solution to the transistor sizing problem for cmos circuits using convex optimization,” IEEE Transactions on Computer Aided Design of Integrated circuits and Systems, vol. 12, no. 11, pp. 1621–1634, 1993.
[11] H. Tennakoon and C. Sechen, “Efficient and accurate gate sizing with piecewise convex delay models,” in 42nd Design Automation Conference, Anaheim, California, USA, 2005, pp. 807–812.
[12] A. Ziesemer, C. Lazzari, and R. Reis, “Transistor level automatic layout generator for non-complementary cmos cells,” in VLSI-SOC 2007, Atlanta, GA, USA, Oct 2007, pp. 116–121.

Evaluating Stimuli Generation Using the VEasy Functional Verification Tool Suite

Paulo A. Haacke, Samuel N. Pagliarini and Fernanda L. Kastensmidt
{pahaacke, snpagliarini, fglima}@inf.ufrgs.br

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Programas de Pós-Graduacão em Microeletrônica e Computação
Porto Alegre, Brasil - 91501-970

Abstract

This paper describes and evaluates a stimuli generation methodology for testbench automation. The methodology is part of a tool suite developed specifically for aiding the process of Functional Verification (FV). A brief description of some features from the tool, referred as VEasy, are presented. Then, the focus goes to the stimuli generation, where the layering methodology used is described. After that, some simulation results are presented showing that the VEasy's methodology enables a speed-up on the simulation and, consequently, enables a faster verification.

1. Introduction

Today, in the era of multi-million gates ASIC and system-on-a-chip (SoC) designs, verification is known as a major bottleneck in the development of a system, consuming about 70%[1] of the design effort. Indeed, it is a fundamental step in the development of digital circuits. Hardware complexity keeps growing and that obviously impacts the verification complexity since it is leading to an even more challenging verification. In fact, it has been demonstrated that verification complexity theoretically rises exponentially with hardware complexity [2].

The main goal of FV is to demonstrate that the intent of a design was preserved in its implementation [1]. To achieve that, often it uses a combination of simple logic simulation and test cases generated for asserting specific features of the design. All the test cases are compiled into testbenches. In order to perform the FV, the Register Transfer Level (RTL) representation of the design is going to be used in the context of this paper. On top of the simulation, FV applies specific and specialized constructs, like assertions, constrained randomness and coverage metrics. Yet, functional/logic flaws are still the main reason for silicon re-spins .

The data generation for a Design Under Test (DUT) is basically the first goal of a testbench and one of the largest challenges of the FV. At the beginning, FV was done with simple direct or random stimulus. But, because of the growth in the complexity of digital circuits, it becomes more difficult to cover all the situations of interest. To solve this, some high level languages that allow the creation of constrained random stimuli were developed [3].

With the increase of the hardware complexity more challenges have been faced by the verification engineers. Some of these challenges have been addressed with automation and new verification methodologies but there is a claim for more innovative tools. In order to achieve this goal this paper describes and compares VEasy, an Electronic Design Automation (EDA) tool suite developed by the authors. Such tool suite is described in [4, 5]. VEasy's aim is to be a FV solution, including a simulator and a Testcase Creation Graphical User Interface (GUI), hence the main domains of improvements presented are the simulator as a simple EDA tool and the Testcase Creation solution jointly with the GUI as a methodology.

This paper explores the stimuli generation challenge. Regarding such topic, VEasy's features are explained, focusing on the solution used to perform the stimuli generation. The next sections are organized as follows: Section 2 explain the generalities of VEasy. Section 3 explains the state-of-the-art stimuli generation and later shows VEasy's approach to perform this task. Some results comparing VEasy with a Commercial simulator are presented on Section 4. Finally, conclusions are drawn on Section 5.

2. VEasy

The tool suite comprehends four main modules: Verilog RTL linting, Verilog RTL simulation, the Testcase Creation methodology and the coverage collection and analysis. Also, the tool suite has two distinct work-flows: the assisted flow and the simulation flow. The Verilog linting [6] is available only in the assisted flow of the tool, which starts when the Verilog description of the DUT is parsed and analyzed. The simulation flow is only enabled when the description complies with the linting rules. Linting guarantees that the input is written using strictly RTL Verilog constructions.

The input of the simulation flow then is no longer a hardware description, instead it is a verification plan file. The verification plan file used by VEasy is actually a complete view of the verification. It includes the traditional lists of features and testcases but it also contains simulation and coverage data. This approach makes it an unified database of the current verification progress. This approach is sometimes referred as an executable

verification plan.
The quality of the verification relies on coverage metrics [7]. VEasy has integrated three different structural coverage metrics: block, expression and toggle coverage. All these metrics are widely used by verification teams and are quite simple.
For the purpose of creating stimulus for the DUT, VEasy uses a layered methodology. This methodology enables testcase creation using a GUI and/or a specialized language, where the user is able to create complex sequence scenarios by using layers of abstraction. VEasy will automatically extract the design interfaces and create a basic sequence that contains all the physical inputs of the design. Such sequence belongs to *layer0*. All the other sequences (and layers) must be built by the user.

A layer is just a container used in the construction of sequences. All the layers constitute a hierarchy, where the *layer1* sequences contains items from *layer0* while a *layerN* sequence contains items from *layerN-1* and below. Each sequence has a name and a list of members. Each member can have a list of rules. All the layers except the *layer0* one can also have a list of sequences. The ability of combining more and more layers allow the construction of sophisticated testcases. One example of a verification environment built using the layered methodology is shown in fig. 1.

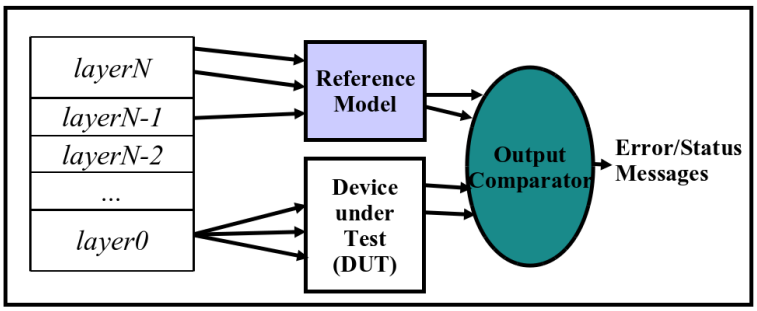


Fig. 1 – Layered methodology.

The layered methodology is the most complex aspect of VEasy and it creates the backbone that supports the simulation since it will provide the simulation stimuli. On top of the methodology there is also a constraint solving engine that allows a hierarchical control of the data being generated. More details regarding the methodology are found in [4]. Our results regarding generating stimuli data are reported in the next sections.

3. Stimuli Generation

The traditional way to verify a circuit is to apply signals to the inputs and check the output response. Such signals are organized in a list of tests, usually referred as testbenches. Such list is commonly generated and managed, although some random and automated test generators are described in the literature [8, 9]. Regarding design complexity, the use of this approach cannot ensure that all the cases referred in the test list were implemented. So, most of the effort is still manual.

Recently, the industrial practice is turning their thinkings to a constraint based approach [10], where some constraints limits and manages the input values that are sent to the DUT. This allow the simulation to reach more distributed cases within a testbench. Note, however, that the employment of constraints usually requires the use of a constraint solving engine. This raises the complexity level of the FV since it requires the verification team to have a deeply knowledge of the design, otherwise it may cause the simulation of a non-desired scenario. There is also the issue of constraint contradiction, i.e., one constraint inhibits the other. Either way, simulation cycles could be wasted if the constraints were incorrectly defined.

Moreover, there is another approach used for generating stimuli. It is called Coverage Directed Generation (CDG), which is based on detecting the occurrence of events in the simulation (by means of coverage) and providing information related to the progress of the same [11]. Then, the coverage results might be analyzed by the verification engineers, which could modify the directives for the test generators as necessary.

VEasy has its own simulation engine, which manages the stimuli generation with its own approach. In order to achieve a larger number of cycles simulated per second, VEasy integrates the testcase generation, the coverage collection and also the circuit description within the same simulation snapshot. This simulation engine is based on a cycle-accurate approach, which enables the user to focus on generating data instead of controlling the timing of the testbench.

4. Simulation Results

In order to evaluate the stimuli generation of the tool, some simulations were done using circuits with different profiles. The circuits were chosen based on the logic construction they contain: *seq* is a finite state machine activated by a sequence, *t6507lp* is a full microprocessor with 100 opcodes and 10 different addressing modes, *dffnrst* is a D-type flip-flop, *fsm* is a finite state machine with 8 states in which each state performs an 8-bit

wide operation, while *adder* is a 16 bit adder. The circuits are described in Tab. 1, which contains the amount of functional inputs of each circuit, i.e., excluding reset and clock signals.

Tab.1 – Amount of inputs of each simulated circuit.

DUT	*seq*	*t6507lp*	*dffnrst*	*fsm*	*adder*
Inputs	10	1	1	2	3

One Commercial simulator from a major vendor was chosen. For each circuit a testbench was created using a set of configurations: all the input signals are kept completely random except for clock and reset, which is triggered only once during the first cycle. All testbenches were configured to run up to 10 million clock cycles. The simulation was performed by first executing the testbench only. Only later the DUT was connected. The results are presented in fig. 2.

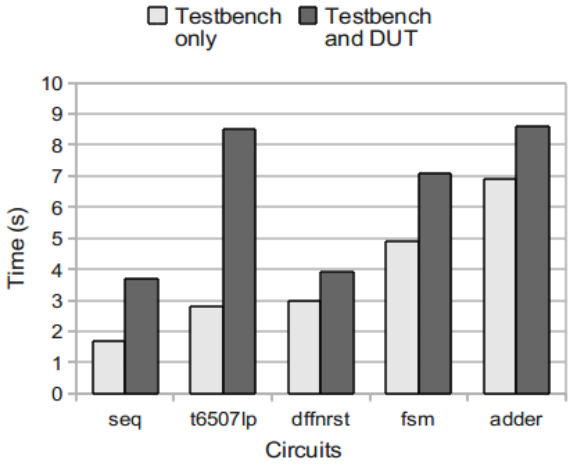


Fig. 2 - Simulation on Commercial simulator.

Fig. 2 presents the data of the simulation performed with and without the actual DUT. One might notice that the time required to simulate the testbenches is meaningful with respect to the time required to simulate the DUT. The testbench might represent, in the worst case scenario, 80% of the overall simulation time.

Fig. 3 shows the data of the simulation performed on VEasy. From this simulation one might notice that the testbench simulation time does not have a direct correlation with the complexity of the DUT. On simple circuits like *dffnrst*, *fsm* and *adder* the time spent on testbench constitutes a major cost on VEasy simulation. However, regarding *t6507lp*, the largest of the circuits, the simulation time of the testbench is not that high. An explanation for such behavior lies in fig. 4.

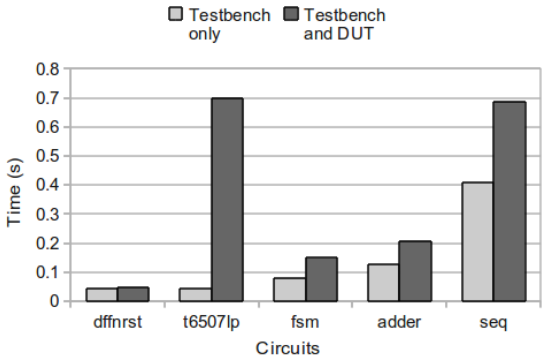


Fig. 3 - Simulation on VEasy.

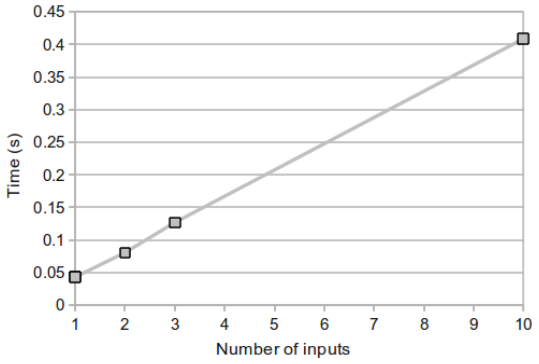


Fig. 4 – Simulation time in respect to the number of inputs of the circuits.

First, one should notice by comparing fig. 2 and fig. 3 that VEasy performs, on average, the same simulation within less than 10% of the time Commercial simulator requires. This gain occurs because the simulation is performed in a cycle-accurate simulator. The hierarchy and organization of layers enables a simple and faster stimuli generation which, when linked with the cycle-accurate simulator, enables a high performance simulation. In other words, the stimuli generation mechanism also benefits from the cycle-accurate simulation.

Regarding fig. 4, the amount of time that is required to simulate a testcase in VEasy is evaluated with respect to the actual number of inputs of the circuits. One might notice that the simulation time depends almost only on the amount of inputs that each DUT has, in a linear fashion. The explanation for this behavior lies in the

actual methodology used: deciding the next meaningful data to be simulated is fairly simple and has a fixed cost when the layered methodology is used. So, the major component of the stimuli generation is actually the draw of random numbers. The actual draw of a random number in VEasy was also addressed in our experiments. Such task is efficiently handled by the algorithm described in [12]. These two components, the random number generator and the hierarchical methodology, enable a high-performance stimuli generation.

5. Conclusion

Regarding the verification of a circuit, many solutions have been proposed. However, the technique that has been actually used in industry is FV. In that context, this paper described VEasy, which contributes with the growing of this area by providing a layered stimuli generation mechanism.
The goal of this paper is not to describe each detail of the tool, but mainly to explain the stimuli generation. On this issue, simulation comparisons between VEasy and a Commercial simulator were presented. For every simulated circuit, VEasy was able to realize the same simulation within a smaller time. Also, on each testbench, the time spent on simulation is almost directly linked with the amount of inputs, i.e., the inner complexity of the DUT does not affect the testbench simulation time when the methodology is used. Only the amount of random numbers to be drawn is affects the simulation time.

6. References

[1] D. Dempster, M. Stuart, and C. Moses, Verification Methodology Manual: Tchniques for Verifying HDL Designs, 2nd ed. Teamwork International, 2001.

[2] A. Piziali, “Functional Verification Coverage Measurement and Analysis”. Kluwer Academic, 2004.

[3] A. Molina and O. Cadenas, “Functional Verification: Approaches and Challenges”, in Latin American Applied Research, 2007.

[4] S. N. Pagliarini, “VEasy: a Tool Suite Towards the Functional Verification Challenge”, Master´s thesis, PGMICRO, UFRGS, Porto Alegre, 2011.

[5] S. N; Pagliarini. (2010) VEasy – a Functional Verification Tool Suite. [Online]. Available: http://www.inf.ufrgs.br/~snpagliarini/veasy

[6] L. Bening and H. Foster, “Principles of verifiable RTL design: a functional coding style supporting verification processes in Verilog”. Springer, 2001, ch. 4.

[7] Grinwald, R. et al. User Defined Coverage, “A Tool Supported Methodology for Design Verification. In: Design Automation Conference, San Francisco, United States, Proceedings... [S.l.: s.n.], 1998. p.158-163.

[8] R. Emek, I. Jaeger, Y. Naveh, G. Bergman, G. Aloni, Y. Katz, M. Farkash, I. Dozoretz, and A. Goldin. “X_Gen: A random test-case generator for systems and SoCs”. In IEEE International High Level Design Validation and Test Workshop, pages 145-150, October 2002.

[9] J.-T. Yen and Q. R. Yin., “Multiprocessing design verification methodology for Motorola MPC74XX PowerPC microprocessor”. In Proceedings of the 37th Design Automation Conference, June 2000.

[10] “Constrained-random test generation and functional coverage with Vera”, Technical report, Synopsys, Inc, Feb, 2003.

[11] Fine, S.; Ziv, A., “Coverage directed test generation for functional verification using Bayesian networks”. In: Design Automation Conference, 2003. Proceedings. 2003. p.286 – 291.

[12] Saito, M.; Matsumoto, M. “SIMD-Oriented Fast Mersenne Twister: a 128-bit pseudorandom number generator”. In: Monte Carlo and Quasi-Monte Carlo Methods 2006. [S.l.]: Springer, 2008. p.607–622.

Using Transistor Networks to Reduce Static Power in CMOS Circuits

[1]Gerson Scartezzini, [2]Ricardo Reis
{gerson.scartezzini, reis}@inf.ufrgs.br

[1]PPGC-[2]PGMicro, Instituto de Informática
Universidade Federal do Rio Grande do Sul – UFRGS
Porto Alegre, Brazil

Abstract

The optimization of circuits to reduce power consumption is very important. To improve the reduction of power it is becoming important a physical design methodology where any logic function should be able to have its layout generate. Considering this physical design methodology, this paper is focused to show that the use of complex gates gives a better solution in terms of power and delay, than the traditional use of basic gates available in commercial cell libraries. The comparisons show an average reduction of 74% in leakage power and 21% in delay.

1. Introduction

With the development of new technologies for integrated circuit manufacturing, it became possible to create ever smaller components, enabling the integration of an increasing number of transistors on a same silicon piece. With each new technology generation, the circuits become smaller and faster. But the increasing number of transistor in a chip brings new design challenges to reduce power consumption and it has become a major source of concern [1][2]. Many research and methods have been developed in order to reduce power consumption without denigrating the performance of the device, as in [3]. The power consumption in CMOS circuits consists mainly of three factors [4]: (i) dynamic power, (ii) short-circuit power and (iii) static power.

For a long time, only dynamic power was a significant source of power consumption in CMOS circuits, thus, many techniques have been found to reduce the consumption. However, as transistors become smaller and faster, the consumption related to static power (leakage power) has become more and more significant.

According to the International Technology Roadmap for Semiconductors 2009, static power consumption is an important factor in the total power consumption of a CMOS circuit, increasing at a rate of 10 % per generation technology. Figure 1 shows a graph with the prediction of [5] for the overall power consumption of an integrated circuit, where can be noticed that the static consumption tends to increase over the year as dynamic power consumption.

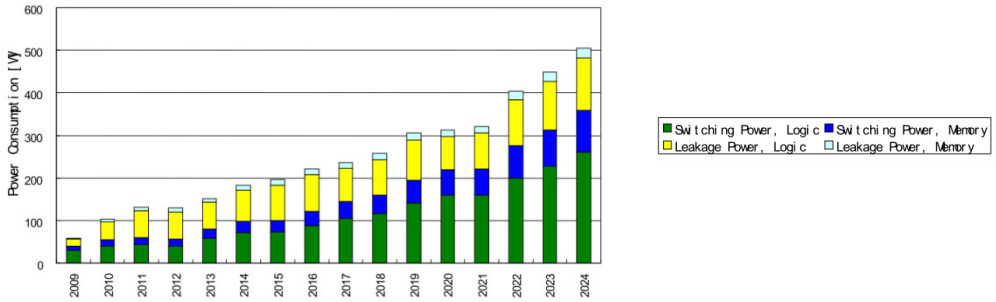


Figure 1: Power consumption prediction by the ITRS 2009.

In order to avoid high static power consumption, researchers have tried to search the sources of this consumption, and looked for new techniques to reduce them. Currently, a large number of designs are developed using methodologies such as the classic Standard Cell. That is, the layout of the circuit is made from the assembly of pre-designed cells taken from an existing cell library [6]. One major problem when using a cell library is that the number of logic functions in commercial cell libraries is limited, reducing the space of optimization of a circuit and reducing the possibilities do have a strong reduction on power consumption.

2. Use of Complex Gates

According [7], the static power consumption in CMOS circuits can be approximated by:

$$P_{leakage} = V_{DD} \cdot N \cdot k_{Design} \cdot I_{leak}$$

Equation 1: Architectural static power model, described in [7].

This works was partially supported by CNPq, Capes and Fapergs

Where $P_{leakage}$ is the static power consumption, N is the number of transistors, K_{design} is a design dependent parameter, and I_{leak} is a technology dependent parameter. I_{leak} depends on technology parameters like V_{th}, while K_{design} depends on design parameters like the fraction of transistors on any time.

From Equation 1, it is clear that the static power consumption is directly proportional to the number of components (N), thereby reducing the number of transistors in the system appears to be a very effective technique to reduce the static power consumption.

One way to reduce the number of transistors in a circuit requests a changing in the design methodology. A real physical design optimization cannot be obtained using the classical standard cell methodology. A true physical design optimization needs a methodology to allow the realization of any transistor network generated by the synthesis tools. To do so, it is necessary a set of tools to do the physical design of any transistor network. The use of complex cells (with several levels of AOI) can replace well a set of basic cells (NOR2, NAND2, for example).

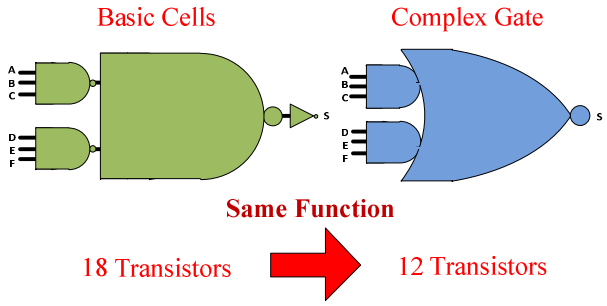


Figure 2: Different options for the design of a same function.

In Figure 2 is showed an example of a logical function described using basic cells and the same function described as a complex gate. It is clear in this example an important reduction in the number of transistors. The complex gate has 12 transistors and the equivalent function done using three basic gates (1 NOR2 and 2 NAND3) has 18 transistors.

3. Development and Results

In order to verify that the use of complex gates in integrated circuits provides reduced power consumption, we carried out an experiment, comparing complex gates with similar function using basic cells.

The process of comparison was conducted in four steps, namely: (A) Design & Sizing, (B) Layout & Extraction, (C) Simulation & Characterization and (D) Inspection & Comparison.

A. Design & Sizing

It was used a commercial 0.35µm PDK (Process Design Kit) from AMS (Austria Micro Systems). The first step of this work was the design of a set of cells. It was defined a set of 14 complex cells of type AOI (And, Or, Inverter) and 14 of the type OAI (Or, And, Inverter), in a total of 28 logic functions. For each one of this 28 cells it was described its transistor network using the SPICE language.

For each cell, pull-up and pull-down networks was sized with the same pair of "w" (w_n=1µm and w_p=1.6*w_n). In order to keep the same delay for both, pull-up and pull-down network, the Logic Effort [8] method was used in each cell.

B. Layout & Extraction

After the definition and sizing of cells, it was automatically generate the layouts of each one of the complex gates. The layout synthesis was done using the ASTRAN [9]. This tool was used to generate the network of transistors, as well to place and route the transistors of the transistor network described in SPICE format. In Figure 3 is illustrated the layout of the cell AOI232 generated using ASTRAN.

With the layout implemented, we used Virtuoso (from Cadence) to perform the electrical extraction of each cell. This extraction looks for parasitic elements, such as capacitances and resistances, generating a transistor level description of the circuit (including diodes, resistances and capacitors) from its layout, described by SPECTRE format.

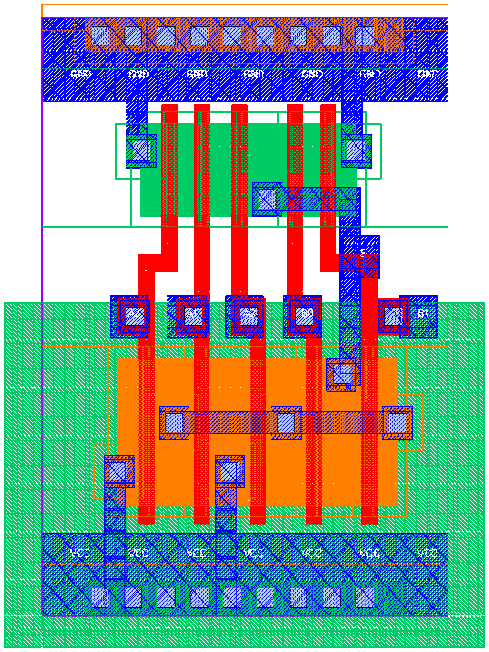


Figure 3: Layout of a complex gate (AOI232)

C. *Simulation & Characterization*

Using the SPECTRE description, each one of the cells was simulated using the SPECTRE simulator considering different input signals, in order to analyze rising time, fallen time, transition time, power consumption and delay.

It was generated a total of 3038 vectors to characterize the 28 cells. Each cell was characterized for a temperature of 25º and a voltage of 3.3V.

For each one of the vectors, the cell was simulated using different input slopes and output loads. Totally the circuit was characterized in all configurations of slopes and loads illustrated in Table 1.

Table 1: Input slope and output load used in characterization

Input Slope (ns)	0,05	0,5	1	2	
Output Load (pF)	0,001	0,005	0,02	0,08	0,16

To automate the characterization and generation of the simulation vectors, it was used a commercial tool called Encounter Library Characterizer (ELC), thus creating a file in liberty format (.lib) with the simulation results.

D. *Comparisons*

To compare the simulation results, it was calculated the average of the sums of all the results in each analysis. It was obtained a single value of transition, delay and power for each cell.

The same design process and transistor-level simulation was performed for the circuits composed by basic cells. Thus generating concrete data for comparison of both methodologies (use of any complex gate and only using basic cells). The results for all the logic functions are presented in the Table 2.

It can be observed that there is a reduction of more than 40 % on the number of transistors when using complex cells. Also there is an average reduction of 21 % in delay. The dynamic power has an average a small reduction of 1% but there is an important reduction of the leakage power by an average of 74%.

Table 2: Comparison of functions using basic cells or complex cells.

Cell	Basic Cells				Complex Gate				Reduction of # (%)	Reduction of Average Delay (%)	Reduction of Average Dynamic Power (%)	Reduction of Leakage (%)
	#	Average Delay (ns)	Average Dynamic Power (W)	Leakage (nW)	#	Average Delay (ns)	Average Dynamic Power (W)	Leakage (nW)				
AOI22	10	0,158	0,146	0,102	6	0,126	0,135	0,037	**40,0**	**20,4**	**7,8**	**63,8**
AOI23	12	0,125	0,118	0,104	8	0,103	0,112	0,041	**33,3**	**17,6**	**5,2**	**61,0**
AOI232	18	0,108	0,097	0,177	10	0,080	0,085	0,040	**44,4**	**26,4**	**12,2**	**77,6**
AOI2X22	16	0,128	0,116	0,172	8	0,093	0,105	0,038	**50,0**	**27,7**	**9,1**	**77,8**
AOI2X23	20	0,108	0,099	0,167	10	0,087	0,096	0,039	**50,0**	**20,0**	**3,6**	**76,5**
AOI2X233	26	0,083	0,074	0,246	14	0,064	0,073	0,042	**46,2**	**22,8**	**2,2**	**82,8**
AOI2X32	20	0,094	0,084	0,182	12	0,066	0,079	0,042	**40,0**	**29,2**	**6,1**	**76,9**
AOI2X323	28	0,075	0,067	0,251	16	0,060	0,070	0,043	**42,9**	**19,2**	**-4,3**	**82,8**
AOI2X33	22	0,085	0,077	0,176	14	0,065	0,076	0,040	**36,4**	**24,0**	**1,6**	**77,1**
AOI32	12	0,131	0,119	0,105	8	0,109	0,113	0,042	**33,3**	**17,0**	**5,0**	**60,3**
AOI323	20	0,095	0,087	0,171	12	0,074	0,085	0,040	**40,0**	**22,8**	**1,6**	**76,8**
AOI33	14	0,110	0,102	0,102	10	0,089	0,099	0,040	**28,6**	**19,1**	**3,6**	**60,9**
AOI3X23	26	0,094	0,085	0,242	12	0,073	0,081	0,039	**53,8**	**22,1**	**4,3**	**83,8**
AOI3X33	30	0,068	0,061	0,259	18	0,053	0,063	0,044	**40,0**	**22,8**	**-3,9**	**83,1**
OAI22	10	0,148	0,140	0,100	6	0,124	0,136	0,036	**40,0**	**16,5**	**2,8**	**63,6**
OAI23	12	0,114	0,108	0,104	8	0,097	0,109	0,040	**33,3**	**15,2**	**-0,6**	**61,0**
OAI232	18	0,105	0,096	0,178	10	0,080	0,093	0,040	**44,4**	**23,6**	**3,2**	**77,4**
OAI2X22	16	0,124	0,113	0,173	8	0,094	0,107	0,039	**50,0**	**24,1**	**5,4**	**77,6**
OAI2X23	20	0,101	0,092	0,168	10	0,081	0,093	0,041	**50,0**	**19,6**	**-0,8**	**75,6**
OAI2X233	26	0,081	0,072	0,247	14	0,064	0,076	0,042	**46,2**	**20,3**	**-5,5**	**82,8**
OAI2X32	20	0,091	0,083	0,182	12	0,072	0,083	0,042	**40,0**	**20,6**	**0,2**	**77,0**
OAI2X323	28	0,073	0,065	0,252	16	0,057	0,069	0,043	**42,9**	**20,9**	**-6,9**	**82,9**
OAI2X33	22	0,079	0,072	0,175	14	0,063	0,076	0,042	**36,4**	**20,0**	**-6,2**	**76,3**
OAI32	12	0,123	0,115	0,105	8	0,099	0,111	0,040	**33,3**	**19,6**	**3,4**	**62,1**
OAI323	20	0,089	0,081	0,171	12	0,072	0,084	0,041	**40,0**	**19,9**	**-4,2**	**75,8**
OAI33	14	0,100	0,094	0,103	10	0,081	0,096	0,041	**28,6**	**18,7**	**-2,0**	**60,5**
OAI3X23	26	0,090	0,080	0,242	12	0,069	0,080	0,042	**53,8**	**23,3**	**0,2**	**82,7**
OAI3X33	30	0,066	0,059	0,260	18	0,056	0,067	0,044	**40,0**	**14,2**	**-15,0**	**83,2**
								Average Reduction (%)	**41,4**	**21,0**	**1,0**	**74,3**

4. Conclusions

From Table 2, it is possible to realize a significant reduction in the number of transistors used in the development of complex gates compared to the same circuit function developed in basic cells, as a significant improvement in static power dissipation in form of leakage, reaching an average of 74% of reduction in leakage consumption.

The most important is that this reduction did not degrade the results of dynamic power dissipation and delay. Contrary to expectations, the average reduction of these results turned out positive, that is, on average there was improvement in the dynamic power dissipation and delay.

5. Future Works

As future work we intend to do an analogous work using technologies from 90 nm to 45nm. We expect even better results when using state-of-art technologies where static power has an increased significance.

6. References

[1] Kim, N.S.; Austin, T.; Baauw, D.; Mudge, T.; Flautner, K.; Hu, J.S.; Irwin, M.J.; Kandemir, M.; Narayanan, V. "Leakage Current: Moore's Law Meets Static Power"; In IEEE Computer Society; P. 68-75; Vol. 36, 2003.

[2] Jeong T. T. and Ambler P. A.; "Design Trade-Offs and Power Reduction Techniques for High Performance Circuits and System", In ICCSA 2006, pp. 531-536, vol. 3984.

[3] Borkar, S.; , "Design challenges of technology scaling", Micro, IEEE , vol.19, no.4, pp.23-29, Jul-Aug 1999.

[4] Henzler, Stephan; "Introduction to Low-Power Digital Integrated Circuit Design - Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies"; In: Springer Series in Advanced Microelectronics, 2007, Volume 25, 1-21, DOI: 10.1007/1-4020-5081-X_1.

[5] International Roadmap for Semiconductors 2009.

[6] Reis, R. e Cols., "Concepção de Circuitos Integrados", 2ª Edição. Série Livros Didáticos do Instituto de Informática, Editora Bookmann, Porto Alegre, 2009, 258 Páginas. ISBN 9788577803477.

[7] J. A. Butts and G. S. Sohi. "A static power model for architects", In Proc. of the 33rd Annual Intl. Symp. on Microarchitecture, 2000.

[8] Sutherland, I.; Sproull, B.; Harris, D. "Logical Effort: designing fast Cmos Circuits", San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[9] Zisesemer, A.; Lazzari, C., Reis, R., "Transistor Level Automatic Layout Generator for non-Complementary CMOS Cells", In: IFIP/CEDA VLSI-SoC2007, International Conference on Very Large Scale Integration, Atlanta, USA, October 15-17, 2007. pp. 116-121, ISBN: 978-1-4244-1710-0.

Transistor Sizing Analysis of Regular Fabrics

[1]Felipe Marranghello, [1]Vinicius Dal Bem, [2]Francesc Moll, [1]André Reis, [1]Renato Ribas
[1]{fsmarranghello,vdbem,andreis,rpribas}@inf.ufrgs.br
[2]moll@eel.upc.es

[1]PGMicro, UFRGS, Porto Alegre, Brazil.
[2]UPC, Barcelona, Spain

Abstract

This paper presents an extensive transistor sizing analysis for regular transistor fabrics. Several evaluation methods have been exploited, such as ring oscillators and single-gate open chain structures. Different design aspects are addressed taking into account stacked transistors and circuit critical paths. The performance degradation of using regular fabrics in comparison to standard cells is expected, but it is quite important to evaluate the dimension of such impact. The results were obtained for PTM 45nm CMOS parameters, and the conclusions can be easily extended to other technology nodes and fabrication processes.

1. Introduction

Systematic process variations have become a major issue for integrated circuits manufacturing due to the reduced dimensions associated to modern technologies, which are smaller than the wavelength used in photolithography [1]. These variations result in discrepancies between the designed layout and the manufactured product, leading to unpredictable behavior [2].

Resolution enhancement techniques (RET), such as phase shift mask and optical proximity correction can be used to improve the layout quality for lithography processing [3][4]. However, these techniques are too expensive to be used in huge VLSI design with many distinguished layout patterns. Thus, reducing the number of allowed patterns is desirable.

Several techniques to improve lithography quality using regular layout have been studied and proposed in the literature [5]-[7]. One methodology for the utilization of dummy features to improve regularity is presented in [5]. The work of Smayling *et al.* [6] shows an approach to reduce variability on gates. None of these purposes is completely regular. A fully regular layout technique is the via-configurable transistor array (VCTA) fabric [7]. In this work, a regular transistor fabric (RTF) similar to VCTA is investigated.

Regular transistor fabrics are understood as matrixes of identically sized transistors forming a regular structure. The main purpose of RTF approaches is to minimize systematic process variations. However, due to the fixed and identical size of all transistors, it can lead to a significant circuit penalty in terms of performance and area.

Transistor sizing plays an important role on circuit performance. Since logic gates can not be sized individually when the RTF pattern is targeted, this task becomes even more critical than when addressing the conventional standard cell design methodology. If the transistors width in the RTF arrays is too small, some cells can present poor timing performance. On the other hand, if the transistors width is too large, the power dissipation can become a critical drawback, while timing improvement can be limited due to the increased gate capacitances.

This paper presents an extensive electrical analysis of RTF patterns. The focus is the transistor sizing impact on signal delay propagation and power consumption characteristics. Area overhead is not addressed herein and performance degradation due to local metal wiring parasites is considered a minor effect being overlooked.

The paper is organized as follows. Section 2 presents the single gate sizing with special attention to transistor stacks in NAND and NOR gates. Section 3 analyses the sizing impact on circuit critical paths. Section 4 presents a general analysis for RTF sizing, and Section 5 outlines the conclusions.

2. RTF Pattern Transistor Sizing

RTF is a regular layout style. Unlike usual standard cell design, it is not possible to perform sizing on each cell individually. All transistors of the same type (PMOS and NMOS) have the same channel length and width. Another difference is that all transistor gates are equally spaced.

This section discusses the impact of RTF patterns with different transistors widths on individual cells performance. The electrical simulations were carried out taking into account the PTM 45nm CMOS process parameters [9], where the minimum transistor channel length and width are 50 nm and 90 nm, respectively. The power supply voltage applied was 1.1 V, at 25 Celsius degree as operating temperature. The logic gates addressed in this analysis were the inverter, NAND and NOR gates, with 2- to 4-inputs.

2.1. Standard Cell Approach as Reference

Traditional standard cell design uses several layout compaction techniques to enhance performance. As example, the source/drain areas that have no contact to metal wire can be made narrower (compacted). This layout style leads to patterns that cannot be well processed during lithography in the most advanced technology nodes [1]. Indeed, design for manufacturability (DFM) rules have been developed to restrict the layout design patterns and improve the lithography quality [2]. A few DFM rules are followed in this work. All polysilicon stripes are vertical only and equally spaced.

In this section, logic gates are individually sized as done in traditional standard cells. Values of delay and power consumption are used as reference for the RTF sizing analysis.

The first cell taken into account was the inverter. The minimum ring oscillator period with fanout four was adopted as performance metric. NMOS width (Wn) was arbitrarily kept constant at the minimum allowed value, while the PMOS width (Wp) of all stages was increased until the maximum oscillation frequency was achieved. The result yields a PMOS width equal to 1.6 times the NMOS width.

In the case of NAND and NOR gates, the sized inverter is used as reference. The goal is to achieve an average performance similar to the inverter for both high-to-low (Td_hl) and low-to-high (Td_lh) delay propagations considering the worst case arch. The delays are measured considering the response of each cell under different input transition times and loads.

To prevent the cells from having huge gate capacitance (Cin), it is limited to four times the one of the reference inverter. It represents a hard constraint in this task. Table 1 shows the results, which are normalized in relation to the inverter characteristics Wn, Td_hl and Cin. Certainly, there are other values that could be used and may fit the established criteria, but it is impractical to consider all of them and often the differences are not so significant.

Table 1: Normalized transistor sizing, delay propagation and input capacitances.

Cell	Wn	Wp	Td_hl	Td_lh	Cin
INV	**1.0**	1.6	**1.0**	1.72	**1.00**
NAND2	1.9	1.8	1.0	1.80	1.42
NAND3	3.0	2.0	1.07	1.84	1.92
NAND4	4.3	2.2	1.10	1.95	2.50
NOR2	1.4	4.0	1.08	1.75	2.08
NOR3	1.8	6.2	1.20	1.98	3.08
NOR4	2.2	8.0	1.28	2.10	3.92

Even though there is no PMOS stack in NAND2 gate, in such situation the PMOS transistors are larger than the one in the inverter because the internal cell capacitances (drain/source areas) are increased due to the larger NMOS present in the transistor network arrangement.

As the stack size increases, it becomes harder to equalize the delays. Furthermore, PMOS transistors are less sensitive to width increase than NMOS transistors. Thus, the NOR4 gate is the most difficult cell to size appropriately.

2.2. RTF Sizing for Basic Gates Design

As mentioned before, in the RTF approach all transistors of the same type (PMOS and NMOS) must have the same channel width. Even though, wider transistors cab be obtained using folding at the cost of area. The adoption of small sizing values tends to lead to poor timing performance of cells like NAND4 and NOR4. On the other hand, large transistor width values can result to over sizing the smaller cells, increasing significantly power consumption due to the increased parasitic capacitances related to the input transistor gate and drain/source transistor regions.

In order to evaluate the impact of the transistor sizing definition in the RTF pattern, each cell has been simulated taking into account the seven transistors widths pairs from Table 1, corresponding to the optimal sizing of each gate evaluated (INV, NAND 2-4, NOR 2-4). Each pair is called here as a RTF configuration, being RTF1 the one with Wn and Wp sizing of the inverter in Table 1, RTF2 corresponds to the transistors width of NAND2 in this table and so on.

Since the results strongly depend on the chosen transistor sizing values, three additional Wn and Wp pairs were also considered in this analysis. They are:

- the NMOS width of NAND4 and the PMOS width of the NOR4, in Table 1, referred as ‘RTF_WC’ configuration;
- the average width values from Table 1, for each kind of transistor, referred as ‘RTF_Avg1’ configuration;
- the average width values from Table 1, but excluding the worst cases NAND4 and NOR4 gates, referred as ‘RTF_Avg2’ configuration.

The values obtained for the RTF_WC, RTF_Avg1 and RTF_Avg2 configurations are shown in Table 2. Values are normalized to Table 1. Widths are normalized to the inverter Wn. Cin is the input capacitance of an inverter built with those widths, normalized in relation to the minimum one. The RTF_WC configuration is expected to present the smaller delay propagation but the highest power consumption. RTF_Avg1 and RTF_Avg2 configurations may present a trade-off that compensates the utilization of small transistors on some cells by over sizing transistors on other ones.

Table 2: Additional RTF templates (Wn and Wp) by considering some specific data from Table 1.

	Wn	Wp	Cin
RTF_WC	4.3	8.0	4.73
RTF_Avg1	2.3	3.7	2.31
RTF_Avg2	1.8	3.1	1.88

Seven ring oscillators, each one designed with one of the cells targeted (INV, NAND 2-4, NOR 2-4) were evaluated. All the RTF configurations defined before were simulated. Table 3 shows the measured period. The values are normalized to the period obtained using the configuration of Table 1 for each specific cell under test.

As expected, the use of larger transistors does not guarantee a better result on delay propagation in the ring structure (i.e., minimum oscillating period) due to the increase in input capacitances observed in each stage. For instance, in the case that RTF3 or RTF4 templates are used to implement a NAND2 gate, the ring oscillator period becomes higher than when using RTF2. Even when the period is minimized, the loss on power consumption can significantly increase. For example, considering the inverter gate, if RTF7 template is adopted the period is 13% smaller, but the input capacitance is almost four times higher.

Table 3: Normalized ring oscillator period for different logic gates designed in several RTF sizings.

	INV	NAND2	NAND3	NAND4	NOR2	NOR3	NOR4
RTF1	**1.00**	1.07	1.18	1.20	1.13	1.22	1.40
RTF2	0.97	**1.00**	1.03	1.04	1.15	1.30	1.50
RTF3	1.02	1.01	**1.00**	1.01	1.27	1.43	1.74
RTF4	1.08	1.05	1.01	**1.00**	1.40	1.63	1.99
RTF5	0.99	1.19	1.33	1.47	**1.00**	1.03	1.07
RTF6	1.01	1.26	1.41	1.60	0.99	**1.00**	1.02
RTF7	0.87	1.27	1.43	1.63	0.99	1.01	**1.00**
RTF_WC	0.87	1.02	1.08	1.18	0.96	1.22	1.07
RTF_AVG1	0.90	1.01	1.07	1.15	0.99	1.38	1.16
RTF_AVG2	0.92	1.03	1.11	1.19	1.00	1.65	1.16

3. Critical Path Analysis

So far, the evaluations were only performed on single logic gates. To have a better idea about the RTF impact on circuit performance, six benchmark circuits were randomly created. The cell amount ranges from seven in CKT1 to 50 in CKT6. Table 4 shows delays results normalized to the delay obtained when the cells are individually sized using the values of Table 1. For simplicity, all cells have only one fanout and the load of the circuit is four inverters of the given RTL configuration.

Table 4: Delay propagation in critical paths extracted from benchmarks, for different RTF sizings.

	CKT1	CKT2	CKT3	CKT4	CKT5	CKT6
StdCell	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RTF1	1.83	1.72	1.29	1.35	1.28	1.16
RTF2	1.85	1.89	1.27	1.38	1.30	1.14
RTF3	2.05	2.12	1.37	1.52	1.42	1.23
RTF4	2.29	2.36	1.48	1.67	1.56	1.34
RTF5	1.70	1.57	1.20	1.24	1.19	1.11
RTF6	1.72	1.56	1.22	1.25	1.21	1.13
RTF7	1.72	1.57	1.22	1.25	1.21	1.13
RTF_WC	1.55	1.54	1.10	1.17	1.10	1.00
RTF_AVG1	1.61	1.60	1.14	1.21	1.14	1.03
RTF_AVG2	1.63	1.60	1.15	1.22	1.15	1.05

As previously mentioned, RTF1 has all the gates with unitary capacitance. This leads to both small power consumption and high delay propagation. Also, larger transistors do not guarantee a smaller delay.

RTF1 may be good if delay is not only the main concern. RTF2, RTF3 and RTF4 have higher delay and higher capacitance than RTF1. RTF5 is faster than RTF1, but it has higher capacitance. RTF_WC presented the

best delay results but with very high capacitance. RTF_Avg1 and RTF_Avg2 show delay around 5% more than using RTF_WC, but with almost half capacitance.

RTF_Avg1 and RTF_Avg2 have a good trade-off. RTF_Avg2 presents better results because when the PMOS and NMOS sizes were chosen it was known that some cells would appear more times than others. Some configurations demonstrated no advantages. It is also interesting to note that some configurations could reach delay propagation near the one obtained using standard cell for some circuits.

4. General Analysis

Transistor sizing for regular fabrics is even a harder task than for traditional standard cells, because it is not possible to consider the cells individually. Previous designs may be used to estimate cell utilization if they are available. These gates are expected to have a significant impact on design performance. This way, transistor width can be chosen considering which cells are expected to present a major impact.

Extracting critical paths from circuits and sizing the related gates to meet a given constraint is also an option. It must be noticed that similar timing and power dissipation characteristics compared to standard cells are unlikely to be achieved. Using the last approach it may not be needed to evaluate the cells individually.

5. Conclusions

This paper presented an extensive electrical analysis of regular transistor fabrics. The restriction on allowed patterns in layout leads to better lithography yield, but there is degradation on performance. Adequate transistor sizing plays an important role in minimizing the gap between RTF and traditional standard cells with DFM rules. For this reason, several possible options for transistors widths are investigated as RTF sizing configurations. Their impact on delay propagation and power dissipation was measured comparing with the results from the standard cells. The analysis present in this work can be easily extrapolated to other technology nodes and fabrication processes. An important advantage of regular layout in relation to traditional standard cell design is variability reduction, not evaluated in this work. Integrated circuit designs must have a performance margin to compensate such an unpredictable behavior. With regularity this margin can be minimized. Thus, the performance loss due to layout restrictions may be compensated.

6. Acknowledgements

Research partially funded by Nangate Inc. under a Nangate/UFRGS research agreement, by CNPq and CAPES Brazilian funding agencies, and by the European Community's Seventh Framework Programme under grant 248538 - Synaptic.

7. References

[1] S. K. Springer, S. Lee, N. Lu, E. J. Nowak, J.-O. Plouchart, J. S. Wattsa, R. O. Williams, and N. Zamdmer, “Modeling of variation in submicrometer CMOS ULSI technologies,” IEEE Trans. on Electron Devices, vol. 53, no. 9, pp. 2168–2006, Sep. 2006.

[2] B. H. Calhoun, Y. Cao, X. Li, K. Mai, L. T. Pileggi, R. A. Rutenbar, and K. L. Shepard, “Digital circuit design challenges and opportunities in the era of nanoscale CMOS,” IEE Proccedings, vol. 96, no. 2, pp. 343-365, Feb. 2008.

[3] M. Lavin, F. L. Heng, and G. Northrop, “Backend CAD flows for restrictive design rules,” Proc. of ACM/IEEE Int’l Conf. Computer-Aided Design (ICCAD) , pp. 739–746, 2004.

[4] J. Wang, A. K. Wong, and E. Y. Lam, “Performance optimization for gridded layout standard cells,” Proc. of SPIE 24th Annual BACUS Symp. Photomask Technology., W. Staud and J. T. Weed, Eds., 2004, vol. 5567, pp. 107–118.

[5] P. G. Drennan, M. L. Kniffin, and D. R. Locascio, "Implications of proximity effects for analog design," Proc. of IEEE Custom Integrated Circuits Conf. (CICC), pp. 169-176, 2006.

[6] M. C. Smayling, H.-Y. Liu, and Lynn Cai, "Low k_1 logic design using gridded design rules", Proc. of SPIE 6925, 2008.

[7] M. Pons, F. Moll, A. Rubio, J. Abella, X. Vera, and A. González, “VCTA: a via-configurable transistor array regular fabric,” Proc. of VLSI System on Chip Conference (VLSI-SoC), pp.335-340, 2010.

[8] F. Beeftink, P. Kudva, D. S. Kung, R. Puri, and L. Stock, “Combinatorial cell design for CMOS libraries,” The VLSI Journal on Integration, vol. 29, no. 1, pp. 67-93, Mar. 2000.

[9] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," Proc. of IEEE Custom Integrated Circuits Conf. (CICC), pp. 201-204, 2000.

Computing Minimum Decision Chains of Boolean Functions

[1]Mayler G. A. Martins, [2]Vinicius Callegaro, [1,2]Renato P. Ribas, [1,2]André I. Reis
{mgamartins,vcallegaro,rpribas,andreis}@inf.ufrgs.br

[1]PGMicro, UFRGS, Porto Alegre, Brazil
[2]PPGC, UFRGS, Porto Alegre, Brazil

Abstract

Every Boolean function has a unique property called Minimum Decision Chain (MDC). This paper proposes an effective way to compute this property for arbitrary functions. The proposed method is very efficient when compared to a more direct and intuitive approach, that is used as the reference for performance analysis. Different examples have been evaluated, and the results are discussed. The proposed method is able to compute the MDC value in order of milliseconds, allowing the use of MDC computation to guide logic synthesis algorithms.

1. Introduction

Logic synthesis is a well-established research field that has direct impact in the quality of digital circuit design [1]. Logic synthesis methods may rely on the exploitation of Boolean functions properties that include positive and negative unateness, binateness and symmetry between variables [1]. One example of this is the unate recursive paradigm used in Espresso [2] to decompose functions recursively, leading to easy-to-solve operations on unate sub-functions. Other examples are methods to compute efficiently read-once formulas (equations where each variable appears at most once [3][4]). Advanced logic synthesis methods use Boolean function properties to guide the algorithms. As a consequence, fast methods to compute Boolean function properties are needed to allow the use of these properties in the logic synthesis flow.

There are sum-of-products (SOP) minimizers like Espresso-signature [5] that use the concepts of non-redundant minimal implicants (introduced by Perkowski *et al.* [6]) to avoid the explicit computation of all prime implicants. According to McGeer *et al.* [5], "the work of Perkowski *et al.* did not receive due attention, possibly because the only algorithm given was that of enumerating all minterms, generating the primes for each, forming the cube of their intersection, and casting out the cubes that are singly contained in any one other". This is a clear case where an important Boolean function property had its use avoided in practice due to the lack of an efficient algorithm to compute it.

This paper discusses the computation of 'minimum decision chains' (MDC). The MDC of a logic function is related to the number of transistors in series in switch networks that implement the logic function. Schneider *et al.* [7][8] have presented a method to compute MDCs, based on a modification of the Quine-McCluskey algorithm [9], referred herein as QMC-MDC. Marques *et al.* [10] have used QMC-MDC algorithm as a criterion to evaluate the feasibility of complex gates in a library-free technology mapping approach. However, the execution time of QMC-MDC algorithm was the main drawback. Hence, the need for a method for fast MDC computation is a challenge. This paper presents a novel and execution time efficient method to compute MDCs. The proposed method is based on functional composition [11].

2. Minimum Decision Chains

A prime implicant of an arbitrary function f can be viewed as a variable assignment such that the output of the function is true and if any variable is unassigned the value of f is undetermined. The minimum decision chain (MDC) of a Boolean function expresses the largest (worst case) number of variable assignments (i.e., number of literals in a prime implicant) necessary to cover a minterm of the function. As example consider the function given by Equation (1).

$$f = \bar{a} \cdot \bar{b} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{d} \cdot \bar{e} + a \cdot c \cdot d + b \cdot c \cdot \bar{d} \cdot e \quad (1)$$

From this equation, four variables (b=1, c=1, d=0 and e=1) are assigned to cover both minterm 13 (i.e., [abcde]=[01101]) and minterm 29 (i.e., [abcde]=[11101]). However, minterm 13 can be covered by assigning just three variables (a=0, d=0 and e=1) as the minterms 1 (i.e., [00001]), 5 (i.e., [00101]) and 9 (i.e., [01001]) also belong to the on-set of f. In the same way, to cover minterm 29 is necessary to assign only three variables (a=1, b=1 and c=1). As a consequence, f can be rewritten as Equation (2), where prime implicants have at most 3 literals each. This way, the largest (worst case) number of variable assignments (i.e. number of literals in a prime implicant) necessary to cover a minterm of the function f is 3, and thus the MDC is also 3, instead of 4.

$$f = \bar{a} \cdot \bar{b} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{d} \cdot \bar{e} + a \cdot c \cdot d + a \cdot \bar{d} \cdot e + a \cdot b \cdot c \quad (2)$$

The MDC of a function f is related to the minimum worst case number of series switches required to implement a switch network for the function f. If the MDC is K, then the function requires at least K switches (transistors) in series to be implemented. Additionally, the existence of a solution with at most K series switches is guaranteed. Notice that there is a MDC value for the on-set (i.e., for the minterms) and another value for the off-set (i.e., for the maxterms).

3. FC-MDC Procedure

FC-MDC computes MDCs by functional composition. It is based on a top-down implicant generation paradigm, as used in BOOM [12], in contrast to Quine-McCluskey method, where the implicants are generated bottom-up. Thus instead of increasing the dimensionality of implicants by omitting literals from their terms, the dimension of a term is gradually decreased by adding new literals. The FC-MDC method is based on truth tables that can be represented as integers or BDDs. In this paper, Boolean functions are represented as integers, while logic operations are performed quite fast by using bitwise operations of the processor ALU.

The Functional Composition for MDC computation is illustrated in Figure 1. It is the association of 1-literal element in 1-lit bucket with N-1 elements, creating so the N-lit buckets by only using bitwise AND operation. This process provides the product terms (smaller bucket [11]) that compose the function in SOP form. Thus, the generation of the N-literal bucket can be expressed by Equation (3).

$$B_n = \left(B_1 \cdot B_{n-1}\right) \mid n \geq 2 \tag{3}$$

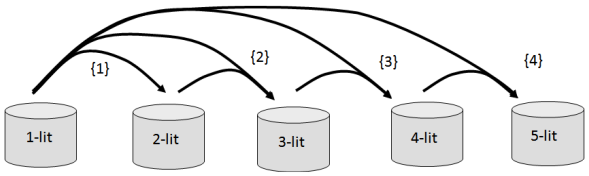


Figure 1 – Functional Composition for MDC computation.

The main idea of the proposed algorithm is depicted in Fig. 2, taking as an example the computation of the on-set MDC for the target function given by $f = a \cdot c + b \cdot c + \bar{a} \cdot b \cdot d$. The first step is to check if the target function is constant. In this case the algorithm returns the result as zero. The algorithm starts by filling the 1-literal bucket and only the literals with the right polarity are created, reducing the computation time. Then the 1-literal functions are multiplied (i.e., through bitwise logic AND) to create 2-literal functions, then 3-literal functions, and so on. When a smaller function is found, it is added (i.e., through bitwise logic OR) to an accumulated smaller function, which is in fact a sum-of-products. A smaller function can be used in a sum-of-products of the target function. It is possible to discover if the newly created functions are smaller than the target function by using bitwise logical operations. The product (bitwise logic AND) of a smaller function *sf* with the (known) target function results again in the smaller function *sf*.

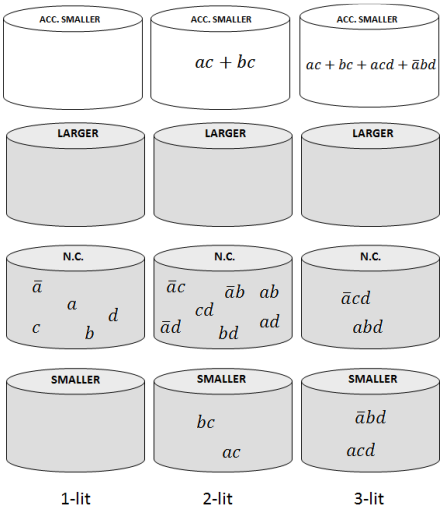


Figure 2 – Illustration of the on-set MDC computation for the function $f = a \cdot c + b \cdot c + \bar{a} \cdot b \cdot d$.

The MDC is found when the accumulated smaller function is equal to the target function. In this case, the (on-set) MDC is the number of literals of the current bucket being processed. In the example illustrated in Fig. 2, the on-set MDC is three, as the accumulated 'smaller' became equal to the target function in the 3-literal bucket. As an additional feature, the algorithm can be modified to terminate when the MDC value of the function being computed exceeds a certain threshold limit established by the user. In this case, the exact value of the MDC does not matter, as it is considered unfeasible. This way, the algorithm can be aborted before

finishing the process if the MDC of the function exceeds the largest value considered feasible by the user, avoiding unnecessary computation of an unfeasible MDC value. Notice that the use of this threshold value is only possible due to the way implicants are computed in FC-MDC, i.e. instead of increasing the dimensionality of implicants by omitting literals from their terms [9], the dimension of a term is gradually decreased by adding new literals [12]. This feature is not possible in the QMC-MDC procedure, due to the bottom up computation of primes.

3. Experimental Results

Experiments have been carried out to evaluate the efficiency of the proposed FC-MDC method to compute MDCs (compared to QMC-MDC [7][8]). The algorithms have been developed in C++ programming language, and test platform used was a Core2Duo 2.4 GHz with 4 GB RAM.

First of all, the QMC-MDC and FC-MDC algorithms were run over different sets of functions. The NPN (i.e., input **N**egation – input **P**ermutation – output **N**egation) representative functions of 4-input and 5-input variables, which contain 221 and 616,125 distinct functions, respectively [13]. A third target set used in the analysis is the library 44-6.genlib distributed in the SIS package [14]. It contains 3,503 functions with maximum four series/parallel transistors in the conventional static CMOS design style.

The average on-set MDC value obtained with these sets of functions and the execution times are shown in Table 1 for both addressed methods: QMC-MDC and FC-MDC.

Table1 - Total execution time of on-set MDC computation.

Function Set	#Functions	Average MDC	QMC-MDC	FC-MDC
4-NPN	221	3.46	34 ms	9 ms
5-NPN	616,125	4.67	92 s	104.3 s
44-6.genlib	3,503	3.87	> 4 h	5.9 s

The main reason for the best performance of the FC-MDC algorithm is associated to the fact that it does not need to fill many buckets when the MDC value is small. Notice that the cases where the MDC is small represent the functions of the most interest in digital design, as they correspond to feasible switch networks (and logic gates) with a small number of stacked transistors [1]. The QMC-MDC was observed to be too slow for computing the MDC of functions in the 44-6.genlib set because the set contains functions with up to 16 inputs where it is quite expensive to compute all the prime implicant terms.

It is very interesting to exploit the property of limitation provided by the FC-MDC, stopping the computation after a certain maximum MDC threshold determined by user. Table 2 shows the number of functions with a MDC value smaller or equal to a user defined threshold and the execution time necessary to compute it. FC-MDC considers the pre-defined MDC threshold to stop computation if the MDC of the function is larger than the limit established by the user.

Table 2 - MDC computation of 5-NPN through FC method, using pre-defined limit value.

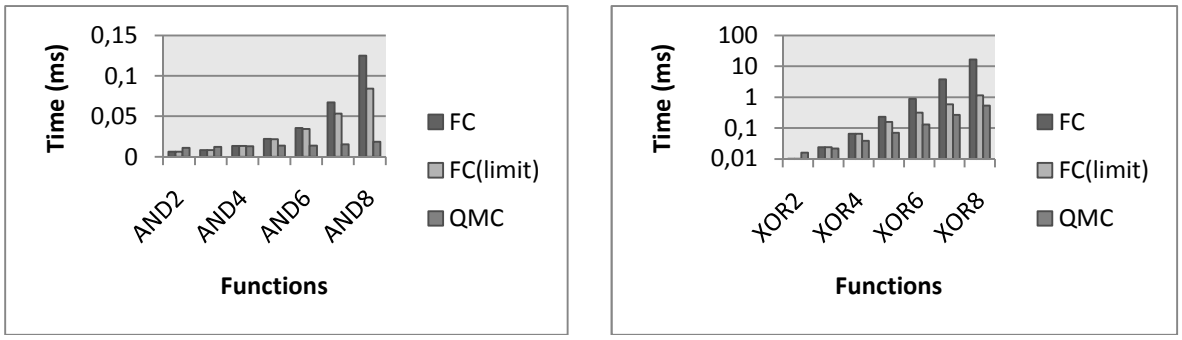
MDC (pre-defined)	# Functions	Time to process all 5-NPN
1	1	8.7 s
2	9	21.3 s
3	3,444	44.7 s
4	318,327	90.2 s
5	294,344	104.3 s

In the second phase of experiments, the behavior of QMC-MDC and FC-MDC were evaluated for specific functions, with special attention to the worst cases obtained with the FC-MDC method that are the best cases of QMC-MDC method. These cases happen when the functions have minterms that cannot be associated to produce larger cubes. AND and exclusive-OR (XOR) functions are examples of this. AND has single minterm, while XOR has many minterms that cannot be associated into larger implicant cubes.

Experiments were performed to investigate the corner cases of FC-MDC method against QMC-MDC, computing the MDC of AND and XOR functions, from 2 to up to 8 inputs, considering two situations: (a) full computation, and (b) using the pre-defined threshold limit of MDC equal to 4, that is the industry practical rule of thumb for the maximum transistor stack in digital CMOS integrated circuits.

Fig. 3a and Fig. 3b provide useful information about the two methods and are shown in log scale to better visualization. In Figure 3a, for AND function evaluation, the QMC method presents almost constant results, because all functions have always one prime implicant minterm. The FC method, in turn, presents an exponential time increasing. However, there is a run time reduction of circa 50% in the AND8 when exploiting the pre-defined limit parameter. In this case, the method is aborted and a limit overflow result is returned.

In the XOR function analysis, the execution time presented too in Figure 3b, the QMC based method presents an exponential run time increasing because the number of minterms is doubled for each XOR, by augmenting the number of input variables. On the other hand, the FC based method presents a more severe increasing than in the AND experiments.



(a) **(b)**

Figure 3 – MDC computation of AND (a) and XOR (b) with 2 to 8 inputs.

4. Conclusions

This paper proposed an efficient method to compute minimum decision chains (MDC) of logic functions. The method is based on functional composition [11] and it is referred as FC-MDC. The FC-MDC method was compared to QMC-MDC method [7][8], which is a slightly modified version of the well-known Quine-McCluskey algorithm [9]. The QMC-MDC method presents some limitations related to the number of prime implicants of the functions, as it has to compute nearly all prime implicants, making it quite computing expensive in some cases. The FC-MDC method is faster in most cases, especially in the cases of CMOS design interest, i.e., logic functions with MDC smaller than 5.

5. Acknowledgements

Research partially funded by Nangate Inc. under a Nangate/UFRGS research agreement, by CNPq and CAPES Brazilian funding agencies, and by the European Community's Seventh Framework Programme under grant 248538 - Synaptic.

6. References

[1] L. T. Wang, Y. W. Chang, and K. T. Cheng, Electronic Design Automation: Synthesis, Verification and Test. Morgan Kaufmann, 2009.

[2] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, 1984.

[3] M. C. Golumbic, A. Mintz, and U. Rotics, "Factoring and recognition of read-once functions using cographs and normality," Proc. of Design Automation Conference (DAC), pp. 109-114, 2001.

[4] M. C. Golumbic, A. Mintz, and U. Rotics, "An improvement on the complexity of factoring read-once Boolean functions," Discrete Applied Mathematics, vol. 156, no. 10, May 2008.

[5] P.C. McGeer, *et al.* "ESPRESSO-SIGNATURE: a new exact minimizer for logic functions," IEEE Trans. on VLSI, vol.1, no.4, pp.432-440, Dec 1993.

[6] L. Nguyen, M. Perkowski, and N. Goldstein, "Palmini - fast Boolean minimizer for personal computers," Proc. of Design Automation Conference (DAC), pp. 615-621, 1987.

[7] F. R. Schneider, R. P. Ribas, S. Sapatnekar, and A. I. Reis, "Exact lower bound for the number of switches in series to implement a combinational logic cell," Int'l Conf. on Computer Design (ICCD), pp. 357- 362, 2005.

[8] F. R. Schneider, A. I. Reis, and R. P. Ribas, "CMOS logic gates based on the minimum theoretical number of transistor in series," Norchip 2006, pp.85-88.

[9] E. J. McCluskey, "Minimization of Boolean functions," The Bell System Tech. Journal, vol.35, no.5, Nov.1956.

[10] F. S. Marques, L. S. Rosa Jr., R. P. Ribas, S. S. Sapatnekar, and A. I. Reis, "DAG based library-free technology mapping," Proc. of GLSVLSI, pp. 293-298, 2007.

[11] M. G. A. Martins, L. S. Rosa Jr., A. B. R. Rasmussen, R. P. Ribas, and A. I. Reis, "Boolean factoring with multiple objective goals," Int'l Conf. on Computer Design (ICCD), pp. 229-234, 2010.

[12] J. Hlavicka and P. Fiser, "BOOM - a heuristic Boolean minimizer," Proc. of ACM/IEEE Int'l Conf. Computer-Aided Design (ICCAD), pp. 439-442, 2001.

[13] V. P. Correia, A. I. Reis, "Classifying n-input Boolean functions," Iberchip 2001, pp. 58-66.

[14] E. M. Sentovich, *et al.* "SIS: a system for sequential circuit synthesis," Technical Report UCB/ERL M92/41, UC Berkeley, 1992.

Video Coding 1

Multiprocessing Acceleration of H.264/AVC Motion Estimation Full Search Algorithm under CUDA Architecture

Eduarda R. Monteiro, Bruno B. Vizzotto, Cláudio M. Diniz, Bruno Zatt, Sergio Bampi
{ermonteiro, bbvizzotto, cmdiniz, bzatt, bampi}@inf.ufrgs.br
Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil

Abstract

This work presents a parallel GPU-based solution for the Motion Estimation (ME) process in a video encoding system. We propose a way to partition the steps of Full Search block matching algorithm in the CUDA architecture, and to compare the performance with a theoretical model and two implementations (sequential and parallel using OpenMP library). We obtained a $O(n^2/\log^2 n)$ speed-up which fits the theoretical model considering different search areas. It represents up to 600x gain compared to the serial implementation, and 66x compared to the parallel OpenMP implementation.

1. Introduction

In the past decade, the demand for high quality digital video applications has brought the attention of industry and academy to drive the development of advanced video coding techniques and standards. It results in the publication of H.264/AVC [1], the state-of-the-art video coding standard, since it provides higher coding efficiency compared to previous standards (MPEG-2, MPEG-4, H.263). In order to achieve higher coding efficiency, the standard introduces increased computational complexity to implement the encoder and decoder.

In this scenario, Motion Estimation (ME) [2] is a key issue in order to obtain high compression gains. It explores the temporal redundancy of a video by searching at reference frames the most similar region in the current frame. When the best 'match' occurs, a motion vector is calculated. It provides the most part of compression gains for H.264/AVC. ME algorithm and associated similarity criteria to determine the best 'match' is an important encoder issue not addressed in H.264/AVC. It requires intensive computation and memory communication for block matching task representing 80% of the total computational complexity of current video coders [3].

However, some block matching algorithms for ME have a great potential of parallelism. Full-search (FS) [1] is one of these algorithms, which performs the search for the best match exhaustively inside a search area. The best match is achieved by calculating the similarity for each block position inside the search area, using a similarity criterion, e.g. Sum of Absolute Differences (SAD). One can observe that the calculation of SAD for one block does not depend on the calculation of the previous block. So these two steps could be calculated simultaneously, in parallel.

By exploring the inherent parallelism potential of ME FS algorithm and the huge computational capacity of recent Graphic Processing Units (GPUs), this work presents a parallel GPU-based solution for the FS block matching algorithm implemented on Compute Unified Device Architecture (CUDA), from NVIDIA [4]. We present here how we efficiently mapped the FS algorithm to the CUDA programming model. Further, the obtained results from the execution of FS implementation with real videos were compared with a serial and parallel OpenMP implementation. We also made a comparison with a theoretical complexity model in terms of computation and communication.

2. Motion Estimation

The main goal of ME is to find in the previously reconstructed frames (reference frames) a block that more closely resembles the block of the current frame, thus reducing the temporal redundancy between frames to be transmitted. The displacements are mapped to motion vectors and associated residue. From one or more reference frames, a motion vector is generated for each block of the current frame and the corresponding block position with highest similarity to the reference frame.

The optimum search to "match" the blocks is carried through exhaustively in a search area using a search algorithm and a similarity criterion. The search area is a region in the reference frame formed around the co-located position of the block to be coded, in the current frame. By the end of the search, the optimum block, i.e. the most similar block using the similarity criterion, is located and a motion vector is generated which indicates the position of this block in the reference picture. This process is better described in Fig. 1.

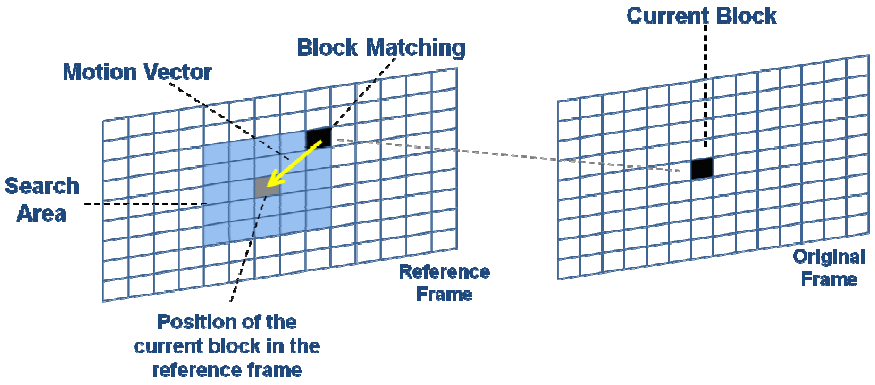


Fig. 1 - Optimum Motion Estimation Algorithm Diagram

3. Proposed Implementation

In this work, we used the FS block matching algorithm for ME. Among the existing search algorithms for ME this algorithm has the highest computational cost and its main objective is to finds the optimum block of the current picture by matching all possible positions (sample by sample) of the search area. In such a way, it can be said that this exhaustive algorithm is capable to generate excellent motion vectors. The similarity criterion used in this work is SAD. Its equation is shown in (1).

$$SAD(R,O) = \sum_{j=0}^{n-1}\sum_{i=0}^{n-1}|R(i,j) - O(i,j)| \quad (1)$$

In (1), *w* is the width and *h* is the height of both the candidate and the current block, *R* is a candidate block in the reference frame and *O* is the current (original) block. The candidate block that presents the lowest SAD value is the best block to represent the current block in the reference frame. The position (x,y) of the best candidate block is represented as a *motion vector*.

Our hardware system target is formed by CPU and GPU. CPU manipulates video to separate the original and reference frames, and feeds the GPU, as shown in Fig. 2. The FS algorithm is divided into two steps: i) SAD of all candidate blocks inside a search area; ii) the comparison to find the best match (lower SAD).

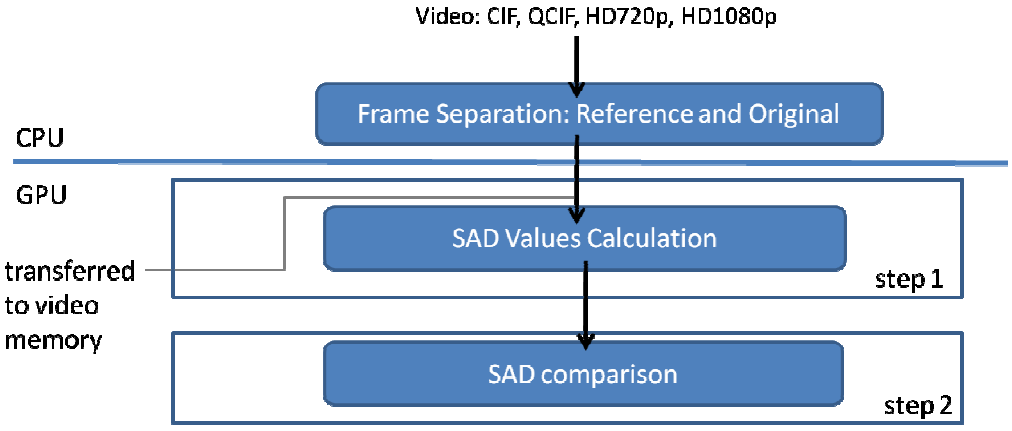


Fig. 2 - Proposed Algorithm Flow

Our target GPU is from NVIDIA which supports CUDA programming. CUDA architecture was proposed by NVIDIA in 2007 [4] with the objective to exploit the high degree of inherent parallelism to their graphical devices. The great computational power offered by this type of technology has made of this architecture a great prominence in diverse areas, in special in the scientific community. Considering the processing hierarchy of CUDA architecture for software implementation we partition our algorithm as shown in Fig. 3. A set of threads (basic processing unit) is organized in blocks, which are elements of a grid. In our work, ME algorithm mapping for GPU is composed by a kernel (procedure to be executed in GPU) where the grid and the block sizes are defined by the video resolution and the search area, respectively. Each thread computes a 4x4 video block.

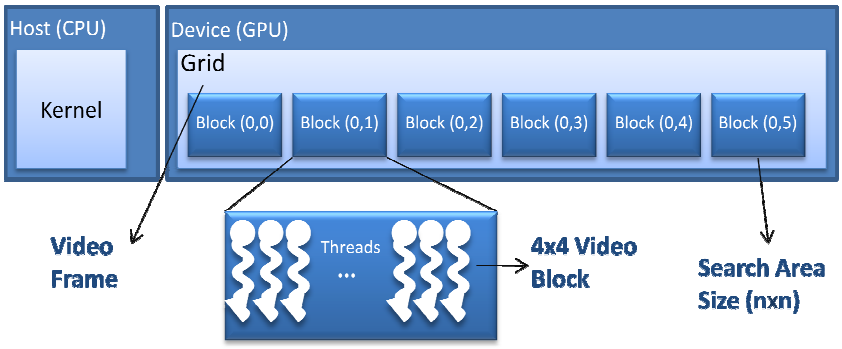
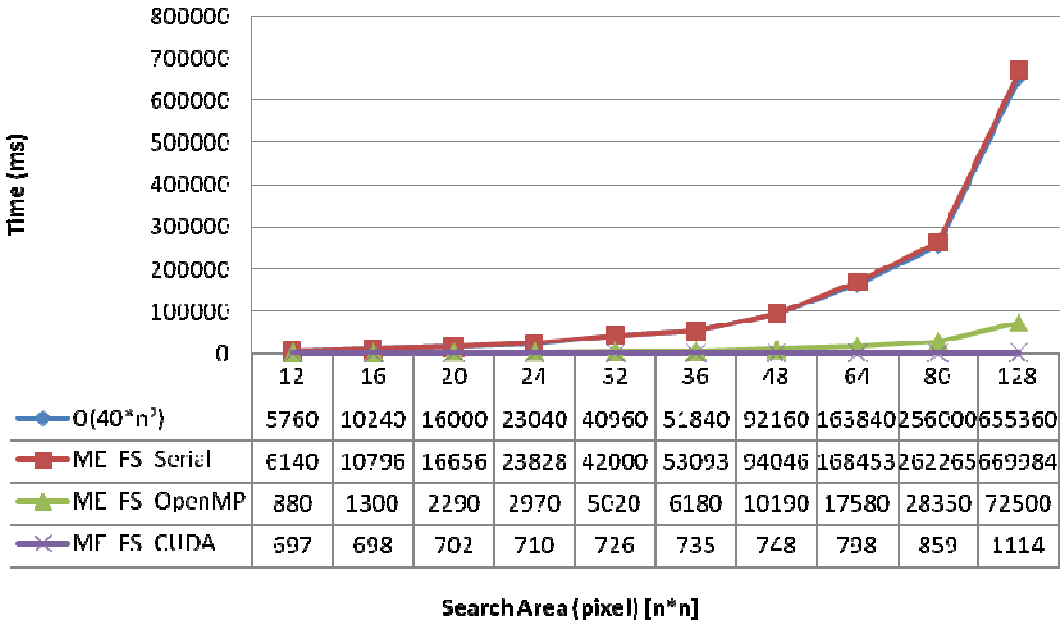


Fig. 3 - CUDA Programming Model – Algorithm Allocation

4. Results

Tests were performed for three video resolutions (CIF, HD720p and HD1080p) comparing the CUDA implementation running on an NVIDIA GTX 480 @ 1.40GHz (480 functional units) to both serial and parallel OpenMP implementation running on a Core 2 Quad Q9550 @ 2.82GHz. Two frames of each of the following video sequences were used: ***(i)*** CIF: Foreman, Flower and Bus; ***(ii)*** HD720p: Mobcal; ***(iii)*** HD1080p: BlueSky.

Results obtained with CIF resolution video shows that the serial implementation is in accordance with the theoretical model. The experimental data fitting is approximately $40n^2$ for n ranging from 12 to 128, where $n*n$ are the number of pixels inside a search area as shown in Fig. 4. Speed-up results were also consistent with the theoretical model $O(n^2/\log^2 n)$ considering different search areas, as shown in Fig. 5. The obtained speed-up represents up to 600x gain compared to the serial software implementation, and 66x compared to the parallel OpenMP software implementation (which generates at most 4 threads in parallel in the quadcore processor). The high speed-up gains achieved were due to the high thread parallelism provided by GPU (compared with OpenMP and serial implementations) combined with the absence of data dependence between blocks in the FS algorithm.



	12	16	20	24	32	36	48	64	80	128
O(40*n²)	5760	10240	16000	23040	40960	51840	92160	163840	256000	655360
ME FS Serial	6140	10796	16656	23828	42000	53093	94046	168453	262265	669984
ME FS OpenMP	880	1300	2290	2970	5020	6180	10190	17580	28350	72500
ME FS CUDA	697	698	702	710	726	735	748	798	859	1114

Fig. 4 - Theoretical model (in blue) versus obtained results for CIF videos

Moreover, the time of communication between GPU and CPU was also considered. Fig. 6 present the total time of execution for Full-HD 1080p resolution in relation to the time of communication between the devices (CPU-GPU). The total execution time was measured using a timer function from *cutil.h* C/C++ library. The timer is initialized before the GPU kernel call and finalized after the execution is completed. The communication time uses a timer function from *ctime.h* C/C++ library. When data is transferred between CPU to GPU the counter is initialized, and likewise for GPU to CPU transfer. It can be noted in Fig. 6 that the time of communication for manipulation of the data between CPU and GPU (and vice-versa) is worthless compared with total time execution.

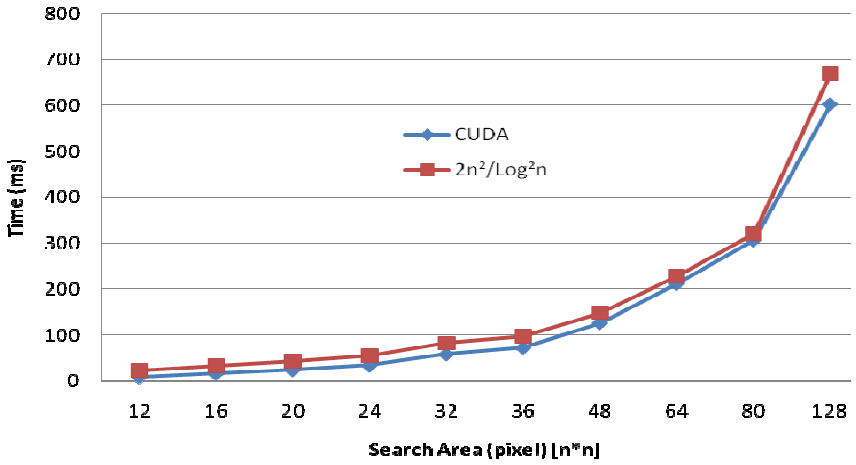


Fig. 5 - Theoretical speed-up (in red) versus obtained speed-up with CUDA (in blue) for different search area

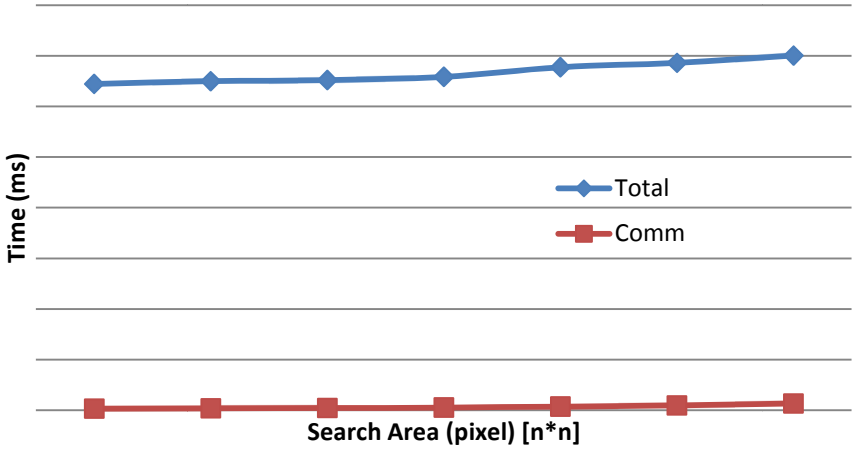


Fig. 6 – Total Execution Time x Communication Time for FullHD 1080p resolution

5. Conclusions

In this paper, we demonstrate that the use of GPU is a good alternative for acceleration of video coding, more specifically the ME. Although we use the FS exhaustive algorithm, the execution in GPU revealed fast compared to serial and OpenMP implementations, as well as to the behavior of the theoretical model previously calculated. Future works could focus on better exploration of data sharing from CPU to GPU, memory hierarchy inside GPU and also the implementation of sub-optimal search algorithms.

6. References

[1] BHASKARAN, V.; KONSTANTINIDES, K. Image and Video Compression Standards: Algorithms and Architectures. 2nd ed. Boston: Kluwer Academic Publishers, 1999.

[2] ITU-T Recommendation H.264 (03/10): advanced video coding for generic audiovisual services, 2010.

[3] HUANG, Y-W; et. al. Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results. *J. VLSI Signal Process. Syst.* 42, 3 (March 2006), pp. 297-320.

[4] CHENG, Y., CHEN, Z. and CHANG, P., "An H.264 Spatio-Temporal Hierarchical Fast Motion Estimation Algorithm for High-Definition Video", IEEE International Symposium on Circuits and Systems, ISCAS, pp. 880-883, 2009.

[5] NVIDIA, Corp. Available at: http://www.nvidia.com

Synthesis and Comparison of Low-Power Architectures for SAD Calculation

Fábio Walter, Sergio Bampi
{flwalter,bampi}@inf.ufrgs.br

PGMICRO - Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS, Brasil

Abstract

This paper presents the standard-cells synthesis and comparison of parallel hardware architectures for the Sum of Absolute Differences (SAD) datapath, focusing on different design points such as the tradeoff between high-performance and low-power dissipation (isoperformance target). Multi-VDD, multi-VT and different combination of parallelism and pipeline architectural techniques were explored in this work. In order to generate the results, we used the IBM 65nm standard-cells library with typical voltage of 1 V and 1.2 V, and the back-end Cadence tools, e.g. Power Analysis, for the power measurements. We achieved significant power reduction for the architectures with low-frequency and high parallelism, with High-VT and mainly with only one pipeline stage and small power source.

Index Terms **–** Low-power, isoperformance, Sum of Absolute Differences (SAD), Multi-VDD, Multi-VT.

1. INTRODUCTION

Video coding is essential to support a range of multimedia applications, from mobile video streaming to high resolution digital television broadcasting. The number of portable, battery-operated devices supporting video coding is extremely large and this is a constantly growing market. H.264/AVC, the state-of-the-art video coding standard [1], introduced a significant compression gain compared to older standards, at the cost of a huge computational complexity. Motion estimation is the most time consuming task on H.264/AVC video encoding process, due to the search of a region in the reference video frame that better matches the region in the video frame to be encoded. Matching process is measured in an objective way using a distortion metric. The Sum of absolute differences (SAD) is the most used distortion metric used in motion estimation, targeting both low to high resolution applications, e.g. Full-HD 1920x1080 pixels.

To achieve high throughput and also to deal with real-time, low power, and high resolution application requirements, many application-specific VLSI designs for the motion estimation and, more specifically, for the SAD calculation were proposed [2]-[7]. Chen et al. [3] presented a H.264/AVC encoder chip for 720p resolution employing 1024 SAD processing units (PE) which use 305k gates (33% of total encoder gate count). Vanne et al. [4] proposed a new SAD processing unit and first included a comparison of various SAD processing units in terms on area and delay. Yufei et al. [5] presented the implementation of SAD architecture with 1- and 2-stage pipeline using 4-2 adder compressors. However, the works in [3]-[5] presented some alternatives to SAD architectures in terms of gate count and delay optimization, but the aspects of power and energy consumption focusing on battery operated devices were not addressed. The work in [6] presents a variable block size full-search motion estimation architecture which employs a 32-parallel SAD tree with 387.2k gates (79% of the total gate count) and the power consumption is showed, but no exploration of different design points was performed. The work in [2] presented some different design points and presented power and energy measurements, but the architectures were not designed focusing on low-power, already the work in [7] present the same different design points, but now focus in low-power and a different measurement technique.

In this work, an extensive comparison was conducted to obtain the best architecture in terms of low-power and isoperformance, i.e. the lowest energy consumption maintaining the throughput of the architecture. It is done with an IBM standard cell logic library in 65nm [8] . The alternatives were implemented, synthesized and compared in terms of power, throughput, maximum clock frequency and source power. This comparative analysis is essential to guide ongoing video encoder hardware designs to meet power and energy requirements, at given performance, targeting battery operated devices that implement video coding algorithms.

The paper is organized as follows. Section II introduces some basic concepts of SAD algorithm and the hardware architectures used in this work. Section III presents the synthesis results of SAD architecture alternatives focusing on low-power. Some comparisons with related works are showed in Section IV followed by the conclusions in Section V.

2. SAD Algorithm and Hardware Architecture

SAD calculation is performed for motion estimation (inter-prediction) and intra-prediction. It is calculated for each macroblock, i.e. 16×16 partition of a video frame, using the information of the macroblock from the current (original) frame of raw video and the predicted macroblock in one of the many candidate modes provided by H.264/AVC. SAD operation is shown in (1).

$$SAD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |P_{i,j} - O_{i,j}| \quad (1)$$

In (1), P is one of the candidate predicted macroblocks; O is the original macroblock; m and n are the dimensions of the macroblock in samples. For inter-prediction, SAD is usually implemented using a deep-pipeline strategy to improve the throughput. However, another strategy is the exploration of parallelism in the SAD architecture, designing well balanced pipeline stages and analyzing the requirements of each prediction (intra/inter) as well as performance and hardware usage goals. Because of that, it is important to explore multiple design points of SAD architectures. In [2], some different SAD architecture solutions with different parallelism (4,8 and 16 8- bit input samples) and pipeline stages (1, 3, 5 and 6) were presented, as shown in Figure 1. , Figure 2. and Figure 3. In the work [7] this analysis was extended by synthesizing the architectures using low-power techniques and performed a comparison focusing on power dissipation and the tradeoff between those two results (which we called isoperformance). In this work we complemented the work in [7] to a new technology (65 nm). This is presented in the next section.

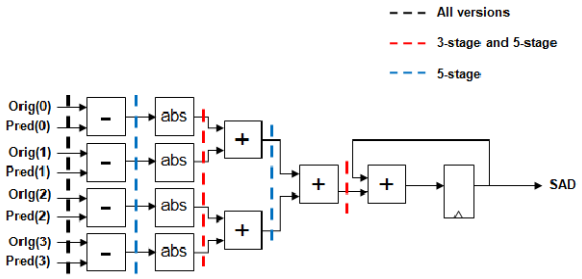


Fig. 1. Pipelines in a SAD with 4 input samples.

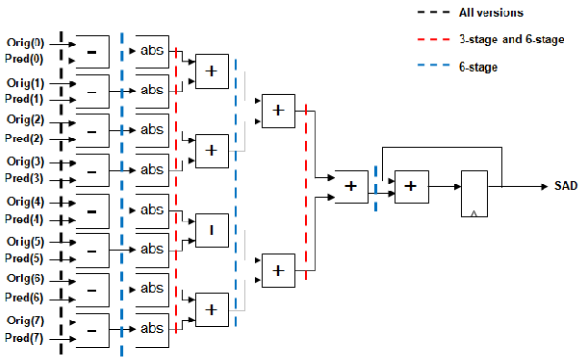


Fig. 2. Pipelines in a SAD with 8 input samples.

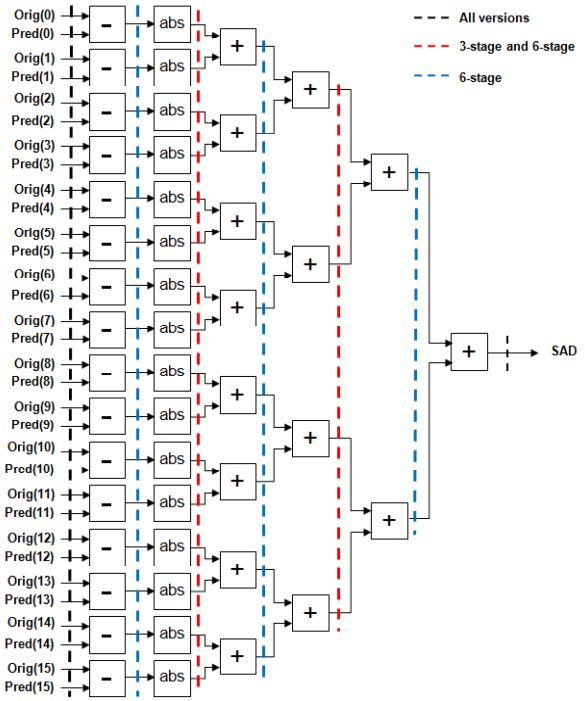


Fig. 3. Pipelines in a SAD with 16 input samples.

3. Synthesis Results and Analysis

For each SAD architectural option presented in the previous section, some iterations of synthesis flow (from front-end to back-end Post-Route step) were performed. The synthesis of each solution was generated with some frequencies to reach the design points to minimum isoperformance. The fast frequency solution, in which mostly low-VT and large cells were used, achieved a larger power dissipation. In the slow frequency solution, slow cells were used, which achieved high-VT cells, small area and power dissipation. The synthesis flow used in this work is composed by a front-end step (logical synthesis, using Cadence RC Compiler), and a back-end step (until Post-Route step, using Cadence Encounter). After back-end, Design Rule Checking (DRC) and Logical Equivalence Checking (LEC) were applied to verify geometry and connectivity, respectively. Power dissipation results were generated using Cadence Power Analysis©.

However, to achieve a low power budget for real-time video encoding applications, it is not required to design the maximum throughput solution, but the lowest power solution which fits the minimum required throughput (in macroblocks per second), e.g. to encode a Full-HD 1080p resolution (1920×1080 pixels) in real-time (at 30 frames per second) it is required to process 244,800 macroblocks per second. Considering that a

macroblock in H.264/AVC contains 256 luminance samples (16×16 block), we need to process 62 million samples per second in average to encode 1080p HD video in real-time. For this reason, we synthesized each architecture option previously developed in VHDL with a different synthesis goal: the logic synthesis targeted the lowest frequency which respect the required throughput for 1080p encoding in real-time. Then, we performed a comparative analysis in terms of energy per operation (Joules per macroblock), which is the ratio between power and throughput. Table 1 shows the results of frequency and power dissipation for each one of the architectures developed in this work which achieved minimum performance to process 256 million samples per second, defined herein as isoperformance.

Tab. 1 – Isoperformance Measurement with Multi-VDD in 65nm

SAD Architecture		Source Power (V)	Frequency (MHz)	Total Power (µW)	Energy per Operation (pJ/MB)
Input Samples	Pipeline Stages				
4	1	0.90	66	108.00	103.68
4	3	0.90	66	152.40	146.30
4	5	0.90	66	223.70	214.75
8	1	0.90	33	98.19	94.26
8	3	0.90	33	145.30	139.49
8	6	0.90	33	208.90	200.54
16	1	0.90	16	89.03	85.47
16	3	0.90	16	127.20	122.11
16	6	0.90	16	192.70	184.99
4	1	1.08	66	148.60	142.66
4	3	1.08	66	210.50	202.08
4	5	1.08	66	307.40	295.10
8	1	1.08	33	138.60	133.06
8	3	1.08	33	197.00	189.12
8	6	1.08	33	303.40	291.26
16	1	1.08	16	124.10	119.14
16	3	1.08	16	174.90	167.90
16	6	1.08	16	277.70	266.59

For the synthesis performed targeting minimum isoperformance, with power source smaller reduce proporcionally in square the total power. Likewise, with less pipeline stages the power consumption is reduced (for example, in the 16 input samples and 0.90 Volts, passing from 1-stage to 6-stage pipeline adds in 116% the power consumption). It is because with more pipeline stages is used more flip-flops, that has a big power consume. In addition to that, using a system with more parallelism and less frequency, the total power is lower, because with the double parallelism increment, the frequency decrease in a half to maintain the isoperformance, and this frequency is proporcional with the power consumption. Already, the double parallelism increase power consumption, but less than the power saved for a half frequency.

4. Comparisons to Related Works

Considering our minimum required isoperformance solutions, this work has sufficient information to compare to the works in [2], [6] and [7], as shown in Table 2. The work in [6] consume 36256 times more energy per operation than our design for minimum required performance. With the paper [2], our work is 81 % smaller in energy per operation and compare with the paper [7], it is 91% smaller. This reduction in the energy per operation in compare with the papers [2] and [7] is because our work use a smaller power source, what reduce in almost 70% the power consume and the other points to power consume reduction is because of the new technology (65nm), being responsible for the reduction in 71% in compare with the paper [7], that use the same measurement techniques.

Tab.2 - Comparison Of Our Minimum Isoperformance Solution With Related Works

	CMOS Technology	Input Samples	Pipeline Stages	Power Source (Volts)	Throughput (MB/s)	Energy per Operation (Joule/MB)
[2]	TSMC 0.18um	8	1	n. a.	3.12 E + 06	4.6 E – 10
[6]	TSMC 0.18um	16	n. a.	n. a.	1.4 E + 05	3.1 E – 06
[7]	TSMC 0.18 um	16	1	1.62	1.25 E + 07	9.82 E – 10
This work	IBM 65nm	16	1	0.90	1.25 E + 07	8.55 E -11

5. Conclusions

This work presents many vantages in energy per operation having the same functionality of the work in [7], but consuming 91% less power. It is because the technology (65 nm) that consumes much less power and accept to use smaller voltages. Also the use of Multi-VT reduces the leakage current, which have a higher contribution related with older technologies. Besides that, the use of parallelism and pipeline stages could reduce the power consumption. Therefore, new technologies help to reduce the power consumption in a much more significant percentage, being used in future development join with others techniques to reduce the power consumption.

6. References

[1] ITU-T Recommendation H.264/AVC (03/10): advanced video coding for generic audiovisual services, 2010.

[2] C. Diniz, G. Corrêa, A. Susin and S. Bampi. “Comparative Analysis of Parallel SAD Calculation Hardware Architectures for H.264/AVC Video Coding”, IEEE LASCAS, 2010. pp. 132-135

[3] T. C. Chen, et al., “Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder”, IEEE TCSVT, v. 16, n. 6, Jun. 2006, pp. 673-688.

[4] J. Vanne, et al., “A High-Performance Sum of Absolute Difference Implementation for Motion Estimation”, IEEE TCSVT, v. 16, n. 7, Jul. 2006, pp. 876-883.

[5] L. Yufei, et al., “A High-Performance Low Cost SAD Architecture for Video Coding”. IEEE TCE, v. 53, n. 2, May 2007. pp. 535-541.

[6] Z. Liu, et al., “32-Parallel SAD Tree Hardwired Engine for Variable Block Size Motion Estimation in HDTV 1080P Real-Time Encoding Application”, IEEE SiPS, 2007, pp. 675-680.

[7] F. Walter, C. Diniz and S. Bampi. “Synthesis and Comparison of Low-Power High-Throughput Architectures for SAD Calculation “, IEEE LASCAS, 2011.

[8] SiWare™ High-Density Tapless Standard Cell Logic Library for Common Platform 65nm LPe LowK Standard VT Process, Release A1, march 2009.

A Real Time HDTV Motion Estimation Architecture for the New MPDS Algorithm

[1]Gustavo Sanchez, [1]Diego Noble, [1,2]Marcelo Porto, [2]Sergio Bampi, [1]Luciano Agostini
{gfsanchez,dvnoble,agostini}@inf.ufpel.edu.br, {msporto,bampi}@inf.ufrgs.br

[1]Federal University of Pelotas – UFPEL
[2]Federal University of Rio Grande do Sul - UFRGS

Abstract

This paper presents the architectural design for Motion Estimation (ME) based on the new Multi-Point Diamond Search (MPDS) block matching algorithm. This algorithm reduces the local minima falls in the ME search process, increasing the quality of the ME prediction results. This paper presents a software evaluation about the MPDS which presented an average PSNR gain of 3.57dB and a maximum PSNR gain of 7.86dB when compared with the well known Diamond Search algorithm. The designed architecture was synthesized to an Altera Stratix 4 FPGA and, in the worst case scenario, this architecture is able to process HD 1080p videos in real time at 30 frames per second.

1. Introduction

Motion Estimation (ME) presents the highest computational complexity among all steps of the current standards for video coding, contributing with more than 80% of the encoder complexity [1]. The ME searches, in the previously processed frames, for the best similarity with the current frame region that is being encoded and generates a motion vector indicating this position. The Full Search (FS) algorithm reaches the optimal results, since it explores all possibilities for a given search window. This implies in a very high computational cost, especially for high resolution videos. Based on this fact, it is important to explore new solutions which bring a good tradeoff between objective quality (PSNR) and complexity.

There are many fast algorithms and techniques which handle this complexity at different levels of impact in objective quality (PSNR). Generally these algorithms exploit the characteristic of locality among temporal correlated blocks and achieve good results in terms of numbers of calculations. However, the majority of these algorithms have a weak point which is the increase in local minima falls with the increase of the video resolutions. This causes the generation of motion vectors with perceivable quality losses when compared to FS.

In the H.264/AVC standard [2], the current most efficient video coding standard, there is no restriction about how the block matching is done in the ME process, so there is a lot of space to explore new ideas.

In this paper, we propose a new fast ME algorithm targeting high resolution videos. This algorithm is able to reduce the local minima falls, increasing the quality of the processed ME named Multi-Point Diamond Search (MPDS) and its hardware design targeting real time HDTV (HD 1080p) encoding.

The MPDS algorithm was evaluated in software using ten HD 1080p video sequences. The results showed an average gain of 3.57 dB in PSNR in comparison with the original Diamond Search (DS) algorithm. The MPDS also reaches a very lower number of calculations than that required for FS when the same quality is considered. The designed MPDS architecture is able to process HD 1080p videos at 30 frames per second when synthesized for an Altera Stratix 4 FPGA.

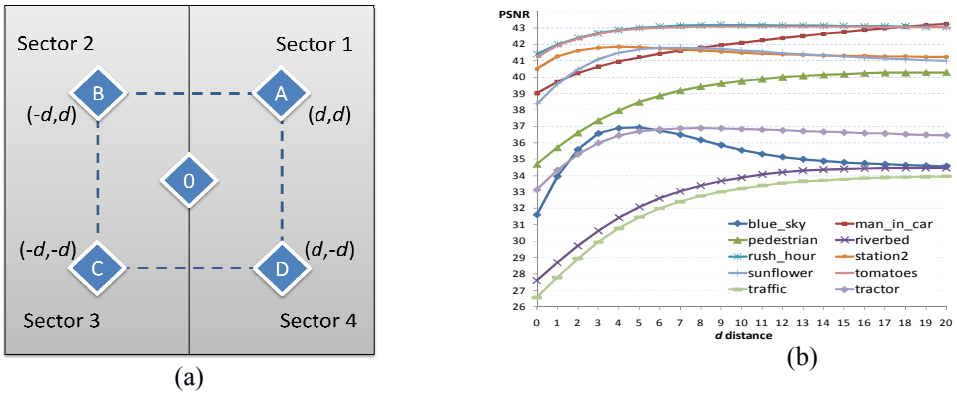
The rest of this paper is organized as follow: section 2 introduces the MDPS algorithm; section 3 shows the MPDS architecture; section 4 presents a comparison with related works; section 5 renders the conclusions and the future works.

2. MPDS Algorithm

The MPDS algorithm uses the search engine of the well known Diamond Search (DS) algorithm [3]. For higher video resolution, higher is the probability of the DS chooses local minima. Then, as higher is the video resolution as higher is the losses of DS when compared to FS. The MPDS algorithm was meant to be resilient to local minima because it starts the search for the best matching in five different and independent positions of the same search area. The search is done at the center of the search window (as the original DS) and at four more positions. Each position, except the central one, is inside of a sector (1, 2, 3 and 4), as presented in Fig. 1(a).

As MPDS algorithm uses the search engine of the DS algorithm, it starts doing the Large Diamond Search Pattern (LDSP) until the best match be found in the center of the LDSP diamond, so the Small Diamond Search Pattern (SDSP) is applied to obtain the final refinement [3]. However, the MPDS is not restricted to the central start point, exactly to avoid the same local minimum which the DS would reach.

The distance parameter ***d*** defines the sector (1, 2, 3 and 4) where each instance (A, B, C and D) of the other four DS will start the search. The ***d*** parameter is determined by the number of pixels in X and Y axis from the central point (0,0). The sectors A, B, C and D start searching at positions (*d*,*d*), (-*d*,*d*), (-*d*,-*d*) and (*d*,-*d*) respectively. When the search ends for all evaluated regions, the MPDS algorithm selects the best result from the five applied diamonds. Fig. 1(b) shows the curves of PSNR for the MPDS algorithm with 4x4 blocks in the ten HD 1080p tested video sequences [4], considering the variation of the ***d*** parameter. The error was computed as the difference between the reference frame and the motion compensated frame.



(a) (b)

Figure 1. (a) MPDS Sectors (b) PSNR curves for the MPDS algorithm with variation of *d* parameter, ***d*** = 0 means original DS

The curves in Fig. 1(b) show an important quality gain, in comparison with original DS (***d***=*0*). The software evaluation that generated these results used blocks with 4x4 samples, because a higher number of vectors would be generated and, consequently, the effect of the multi-point search would be more precise and concise.

Through the analyses of the ***d*** parameter variation, presented in Fig. 1(b), it was possible to define the optimal distance for the average case. Considering the results for the ten HD 1080p video sequences, the best distance for ***d*** parameter is 15, in the average case. It means that the MPDS presents an average gain of 3.57dB in comparison with the original DS, using ***d*=15** for this set of video samples.

This algorithm was developed focusing on the quality of the results generated by the ME process. The inherent parallelism of the MPDS algorithm is a very important characteristic that can be explored in hardware implementations to achieve real-time when processing HD videos.

3. Designed Architecture

The designed architecture for MPDS algorithm was described in VHDL and synthesized to an Altera Stratix 4 FPGA [4]. This architecture works with 16x16 blocks, and uses 4:1 pixels sub-sampling. In this configuration, the average PSNR gain in comparison with DS algorithm, is about 1,5dB. As the MPDS needs to process the DS in five different sectors, an architecture was developed to perform the usual DS algorithm and it was used to implement the MPDS algorithm.

The ME core architecture performs a normal DS with block size of 16x16 sub sampled 4:1 and its block diagram is presented in Fig. 2. In the beginning, the core is idle, and waits until the input data is ready (when the flag start is set). The architecture must fill the Reference Memory and Current Memory in advance to start the search process. These memories are fed by data from an external memory, which contains the original frame (without compression) and the reference area. The reference data is divided in 13 local memories, and each local memory contains the information that is necessary for each processing unit (PU). The processing units are responsible for the calculation of the Sum of Absolute Differences (SAD) between the current block and each reference block.

The local memories are numbered from 1 to 9 (MEM 1 to MEM 9, respectively in Fig. 2) and they contain the data used for SAD calculation for the LDSP. Memories from A to D (MEM A to MEM D, respectively in Fig. 2) contain the data that will be used for the refinement, i.e., for the SDSP. This hierarchical structure of internal memories were designed intending to increase the architecture performance, since all PUs can process their LDSP or SDSP calculations in parallel, receiving the necessary data from the Current Memory and the data of each reference region stored in Local Memories. During the time that the PUs are being used to calculate one LDSP, the Reference Memory will communicate with external memory to get the new samples which will be necessary in the next LDSP calculation. In this case, the Reference Memory will store the reference areas for the eight possible options of the new DS step: four LDSP starting from a vertex and four LDSP starting from an edge. The decision of which area will be used depends on where the current LDSP will find the lowest SAD. If the current LDSP finds the lowest SAD in the center, then the SDSP is calculated, since the data necessary was previously stored in memories.

When all local memories are filled, the data is dispatched to the PUs where the SAD calculation is performed. The proposed architecture uses nine PUs (as illustrated in Fig. 2), one for each position of the

LDSP. Then, one LDSP can be completely processed in parallel. To reduce the hardware consumption, four of these nine PUs are also used to calculate the SDSP.

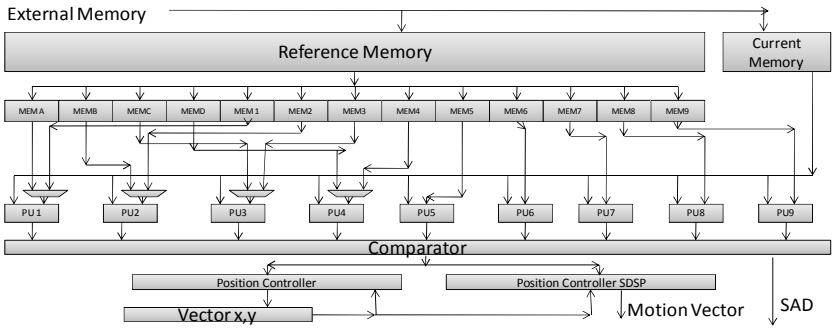


Figure 2. Block diagram of the designed architecture.

Each PU is able to calculate the SAD of a complete 16x16 sub sampled 4:1 block in parallel. The PU was designed in a pipeline of five stages and it is basically an adder tree to calculate the SAD.

The comparator sends the best block to the Position Controller in Fig. 2 which is responsible to update the new position of this block, since this is necessary to generate the motion vector. This process is repeated until the best SAD is found in the center of the LDSP (PU 5). In this moment, the SDSP is triggered and the Position Controller SDSP in Fig. 2 will generate the final integer motion vector for this block in this DS core.

Fig. 3 presents the block diagram of the MPDS architecture. At the beginning the MPDS starts to fill the CORE C with lines from reference frame and lines from current frame, as soon as CORE C is completely filled, the architecture continue to fill the next core while the CORE C receives the signal to start the search. This process is repeated until the CORE 4 is filled. The architecture starts this process again but this time it is necessary to check if each core is in an idle state. If the first case occurs, that core is filled with new data to perform next iteration. Otherwise, the control jumps to the next core. This process stops when every core have ended the ME (the end flag is set), or when the iteration limit is reached.

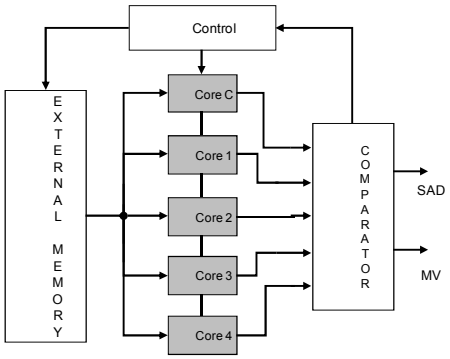


Figure 3. Block diagram of the MPDS architecture.

The total iterations of LDSP is limited up to 25 iterations (an average of 5 iterations per DS core). When all cores of the MPDS conclude the DS (including the SDSP), the best match of each core must be compared. This means that the block with the lowest SAD among the five cores must be selected as the best matching of MPDS. The comparator in Fig. 3 is responsible for this.

4. Results and Discussions

The MPDS architecture was described in VHDL and synthesized for the EP4S40G2F40I2 Altera Stratix 4 device [4]. The synthesis results of the MPDS and the core DS architecture are presented in the Table 1. The high number of registers and memory bits in the complete architecture and the frequency degradation compared to the core were already expected since the MPDS architecture uses 5 cores and also has a control system.

The processing rates achieved by the MPDS architecture considering the worst case are also presented in Table 1. The MPDS designed architecture is able to process HD 1080p videos in real time. In fact, the MPDS architecture can reach real time for HD with a lower operational frequency, it can be an excellent option for power saving for an ASIC synthesis.

TABLE 1. SYNTHESIS RESULTS AND PROCESSING RATES OF THE MDPS ARCHITECTURE

	Maximum Frequency	ALUTs	Registers	Memory Bits	HD 1080p FPS
DS Core	237,7 MHz	6472 (3%)	9024 (5%)	3200 (<1%)	166
MPDS	222,52 MHz	32453 (15%)	44648 (25%)	16000 (<1%)	44

5. Related Works and Comparisons

The ME is a very active research area inside video coding community. However, most of the works that present new algorithms usually does not present a technological study on the viability of their solution. So, there are not many works that could be fairly compared with our work.

The work [5] performs the Quarter Sub-Sampled Diamond Search with Dynamic Iteration Control (QSDS-DIC) as the search algorithm. That architecture is a DS with a dynamic iteration control and a 4:1 pixel sub-sampling rate. The block size and the similarity criteria are the same of that used in our work and the average number of iterations is 20 per motion vector with a dynamic iteration control to better explore the available iterations. However, our work presents best quality results when compared to [5], since our work uses a more sophisticated block matching algorithm. On the other hand, our solution consumes more hardware resources. The work [6] performs the Large Diamond Parallel Search (LDPS), which is another sub-optimal algorithm, with 16x16 blocks. This algorithm is very simple, and its quality results tend to be lower than the DS algorithm, due it implements only a part of the whole DS algorithm. The quality results are not presents in [6], so the fair comparison is not possible.

Table 2 presents a comparison among this work and the works [5] and [6]. Even using different FPGA families, it was possible to conclude that our solution uses much more hardware resources. However, our architecture has a better compression quality when compared to both works [5] and [6]. Comparing to both works ([5] and [6]), this work is better in terms of quality because the MDPS algorithm was developed with the goal to reduce the probability of the search falls in local minima, which is normally a weak point of sub-optimal algorithms. Our solution has similar performance of the architecture presented in [5]. The architecture presented in [6] has higher performance. However a fair comparison must be done with the DS core of our architecture. In this case, our DS architecture has also higher performance than [6].

TABLE 2. COMPARISON WITH RELATED WORKS

Criteria	MDPS	Porto, 2009 [5]	Kthiri, 2009 [6]
Algorithm	MPDS	QSDS-DIC	LDSP
FPGA Family	Stratix IV	Virtex-4	Stratix II
Memory bits	16,000	2,048	10,976
Frequency (MHz)	230,26	213.3	100
Performance (HD 1080pFPS)	44 (worst case)	45(worst case)	143(average case)

6. Conclusions and Future Works

This paper presented a ME architecture with the new MPDS search algorithm. The MPDS was proposed to be resilient to local minima, an effect more evident on high definition videos. The designed architecture was described in VHDL and synthesized to a Stratix IV FPGA. The synthesis result shows that the designed architecture is able to process HD 1080p videos in real time.

As future works, we intend to optimize the memory organization to increase the performance and to reduce the hardware cost.

7. References

[1] Cheng, Y., Chen, Z. and Chang, P., “An H.264 Spatio- Temporal Hierarchical Fast Motion Estimation Algorithm for High-Definition Video”, IEEE ISCAS 2009, pp. 880-883, 2009.

[2] JVT Editors (T. Wiegand, et al), Draft ITU-T Recommentadion and final draft international standard of joint video specification (ITU-T Rec.H.264 |ISO/IEC 14496-10 AVC), JVT-G050r1, Geneva, May 2003.

[3] J. Tham, et al, “A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation”, IEEE Transactions on Circuits and Systems for Video Technology, 1998.

[4] Altera Corporation. “Altera: The Programmable Solutions Company”. Available at: www.altera.com.

[5] M. Porto, et al. “Motion Estimation Architecture Using Efficient Adder-Compressors for HDTV Video Coding”, JICS, 2009

[6] M. Kthiri, et al. “Hardware Implementation of fast block matching algorithm in FGPA for H.264/AVC”, International Multi-Conference on Systems, Signals & Devices, 2009.

Multilevel Data Reuse Scheme for Off-Chip Memory Accesses Reduction Applied to a Motion Estimation Architecture

[1]Mateus Grellert, [2]Felipe Sampaio, [1]Julio C. B. Mattos, [1]Luciano Agostini
{mgdsilva,fmsampaio,julius,agostini}@inf.ufpel.br

[1]Federal University of Pelotas – UFPel
[2]Federal University of Rio Grande do Sul - UFRGS

Abstract

Motion Estimation (ME) in video coding is a vital component that excels not only in computational complexity, but on-chip memory bandwidth as well. These two issues are considered critical constraints in terms of High Definition (HD) video coding, since a large number of data must be processed. The multilevel data reuse scheme proposed in this paper is able to reduce the off-chip memory bandwidth, with direct impact in throughput and energy consumption. This scheme explores the concept of overlapped Search Windows (SW) in more than one level and poses no harm to video quality. Comparisons with related works show that this solution provides the best tradeoff between the use of on-chip memory and reduction of the off-chip memory bandwidth. The data reuse scheme was applied in a ME architecture and the synthesis results show that this solution presented the lowest use of hardware resources and the highest operation frequency among related works. The proposed architecture is able to process 1080p videos at 25 fps, and the reduction ratio of off-chip memory access achieved by the architecture is greater than 95% when compared to the traditional method.

1. Introduction

Motion estimation is an important component of video coding systems, representing a significant factor in the overall performance of an encoder, as well as in image quality and bit rate. The performance of ME is dictated by many varying factors, including block matching algorithm, search window (SW) size and number of supported reference frames. When the Full Search Block Matching Algorithm (FSBMA) is considered, computational complexity and off-chip memory bandwidth cause hard restrictions, especially if a large SW is scanned. Though computational complexity can be diminished by exploiting the inherent parallelism of FSBMA, memory bandwidth remains an issue that affects not only throughput, but also energy consumption.

Traditionally, the process of video coding involves two memory units: one to store a current frame and another one to keep reference frames, i.e., frames that were previously processed and reconstructed to be used as references for the next coding steps. The latter memory, particularly, is accessed many times during this process, making solutions that aim to reduce accesses to this memory imperative to achieve real time processing and to reduce the encoder energy consumption. The reference frame memory bandwidth is so limited that the latest standardization effort for video coding, called HEVC (High Efficiency Video Coding) [1], is considering to define an additional compression path dedicated only to compress the reconstructed frames before storing them in the memory.

This paper presents a scheme for multilevel data reuse for the ME process. A VLSI design for the ME with this data reuse scheme is also presented. The ME architecture processes multiple reference frames (MRF-ME) and it is compliant with the H.264/AVC standard [6]. The MRF-ME supports four reference frames, a SW of *19x19* samples and a block size of *4x4* samples. The architecture presents a memory bandwidth reduction ratio of 95.73% when compared to the traditional method, i.e., accessing every single sample from the off-chip memory. This work considers a comparison in two aspects: (1) the designed architecture with related works; and (2) the proposed multilevel scheme with other reuse schemes.

This paper is organized as follows: section II explains the proposed data reuse scheme; section III describes the designed architecture; section IV presents synthesis results and comparisons with related works; finally, section V closes this paper with conclusions and future works.

2. Multilevel Data Reuse

The enormous amount of memory accesses required to process ME for high resolution videos is a critical constraint in an encoder performance, since memory bandwidth is a limited resource. Furthermore, memory access increases the energy consumed by an architecture, aggravating the scenario for embedded devices. Therefore, solutions that aim to reduce the number of memory accesses are essential to increase the efficiency of video coding systems. Therefore, this work proposes a scheme that applies reuse strategies in two levels of memory hierarchy: local cache and local buffer. On local buffer level, data redundancy involving the SW among four neighboring current blocks is exploited. By doing so, the bandwidth between local cache and the local buffer is reduced. On local cache level, the same strategy is used with a higher granularity, considering groups of four current blocks as a single unit. This last solution drastically decreases the bandwidth between the

off-chip memory and the local cache. Therefore, this multilevel solution is able to reduce the bandwidth of off-chip and on-chip memories, with positive impact in throughput and in energy consumption.

2.3. Local Cache Level Data Reuse

In order to achieve a high level of data reuse, a local memory that works as a cache is required. This cache stores the entire SW area for sixteen blocks, which brings an impact on hardware consumption, but, at the same time, greatly reduces off-chip memory bandwidth. Furthermore, this local cache may work as a pre-fetch unit, anticipating samples that will be accessed while the ME core works with the samples already available.

The replacement strategy for this cache is illustrated on fig. 1(a). Initially the cache must be filled with an entire SW area from off-chip memory to process the first group of blocks (GoB(0)). One GoB is a group of data from the reference frame that is necessary to process a number (in this case, four) of candidate blocks. While GoB(0) is processed by the ME the additional information for GoB(1) to GoB(3) is loaded from the off-chip memory. When GoB(0) was processed, then the values that will not be used anymore are replaced by the new data that will be necessary to process the GoB(4). In this case, only ¼ of the data necessary for GoB(4) must be loaded from off-chip memory, since the other ¾ are already stored in the cache and they will be reused. The data that will be discarded is that data from GoB(0) that will not be used anymore for any other GoB. This process is repeated for all GoBs.

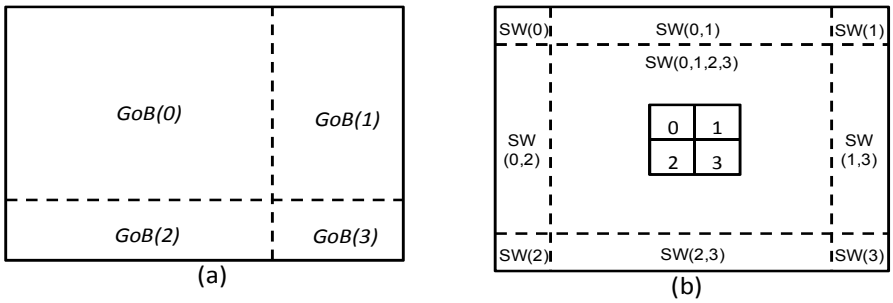


Fig. 1 - Different levels of data reuse: local cache (a) and local buffer (b).

2.4. Local Buffer Level Data Reuse

The scheme proposed in this paper also uses the concept of overlapped SW among neighboring blocks, but it is used on a lower level, to reduce the bandwidth of the local cache. This requires local buffer that stores all samples required to process a number of blocks of size *NxN* in parallel, and this number has direct impact on both, buffer size and reduction ratio. It is also important to emphasize that the architecture of this buffer, as well as access reduction, depends on which block matching algorithm is applied.

A considerable amount of data is accessed more than once from the cache during ME, on account of common SW among processed blocks. With that in mind, the local buffer stores the SW samples related to four current blocks instead of one. The reuse implemented in this level is very important to reduce bandwidth with the local cache, since accesses to this cache also represent unwanted power consumption. Fig. 1(b) illustrates the overlapped SW area among four neighboring blocks (0-3).

2.5. Bandwidth and Size calculation

Given the specifications of both, local cache and local buffer, it is possible to define equations to model the size and bandwidth (BW) of both memory levels. Tab. 1 displays these equations, considering a parallelism of four blocks, where SR_v and SR_h are the search ranges vertically and horizontally; *N* is the horizontal or vertical dimension of the block and *W* and *H* are the width and the height of the frame.

To exemplify, if a SW with *64x64* samples and blocks with *8x8* samples are considered, 5.4 Mbytes are accessed from the external memory to the local cache to process a single 1080p frame. On the other hand, the bandwidth between the cache and the buffer to process this frame is 40 Mbytes. The traditional method of accessing every sample from the off-chip memory demands a bandwidth of 126.56 Mbytes. Therefore, the proposed local cache causes a reduction of 95.73% on the accesses to the off-chip memory. Additionally, the data reuse explored by the local buffer can reduce in 68.4% the number of accesses to the local cache memory.

Tab. 1 - Bandwidth and Size Calculations

Reuse Level	Size (Bytes)	BW_{1frame}(Bytes)
Cache	$(SR_v + 3N) * (SR_h + 3N)$	$\frac{W * H}{4N} * (SR_v + 3N)$
Buffer	$(SR_v + 2N - 1) * (2N)$	$\frac{W * H}{4N^2} * (SRv + N)^2$

3. MRF-ME Architecture with Multilevel Data Reuse

The multilevel data reuse scheme was coupled to a MRF-ME architecture proposed in [7]. The MRF-ME architecture was described in VHDL, and its specifications are: four *4x4* current blocks processed in parallel and FSBMA with a SW of *19x19* samples and four reference frames, each one sequentially processed. Regarding block matching algorithms, FSBMA offers two useful features: it is highly parallelizable and the SW of neighboring blocks have large overlapped areas, which is useful to implement data reuse methods.

The high level memory hierarchy that implements the data reuse consists basically of three modules: (1) a Local Buffer, which stores all samples needed to process four neighboring current blocks in parallel; (2) a Local Cache, which communicates with the off-chip memory and stores the SW of sixteen neighboring current blocks; and (3) a Control Unit, which generates memory addresses that must be accessed and triggers the MRF-ME core. An overall view of the complete architecture is shown on fig. 2.

The local buffer works as a circular register file and it stores all samples required to process four current blocks in parallel. Since the architecture processes *4x4* blocks, the buffer contains eight register lines. Additionally, given that a SW with *19x19* samples is adopted, each line is formed by 23 8-bit registers. This component is controlled respecting three steps: (1) initial charge, where the buffer is completely filled; (2) rolling, where all columns are cyclic-left-shifted; and (3) new charge, where a new line from the local cache is loaded. Each line takes one cycle to be filled, requiring a large bus with the local cache, but memory bandwidth is greatly reduced, since local buffer level data reuse is applied.

The local cache stores all SW data for sixteen current blocks. According to definition, the memory design must store the amount of 31x31 samples. In order to permit the access and the writing of more than one word size, the local cache was designed as a set of smaller memories (called partitions).

Each partition represents a memory with 32 lines (5 bit-addressed) with a word size of 8*8 bits, storing 8 samples of the SW. This memory structure is useful to replace the unnecessary information of the previous data by the next SW offset in the local cache. Since RAM memory bits costs less than registers, using a larger memory to apply data reuse may represent an efficient and low cost approach.

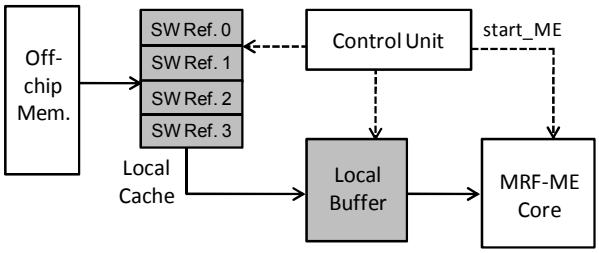


Fig. 2 - Overall view of the developed architechture.

The local cache works with the local buffer, so it must be designed accordingly. Thus, the input/output of this memory have a bit width of 23 bytes. Data reuse and replacement are managed via address calculation. As consequence, the output must be eventually reordered. To do that, a multiplexer was added between the output and the local buffer. Since the MRF-ME supports four reference frames, the local cache consists of four identical memories, each one for one reference frame. Therefore, it is natural to conclude that the overload in area consumption is defined by how many reference frames are targeted.

4. Results and Discussions

There are two comparative approaches possible on this topic: architectural results and reuse scheme efficiency. Both approaches are explored in this section, starting with a reuse scheme evaluation.

Tab. 2 - Bandwidth and Size Calculations

Specifications	**SRv = SRh = 256, 2 Ref. Frames**		**SRv = SRh = 256, 2 Ref. Frames**		**SRv = SRh = 256, 4 Ref. Frames**	
Work	Scheme C+ [2]	This Work	Scheme D [2]	This Work	Scheme C [3]	This work
On-Chip Memory(Kbytes)	198.76	185.42	812.48	185.42	20.99	121.25
Bandwidth(Mbytes/sec)	332	250.49	74.4	250.49	364.28	290.04
Additional Processor and Memory	NO	NO	NO	NO	YES	NO

In [2], a data reuse scheme entitled Level C+ is applied. This scheme was used as a model for that adopted in our work, but our multilevel data reuse solution is able to reach better results than that work, as is possible to notice in tab. 2. The authors also present a solution called Level D scheme. In this case, our work presents a larger bandwidth, but with a very smaller on-chip memory. Chen [3] proposes a frame-level rescheduling method, which consists of calculating every possible match within a reference frame to a number of current macroblocks. This strategy can greatly reduce on-chip memory size, but to achieve frame-level rescheduling,

partial results must be processed in an additional RISC processor and stored in an additional external memory. This will consume not only chip area, but bandwidth as well, so those results are unclear and the comparison unfair. An important conclusion about the results presented in tab. 2 is that our solution presents the best trade-off among off-chip memory bandwidth reduction, on-chip memory utilization and use of extra hardware components. This is a good metric to evaluate the performance of different works and, accordingly with this metric, our work presents the best results.

In addition to reuse schemes evaluation, architectures that employ these schemes can be compared. Tab. 3 displays this comparison, where gate count and frequency were obtained with Leonardo Spectrum, using TSMC 0.18 technology [8]. When HD resolution is targeted, on-chip memory can consume a large amount of hardware, so Level D reuse scheme may not be worth the bandwidth reduction it achieves. For this reason, this work is based on Level C+ scheme, which is a balanced solution between efficiency and hardware consumption. As tab. 3 shows, this work presents an on-chip memory that is six times smaller than [5] and eleven times smaller than [4]. The solution also presented the highest operation frequency and the lowest gate count among them. Though our memory bandwidth is higher, it is import to emphasize that [4] and [5] support a lower number of reference frames (1 and 2 respectively), and [5] has a low throughput. These characteristics decrease the amount of data processed and the required bandwidth. In other words, if [4] and [5] were adapted to follow our specifications, our work would present the best results also in terms of bandwidth reduction.

Tab. 3 – Architectural Comparison

Work	Search Range	Reuse Scheme	Ref. Frames	Off-Chip BW (Mb/sec)	Freq. (MHz)	Through put	#Gates (KGates)	On-Chip Mem. (Kb)
[4]	16x16	D	1	110.6	55.6	1280x720 @60 fps	176	41.6
[5]	65x33	D	2	24.9	168	720x576 @30 fps	168	23.75
This Work	19x19	Multilevel C+	4	204.35	265.2	1280x720 @56 fps	127.83	3.96

5. Conclusions

This paper presented a multilevel data reuse scheme based on overlapped search windows among neighboring blocks. The proposed scheme can greatly reduce the number of off-chip and on-chip memory accesses with no penalty on video quality. The data reuse scheme was designed using VHDL and coupled to a MRF-ME architecture without any impact on performance. This architecture presented a high throughput with low off-chip memory bandwidth, being able to process 720p videos at 56 fps and 1080p videos at 25 fps. The use of the presented multilevel data reuse scheme reduced more than 90% the number of off-chip memory accesses and more than 60% the number of on-chip memory accesses (cache). These results improve the architecture throughput, reducing significantly the energy consumption. When compared to other reuse schemes found on the literature, this work presents better trade-off among bandwidth reduction, use of on-chip memory and use of extra hardware resources. The architectural results are also competitive, since our solution used the low amount of on-chip memory, reached the highest operation frequency and used the lowest number of gates among related works, even processing more reference frames.

6. References

[1] Sullivan, G.; Ohm, J. Meeting report of the first meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Joint Collaborative Team on Video Coding of ITU-T SG16 WP3 and ISO/IEC

[2] C.-H. Chen, et al, “Level C+ Data Reuse Scheme for Motion Estimation With Corresponding Coding Orders”, IEEE TCSVT, vol.16, n. 4, Apr. 2006, pp. 553 – 558.

[3] T.-C. Chen, et al. “Single Reference Frame Multiple Current Macroblocks Scheme for Multi-Frame Motion Estimation in H.264/AVC”, ISCAS, May 2005, pp. 1790 – 1793.

[4] Z. Zhaoqing, et al, “High Data Reuse VLSI Architecture for H.264 Motion Estimation”, ICCT, Nov. 2006, pp. 1-4.

[5] D.-X. Li, W. Zheng, M. Zhang, “Architecture Design for H.264/AVC Integer Motion Estimation with Minimum Memory Bandwidth”, IEEE TCE, vol. 53, n. 3, Aug. 2007, pp. 1053 – 1060.

[6] J. V. Team, Draft ITU-T Rec. and Final Draft Int. Standard of Joint Video Spec. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[7] M. Grellert, et al, “Memory-Aware Multiple Reference Frame Motion Estimation for the H.264/AVC Standard”, ICECS, Dec. 2010 (accepted for publication.).

[8] TSMC, “TSMC 0.18 and 0.15-micron Technology Platform.”, Available at www.tsmc.com/download.

Fast Distortion-Based Heuristic and Hardware Design for the H.264/AVC Intra-Frame Decision

[1]Daniel Palomino, [3]Guilherme Corrêa, [2]Luciano Agostini, [1]Altamiro Susin
{dmvpalomino,altamiro.susin}@inf.ufrgs.br, guilherme.correa@co.it.pt,
agostini@inf.ufpel.edu.br

[1]Microeletronics Group - Federal University of Rio Grande do Sul
[2]Group of Architecture and Integrated Circuits - Federal University of Pelotas
[3]Institute for Telecommunications - University of Coimbra

Abstract

In the Rate-Distortion Optimization (RDO) technique for H.264/AVC, the process of choosing the best intra-prediction mode is performed through exhaustive executions of the whole encoding process, which increases significantly the encoder complexity, sometimes even forbidding its use in real time video applications. In order to reduce the number of calculations necessary to determine the best intra-frame mode, this work proposes a fast distortion-based heuristic and its hardware design for the H.264/AVC intra-frame decision. The application of the proposed heuristic reduces in 13 times the number of encoding iterations for choosing the best intra-frame mode when compared with RDO-based decision, at cost of relatively small bit-rate increase and image quality loss. The proposed heuristic was designed in hardware targeting two technologies: (1) FPGA and (2) Standard Cell. This architecture achieved an operation frequency of 129.1 MHz when synthesized to Standard Cell, being able to process until 465 HD1080p frames per second.

1. Introduction

H.264/AVC, the state-of-art video coding standard proposed by Join Video Team (JVT) [1], provides up 50% of compression gain when compared with previous standards, like MPEG-2. In order to achieve this goal, a huge amount of coding options have been included in the prediction modules (Inter and Intra). Considering the intra prediction, there are two possible block sizes to encode one macroblock (MB): (1) I16MB, with four possible prediction modes applied to 16x16 luminance blocks and (2) I4MB, with nine possible prediction modes applied to 4x4 blocks. All these possibilities must be evaluated to select the best one for each MB inside a frame, in a way to achieve high compression rates preserving the video visual quality.

Rate-Distortion Optimization (RDO) [2] is a well known technique used to achieve the best coding efficiency for an MB considering the relation bit-rate and video quality. However, the computational complexity of evaluate all H.264/AVC coding modes is extremely high, since the RDO technique performs a complete encoding process to choose the best prediction mode for each MB. Considering an HD1080p video sequence with a group of pictures (GOP) IPPPPP (an I frame every six frames), more than 8 million iterations of prediction, transform, quantization, inverse quantization, inverse transform and entropy coding (called in this work as *encoding loop*) are needed for each frame. This way, it is hard to use RDO technique when high resolution and real-time applications are considered. Fig. 1 shows the diagram of RDO based decision. Gray blocks are performed once for each prediction mode, while the mode decision block (in white) receives all candidate mode bit-rates and distortions (dashed lines) and selects the mode with the lowest RD cost to be used in the encoding process.

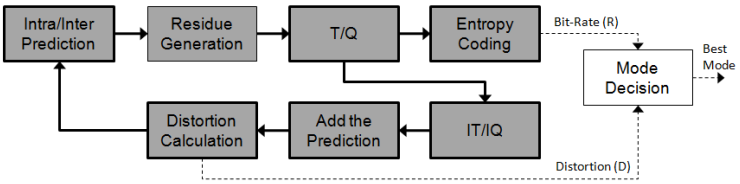


Fig. 1 – Diagram of RDO based encoding process.

Due to this complexity, some works as [3], [4], [5] and [6] have proposed fast intra-frame mode decisions to decrease the time of encoding one MB. The work [3] uses the SATD calculation to decide whether or not the rate-distortion cost over one mode will be performed. In the work [4], it is performed an analysis over the transformed coefficients and according its arrangement a calculation using some equations is performed to determine the probability of each mode. The modes with low probability are not evaluated by the RDO-based decision. The work [5] proposes a fast intra-frame decision based on the low frequency coefficients generated after the transformation. These coefficients are used to determine the homogeneity level of one MB. However, this work only performs the decision among different block sizes (I4MB or I16MB). The decision among same block sizes is performed using the RDO technique. Finally, the work [6] performs the intra-frame decision

using directional gradients. These gradients are calculated using equations presented in the paper. The gradients are compared with previous thresholds generated by simulations and then the two lowers are chosen to be performed by the RDO-based decision. Even though these works present computational cost reduction in the intra-frame decision process, all of them still use the RDO technique.

The main goal of this work is to develop a fast heuristic based only on distortion for the intra-frame mode decision module, in order to completely eliminate the use of RDO based decision, decreasing the time needed to encode one MB. The hardware design for this solution is also presented in the next sections. The paper is organized as follows: Section 2 presents the fast intra-frame mode decision proposed in this work. Section 3 shows the hardware architecture for this solution. Section 4 shows some results generated through simulations using the proposed method and compares with related works. Finally, section 5 concludes this work.

2. Fast Intra-Frame Mode Decision

In this work, the proposed fast intra-frame decision is performed in a hierarchical way in two steps: (1) decision among same block sizes and (2) decision among different block sizes. The first decision step is based on distortion to choose the best I16MB partition considering the four possible modes and the best I4MB partition among the nine possible modes. Several simulations considering three distortion metrics were performed: Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD) and Sum of Absolute Transformed Differences (SATD). The results (for bit-rate and video quality) generated using these three metrics were compared among each other. Besides, a comparison among the three metrics was performed considering computational complexity measured by number of sums. Tab. 1 shows these results. The simulations were performed using CIF (352x288 pixels) videos.

.

Tab. 1 – Comparison between SAD, SSD and SATD.

SSD vs. SAD			SATD vs. SAD		
PSNR (dB)	Bit-rate (%)	# Sums (%)	PSNR (dB)	Bit-rate (%)	# Sums (%)
+0.008	-0.63	+361.29	+0.079	-1.13	+348.39

SATD and SSD metrics show better results when bit-rate and video quality are considered. However, the computational complexity of these two metrics is very much bigger than SAD (about 361% bigger when the SSD metric is considered). As the main goal of this work is to perform the intra-frame decision as fast as possible, the SAD metric is used in the first decision step.

The second step of the intra-frame decision is to choose which partition (I4MB or I16MB) will be used to encode the MB. This decision is made using the information generated by the first step: the distortion of the best I4MB partition and the distortion of the best I16MB partition measured in SAD. A simple comparison between these two metrics would cause in most cases the choice for I4MB partitions. However, analyzing the difference between these two metrics it is possible to make a good choice of what partition will be used for each MB. Several simulations were performed to classify the difference of distortions (DD) with the intra-prediction mode selected by the RDO technique. Fig. 2 shows a graph where the amount of chosen modes (I4MB and I16MB) for each MB selected by the RDO technique is compared with the difference of distortion generated by the first step.

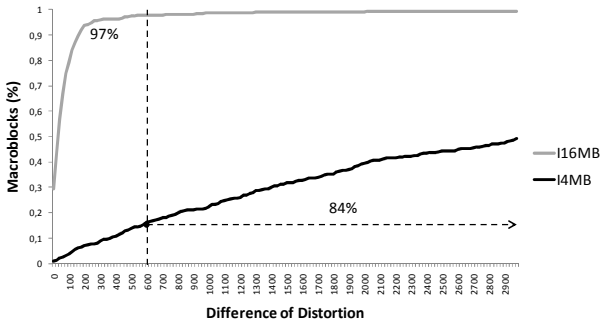


Fig. 2 – Amount of chosen modes for each MB selected by the RDO technique and difference of distortion.

With the difference of distortion set to 600, for example, it is possible to see that in most cases when the I16MB partition is chosen (97%) the difference of distortion is very small (lesser than 600), while when the I4MB partition is chosen the difference of distortion is very large (84% are bigger than 600). This way, it is possible to use this information in comparison with a threshold value to choose the partition size for intra-frame MBs.

The threshold value that presents the best results in terms of PSNR and bit-rate was obtained as follows: (1) all videos used in the simulations were first encoded using the RDO technique for the decision among different block sizes. Meanwhile, the distortion values and the chosen partition were saved. Then, the differences of distortion were compared with the chosen partition to define the threshold value. Several

simulations were performed with a threshold ranging from 0 to 1000, being 600 the threshold value which generated the best results considering the bit-rate and video quality relation.

3. Designed Architecture

The designed architecture for the proposed intra-frame decision is presented in Fig. 3. The decision is divided in chrominance decision (8x8 modes) and luminance decision (I4MB and I16MB modes). The chrominance decision is performed considering only the distortion (SAD) between the predicted chroma block (PCB) and the original chroma block (OCB). The luminance decision is divided exactly like the proposed heuristic (in two steps): (1) decision among same block sizes, considering the distortion (SAD) between the 4x4 and 16x16 predicted luma blocks (PLB) with the original luma block (OLB), and (2) decision among different block sizes, considering the difference of the best distortions generated in the step 1.

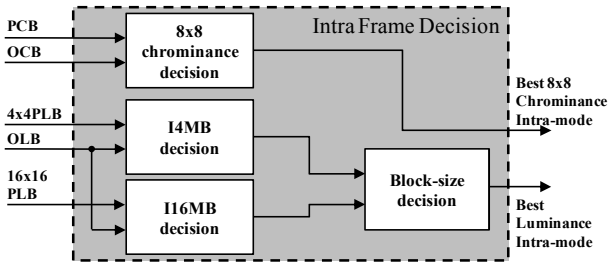


Fig. 3 – Architecture of the proposed intra-frame decision.

The architecture to perform the SAD calculations between the original block and the predicted block was designed with a parallelism of eight samples and two pipeline stages. It means that only eight samples of the original block and the predicted blocks generated by all intra-frame modes can be available per cycle. This way, 34 clock cycles are needed to perform the decision over one MB. The architecture was synthesized targeting two different technologies: (1) Altera EP2S130F1508C3 Stratix II FPGA [7] and (2) TSMC 0.18µm Standard Cell [8], in order to allow its use in several applications. Considering hardware resources, the FPGA synthesis used 3,267 ALUTs (Look-up Table) and 2,312 DLR (Dedicated Logic Register) while the standard cell synthesis used 28,518 gates. The maximum operation frequency was achieved by the standard cell synthesis: 129.3MHz. Considering this frequency and the number of clock cycles needed to perform one MB (34), the architecture is able to process until 465 HD1080p frames per second.

4. Results and Comparison

The results obtained by using the proposed heuristic are presented in Tab. 2. The first columns present the results using RDO-based decision. The central columns present the results obtained by the proposed decision. Finally, the last columns show a comparison between the two approaches in terms of bit-rate increasing and image quality (PSNR) difference. These results were obtained through simulations performed with HD1080p video sequences.

Tab. 2 – Results and Comparison with RDO-based decision.

Video	RDO		Proposed Heuristic		RDO vs. Proposed Heuristic (Threshold 600)	
	PSNR (dB)	Bit-Rate	PSNR (dB)	Bit-Rate	PSNR loss	+ Bit-Rate (%)
STATION 2	39.643	34022k	39.338	35729k	0.305	5.02
SUNFLOWER	42.766	33030k	42.413	35391k	0.353	7.15
TRACTOR	39.400	58208k	39.035	60939k	0.365	4.69
TRAFFIC	39.643	50884k	39.308	53070k	0.335	4.30
MANINCAR	42.609	8818k	42.518	9018k	0.091	2.27
PEDISTRIAN AREA	40.878	24949k	40.699	26875k	0.179	7.72
RIBERBED	38.652	56735k	38.280	58962k	0.372	3.93
ROLLING TOMATOES	40.475	12170k	40.387	12537k	0.088	3.02
RUSHHOUR	41.694	20013k	41.484	21306k	0.210	6.46
AVERAGE	**40.640**	**33203k**	**40.385**	**34870k**	**0.255**	**5.02**

The application of the proposed heuristic resulted in an average increase of 5.02% in the bit-rate and an average decrease of 0.255dB in the image quality (PSNR). The increase of bit-rate and the decrease of image quality are very small when compared to the enormous computational complexity reduction in the decision process. As presented in Fig. 1, the RDO-based encoding process is finished only after the execution of all

possible intra-frame prediction modes. The decision proposed in this work is performed after the generation of the predicted blocks by the intra-prediction followed by the SAD-based distortion calculation and then the difference of distortion operation. This way, the encoding loop presented in Fig. 1 is completely eliminated, simplifying the intra-frame decision process. When RDO-base decision is performed, four 16x16 and nine 4x4 intra-frame modes must be evaluated, totalizing 13 encoding iterations per MB. Considering the proposed decision method, the encoding process is performed only once for each MB. Tab. 3 presents a comparison with related works in terms of bit-rate, image quality (PSNR) and reduction in RDO calculations.

Tab. 3 – Comparison with related works.

Works	PSNR loss (dB)	Bit-rate Increase (%)	# of Iterations Reduction (times)
SUN (2008) [3]	0.136	0.69	4.7
FENGQIN (2008) [4]	0.020	3.00	5.9
LEE (2009) [5]	0.088	0.09	4
JEON (2009) [6]	0.148	4.00	6
This Work	**0.255**	**5.02**	**13**

While other works have shown a reduction of coding iterations from 4 to 6 times in comparison with RDO-based decision, the proposed decision allows a reduction of 13 times. The cost of this gain resides, however, in the bit-rate increase of 5.02% and image quality loss 0.255dB which do not represent a large loss when the gain in terms of computational complexity reduction is considered.

5. Conclusions

This work has presented a fast distortion-based heuristic and hardware design for the H.264/AVC intra-frame decision. The main goal is to perform the intra-frame decision process without using the RDO technique, strongly decreasing the time needed to encode one MB. When the proposed intra-frame decision is applied, an increase of 5.02% in bit-rate and a decrease of 0.255dB in image quality were noticed in average. On the other hand, the number of encoding iterations is 13 times smaller than that used by RDO. Besides, none of the works found in the literature reduce computational complexity more than 6 times in comparison with RDO technique. The designed architecture achieved an operation frequency of 129.3MHz when synthesized by standard cell, being able to process until 465 HD1080p frames per second, enough for encoding high resolution videos in real time.

As future works, it is planned to integrate the designed architecture with an intra-predictor, transforms and quantization loop and an entropy coder, in order to design a full intra coder.

6. References

[1] ITU-T Recommendation H.264/AVC (05/03): advanced video coding for generic audiovisual services. 2003.

[2] G. Sullivan and T. Wiegand, “Rate-Distortion Optimization for Video Compression”, IEEE Signal Processing Magazine, v. 15, Nov. 1998, pp. 74-90.

[3] Sun, Y. T.; Yinyi, L. SATD-Based Intra Mode Decision For H.264/AVC Video Coding. IEEE International Conference on Multimedia and Expo, Hannover, 2008. 61-64.

[4] Fengqin, W.; Yangyu, F. Fast Intra Mode Decision Algorithm in H.264/AVC Using Characteristics of Transformed Coefficients. 5th International Conference on Visual Information Engineering. [S.l.]: [s.n.]. 2008. p. 245-249.

[5] Lee, Y.-M. An Improved SATD-Based Intra Mode Decision Algorithm for H.264/AVC. International Conference on Acoustics, Speech, and Signal Pocessing, ICASSP. Taipei: [s.n.]. 2009. p. 1029-1032.

[6] Jeon, Y.-I. Fast Intra Mode Decision Algorithm Using Directional Gradients for H.264/AVC. International Congress On Image and Signal Processing, CISP. Tianjin: [s.n.]. 2009.

[7] Altera Corporation. “Altera: The Programmable Solutions Company”. Available at: www.altera.com.

[8] Artisan Components. *TSMC 0.18 µm 1.8-Volt SAGE-X™ Standard Cell Library Databook*. 2001.

Data Reuse Scheme for an Out-of-Order Motion and Disparity Estimation Targeting the Multiview Video Coding

[1]Felipe Sampaio, [1]Bruno Zatt, [1]Sergio Bampi, [2]Luciano Agostini
{felipe.sampaio, bzatt, bampi}@inf.ufrgs.br, agostini@inf.ufpel.edu.br

[1]Universidade Federal do Rio Grande do Sul - UFRGS
Programa de Pós-Graduação em Computação - PPGC

[2]Universidade Federal de Pelotas - UFPel
Grupo de Arquiteturas e Circuitos Integrados - GACI

Abstract

This paper presents a data reuse scheme for memory-aware Motion and Disparity Estimation (ME and DE) targeting the Multiview Video Coding. The strategy, called Search Window Centric Strategy (SWCS-MVC), ally several techniques to exploit the data locality: (a) reuse of overlapped regions between neighbor search windows, (b) GDV adoption to determine the DE start point and (c) out-of-order processing to increase the cache efficiency. In the best case, the data reuse scheme causes a reduction of 42,8% in the required off-chip memory bandwidth when compared with the use of the traditional in-order frame level encoding (called Current Block Centric Scheduling). In the other hand, the penalty of using the out-of-order frame level processing impacts in the on-chip memory required to store the encoding partial results. However, with the search window size growth the penalty is amortized.

1. Introduction

In the last years, the improvements in the multimedia technologies have caused the development of high realism applications. These applications aim to explore the characteristics of the human vision in order to increase the user experience [1]. As results of this effort, the 3D TV applications are even more popular and available for the final costumer.

The basis for these applications is not the usual digital videos taken by only one camera viewpoint. The 3D applications, for example, combine two or more different viewpoints of the same scene in order to allow the depth perception. These videos generated by more than one camera are called multiview videos and each individual video is referred as view.

However, the transmission and storage of multiview videos are practically unfeasible without a compression process, since the large amount of data that are needed to be treated. The simplest way to compress a multiview video is to simply apply a mono-view encoder for each one of the views. However, this approach does not explore the data redundancies between frames at different views. As said in the work [2], almost 30% of the dependencies in a multiview video were detected between frames of different views.

In order to eliminate this redundancy, called disparity redundancy, the MVC Standard (Multiview Video Coding) was defined as the most recent video compression standard that deals with multiview videos [3]. It is an extension of the H.264/AVC standard [4] and it inherits all of the high complexity tools that are used in the H.264/AVC codecs.

The MVC encoder with all its complex coding tools requires high throughput rates and large bandwidth with the main memory. This work focuses on reducing the bandwidth rates proposing a data reuse scheme for the two modules that represent the memory access bottleneck in the MVC encoders: the Motion and Disparity Estimation (ME and DE). These modules perform searches in a delimited search area in order to determine the best way to encode a block of pixels in the target video. The ME takes as reference for its search past or future frames in the same view. This way, the ME is trying to find the motion given a current and a reference frame. The DE is the ME extrapolation but using frames of different views as reference. In this case, the goal is to determine the disparity given by the different cameras position. These modules are responsible for more than 70% of the memory access during the encoding process.

The goal of this work is to reduce the required bandwidth during the ME and DE steps. Then, a frame order processing scheduling is proposed to increase the data reuse when the off-chip memory is accessed, called SWCS-MVC (Search Window Centric Scheduling). Besides, the data locality between two adjacent search windows is also explored. Such works in the literature already explore this level, like [5] and [6]. They are used in a combination with the SWCS strategy. The penalties involved with the use of the proposed strategy are also evaluated and an extra on-chip memory was designed solve it.

This paper is organized as follows: Section 2 presents the proposed data reuse scheme and the out-of-order processing for the ME/DE modules; Section 3 discusses the results in terms of on-chip memory and off-chip memory bandwidth requirements and performs a comparison with related works; finally, the Section 4 concludes this work and points some future works.

2. Data Reuse Scheme

This work proposes a data reuse scheme for the multiview video coding to reduce the off-chip memory bandwidth impact during the references fetching. As already mentioned, the strategy is focused in the ME/DE since they are the modules that most access the off-chip memory among all encoding parts.

The main contributions of this work are: (1) off-chip memory bandwidth reduction in the encoding references access; (2) out-of-order frame processing to increase the data locality and, consequently, the cache efficiency; (3) local storage of the encoding partial results; (4) GDV usage to determine the DE start point.

In the first level, the overlapped area between adjacent search windows is explored. In the literature there are several works that proposes search window data reuse between neighbor blocks. The works [5] and [6] presented several reuse schemes, named Level A-D and Level C+. For further information, detailed descriptions are encountered in the original work.

Changing for a frame level reuse, which is the focus of this work, basically there are two frame-scheduling for a MVC processing. The traditional one, named as Current Block Centric Scheduling (CBCS), considers the current block as the central point of scheduling. In this approach, given a block that needs to be processed by the ME or DE, all its corresponding search windows in all reference frames must be accessed in the reference frame memory. The alternative scheduling, called in this work as Search Window Centric Scheduling (SWCS-MVC), is a memory-aware approach that performs the frame processing under other central point: the search window. In this case, a reference frame is scanned and all the ME/DE operations that use the given frame as reference are performed immediately. The Fig. 1 demonstrates these two different approaches.

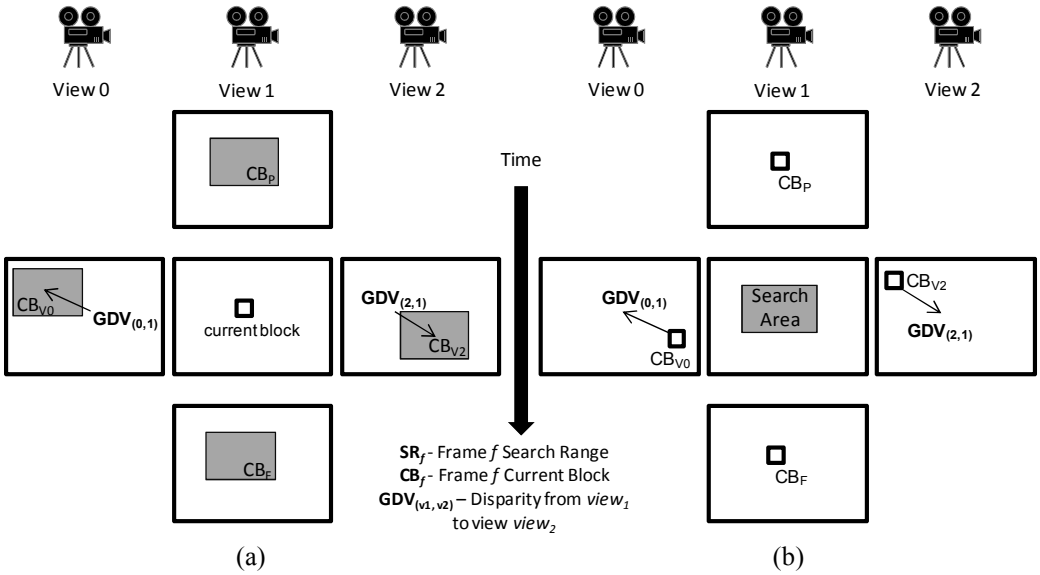


Fig. 1 - (a) CBCS and (b) SWCS-MVC.

Considering a prediction structure which specifies that a frame is used as reference twice (once by ME and once by DE), the SWCS can save 44,7% bandwidth when compared with the CBCS approach. When more reference frames are used, better are the gains of the SWCS. However, since the use of the SWCS implies in an out-of-order processing, partial results are generated during the encoder process and they cannot be decided instantly. Then, they need to be stored to be compared with the final results. At this moment, they can be discarded and replaced for new partial results. To deal with this penalty, this work proposes the use of a Partial Results Memory (PRM). The PRM sizing is expressed by the Equation (X), where $DIST_{bits}$ is the distortion bit-width, VET_{bits} is the motion vector bit-width, IDx_{bits} is the reference index bit-width, W and H are the horizontal and vertical frame dimensions, N is the block size and Q_{range} is the number of frames which its information needs to be stored.

$$PRM_{size} = (DIST_{bits} + VET_{bits} + IDx_{bits}) \times \frac{W \times H}{N^2} \times Q_{range} \quad (1)$$

3. Results

Tab. 1 presents a comparison between the required bandwidth for real time processing (30 frames per second) considering the two scheduling strategies: (a) the traditional CBCS and (b) the memory-aware SWCS adopted in this work. The overlapped regions of adjacent search windows are reused in according with three schemes proposed in [6]: Level C, Level D and Level C+ with four different block processing order. These processing orders in Tab. 1 are better explained in the corresponding work.

Tab. 1 - Bandwidth Results

Block Size 16x16	*XGA Resolution (1024x768 pixels)*	*4 views* SW_H=SW_V= *128*	*@30fps* |*GOP*| = *8*
Scheme	CBCS Bandwidth (Mbytes/s)	SWCS-MVC Bandwidth (Mbytes/s)	Ratio
Level C	1582,24	904,14	- 42,86%
Level D	213,69	122,11	- 42,86%
Level C+ HF2V2	879,64	502,65	- 42,86%
Level C+ HF3V2	879,64	502,65	- 42,86%
Level C+ HF2V3	645,44	368,82	- 42,86%
Level C+ HF2V4	528,33	301,91	- 42,86%

For all cases, the use of the SWCS scheme provides reduction of around 42% in the off-chip memory bandwidth when compared to the CBCS. This gain can be even better if more reference frames are used.

Considering a solution without any cache hierarchy, the gain provided by the SWCS employment can save, in the best case, more than 350 times the required bandwidth for a XGA video and more than 320 times for a VGA video considering the same scenario of the comparison presented in Tab. 1.

The Tab. 2 performs a comparison in terms of on-chip memory requirements for both strategies (CBCS and SWCS).

Tab. 2 - On-chip Memory Size Results

Block Size 16x16	*XGA Resolution (1024x768 pixels)*	*4 views* SW_H=SW_V= *128*	*@30fps* |*GOP*| = *8*
Scheme	CBCS On-chip Mem. Size (Kbytes)	SWCS-MVC On-chip Mem. Size (Kbytes)	Ratio
Level C	39,94	70,97 (19,97 + 51,0)	+ 77,6%
Level D	285,50	193,75 (142,75 + 51,0)	- 32,1%
Level C+ HF2V2	49,38	75,69 (24,69 + 51,0)	+ 53,3%
Level C+ HF3V2	54,35	78,17 (27,17 + 51,0)	+ 43,8%
Level C+ HF2V3	59,81	80,91 (29,91 + 51,0)	+ 35,3%
Level C+ HF2V4	71,25	84,63 (35,63 + 51,0)	+ 21,6%

Besides the high memory access reduction, the SWCS scheduling presents another advantage: the number of used reference frames in the target prediction structure does not affect in the on-chip memory required to store the search window samples. In the other hand, the CBCS require an instance of a search window on-chip memory for each reference frame scanned by both ME and DE. For all cases in the Tab. X, the on-chip memory used to store the reference samples is 50% smaller in the SWCS strategy.

As already explained, the SWCS strategy causes a penalty in the on-chip memory because of the necessity of storing the encoding partial results (the PRM). Considering the analysis scenario and the expression defined in the Equation (1), the PRM needs to have 51Kbytes. This way, in almost all cases in Tab. X the SWCS strategy presents worst results, excepting when Level D is used (gain of around 32%).

However, if a projection is performed by increasing the search window dimension it is possible to notice that the PRM represent a smaller part into the total on-chip memory required. This analysis is presented in the graphics of the Fig. 2.

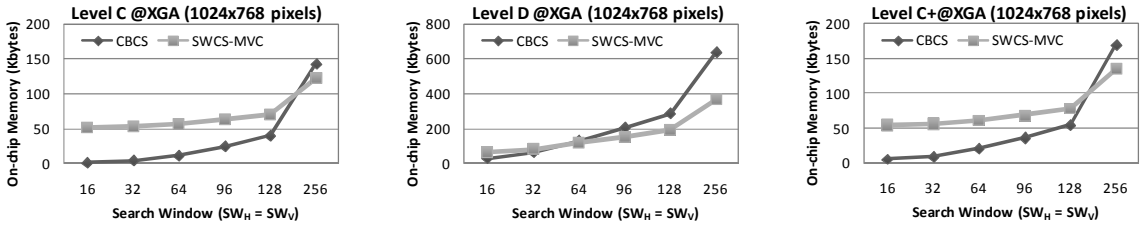


Fig. 2 - On-chip memory size growing analysis.

For all cases in Fig. 2 (Level C, Level D and Level C+), the SWCS on-chip memory growth curve present worst results for smaller search windows. It is explained by the high impact of the PRM. When considering the bigger search windows, the CBCS growth curve is sharper and present worst results from a certain search window size, depending on the adopted data reuse scheme.

4. Conclusions

This work presented a data reuse scheme to reduce the required bandwidth for the Motion and Disparity Estimation processing for the Multiview Video Coding. The strategy considers some scenarios: (1) reuse of overlapped regions between search windows of neighbor blocks (current block reuse level), (2) frame scheduling to increase the data locality and, consequently, the cache efficiency (reference frame level reuse level) and (3) the use of GDV to determine the DE start point. The scheme, called SWCS-MVC, brings some important gains in the off-chip memory reduction. In the best case, for the considered scenario, the use of the SWCS-MVC reduced by around 42% the number of memory access when compared with the traditional in-order strategy (called CBCS). Due the penalties of the SWCS-MVC, the required on-chip memory increases since encoding partial results must be stored until they are decided (called Partial Results Memory). However, with the search window increasing (essential for high resolution videos), the PRM size is amortized and from a certain point does not represent additional cost.

5. References

[1] N. Dodgson. A. “Autostereoscopic 3D Displays”. Computer, v. 38, n. 8, p. 31-36, aug. 2005.

[2] P. Merkle, et al. “Efficient Prediction Structures for Multiview Video Coding”. IEEE Transactions on Circuits and Systems for Video technology, v. 17, n. 11, p. 1461-1473, nov. 2007.

[3] JVT Team. “Editors' draft revision to ITU-T Rec. H.264 | ISO/IEC 14496-10 Advanced Video Coding – in preparation for ITU-T SG 16 AAP Consent (in integrated form)”. Doc. JVT-AA07. [S.l.]: [s.n.]. 2009.

[4] ITU-T. “ITU-T Recommendation H.264/AVC (05/03): Advanced video coding for generic audiovisual services”. 2003.

[5] J.-C. Tuan, et al. “On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture”. IEEE Transactions on Circuits and Systems for Video Technology, v. 12, n. 1, p. 61-72, jan. 2002.

[6] C.-Y . Chen, et al. “Level C+ Data Reuse Scheme for Motion Estimation With Corresponding Coding Orders”. IEEE Transactions on Circuits and Systems for Video Technology, v. 16, n. 4, p. 553-558, april 2006.

Design Automation Tools 2

Area Overhead and Performance Impact of Regular Transistor Layout Design in Digital Integrated Circuit

[1]V. Dal Bem, [1,2]P. F. Butzen, [1]F. S. Marranghello, [1]A. I. Reis, [1]R. P. Ribas
{vdbem,pbutzen,fsmarranghello,andreis,rpribas}@inf.ufrgs.br

[1]PGMicro, UFRGS, Porto Alegre, Brazil
[2]Center of Computational Science, FURG, Rio Grande, Brazil

Abstract

Regular transistor layout (RTL) is expected to reduce the process variation, increase the fabrication yield and improve the device reliability in nanometer CMOS technologies. Although the penalty in design flexibility seems to be obvious, in terms of area optimization and circuit performance, deep and extensive investigations must be done in order to evaluate the actual design cost and trade-off in using such a more lithography-aware strategy. This paper presents a detailed comparison between RTL and standard cell methodologies. Experimental results have shown that the impact of RTL technique in digital circuit design is quite manageable, presenting acceptable area overhead and even power consumption reduction for certain circuit timing constraints.

1. Introduction

CMOS processes have been continuously scaled down in order to improve the integrated circuit (IC) timing and power consumption characteristics, as well as to increase the integration density of semiconductor devices in a single die. In nanometer technologies, the critical feature sizes printed on a chip is significantly smaller than the light wavelength used in photolithography steps [1]. As a consequence, the resolution between the drawing layout and the printed pattern is affected..

New design methodologies based on regular transistor layout (RTL) have received special attention in the last years [2]-[5]. The main goal of those methodologies is the reduction of layout patterns, resulting the improvement in the lithography resolution efficiency. The most common RTL approach consists basically of pre-defined logic block arrays [4][5]. Such design strategy exploits the use of via masks in the programmability of different logic functions over RTL patterns and wiring arrangements. By doing so, mask writing efforts are significantly reduced during the circuit fabrication.

As occur in old mask-programmable gate-arrays, the area overhead and circuit performance penalty are the most important drawbacks of regular layout design. It represents a compromise between the lithography yield improvement and a more restrictive design space.

This paper presents an extensive analysis and helpful discussion about the RTL circuit design in comparison to the most applied standard cell approach. Experimental results have shown that the evaluated RTL templates can achieve good timing performance and acceptable power dissipation levels with non-prohibitive area penalty.

2. Regular Transistor Layout

Several RTL approaches for lithography-aware integrated circuit design have been recently proposed in the literature [2]-[5].

In [2], Pileggi *et al.* present a structure named VPGA (via patterned gate-array) that represents a compromise between FPGA and standard cell design flexibility and performance. The proposed VPGA unit is composed by a pre-defined set of basic cells, where the customization of interconnections and logic functions is performed by via mask patterns.

In another work, Jhaveri *et al.* propose the construction of cell library containing just a few regular cells, called 'logic-bricks' [3]. The bricks are design-specific and the information about good candidates is obtained through a regularity extraction step over the target circuit. Some usual brick requirements are to be able of reproducing complex functions and to present some configurable vias.

In [4], Ran *et al.* propose the use of VCCs (via-configurable cells). This approach seems to be less flexible than the one previously discussed [2], since some attributes like cell dimensions, number of transistors and metal lines available at each cell are fixed. In this design strategy, the only difference between distinct logic gates is the vias positioning over their layouts.

A similar purpose to VCC has been presented by Pons *et al.* in [5]. Such approach, called VCTA (via-configurable transistors array), presents also the fixed number of physical resources (transistors and metal lines), and the logic gates are configured by via insertion. The major difference to [4] relies on the layout topology. In VCTA, considering polysilicon stripes (transistor gate) drawn vertically, the P and N active areas are drawn in the same horizontal line. It is a particular strategy that presents some advantages like unnecessary

polysilicon alignment between PMOS and NMOS transistors of pull-up and pull-down logic planes, respectively, as observed in standard cells.

All these related works intend to provide efficient lithography-aware regular layout patterns by restricting the layout construction flexibility and providing enough resources for minimal circuit design impact. However, none of them actually present a fair comparison to the most adopted standard cells IC design methodology.

The analysis presented in this work aims to address this lack. It is based on general RTL templates, like the VCC and VCTA ones [5].

3. RTL Evaluation Flow

The comparison between RTL and standard cell approaches has been performed on the technology mapping step in the IC design flow. This task requires as input data the circuit description (usually in HDL format) and the target cell library. Thus, each benchmark circuit has been mapped using both a standard cell library and one (or more) RTL-based library. Then, the mapping results of area, timing and power dissipation have been compared. The evaluation flow is depicted in Fig. 1. The generation of the RTL-based library implies a series of choices including the unit template, the transistors sizing and the eventual transistor layout folding.

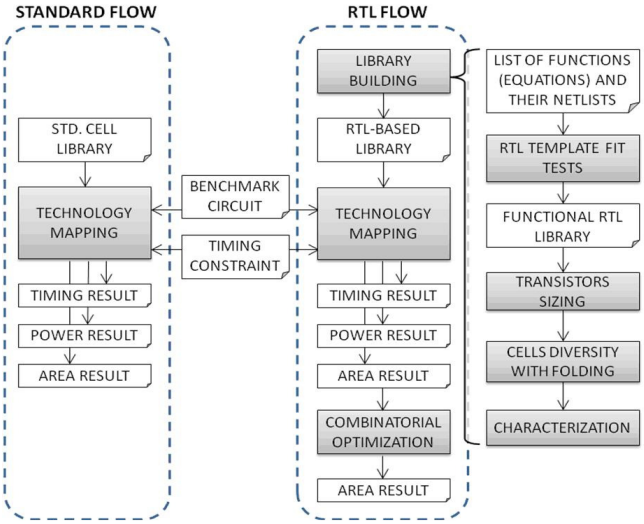


Figure 1 – RTL evaluation procedure.

3.1. Array unit template

Each RTL unit has a fixed number of metal lines (interconnection wiring) and transistors (PMOS and NMOS). The chosen amount of available resources in an RTL unit defines the 'template'. Each circuit must use only one template. Thus, the cells implemented over a certain template differ exclusively by their vias positioning while the high level of layout regularity is kept.

Since the circuit design is directly dependent on the RTL template, the characteristics of the candidate templates must be carefully analyzed. For instance, the largest ones are usually able to implement more complex CMOS gates. Thus, the mapped circuit tends to present less cell instances, but at same time each instance consumes more area. In order to provide a fair evaluation of this trade-off, different RTL templates have been investigated. The templates are labelled according to their resources: '2M6T', '4M4T', etc. The '2M6T' template, for instance, means that every built cell in this respective library has 2 lines of metal 1 available for gate input connection and 6 transistors in each plan (a total of 12 transistors).

Hence, each template acts as a filter for building an RTL-based cell library, containing only the functions whose implementation fits on the template resources.

It must also be pointed out that an efficient use of the available RTL unit resources suggests the building of more than one (independent) logic gate. For example, an inverter and a 2-input NAND gate can usually be implemented in the same unit. In this case, the RTL unit can be understood as a multiple-output cell, where each output is a function of an exclusive subset of inputs. The conventional mapping engines are not able to deal with such kind of cells. Thus, a post-processing step of combinatorial optimization has been applied. This step provides the best set of combinations between cell instantiations resulting from mapping that could share the same RTL unit.

3.2. Transistors sizing

All transistors of an RTL unit and, therefore, all transistors in a circuit using such a kind of design approach have the same width. It means that if the electric current capacity, or drive strength (i.e., the transistor channel width), is increased for benefiting the critical path propagation delay, it naturally penalizes the area of all cell instantiations in the circuit. Two possibilities have been taken into account: (a) using the minimum

transistor width associated to the technology node, and (b) using transistor widths which present good compromise between area and delay, obtained from inverter ring oscillators and from samples of circuit critical paths. Once all the cell netlists selected for a specific template were sized, this RTL-based library was electrically characterized in order to be used as input data in the mapping procedure.

3.3. Transistor layout folding

Besides the transistors width sizing, another way to deal with the electrical current capacity of cells is using the layout folding technique. It is attained by sharing the four terminals of a transistor with other one(s). It means, connect transistors in parallel to perform as a larger one. It is noteworthy that, as occur in mask-programmable gate-arrays, in the RLT approach only folding can be used for tuning the PMOS and NMOS sizing ratio (PN ratio) and for sizing compensation of stacked transistors, as well as for creating any other relationship between the pull-up PMOS and pull-down NMOS planes. Moreover, due to the area penalty, usually only inverters are provided in different drive strength options to act as buffers in critical paths.

4. Experimental Results

The results come from the application of the flow described in Section 3 (see Fig. 1), over specific choices of list of functions, target technology, the standard cell library as reference, and the set of benchmark circuits.

4.1. Cell libraries

The RTL-based cell libraries represent the libraries building through the flow described in Section 3, by considering a list of 4058 Boolean functions. This list encompasses all possible functions from 1 to 4 inputs, excluding the ones equivalent by inputs permutation (p-class [6]), plus the 76 functions of 5 and 6 inputs from the reference library known as ‘genlib_44-6’ [7].

A set of 21 different RTL templates has been tested. It is composed by all possible combinations between 2, 4 or 6 ‘metal 1’ lines used for input connection, and 2 to 8 transistors at each plan, i.e., the pull-up PMOS and pull-down NMOS ones.

In Table 1 is shown how many functions, among the 4058 ones mentioned above, are able to be inserted in a single RTL template unit when considering their transistors netlists.

Table 1 – Number of logic gates which fit each RTL template.

	Transistors at each plan						
M1 lines	2	3	4	5	6	7	8
6	**4**	**9**	**19**	**53**	**126**	**151**	**189**
4	**4**	**9**	**19**	**33**	**55**	**69**	**81**
2	**4**	**6**	**6**	**6**	**8**	**8**	**8**

4.2. Benchmark circuits mapping

For the technology mapping task, some ISCAS’85 benchmarks were applied [8]. The target technology was the PTM 45nm CMOS process [9], and the chosen standard cell library as reference for comparisons was the Nangate FreePDK45 Open Cell Library [9].

From the previous section, only three templates were considered appropriate for the execution of the whole RTL evaluation flow: 2M6T, which represents the smallest one able to implement an XOR gate, 4M4T and 4M6T, which were found to be the best compromise between unit area and number of possible functions.

In the following results, all transistors in the RTL units were sized to the minimum width allowed by the addressed CMOS process. Using this sizing choice, the timing performance of RTL templates are not as efficient as standard cell, as expected. However, the average increment in the minimum required period was only 10.26%.

Table 2 shows some area and power results of benchmarks mapped using the standard cell and RTL libraries. This table presents the results related to the tightest timing constraints, respected simultaneously by all libraries, considering steps of 50 ps.

According to Table 2, it is possible to obtain two different area comparisons. The first one is following the simplest flow, where each RTL unit presents a single transistors netlist and logic function, and the benchmarks area is directly resulted from the mapping. In this comparison the average area increment, when considering always the best RTL against the standard cells, is of approximately 110%. The second one is after the combinatorial optimization, representing a flow where different transistors netlists could be combined in the same RTL unit, as explained in Section 3. In this comparison the average area increment is reduced to around 65%. It is also evident that this optimization makes larger templates to become better choices, since they are more prone of using the functions combinations in order to reduce their average amount of unused transistors.

In this same Table 2, it is shown an average power decrement of 69.4%. It can be explained by the minimum transistor sizing applied in the RTL patterns.

Table 2 – Results of ISCAS'85 benchmarks mapped in standard cell (Nangate Lib.[9]) and RTL libraries.

		Nangate Lib		2M6T RTL			4M4T RTL			4M6T RTL		
Circuit	**TC[a] (ps)**	**Area (µm)**	**Power (nW)**	**Area (µm)**	**Area[b] (µm)**	**Power (nW)**	**Area (µm)**	**Area[b] (µm)**	**Power (nW)**	**Area (µm)**	**Area[b] (µm)**	**Power (nW)**
c17	50	6	128	17	15	39	14	9	31	17	12	31
c432	450	161	5225	483	418	2314	488	388	2053	533	357	2027
c880a	300	316	9445	792	707	3557	791	581	2834	930	588	2933
c1908a	450	331	19428	414	367	5157	625	495	5111	625	438	5258
c499	400	364	24180	536	469	8542	830	657	8004	720	530	9308
c1355	400	372	24155	555	488	9368	832	657	7917	726	530	9099
c1908	500	380	19393	730	625	7194	983	764	7745	999	683	6506
c2670a	300	550	20734	1235	1096	7085	1411	1083	6181	1526	976	6119
c2670	350	574	21373	1197	1066	7153	1252	976	5891	1403	916	5962
c3540a	600	682	28226	1695	1503	9898	1623	1284	8240	2004	1308	9040
c3540	650	849	30636	2208	1999	12925	2214	1733	10714	2313	1578	10073
c5315a	500	1136	50509	2599	2341	17995	2601	2027	15661	2670	1852	15297
c5315	550	1171	48292	2531	2310	16908	2752	2123	15069	2765	1915	14342

[a]TC = The smallest timing constraint respected simultaneously by all libraries.
[b]The second 'Area' column represents the circuit area after the combinatorial optimization step.

5. Conclusions

A comparison analysis between the regular transistors layout (RTL) design strategy and standard cell methodology was presented. Experimental results were obtained using different RTL templates, and the templates efficiency varied from one benchmark circuit to another. The mapping results of benchmarks shown that RTL designs are able to reach a good power performance and an acceptable timing performance at a reasonable cost of area penalty. However, the area overhead can be significantly minimized if the design exploits efficiently the resources available in RTL patterns, for instance, by sharing of the same unit by more than one netlist.

6. Acknowledgements

Research partially funded by Nangate Inc. under a Nangate/UFRGS research agreement, by CNPq and CAPES Brazilian funding agencies, and by the European Community's Seventh Framework Programme under grant 248538 - Synaptic.

7. References

[1] A. K.-K. Wong, Resolution Enhancement Techniques in Optical Lithography (SPIE Tutorial Texts in Optical Engineering Vol. TT47). SPIE Publications, 2001.

[2] L. Pileggi, H. Schmit, A. J. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Y. Tong, "Exploring regular fabrics to optimize the performance-cost trade-off," Proc. of Design Automation Conference (DAC), pp. 782- 787, 2003.

[3] T. Jhaveri, V. Rovner, L. Liebmann, L. Pileggi, A. J. Strojwas, and J. D. Hibbeler, "Co-Optimization of Circuits, Layout and Lithography for Predictive Technology Scaling Beyond Gratings," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, ,vol.29, no.4, pp.509-527, Apr. 2010.

[4] Y. Ran and M. Marek-Sadowska, "On designing via-configurable cell blocks for regular fabrics," Proc. of Design Automation Conference (DAC), pp. 198- 203, 2004.

[5] M. Pons, F. Moll, A. Rubio, J. Abella, X. Vera, A. González, "VCTA: a via-configurable transistor array regular fabric," Proc. of VLSI System on Chip Conference (VLSI-SoC), pp.335-340, 2010.

[6] T. Sasao, Equivalence Classes of Logic Functions. Switching Theory for Logic Synthesis. Kluwer Academic Publishers. 1999.

[7] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: a system for sequential circuit synthesis," Technical Report UCB/ERL M92/41, UC Berkeley, 1992.

[8] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark design and a special translator in Fortran," Proc. of IEEE Int'l Symp. on Circuits and Systems (ISCAS), 1985.

[9] Nangate Open Cell Library FreePDK 45nm. Avaliable at http://www.nangate.com/. Version 1.3 07/2009.

SET and SEU Simulation Toolkit for LabVIEW

Walter Calienes Bartra, Fernanda G. de Lima Kastensmidt, Ricardo Reis
{wecbartra, fglima, reis}@inf.ufrgs.br

PGMICRO, Instituto de Informática
UFRGS

Abstract

Nowadays, the fault simulation process is an important step for any project. Predicting the behavior faults of any process step is essential to ensure that the project is well designed. During the simulation various problems can be detected and corrected. A tool capable of simulate the effects that occur when a source of fault is inserted in a digital circuit, in specific SEU faults, is presented. In addition to modeling a fault, is implemented a TMR method capable of verifying the existence of a fault and not let it spread through the whole circuit. Also is implemented a Voltage Controled Oscilator (VCO) for view fault effects in this analog circuit. LabVIEW tool is used to create a set of virtual instruments to simulate SEUs, it is efficient for modeling the characteristics of its, SETs and others. It is possible with this library to replicate the effects of SEUs and SETs described in the literature.

1. Introduction

With the advancement of technology and the shrinking of device size makes integrated circuits more susceptible to errors due to radiation effects. The radiation sources may come from the space (solar flares, Van Allen belts, solar wind or cosmic rays)[1], or from radioactivity or electromagnetic sources generated on Earth. Once the circuit is exposed to a radiation source, it can have its logical value changed altering the characteristics of the circuit or even permanently disabling the device, depending on the amount of radiation to which the circuit is exposed.

In the literature, these faults are represented by the designation of Single Event Upset (SEU) and Single Event Transient (SET). SEU is a fault that changes a bit from a register, for example, a register with a logical value 1 is replaced by a logical value 0 after being affected and vice versa. SET affects the functionality of the transistor, creating an anomaly that can affect the output of a logic gate. The fault occurrence probability changes with the increase of simultaneous faults in both sequential or combinational circuits [2].

Programs such as *AMATISTA* [3] are used in the industrial area to develop fault-tolerant circuits. Another software, *FPGA-based fault simulator* [4] using partial reconfiguration to simulate SEU faults by changing the interconnections in FPGAs. Programs like *Fsimac* [5] simulate faults on combinational asynchronous circuits. *LIFTING* [6] use Stuck-at fault models for fault simulation in circuit interconections. All these simulation programs require prior knowledge of hardware description languages such HDL or C code to develop their functions. LabVIEW, a graphical and parallel language, is easy to learn and fast to debug, which makes it ideal for engineers and scientists, even those with limited programming skills [7].

This work consists in the development of a toolkit for LabVIEW. LabVIEW allows the prototyping of tools and accessories in an agile and efficient way [7]. One of the features that this tool has is its ability to run the program inserted concurrently, in addition, its programming is entirely visual. The programs written with LabVIEW are called virtual instruments (VIs). In the literature a fault simulation toolkit of integrated circuits, for this graphical programming environment was not found.

The rest of the paper is organized as follows: secton 2 describes the toolkit developed, section 3 shows the results obtained in the experiments, section 4 shows the conclusion and the future works are in the section 5.

2. Toolkit Development

VIs are based on the use of arrays and are clasified in of three groups: Logic Gate Simulation Tools, Boolean Logic Simulation Tools and Analogic Simulation Tools. The toolkit was developed on NI LabVIEW 8.20 Professional Development System platform (it is possible to adapt these tools to later versions by using the Mass Compile option of LabVIEW).

2.1. Logic Gate Simulation

This part of the package contains several tools to simulate logic gates under certain faults (like SET). This toolkit contain components like logic gates, registers and clock generators. These components have variable range of voltage, that's is the reason why they have defined forbidden gaps. The input and output formats are Arrays of Double Precision Floats.

2.2. Boolean Logic Simulation

Is a part of the toolkit that contains tools to simulate effects on systems of bivalent logic (logic "1" or logic "0"). It is based on the use of logic functions of LabVIEW and their data is the Boolean type. Some of these VIs require structures with Shift Registers (FOR or WHILE loops) to operate properly.

2.3. Analog Simulation

The tootkit also has some analog circuits (such as switched capacitors, current mirrors, transistors, etc..) in order to simulate more complex circuits and to study the effect of failures on them. This part is based on using Point-by-Point functions (very useful for real-time simulations). This part of the package is still in testing stage.

3. Experiment Results

The purpose of using LabVIEW to do the simulations is trying to build a VI that is most similar to the required circuitry using VIs developed for the toolkit. The faults are inyected using special VIs created for this purpose. The colected data are shown with the LabVIEW Front Panel tools.

3.1. Inverter with Capacitive Load and SET

Figure 1 shows a circuit that has only an inverter and a capacitive load that is affected by the SET, which can be misinterpreted as a "1" when it should be a logical "0". The inverter input signal source is a square signal of f=1MHz and has a supply voltage in 2.4V. The load of this circuit is a capacitance of 100fF that simulates the fan-out of an inverter. This figure also shows the Block Diagram simulation of the circuit. The input signal is generated by the function Square Waveform provided by LabVIEW; for this case to use a sampling rate of F_s=60G samples per second to generate the signal #s=1M samples for simulation #s/F_s= 16.7µs (must remember the Nyquist criterion: f ≤F_s/2). The output voltage is linearly affected by the resulting voltage between SET current and resistance in the inverter output. These effects can be seen on the Front Panel of the simulation. The SET pulse can be interpreted as an wrong logic "1" if a capacitive load is replaced with a logic gate or register.

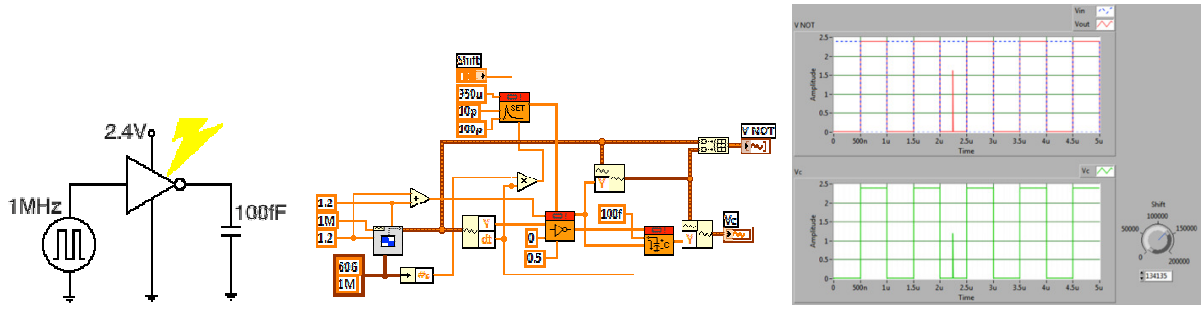


Figure 2. Inverter Circuit with SET, Block Diagram and Front Panel of the simulation.

3.2. Bit-Flip Simulation in TMR Time Redundancy Memory Cell.

Bit-Flip can happen in different parts of a circuit, commonly found in registers and data input circuit. Its nature is totally random and it may happen at any time. Figure 2 shows a circuit that uses the TMR technique with redundancy in time to make the circuit tolerant to faults. We tripled the D registers and added a Voter in the output of triplicated circuits. Each of the three registers use the same clock frequency, but each has a clock time offset to a certain amount of degrees (this shift is represented by Δt). To represent the randomness and change the data value due to a Bit-Flip using a Poisson Noise Generator with a Buffer with hysteresis to prevent the noise to be very aggressive. With this, it is possible to observe better the effects of Bit-Flip.

In the Block Diagram, it is possible to see the VIs responsible for the production of Poisson Noise, the Data Generator and the Clock Generator in Triplicated Delay Wave. Moreover, one can see the Bit-Flip Generator that is inside on a FOR structure along with the three D Registers and a Voter. Figure 3 shows the Front Panel of this VI, that have a Clock Frequency, Data and Clock Shift controls. The outputs is a graph with four signals: an Input Data, the Random Fault Signal, the output of the TMR circuit and the Clock Signal.

After running this simulation with different values of clock offset Δt and frequency, making a comparison between this circuit and a circuit without Bit-Flip, we draw a trend in the number of average errors vs. the frequency for different values of delay clock. It can be concluded that this technique is good for relatively low frequencies (between 20 and 90kHz) and with a displacement Δt of 120 degrees. If the frequency is too low or too high, you can see the increase of errors in an alarming rate. A very low Δt shift implies a amount of large errors in the data output circuit.

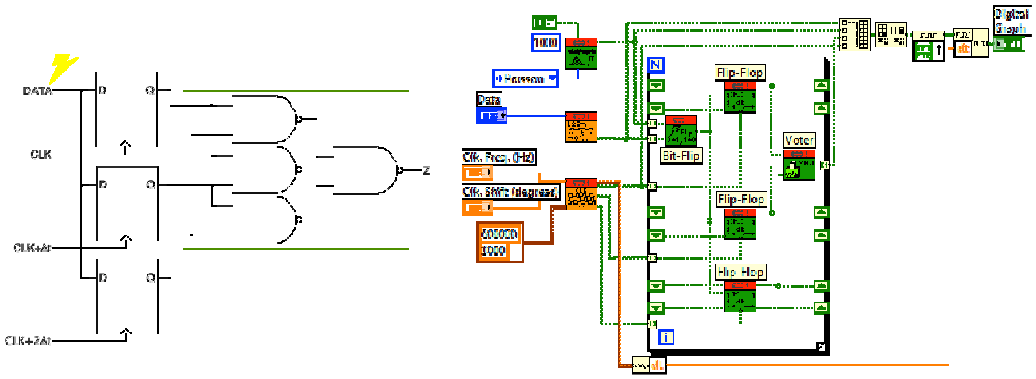


Figure 2. TMR Time Redundancy Circuit and LabVIEW Block Diagram.

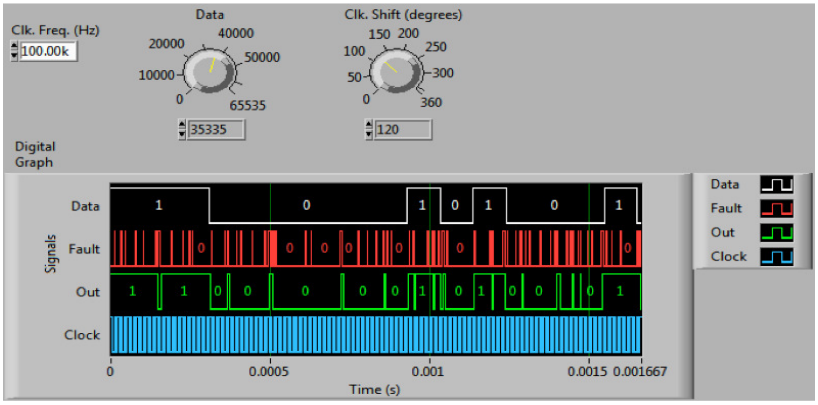


Figure 3. Front Panel of the simulation of TMR Memory Cell Circuit with Bit-Flip.

3.3. VCO Simulation with LASER Pulse

For this simulation, functions Point-By-Point are used to simulate a voltage controlled oscillator (VCO) affected for a short time LASER exposition or LASER pulse. For this simulation, a FOR structure is needed because your shift registers are used for feedback a data [7]. A VCO is based primarily on two tank circuits, which consist in an inductance in parallel with a capacitor or varicap diode. These tank circuits generates a sine wave along with two NMOS transistors crossed to generate a negative resistance that makes the VCO oscillate. Figure 4 shows this basic circuit. All particle impact lasts a short time. The effects are almost undetectable in analog circuits because the particles contribute little (around 50fC), the effect is very similar to pink noise of short duration. To see effects of faults in these circuits, must be simulated a pulse with high duration and high current equivalent to more than 30pC on the cross-pair transistors that form the negative resistance using an asynchronous LASER pulses [8].

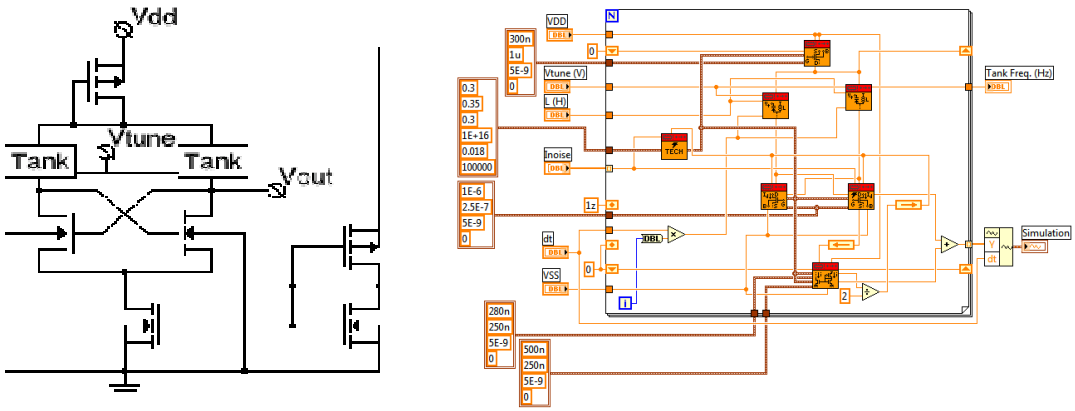


Figure 4. VCO Circuit and its correspondent Block Diagram.

For this simulation, an inductance of 1nH and a control voltage (V_{tune}) of about 1.77V is used, this produce a sine wave of 1.25GHz in the VCO output. Figure 5 shows the simulation when a SET fault with 25mA of maximum current and charge equivalent to just over 37pC is inyected. The behavior shown is similar to the

SPICE simulation results in [8]. The lack of modeling parasitic capacitances in transistors add inaccuracy to the simulation done with this toolkit.

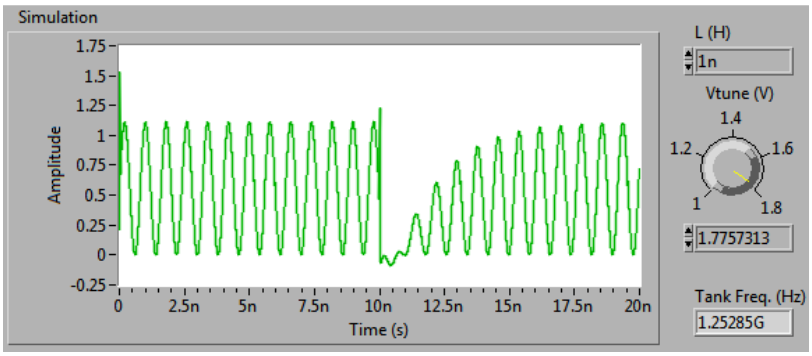


Figure 5. Front Panel of the VCO simulation

4. Conclutions

Several methods of insertion and mitigation of faults in digital and analog circuits were tested in accordance with previous work on these issues.

After developing VIs for the simulation, generation of faults and test of fault tolerance technics, it is demonstrated that LabVIEW is a good platform for prototyping, programming and simulation. Moreover, the methods implemented in this work have served to understand these topics that are so important for modern technology.

5. Future Works

As future works, the model of logic gates should be improved. Also fault models need to be ajusted according to current technologies. It is also thought to add more functions, such as Hamming code generators, EDAC and functions for failure analysis in ICs and RAM memories. The plan contemplate to use some VIs that simulate BIST circuits based in LFSR circuits type Fibonacci and Galois.

The modeling of transistors Point-By-Point will be improved, to include an efficient model of parasitic capacitances. In addition, the use of a more sophisticated transistor model, such as ACM model [9] is needed, because transistor models affect both analog and digital fault modeling.

6. References

[1] R. Velazco, P. Fouillat, R. Reis, *"Radiation Effects on Embedded Systems"*. Dordracht, The Netherlands: Springer, April 2007.

[2] N. Miskov-Zivanov, D. Marculescu, *"Multiple Trasient Faults in Combinational and Sequential Circuits: A Systematic Approach"*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, October 2010.

[3] I. Gonzáles, L. Berrojo, *"Supporting Fault tolerance in a industrial environment: tha AMATISTA approach"*. Seventh International On-Line Testing Workshop, 2001. Proceedings, pg. 178-183.

[4] L. Kafka, O. Novák, *"FPGA- based fault simulator"*. 2006 IEEE Design and Diagnostics of Electronic Circuits and Systems. Proceedings, pg. 272-276.

[5] S. Sur-Kolay, M. Roncken, K. Stevens, P. P. Chaudhuri, R. Roy, *"Fsimac: A Fault Simulator for Asynchronous Sequential Circuits"*. 9th Asian Test Symposium, 2000. Proceedings, pg. 114-119.

[6]. A. Bosio, G. Di Natale, *"LIFTING: a Flexible Open-Source Fault Simulator"*. 17th Asian Test Symposium. Proceegings, pg. 35-40.

[7] G. W. Johnson, R. Jennings, *"LabVIEW Graphical Programming"*, 4th ed. New York, USA: McGraw-Hill, 2006.

[8] W. Chen, V. Pouget, H. J. Barnaby, J. D. Cressler, G. Niu, Pascal Fouillat, Y. Deval, D. Lewis, *"Investigation of Single-Event Transients in Voltage-Controled Oscillators"*. IEEE Transactions on nuclear Science, Vol. 50, Num 6. December 2003.

[9]. O. da Costa Gouveia Filho, A. I. Araujo Cinha, M. Cherem Schneider, C. Galup Montoro, *"The ACM Model for Circuit Simulation and Equations for Smash"*. Florianópolis, Sep. 1997.

Prematurely Aborting Linear System Solver in Quadratic Placement

Guilherme Flach, Marcelo Johann, Ricardo Reis
{gaflach, johann, reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul (UFRGS)

Abstract

In this work we develop a new quadratic placement technique where we prematurely abort the linear system solver to save runtime. This has low impact in the placement flow due to the expand-contract phenomena intrinsic to the ICCG solver outlined in this paper. A coloring scheme for placement useful for comparing visually placement results is described. The coloring scheme then is used to create a partitioning strategy to insert more global view in our placement technique. Our placer based on the ideas developed in this paper is 25% faster for largest ISPD 2002 benchmarks than FastPlace 3 with no significant penalty in wirelength.

1. Introduction

By definition, the placement problem is easy to be stated, although solving it efficiently is commonly a challenging task. Simply put, its goal is to place circuit components evenly on the circuit area so that wirelength and other parameters are targeted.

Among several placement techniques, quadratic placement [1] has taken most of academic efforts today since it is in general faster and achieves comparable results to other state-of-the-art strategies. In this paper, we present a new standard cell placement tool inspired by quadratic placement technique.

Our main contributions are:

- a new fast placement technique able to deal with 100K cells in few seconds;
- a new coloring placement scheme to aid placement result comparisons;
- a placement result improvement through partitioning;
- an easy to parallelize placement tool.

2. The Anatomy of a Quadratic Placer

A quadratic placement tool is built around a linear system which describes the cell connectivity as well as the forces that are used to remove overlap between cells as depicted in Figure 1. The solution of the linear system provides the equilibrium cell positions where the total force acting on any cell is zero. When no force is applied besides the connectivity, the system solution is equivalent to the one which minimizes the sum of quadratic distance between connected cells, hence the name quadratic placement.

Since the linear system without overlap removal forces provides invalid solution with large amount of overlap, spreading forces must be added in order to spread cells evenly over circuit area so that overlaps are reduced. The spreading forces are added based on the current placement solution. This creates an iterative process where one interleaves the system resolution and the system modification until the cells are evenly spread over circuit area.

The differences between quadratic placers come from the way the cell connectivity is modeled and mainly from the the way the spreading forces are introduced in the linear system.

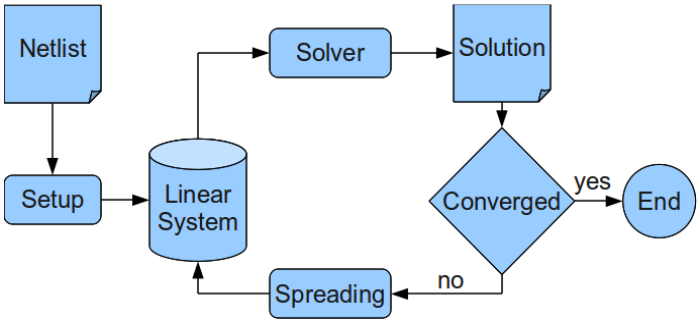


Fig. 1 – The Anatomy of a Quadratic Placer.

3. A New Quadratic Placement Technique

As mentioned previously, quadratic placers differs from each other mainly in the way spreading forces are added to the system. Our tool, on the other hand, also adapts the quadratic placement flow by modifying the

way the linear system is solved. In fact, in our tool, the linear system is not entirely solved as the solver is interrupted in its very first iteration. At first glance, this may seem a little disruptive, but it is grounded by our experimental observation that one iteration of the solver suffices to provide a good approximation to where cells should be moving. When interrupting the solver at first iteration, the placer is able to save runtime since in general many iterations are required to converge to the solution.

To come to this idea, we set the following experiment. Using the Incomplete Cholesky Conjugate Gradient (ICCG) method [2] for solving the linear system, we have plot cells at every solver iteration instead of just plotting after the solver converges. This shown a very interesting phenomena: cell positions initially expand and then contract back to their final position. Figure 2 illustrates this phenomena.

The expansion is caused by the spreading forces added to the system whereas the contraction is due to cell connectivity. At initial solver iterations spreading forces cause cell positions to overestimate their final positions. As the system converges, the cells are contracted back until they reach their final position. Although, cell position overestimate final position, the overestimation is bounded by cell connectivity.

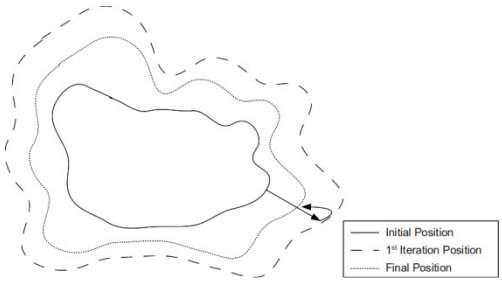


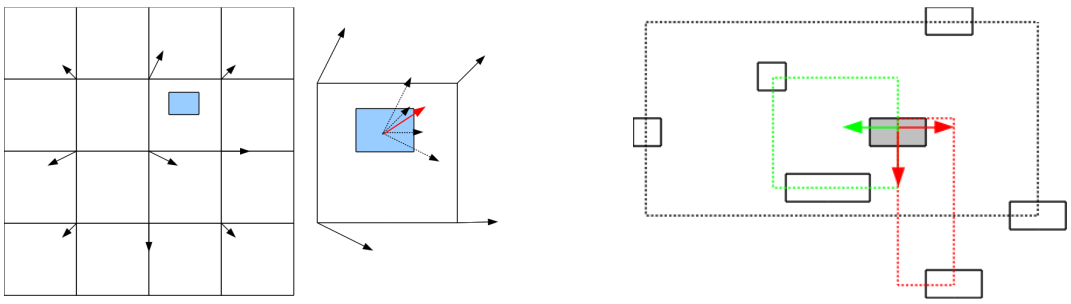
Fig. 2 – Expansion-Contraction ICCG Phenomena.

Looking at flow in Figure 1, after cell have contracted back we apply techniques to spread them. This indicates that we are repeating a process that is intrinsic to the ICCG in its initial iterations. Based on such observation, we develop our placer executing only the first iteration of the solver and then applying the spreading force techniques to reflect the new circuit utilization.

3.1. Spreading Forces

Spreading forces are added to the system to reduce overlap between cells. To compute spreading forces in our tool, first the placement area is divided hierarchically in rectangular regions. At first level, the placement area is divided in four same-sized regions. At next level, each region of the previous level is divided in four same-sized regions. The number of levels is chose such that the region area of the last level meet approximately the cell average area.

After placement area is divided, gradient forces are computed at corner of each region in each level. Gradient forces point to high density regions to low density ones. Finally, to compute the spreading force acting on a cell, we interpolates the gradient forces of the four corners at the center of the cell as shown in Figure 3a. This process is done in each level and the final gradient force acting on the cell is the average gradient force for all levels.



(a) (b)

Fig. 3 – Linear system modification: (a) spreading forces; (b) HPWL forces.

3.2. HPWL Forces

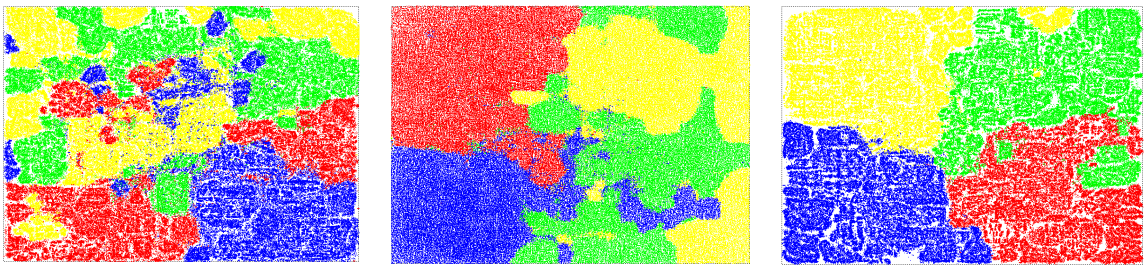
A common reported drawback of quadratic placement technique is that it only indirectly accounts for half-perimeter wirelength (HPWL) since the system tends to reduce quadratic wirelength. Our placer applies HPWL forces to aim more accurately the HPWL.

HPWL forces affects only cells on the boundary of the bounding-box. For such cells, a force is added in order to push it to the opposite border. This force is fixed, i.e., does not vary on the net cardinality neither the bounding-box area. The computation of HPWL forces is exemplified in Figure 3b.

4. Improving Placement Results Using Partitioning

During the development of our placer, we noticed that our approach is faster, but was not capable to meet wirelengh results from other state-of-the-art placer. To try to discover why our placer could not beat other placer in wirelength, we developed a technique for placement coloring to compare visually different placement results. It is known that placement and partitioning are in some extent correlated [3]. In fact, many placers use partitioning as their main algorithm or as a heuristic to improve placement results. Supported by this correlation, our coloring technique use partitioning to color each cell. Basically cells are partitioned into *n* groups and to each group a color is set. Due to the correlation between placement and partitioning, it is expected that same-colored cells will be clustered together.

Finally, we compared our placement results with the FastPlace 3 [5] results visually. The comparison for the *ibm18* benchmark [4] is shown in Figure 4 where we can notice that our placer (a), in fact, cluster same-colored cells, but clusters are more messy spread than the result from FastPlace (b). This fact indicates that our placer was not able to deal with the global view of the problem, although it was doing a good job in the local view.



(a) (b) (c)

Fig. 4 – Placement coloring. (a) Our prior placer result; (b) FastPlace 3 result; (c) Our final placer result using partitioning for initial placement.

Using the results of this coloring techniques, we used partitioning to impose a initial relative order between cells of different color. We partitioned cells in four groups and placed each group in the center of the four placement area quadrants instead of putting all cells in the middle of the circuit. This increased the total run-time of the placement, but allowed our placer to provide state-of-the-art results. The result after partitioning is presented in Figure 4c.

5. Results

To check our placer performance we run it over the ISPD 02 benchmark set [4] and compare it to the state-of-the-art placer FastPlace 3 [5]. FastPlace 3 is currently one of the faster academic placers providing comparable results with other state-of-the-art placers. Our placer receive as input the 4-way partitioning result provide by the hMetis [6]. The run-time of the hMetis is not accounted in the results, however, we point out that for the largest benchmark, *ibm18*, it took only *9s* to run the partitioning.

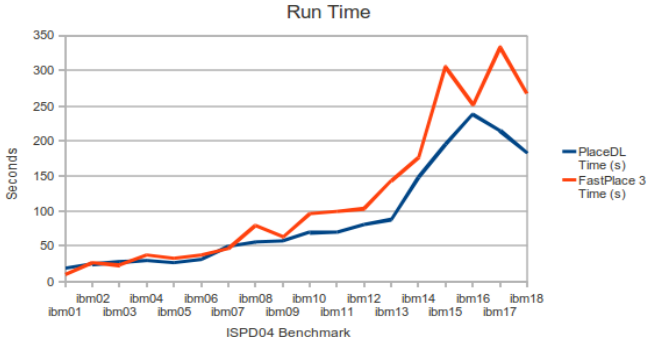


Fig. 5 – Results on the ISPD 02 benchmark set.

In the Table 1 we see that our placer is in average 13% faster while providing the same average wirelength. It is important to highlight that as the benchmark size increases, our placer achieve larger runtime gains, although a little increase in the wirelength is noticed. If we average only the nine largest benchmark w.r.t the number of nodes, the average runtime reach up 25%. This effect can be seen in the Figure 5.

Tab.1 - Our placer results on ISPD 2002 benchmark set.

Benchmarks	Our Placer		FastPlace 3		Our Placer / FastPlace3	
	HPWL	Time (s)	HPWL	Time (s)	HPWL	Time
ibm01	1.69E+006	18.8	1.72E+006	10.05	0.98	1.87
ibm02	3.59E+006	24.32	3.60E+006	26.39	1.00	0.92
ibm03	4.96E+006	27.34	4.66E+006	22.71	1.07	1.20
ibm04	5.64E+006	29.64	5.73E+006	36.8	0.98	0.81
ibm05	9.45E+006	26.72	9.88E+006	32.76	0.96	0.82
ibm06	4.94E+006	31.49	4.94E+006	37.16	1.00	0.85
ibm07	8.37E+006	49.28	8.23E+006	46.66	1.02	1.06
ibm08	8.99E+006	55.96	9.05E+006	79.18	0.99	0.71
ibm09	9.48E+006	57.77	9.67E+006	63.13	0.98	0.92
ibm10	1.73E+007	69.33	1.73E+007	96.21	1.00	0.72
ibm11	1.41E+007	69.89	1.46E+007	99.71	0.96	0.70
ibm12	2.19E+007	80.46	2.28E+007	103.49	0.96	0.78
ibm13	1.69E+007	87.91	1.69E+007	143.33	1.00	0.61
ibm14	3.32E+007	148.82	3.19E+007	176.38	1.04	0.84
ibm15	3.90E+007	195.09	3.85E+007	305.52	1.01	0.64
ibm16	4.49E+007	238	4.42E+007	251.51	1.01	0.95
ibm17	6.20E+007	214.62	5.99E+007	333.18	1.04	0.64
ibm18	4.18E+007	182.27	4.07E+007	267.58	1.03	0.68
				AVG	*1.00*	*0.87*

6. Conclusions and Future Work

In this work, we presented a new quadratic placement technique where we prematurely abort the linear system solver at the very first iteration. This has low impact in the placement flow due to the expand-contract phenomena intrinsic to the ICCG solver outlined in this paper. Prematurely aborting the solver save runtime allowing our placement tool run faster than state-of-the-art placers.

We also presented the use of partitioning to insert global view in quadratic placement. The lack of global view in our prior placer was detected thanks to a new scheme for placement coloring developed in this work. The coloring scheme set the cell color based on the partition that cell belongs to and can be used for researches to visually compare their results.

Besides being fast, most of the pieces of our placer are easily to parallelize as the linear algebra operations and gradient computation. This allows our placer to scale well as the circuit size increases.

7. References

[1] C. J. Alpert , T. Chan , D. J.-H. Huang , I. Markov , K. Yan, “Quadratic placement revisited”, Proceedings of the 34th annual conference on Design automation, p.752-757, June 09-13, 1997, Anaheim, California, United States.

[2] R. Beaumens. “Iterative solution methods”, Applied Numerical Mathemetics. vol.51, No 5, 2004.

[3] M. A. Brever,“Min-Cut Placement”, Journal of Design Automation and Fault Tolerant Computing, Oct., 1977, pp. 343-362.

[4] N. Viswanathan , C. C. Chu, “FastPlace: efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model”, Proceedings of the 2004 international symposium on Physical design, April 18-21, 2004, Phoenix, Arizona, USA.

[5] N. Viswanathan , M. Pan , C. Chu, “FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control”, Proceedings of the 2007 Asia and South Pacific Design Automation Conference, p.135-140, January 23-26, 2007.

[6] G. Karypis and V. Kumar. “hMETIS 1.5: A hypergraph partitioning package”. Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the Web at URL http://www.cs.umn.edu/~metis.

Decreasing Transistor Count Using an Edges Sharing Technique in a Graph Structure

Vinícius N. Possani, Luciano V. Agostini, Felipe S. Marques, Leomar S. da Rosa Jr.
{vnpossani, agostini, felipem, leomarjr}@inf.ufpel.edu.br

Group of Architectures and Integrated Circuits – GACI
Federal University of Pelotas – UFPel
Pelotas – Brazil

Abstract

Increasingly, in VLSI designs, the integrated circuits have higher density of transistors on the small physical area, power consumption reduced and greater performance. An important factor that has contributed for this is the representation of logic functions with a reduced number of transistors. Thus, we sought an alternative solution to common methods, such as factorization, to generate optimized networks. This paper presents a graph-based structure to represent a transistor network and a technique to reduce the number of transistors by edges sharing. Our method can achieve non-series-parallel arrangements while methods based in factorization can only derive series-parallel arrangements, which may not be the best solution. Thus, when applied to the set of 4 input p-class logic functions, our method has advantages if compared to the good-factor algorithm implemented in SIS Software. Also, in other logical functions our algorithm can achieve results as good as those generated by techniques based in BDD.

1. Introduction

The micro electronics industry has brought great advances in last years, no doubt, designing digital circuits VLSI becomes an increasingly task of extreme complexity and high cost of resources and time. In this context, aid tools are applied to support these projects, contributing to the designers manipulate more transistors and decreasing the development cycle. Therefore, the automatically generation of transistor networks makes simple some arduous tasks. Moreover, it also reduces the aggregate cost to the final product.

This paper proposes an edge sharing method, on a graph structure, to generate optimized transistor networks. In our approach, the input Boolean expression is translated into a graph that is later optimized through edges sharing. Nowadays, alternative methods which are available in the literature has been study and applied in this context. They are based on graph optimizations, were each edge in the graph keeps an association with a transistor in the network. The main idea is try to minimize the edges in an existent graph [1] or to compose a new graph with a reduced number of edges [2]. These alternative methods are used because the common technique to optimize a transistor network is based on factorization [3-4] and this may not be an optimum solution [5]. In factorization method an input Boolean expression is manipulated in order to reduce the number of literals that compose the expression. Subsequently, this optimized expression is translated in a transistor network composed by a reduced number of switches. In this sense, our sharing method intent derives non-series-parallel arrangements in order to deliver better results than the common technique.

2. Edges Sharing Method

The edges sharing method considers as input a sum-of-products (SOP) expression. In order to translate the expression to a graph, a parser is needed. The parser will deliver one vector of literals for each product storing these vectors in a list. Afterward, it is started the assembly of the graph by removing vectors one at a time from this list and creating an edge in the graph for each literal found in the vector. As an example we will use the Exp. (1) which represents a 'XOR' with 4 inputs. Fig. 1.a shows the graph obtained of this expression.

$$!A*!B*!C*D + !A*!B*C*!D + !A*B*!C*!D + !A*B*C*D + A*!B*!C*!D + A*!B*C*D + A*B*!C*D + A*B*C*!D \quad \text{(Exp.1)}$$

In the sequence, all paths in the graph are traversed in order to recognize identical edges (edges that represent same literals and have at least one vertex in common). If this condition is verified in the graph, then the identical edges are shared. This procedure consists in keeping only one of these identical edges, eliminating the remaining edges and merge the vertices that connect them. The vertices that will be merged are detached with the circumferences without fill in the figures below. This is exemplified in Fig. 1.b where the edge '!A' was shared and the vertices 1, 5, 8 and 11 were merged. Now the edge 'A' will be shared generating the graph shown in Fig. 1.c. So, in this moment the vertices 8 and 17, one at a time, are considered the new starting point of the optimization process, where the algorithm sought identical edges between these two vertices and the

vertex 4. This way the edge '!B' connected to the vertex 8 will be shared and in sequence this occurs with edge 'B'. Afterward that, the same process is applied to the edges '!B' and 'B' attached to the vertex 17. This is demonstrated in Fig. 1.d.

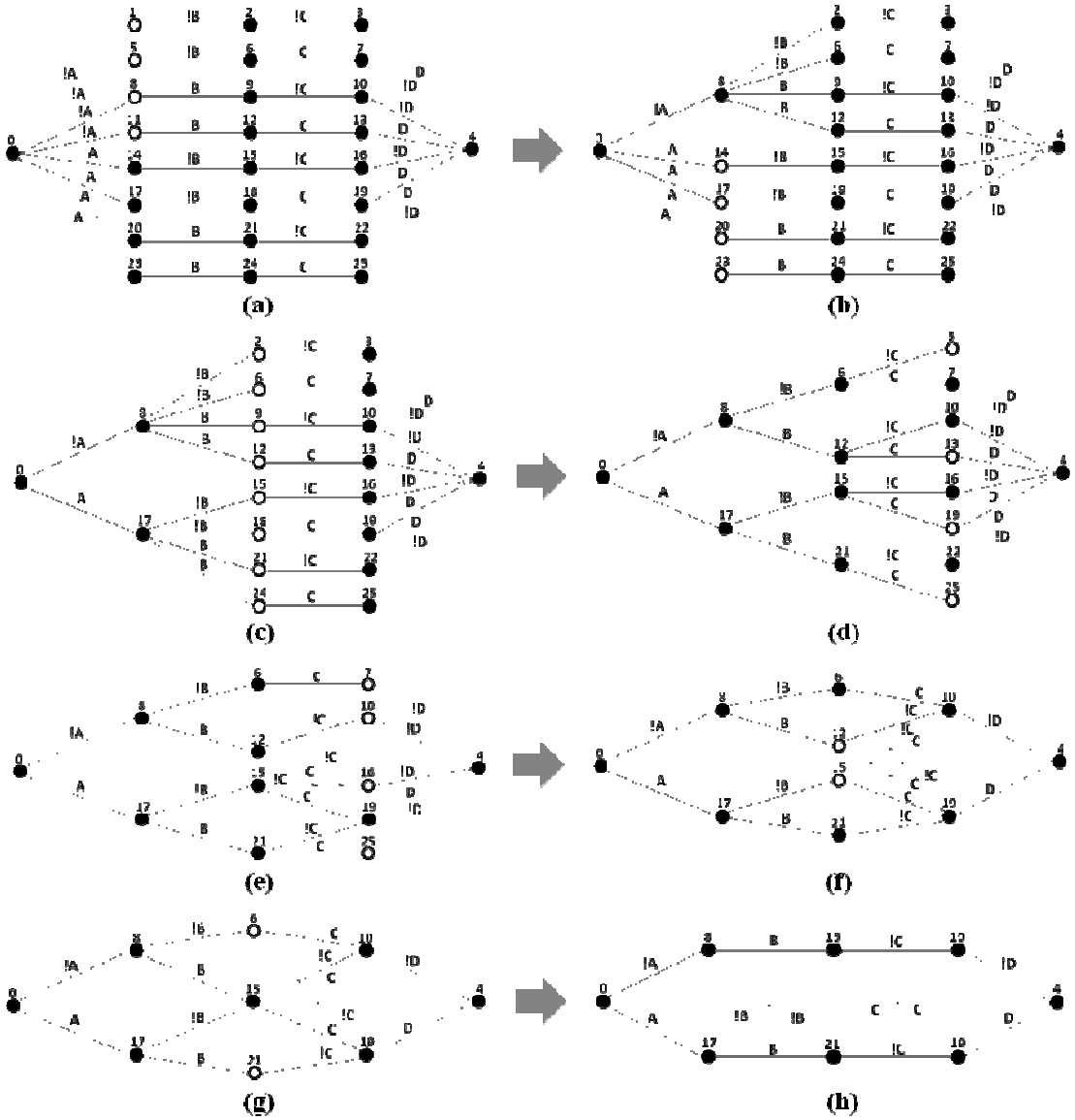


Fig. 1 – Steps of the sharing method to a 'XOR' with four inputs.

Considering the Fig. 1.d, departing from the vertices 6, 12, 15 and 21, one at a time, and traversing the graph toward the vertex 4, it is not possible to perform new optimizations because identical edges are not found between these vertices and the vertex 4. To perform new optimizations the sharing algorithm is applied from the end to the beginning of the graph. Thus, departing from the vertex 4 the edges 'D' are identified and were shared as the Fig. 1.e. demonstrates. In the next step the edges '!D' will be shared resulting in the graph of the Fig. 1.f.

Now, consider the vertices 10 and 19 as new start points of the sharing algorithm. There are two edges '!C' connected on the vertex 10, as Fig. 1.f shows. Then it is possible to remove the edge '!C' attached to the vertices 10 and 12 merging the vertices 12 and 15. In this case, the merging of the vertices 12 and 15 will derive two edges 'C' between the vertices 19 and 15. When this is detected, just one of these edges remains in the graph. Fig. 1.g shows this state of the graph. This process is applied again, but this time to the edges 'C' that are connected to the vertex 10, merging the vertices 6 and 21. This will derives two edges '!C' between the vertices 19 and 21, one of this edges will be removed resulting in the final graph illustrated by Fig. 1.h. Afterward, starting from the vertex 19, it is not possible to perform other optimizations. Thus, the optimization process ends. If any of these processes generates an invalid path, a recovery routine is invoked and the process is reversed. In the next session this procedure will be explained.

3. Validation Procedure

To guarantee that the optimized transistor network will be the faithful representation of the original expression, making sure that all products described in SOP are present in the resultant graph and sure that sneak-paths (forbidden paths) are not introduced on the network, a validation procedure is applied. Thus, it is necessary to validate all paths of the graph each time an edge is shared. The paths of the graph are generated through a recursive algorithm applied on the adjacency matrix that represents the graph. The algorithm uses a list structure to store the indexes of the matrix that make up a path of the graph.

Consider as example the simple graph illustrated by Fig. 2.a. The first step is to insert in the list the index '0' that indicates the row of matrix that represents the initial vertex of the graph. Then this row is traversed in searching of the literals. When the literal 'A' is found in the row '0' and column '1', as shown in Fig. 2.b, the index of this column is inserted into the list, if this index has not yet been inserted. Then, the algorithm immediately switches to the line '1' indicated by the column of the element found. Each row changing is a recursive call of the algorithm as Fig. 2.b demonstrates. This process is repeated recursively until the index '2' of the final vertex is reached, meaning that a path was formed. This procedure can be seen through the arrows in Fig. 2.b. Notice that literals in a column whose index has already been inserted in the list are ignored. Afterward, through the contents stored in the list it is possible to compose the path by indexing the matrix. All paths formed are stored in a list in order to compare with the original expression.

Once a path is formed, the last index '2', inserted into the list, is removed and the algorithm returns to the last cell of the matrix that was visited. Then, it continues traversing this row until finding another element, in this case the literal 'C', and it switches to the row '3' indicated by the column of this element.. If the row ends and any other element is found the algorithm returns to the last row visited and continues searching elements. After forming the two paths that start with literal 'A' the algorithm returns to the cell with row '0' and column '1', keeping in the list just the index '0'. Of course the algorithm travels this row finds the element 'B' in row '0' and column '3'. So, all procedure explained above is applied again as shown the Fig. 2.d and Fig. 2.e. The process ends when it returns to the initial row '0' and all cells this row were visited.

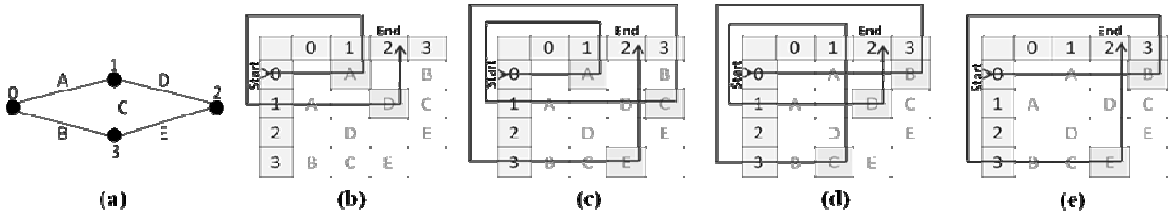


Fig. 2 – Matrices for explaining the algorithm able to generate all paths of a graph.

The next step consists in comparing each path with the products that compose the original expression. If a path that does not belong to SOP has been introduced, a routine checks if this path is sensitized or not sensitized. When thinking in a transistor network, a path cannot be sensitized if it contains both polarities, for example 'A' and '!A'. In order words, this path is not a valid path. If the new path introduced is not sensitized he is accepted, because it does not change the logical behavior of the circuit. Otherwise, the graph needs to be restored to step before the optimization that generated the new path, and this optimization is discarded. For this, a restore routine is invoked, this is responsible for recovering the edges and vertices that were eliminated from the graph and reconnect them.

To perform the recovery process it is necessary to store some information as the literal represented by removed edges and what vertices were merged. Finally, if necessary, with this information the graph can be recovered without compromising the functionality of the circuit which it represents. Notice that all original products of the Expression (1) are present in the graph of the Fig. 1.h. However, by sharing edges, some new paths were also introduced. All these paths are allowed because are paths that cannot be sensitized. Another interesting fact is that the proposed approach may derive Wheatstone bridge networks like methods proposed by [1] and [2]. The example illustrated in Fig. 1.h presents some bridge configuration. It is a benefit over optimization approaches based on factorization that can only derive series-parallel networks.

4. Experimental Results

Our algorithms were implemented in Java language. As output, the technique shows the optimized networks using the Prefuse graphics library [7] and generates a Spice netlist of the optimized circuit. To describe our edges sharing method we used the Exp. (1), referring to a 'XOR' with four inputs. The achieved network was compared to the result obtained by others techniques described in [5], as BDD, OpBDD, LBBDD, CSP and to the good-factor algorithm from SIS Software [8]. Our method reaches the same result like the BDD, OpBDD and LBBDD methods, with 12 transistors, overcoming the SIS with 16 transistors and CSP, NCSP with 22 transistors.

Finally, the set of 4 input p-class logic functions was used as benchmark to evaluate our proposed algorithm. This set is composed by 3982 logic functions. Each logic function was applied to SIS software as

well as to our solution. When running in SIS, the two available algorithms were used, the quick-factor and the good-factor. However, our proposed method was able to deliver better solutions, reducing the total number of switches in the networks as Tab. 1 illustrates. This is due to the ability of generating networks with Wheatstone bridge arrangements. Our approach was capable to reduce up to 4 transistors in some generated networks if compared to the SIS. On the other hand, the good-factor achieves some smaller transistor networks. On these 130 cases our algorithm was not able to generate bridge configuration, generating networks that are purely series-parallel arrangements.

Tab. 1 – Results for the set of 4 input p-class logic functions.

	Total transistor count
Our solution	35598
Good-factory	37723
Quick-factory	38341
Our solution compared to SIS	**# of logic functions**
decreasing transistor count	1644
exact same transistor count	2208
increasing transistor count	130

5. Conclusions and Future Works

This paper presented an edges sharing method to derive optimized transistor networks. The algorithm was implemented in Java language and a graph interface using Prefuse library is available. To describe our algorithm step by step, we use an 'XOR' with 4 inputs. The optimized network presents the same result of the methods in [5] for a 'XOR' with 4 inputs, surpassing the SIS solution. Nevertheless, when using the set of 4 input p-class logic functions, our solution is able to perform a considerable reduction of the total transistor count. Moreover, it is capable to deliver 1644 networks with less transistor count, if comparing to the good-factor solution, reduce up to 4 transistors in some networks. The optimized transistor networks generate by our approach are validated ensuring the logical behavioral of the network. As future work we intend to evaluate the complexity of the algorithm. Also, we intend to compare the proposed solution with the method described in [4].

6. References

[1] J. Zhu et al. On the Optimization of MOS Circuits. IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications. (1993), 412-422.

[2] D. Kagaris et al. A Methodology for Transistor-Efficient Supergate Design. IEEE Transactions On Very Large Scale Integration (VLSI) Systems. (2007), 488-492.

[3] Brayton, R. K. Factoring logic functions. IBM J. Res. Dev. 31, 2 (1987), 187-198.

[4] Mintz, A. and Golumbic, M. C. Factoring boolean functions using graph partitioning. Discrete Appl. Math. 149, 1-3 (2005), 131-153.

[5] Da Rosa Jr, L. S. Automatic Generation and Evaluation of Transistor Networks in Different Logic Styles. PhD Thesis PGMicro/UFRGS, Porto Alegre, Brazil. (2008), 147 p.

[6] Da Rosa Jr., L. S.; Marques, F. S.; Schneider, F.; Ribas, R.P.; Reis, A. I. A Comparative Study of CMOS Gates with Minimum Transistor Stacks. Symposium on Integrated Circuits and Systems Design. (2007), 93–98

[7] Prefuse.org. The Prefuse Visualization Toolkit. [Online] Avaliable: http://prefuse.org/ [Acessed: Mar. 25, 2010].

[8] Sentovich, E.; Singh, K., Lavagno; L., Moon; C., Murgai, R.; Saldanha, A., Savoj; H., Stephan, P.; Brayton, R.; and Sangiovanni-Vincentelli, A. SIS: A system for sequential circuit synthesis. Tech. Rep. UCB/ERL M92/41. UC Berkeley, Berkeley. (1992).

Sroute: A Router Tool for Structured ASICs

[1]Érico de Morais Nunes, [1]Reginaldo da Nóbrega Tavares
{nunes.erico,regi.ntavares}@gmail.com

[1]Universidade Federal do Pampa – UNIPAMPA
Bagé - Brazil

Abstract

This paper presents a routing tool and its integration on a synthesis flow for automatic generation of structured ASICs. The most important contribution of this tool is that it can be used to generate interconnections with less vias. The router implementation is described and some results from the physical synthesis are shown.

1. Introduction

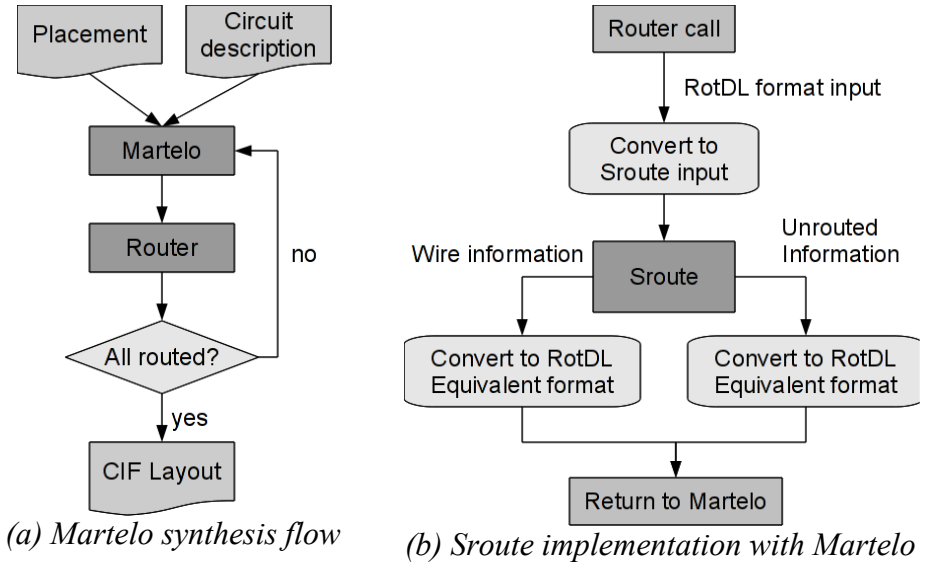
The design of integrated circuits (IC) is becoming increasingly complex. The design complexity has increased due to the continuous scale reduction of the circuit components. Although IC components scale reduction brings the advantage of allowing more functionality in a single chip, it usually demands higher design cost to achieve higher performance [1].

Design complexity has progressively increased because of higher impact of process variability and parasitic electric effects. In deep submicron design, fabrication is more sensible to process variability that causes defects or circuit malfunction after fabrication [2].

Several techniques have been employed to increase the circuit performance when the circuit design is exploring new IC process technologies. Such techniques are called design for manufacturability (DFM). One DFM techinique is based on structured or regular layouts called structured ASICs. Structured ASICs uses few logic cells placed in a regular and structured form to improve layout predictability. This strategy may help to improve the fabrication process because the same cell layouts are employed to construct the circuit.

In this regular scenario the wires would be implemented with more attention because the interconnection implementation is more unpredictable. A number of factors contributes to affect the wire layout scenario. For example, the cell placement can generate long wires or even congested areas and it may create a disordered layout scenario. Therefore, routing also is an important design task to DFM.

This paper presents a new router called Sroute. Sroute is a grid router which uses A* algorithm plus some improvements. This new router is able to reduce the number of vias. Reduction on number of vias may help to improve the layout scenario complexity. To check this capacity, Sroute was inserted in a regular layout design flow called Martelo [3][4]. Martelo is a physical design tool that provides an automated synthesis flow able to generate structured and regular layouts. The layout regularity is achieved because Martelo uses only 2-input nand gates. The 2-input nand gates are placed in a regular matrix. Fig. 1(a) shows the Martelo design flow.



(a) Martelo synthesis flow *(b) Sroute implementation with Martelo*

Fig. 1 – Synthesis flow overview

Martelo synthesis originally uses the RotDL [5] router that implements all interconnections of the circuit. RotDL is a maze router based on A* path search algorithm. Because of Martelo matrices rely on full over the cell (FOTC) routing, the routing step is an important work on the Martelo flow.

This paper is organized as follows: section 2 describes the Martelo flow synthesis and interaction with Sroute. Section 3 is about the Sroute implementation and features. Section 4 describes a set of experiments performed to compare Sroute and RotDL in terms of the total wirelength and the number of vias. Then section 5 is about some conclusions.

2. Integration with Martelo

Martelo [3][4] is a layout generator based on regular matrices of 2-input nand cells. Only 2-input nands are allowed in this matrix. Martelo uses the synthesis flow described in Fig. 1(a). The synthesis receives two inputs through files: the netlist and the placement. In order to enable transparent exchange of data between Martelo and Sroute, it was necessary to implement programs that convert the input and output descriptions. Martelo calls the router with a RotDL input file format as argument. If the enabled router is Sroute, this input must be converted to a Sroute format. The router result is also expected to be in RotDL format. So, it was also necessary to implement programs that convert Sroute to RotDL format. This flow is presented in Fig. 1(b). The few converters were implemented using Python script language and they are inserted in the automatic process.

3. Sroute

Sroute is a grid router tool based on a maze router approach.

Sroute uses an A* heuristic for path search in the grid. Sroute makes use of some features that improve the quality of the results. For example, Sroute tries to reduce the number of corners and vias which are inserted. The Sroute input is a file describing the grid and the nets. Nets are described by only an ID string and a list with 3 coordinate points. The 3 coordinate points describe respectively the x, y and z coordinate of a pin. The axis consider x grows to the right, y grows to up and z is the layer. After reading the input file, Sroute places the pins at their grid positions.

Sroute makes a breadth-first search (BFS) using the A* heuristic. A* is a well established heuristic used to reduce the number of grid positions visited before finding the target on a BFS. This heuristic makes use of a cost in each grid position to determine if using that position is a good choice towards a good result. For routing, the higher the cost of a position, the worse it is to include that point in the wire path. The cost of a position for a routing instance on Sroute is basically the number of expansion steps necessary to reach that position plus the rectilinear distance between a source position and the target. A* makes finding a path significantly faster than using the standard maze routing. This happens because positions in the direction opposite to the target will have a higher cost and are quickly discarded. As a result, A* wastes much less time expanding non used positions.

This routing style only solves routing between two pins (a source and a target), but nets often have more than two pins. Nets with more than two pins are called multi pin or multi terminal nets. Multi pin nets must first be decomposed into a set of two pin connections. By solving all the two pin connections of a net, it is guaranteed that the net is correctly routed.

However, routing the two pin nets blindly does not usually yield good results. Fig. 2(a) shows what can happen if that approach is taken. Fig. 2(a) and Fig. 2(b) are routing results from Sroute.The circular geometries are only demonstrative for the positions of the pins involved.

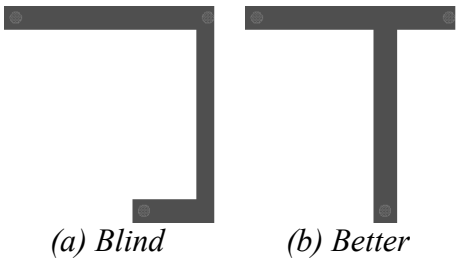


Fig. 2 – Routing Multi pin nets

The case in Fig. 2(a) happens because each two pin connection is solved individually. In this case, each previously placed wire segment is treated as an obstacle, and a new two pin connection cannot cross it, even from the same net.

Sroute makes use of a strategy called net linking to try to improve this solution. When using net linking, the first two pin connection is routed normally. Then, the following two pin connections of the same net have two ways to be marked as solved:

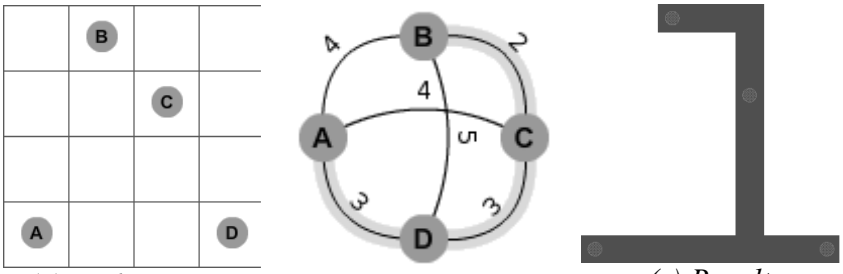
- reach the specified target pin position, or
- reach a position containing a wire segment from the same net.

To accomplish this, whenever a two pin connection is routed, all of its positions must be marked as belonging to the net. When another two pin connection of the same net is being routed, each visited position is checked for both solutions. By using this strategy, at most of the times it is possible to accomplish the routing result as in Fig. 2(b).

It is important to note that the way multi pin nets are decomposed in two pin connections and the order on which these two pin connections are routed is a key factor for this technique. Also, because of net linking, the same two pin connection can produce different results if the source and target pins are swapped. For example, considering a net with three pins: P1, P2 and P3. Considering P1 and P2 are routed first and a wire route is constructed and finished between them. When P3 is to be routed to be included on the net, net linking will only be able to produce improvements if P3 is the source and either P1 or P2 is the target. There is no room for improvement is P3 is the target. This happens because for an improvement to be achieved through net linking, P3 must not reach P1 or P2 but rather a position with a wire segment of the same net before actually reaching the target pin. If either P1 or P2 is the source, this means the position of P1 or P2 is already part of the connection solution and the wire must be routed until it reaches P3.

Sroute tries to maximize the advantage of this by using a decomposition algorithm which favors net linking. The decomposition algorithm, which is discussed below, is based on a Minimum Spanning Tree (MST) algorithm. A MST algorithm builds a spanning tree inside a graph on which the sum of edges is minimum [6]. A spanning tree is a tree which connects all the vertices of the graph.

First, a weighted complete graph is built on which each pin of the connection is a vertex. By being a complete graph, it means that for each vertex, there is an edge connecting to each other vertex of the graph. The weight of the edges of the graph are the rectilinear distance between the positions of the pins. Then, Prim's MST algorithm is run on this graph. Prim's algorithm works by maintaining a set of visited vertices (V). A vertex in this set already belongs to the tree. This set is initially empty and a first vertice must be provided or selected. At each iteration, the algorithm picks the edge with the smallest cost containing at least one vertex in the V. If this edge includes a new vertex to V, the edge is included in the solution and the vertex is added to V. When all vertices belong to the V, the algorithm finishes. In Sroute, the initial vertex is selected as one of the vetices containing the lowest weighted edge of the graph.



(a) Multi pin net *(b) Graph and MST* *(c) Result*

Fig. 3 – Routing Multi pin nets

Sroute runs the decomposition algorithm for each net and, at each edge selection, it reports a new two pin connection. In order to favor net linking, a new two pin connection swaps the order of which vertices were added to the set. That is, the connection always has the last visited vertex as the source, and the vertex which was sent first to the visited vertex set as the target. As connections are routed in the same order of which edges are selected by this process, a new vertex is always the source and this helps avoiding problems as in Fig. 2(a).

An example multi pin net, containing pins A, B, C and D, is shown in Fig. 3(a). The graph built based on it is shown in Fig. 3(b). A MST is built on the graph, consisting of the highlighed edges. Considering the initial vertex for the MST algorithm was taken as B, the output two pin connections and order would be (considering the notation source→target) C→B, D→C and then A→D. Fig 3(c) shows the result after routing.

4. Experiment

An experiment with different circuits was done in order to compare the performance between Martelo-RotDL and Martelo-Sroute combinations. Simple circuits were used in this set of experiment. CLA1 and CLA2 are 4-bits carry lookahead adders. The RPC is a ripple carry adder of 6 bits. The last circuit is able to compare numbers of 4 bits.

The following tables show the wirelength (WL) and the number of vias for some logic circuits generated with Martelo-RotDL and Martelo-Sroute. The same input set of constraints were used to start the synthesis with both routers. The number of vias was measured considering the DFM purpose of the Martelo flow. The ratios in the tables express the of result of Sroute in comparison with the result of RotDL.

Tab. 1 – Wirelength and Vias comparison

Circuit	Layers	RotDL WL	Sroute WL	WL Ratio	RotDL Vias	Sroute Vias	Vias Ratio
CLA 1	3	325060	299170	0.92035	1176	832	0.70748
	4	273650	259810	0.94942	1206	813	0.67412
CLA 2	3	520700	456660	0.87701	1932	1217	0.62991
	4	437630	417850	0.95480	1839	1203	0.65415
RPC Adder	3	62120	55240	0.88924	417	199	0.4772
	4	62120	55240	0.88924	417	199	0.4772
Comparator	3	806050	578650	0.71788	2910	1665	0.57216
	4	617500	578200	0.93635	2652	1676	0.63197
Average				0.89179			0.60303

The total wirelength was reduced by about 10% as it can be seen in Tab. 1. The improvement is mostly noticeable when routing with fewer metal layers.

Tab. 1 also shows that the number of vias has reduced considerably. It happens because RotDL assigns a metal layer to implement wires in a specific direction. In this case, there is a via insertion when the wire changes direction. This reduction is very important because vias can be a challenge to be implemented in the circuit. Also, vias increase the electrical resistance and capacitance, so reducing the number of vias may reduce electrical demands.

Preferred direction per layer is usually done to increase routability [7], however, in this case, routability was not hurt by this. Another reason for using preferred directions is performance. Indeed, RotDL runs faster than Sroute. However, both routers are run in an acceptable ammount of time.

5. Conclusion

Martelo, a regular layout generator proposed by [3], has been extended to work with a new router called Sroute. This automatic flow has been shown functional when Sroute is active. Sroute still has potential for improvement but it can already be used to generate layouts.

Although small circuits were used in the set of experiments, the Sroute results are positive. For example, the reduction of the number of vias is very important. It is important because the layout complexity may be reduced. The total wirelength, one of the most important metrics for physical routing, also has shown a reasonable improvement with Sroute.

6. References

[1] H. Chen and Y. Chang, “Routing for manufacturability and reliability,” IEEE Circuits and Systems Magazine, 2009. pp.20-31.

[2] P. Gupta and A. Kahng, “Manufacturing-aware physical design,” 2003 International Conference on Computer Aided Design, 2003, pp. 681-687.

[3] C. Menezes, C. Meinhardt, R.Reis and R. Tavares “Design of regular layouts to improve predictability,” Proceedings of the 6th International Caribbean Conference on Devices, Circuits and Systems, pp. 67-72, 2006.

[4] C. Menezes, “Geração automática de leiaute através de matriz de células – MARTELO,” Masters's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Jan. 2004.

[5] G. Flach, R. Hentschke, and R. Reis, “Algorithms to improvement of RotDL router,” XIX South Symposium on Microelectronics, pp. 100-107, 2004.

[6] T. H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms. MIT Press, 2001.

[7] N. Sherwani, Algorithms for VLSI Physical Design Automation, Kluwer Academic Publishers, 1995.

An Algorithm for Generating Logical Expressions Using a Graph-based Approach

Julio S. Domingues Jr., Renato S. de Souza, Vinicius N. Possani, Felipe S. Marques, Leomar S. da Rosa Jr.
{jsdomingues, rsdsouza, vnpossani, felipem, leomarjr}@inf.ufpel.edu.br

Grupo de Arquitetura e Circuitos Integrados – GACI
Universidade Federal de Pelotas – UFPel

Abstract

This paper presents a method to extract Boolean expressions from graphs that represent transistor networks. It uses graph compaction techniques to identify whether a transistor network has only series-parallel associations or also has bridge connections. The algorithm was implemented in Java language and integrated to the Soptimizer tool, which is an academic tool to apply optimizations in Boolean functions based on graphs. It has been validated using the set of 4-input functions of the P-class library set. Concerning series-parallel networks, all achieved results are compatible to the SIS tool.

1. Introduction

Nowadays, most of the microelectronics market is mainly composed of Very Large Scale Integration (VLSI) circuits. These electronic devices can have billions of transistors in a single die. Due to the complexity of these circuits, Electronic Design Automation (EDA) tools play an important role in the design time. It reduces the time-to-market. Some of these tools work at the logic synthesis level, minimizing logic expressions through factorization procedures [1][2] in order to obtain equations with fewer literals.

In a previous work, we have proposed a method to generated transistor networks from sum-of-products. This method uses edge sharing techniques to reduce the number of transistors need to implement a given Boolean function. The implementation of this set of method resulted in the *Soptimizer* tool. The *Soptimizer* only works at the graph level, and it is not able to generate a Boolean expression that symbolizes the Boolean function represented by a graph. In order to close this gap, this paper presents an algorithm able to extract expressions from graphs that represent Boolean functions. The method proposed by *Soptimizer* proven to be a good solution, because the optimizations applied could to generate networks with Wheatstone Bridged configuration, considered the most optimal configuration of the circuit for some logic functions.

The remaining of this paper is organized as follows. First, some basic concepts are reviewed in section 2. In section 3, we describe the proposed algorithm. Some results are shown in section 4. Finally, conclusions are presented in section 5.

2. Background

For better understanding of this paper, some related concepts are reviewed on this section. **Sum-of-products** (SOPs) are canonical forms to represent Boolean functions through expressions. These expressions are not optimal concerning the number of literals. **Factorization** [1][2] is an optimization technique that can be applied to Boolean expressions (such as SOPs) aiming the minimization of some criteria. Usually, it aims the reduction on the number of literals in a given expression. There are algebraic and Boolean methods for factorization. Due to the algorithm complexity, algebraic are faster than Boolean methods. However, Boolean factorization can achieve better results.

Boolean functions can be represented in different ways. Graphs can be used to represent them. A **graph** is an ordered pair $G = (V, E)$ comprising a set V of **vertices** or **nodes** together with a set E of **edges** or **lines**, which are 2-element subsets of V. A **directed graph** is an ordered pair $D = (V, A)$ with V being a set vertices, and A being a set of ordered pairs of vertices, called **arcs**, **directed edges**, or **arrows**. An arc $a = (x, y)$ is considered to be directed from x to y; y is called the head and x is called the tail of the arc; y is said to be a **direct successor** of x, and x is said to be a **direct predecessor** of y. If a path leads from x to y, then y is said to be a **successor** of x and **reachable** from x, and x is said to be a **predecessor** of y.

Different kinds of data structures can be used to store graphs in computer systems. The **adjacency matrix** is one of them. This is an n by n matrix A, where n is the number of vertices in the graph. If there is an edge from a vertex x to a vertex y, then the element $a_{x,y}$ is 1 (or in general the number of xy edges), otherwise it is 0. In computing, this matrix makes it easy to find subgraphs, and to reverse a directed graph.

Besides Boolean function representation, graphs can also be used to represent transistor networks [3]. In this case, each edge represents a given transistor and the vertices are points of connections among transistors. Usually, CMOS transistor networks are constructed through series-parallel associations. In this case, each literal

of a Boolean expression becomes an edge (transistor) in the graph representation. Therefore, the fewer literals in an expression the fewer transistors are needed to implement a Boolean function.

Prefuse is a Java graphics library where you can create graphical interfaces to provide networking capabilities. SIS it is a tool for factorization of logic expressions used as reference. PClass is a library cells with known functions that were used as a method to benchmark the proposed algorithm.

3. The Proposed Algorithm

This section describes an algorithm to extract Boolean expressions from graphs that represent transistor networks. The proposed method is integrated with the Soptimizer tool. This tool factorizes SOPs (represented by graphs) using algebraic techniques applied to graphs. It reduces the number of edges required to represent a given Boolean function. As result, the Soptimizer tool presents a minimized graph.

In order to allow integration with other tools and other validation procedures, a Boolean expression is desired. The generation of a Boolean expression requires successive traversals of a graph from a terminal node to another. On each traversal, every edge associated in series to an adjacent edge is "compacted" to a single edge. If there is no more series edges to be compacted, then the algorithm start to traverse the graph aiming the compaction of edges associated in parallel. When there are no more edges in parallel to be compacted, the algorithm is started again looking for series compaction. This iterative process stops when there is no more possibility of series and parallel compaction.

The graph compression steps are demonstrated in Fig. 1. The initial graph is shown in Fig. 1.a. The process begins by searching adjacent edges connected in series in sub-graphs. In this example, the first sets of edges to be compacted are connected to the terminal node *T0*. For instance, the edges *!F*, *!E* and *!C* are associated in series, and can be compacted to a single edge. Therefore, the sub-graph composed by these edges is replaced by a new edge that actually is a list of edges. In this case, this new edge is referred as the product "*!F.!E.!C*". In a similar way, the edges "*F.E.C*" and "*D.B.A*" are created. This process is depicted in Fig. 1.b.

After performing the first iteration of series compaction, the algorithm looks for parallel associations. In Fig. 1.b, there are two sub-graphs in parallel. As shown in Fig. 1.c, the edges "*!F.!E.!C*" and "*F.E.C*" can be compacted in a single edge that is referred as "*!F.!E.!C + F.E.C*". At this point, there are no more edges in parallel. The algorithm is restarted in order to look for edges associated in series. Hence, the graph in Fig. 1.c can reduced to the one in Fig. 1.d. The final graph has a single edge that represents the Boolean function expressed by "*(!F.!E.!C + F.E.C).D.A.B*".

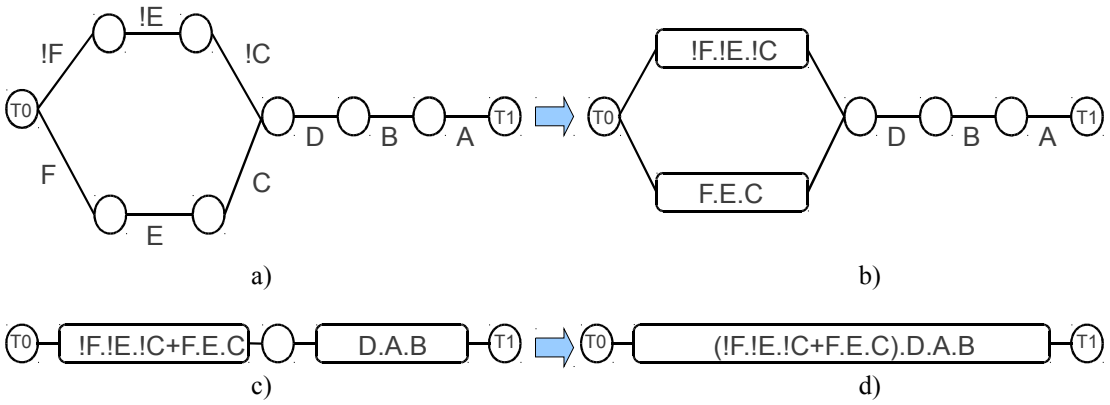


Fig. 1 – Graph compaction example.

Using graph compaction it is possible to identify bridge connections. Accordingly to [3], the minimal number of transistors to implement a Boolean function cannot be achieved by purely series-parallel networks. As an alternative, "*bridge*" transistor networks can be used. In this case, bridge connections are compacted through the compaction algorithm. Therefore, the final graph will have at least five edges on its construction (the minimal construction of a bridge network).

The main objective of our method is to extract a Boolean expression from a graph. When a graph can be compacted into a single edge (meaning that there are only series-parallel connections), the Boolean expression is naturally achieved. This is not the case of bridge networks that requires a more sophisticated algorithm to construct the expression. An example of bridge network compaction can be seen in Fig. 2.

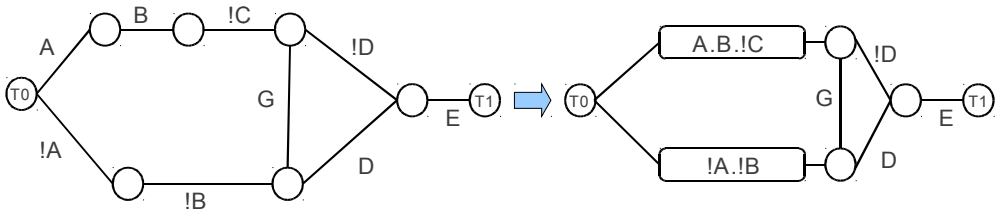


Fig. 2 – Example of bridge network compaction.

Considering the graph compaction of a bridge network, it is necessary to roam it to in order to generate a Boolean expression that represents the whole graph. This final procedure is done using an adjacency matrix to represent the compacted graph. The expression can be extracted applying a depth-first-search algorithm on the adjacency matrix. The search starts on a terminal node and ends on another. Fig. 3 shows the searches on the adjacency matrix that represents the compacted graph of Fig. 2. The paths followed during the iterations are presented in the matrix of the Fig. 3 and differentiated by the colors black and gray. The depth-first-search starts on the terminal node "T0", i.e., on the index zero of the adjacency matrix. This way, two expressions starts to be created from this point, considering the products "*A.B.!C*" and "*!A.!B*". The information of each visited edge is concatenated to the expressions resultant from the previous visited edges. For this example, the expression resultant of this process is "*A.B.!C.(!D.E+G.D.E) + A.!B.(G.!D.E+D.E)*".

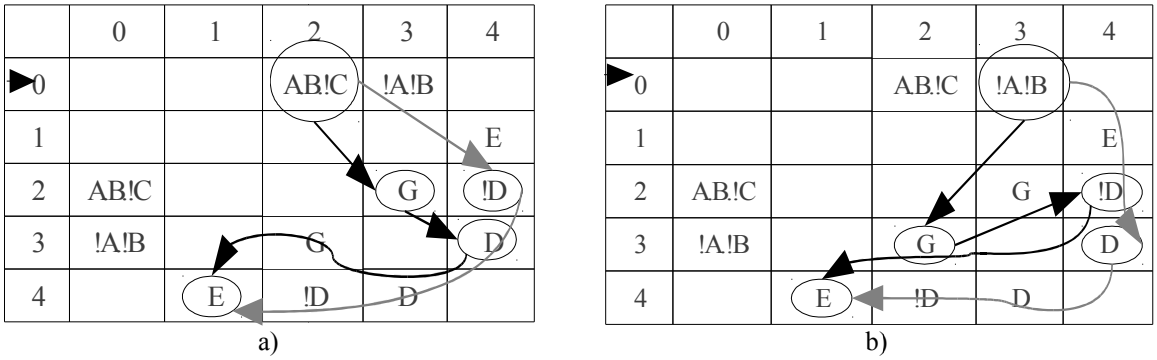


Fig. 3 – Adjacency matrix representation.

The graphics library Prefuse [4] was used as a form of better understanding and viewing of the final graph. Fig. 4 shows the graphical output of the tool Soptimizer.

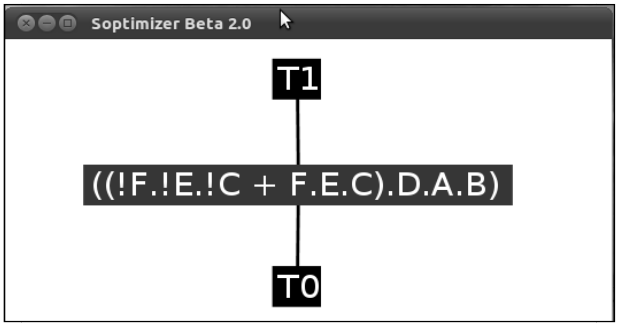


Fig. 4 – Prefuse of the Soptimizer.

4. Experimental Results

Since the Soptimizer tools is implemented in Java language, the proposed algorithm was implemented through a set of Java classes that are already integrated into the tool and the Prefuse [4] tool kit, which is used to visualize all generated graphs.

In order to validate the expressions generated by our algorithm, we have run a set experiments on a computer with a Core2Duo processor and 4GB of RAM, and a 64-bit version of the Ubuntu 10.10 operating system. The first experiment validated 10 functions randomly chosen from the complete set of the 7-input pClass library. The second experiment has validated all 3.982 functions of the 4-input pClass logic functions. All expressions were optimized through the Soptimizer tool and the resultant expressions were generated through the proposed algorithm in approximately 8 minutes. All expressions generated by our algorithm were compared to the input expressions in order to validate the algorithm.

Tab. 1 presents a comparison of the SIS [5] factorization procedure and our method associated to the Soptimizer tool, considering a sub set of 15 functions of the 7-input *pClass* library. The second column shows the number of literals of each SOP that represents each function. The third column shows the number of literals of the factored forms generated by the SIS tool. The last column presents the number literals achieved by our tool. Since these functions have their minimal representation using series-parallel networks, both SIS and Soptimizer give the same result. SIS does not handle bridge networks. This is the advantage of our method.

Tab.1 – Literals count of factored forms

Function	# literals in the SOP	# literals from SIS	#literals from Soptimizer
3	8	6	6
16	6	5	5
18	5	4	4
120	16	11	11
468	12	9	9
469	10	8	8
456	10	8	8
502	13	11	11
508	12	9	9
520	7	6	6
521	12	8	8
525	12	9	9
3567	15	10	10
3569	19	13	13
3809	8	7	7

5. Conclusions and Future Works

This paper presented a solution for the generation of Boolean expressions from graphs. This method fills a gap between Soptimizer and other tools that uses expression as input. Our algorithm is able to generate expression for series-parallel and bridge transistor networks represented by graphs in the Soptimizer tool.

Boolean expressions do not have an operator to denote bridge connections. This way, when expressed using the primitive operators, a lot of redundancies are added into the expression. Our algorithm is able to handle this problem, leading to optimal solutions.

As future work we intent to eliminate non-sensitized paths of the transistor network. This characteristic will lead to the extraction of smaller expressions in terms of literals.

6. References

[1] BRAYTON, R. K. Factoring logic functions. IBM J. Res. Dev. 31, 2 (1987), 187- 198.

[2] MINTZ, A. and Golumbic, M. C. Factoring boolean functions using graph partitioning. Discrete Appl. Math. 149, 1-3 (2005), 131-153.

[3] ZHU, J. et al. On the Optimization of MOS Circuits. IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications. (1993), 412-422.

[4] PREFUSE.ORG. The Prefuse Visualization Toolkit. [Online] Avaliable: http://prefuse.org/ [Acessed: Mar. 25, 2010].

[5] SENTOVICH, E. et al. SIS: A system for sequential circuit synthesis. Tech.Rep. UCB/ERL M92/41. UC Berkeley, Berkeley. (1992).

Video Coding 2

A Media Processing Implementation for ISDTV Middleware with Optional Hardware Acceleration Support

Jean F. G. Quadro, Tiago H. Trojahn, Juliano L. Gonçalves, Luciano V. Agostini, Leomar S. da Rosa Junior
{jeanquadro, thtrojahn, juliano, agostini, leomarjr}@inf.ufpel.edu.br

Group of Architectures and Integrated Circuits – GACI
Technology Development Center - CDTec
Federal University of Pelotas – UFPel
Pelotas – Brazil

Abstract

The media decoding is one of the fundamental components required in a Digital TV standard, mainly because High Definition uncompressed videos cannot be transmitted due to bandwidth constraints. Currently, one of the most advanced Digital TV standards is the Brazilian International Standard for Digital Television (ISDTV), featuring a return channel, high-definition content and a middleware named Ginga. This paper describes a media decoder, with optional hardware acceleration support, compatible with Ginga. Results present a performance analyze of the Media Processing with and without hardware acceleration support when reproducing video stream with different resolutions.

1. Introduction

The video stream, normally broadcasted through the air or using a cable, in the Digital TV standards is a coded version of the original video created directly by the producers. The coding process is used to reduce the video stream size: a single frame using 3-bytes for coloring in the 1920x1080 resolution uses about 5.93 Mega Bytes (MB) of size. In a video with 30 frames per second (FPS), it is required a constant transmission rate of about 178 MB per second, much more than an average internet connection or even a local network can provide nowadays.

In addition, some of the Digital TV standards provide support to multiple audio and subtitle streams and, in some cases, even applications to be used in a single stream, know as Transport Stream (TS), requiring even more bandwidth of the transmission channel. For this reason, coding standards are required to provide high-definition and high-quality content to TV viewers at reasonable costs.

After coded, the data needs to be decoded to be effectively used. In terms of Digital TV, the decoded video and audio stream can be showed right to the viewer or manipulated by other application, increasing the complexity of the decoding procedure.

This work presents a video decoding application, named Media Processing, able to handle both audio, video and subtitle streams using an open-source library. The Media Processing was developed to be compatible with the Ginga, the International System for Digital Television (ISDTV) middleware and its audio, video and subtitle specifications.

The ISDTV is an emergent Digital TV standard developed in Brazil which presents several peculiar aspects that innovates and improves the quality of Digital TV services. For example, the Brazilian Digital TV system adopts the H.264/AVC [1] as video coding standard, differently from the established American, European and Japanese Digital TV standards. Also, the Ginga middleware provides support to interactivity through a return channel, delivering more possibilities to Digital TV applications like bank accounts access, shopping, and mass advertising.

This work is organized as follow: section 2 presents the Media Processing and its main feature. Section 3 presents the performance analysis of the Media Processing. Finally, section 4 presents conclusions and future works.

2. The Media Processing

The libVLC [2] library at version 1.1.5 was used to implements the Media Processing module. The Media Processing was developed to be able to provide all the basic features expected from a video decoder, including the methods:

- Play – The most used and the very fundamental feature of all media decoders. It is responsible to start all the decoding operation and provide the decoded stream to the viewer or to another application.
- Pause – Responsible to pause the operation without deallocating the resources to continue the decoding.
- Stop – The finalization method, used to deallocate all resources and shutdown the decoding process.

However, the ISDTV standard requires more features of the Media Processing than these basic methods. For example, the Media Processing must provide data to other modules and, sometimes, directly to applications

installed by the viewers. This characteristic makes the Media Processing much more complex and consequently prone to eventual failures. Thus, the Media Processing implemented in this paper presents a wide exception control in all methods provided to others modules or applications in order to offer a smooth an easy utilization by applications programmers. Exception like malformed inputs streams and mistakes in calling methods are trivially controlled. Some more complex exception handling, like race conditions and allocation problems are performed directly by libVLC.

Besides the main methods of reproduction, auxiliary methods were implemented to applications and viewer usage. These methods are divided in two large groups, getter and setter methods. The details of each group are explained below:

- Getters methods: A large number of getters of various video, audio and subtitle information like video resolution and audio volume. These methods are important to other modules or applications that may require details of the media being reproduced.
- Setter methods: A set of setters methods that can change the reproduction behavior, like video brightness, audio volume and subtitle stream. These methods are fundamental to viewers to control the reproduction, reducing the volume for example. Applications can also use these methods.

A part from that, the Media Processing presents a large set of implemented features, listed below:

- Video Reproduction: The Media Processing support a wide range of video coding standards, including the H.264/AVC used in the ISDTV.
- Audio Reproduction: In terms of audio, the Media Processing provides full support to the mp3 and the AAC audio coded streams.
- Subtitles: The SubRip (SRT) and the Advanced Substation Alpha (ASS) are supported. Multiple subtitle streams in the same media stream are supported too.
- Demux: The Media Processing has a built-in Demux which can recognize the ISDTV Transport Stream and other input containers like Audio Video Interlace (AVI) and the Matroska Video (MKV).
- Screenshot: The Media Processing provides methods permitting a viewer to take screenshot and save it in a lossless JPEG format [3].
- Optimizations: The Media Processing can use all the optimizations present on the libVLC library like SSE and SSE2. A video card accelerated decoding can be used when the computer have a compatible video card [4], aiming to reduce the decoding stress to processor.
- Input: The input stream can be a file containing the streams which needs to be demuxed, a basic media stream already demuxed by another application or module, or an interface to an input device. A remotely located media can be accessed through a connection using the HTTP, FTP or RTP protocols.
- Output: The current Media Processing implementation features a built-in X.Org and a FrameBuffer video output. ALSA and DirectX WaveOut are the audio outputs currently implemented.

Fig. 1 shows the complete procedure to reproduce an input stream in a Digital TV device.

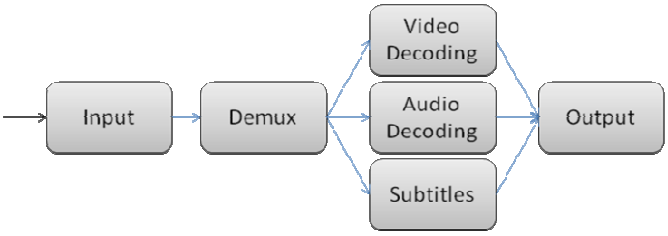


Fig. 1- Basic audio, video and subtitles decoding procedure.

The process described in Fig. 1 presents the Input, where the stream is recognized as valid input. The validated input goes to Demux, where the input stream is demuxed into audio, video and subtitles streams. Each elemental stream goes to the corresponding decoder and, after the decoding process, the output shows the resulting video and subtitle on-screen and the audio on TV speakers, for example.

An alternative Media Processing for the ISDTV is presented in [5], but it does not present audio decoding, video card hardware acceleration and some other features.

3. Performance Analysis

The Media Processing implementation was evaluated to verify the performance when reproducing input video and audio streams during nine minutes. This experiment was repeated three times in order to deliver more accurate results. The performance was measured in terms of processor usage and memory cost. The samples were generated with the procps application in an Ubuntu 10.04 operating system with recommended installation settings.

As benchmarks, the videos, were used the Big Buck Bunny [6], the Elephants Dream [7] and the Sintel [8] animations for input streams in three different resolutions: 848x480 (480p), 1280x720 (720p) and 1920x1080

(1080p). The progressive scan was utilized, dispensing the default libVLC De-Interlace filter. Video streams were coded using the x264 [9] in version 1649, and audio streams were coded with the Nero AAC Codec [10] at version 1.5.4.0.

All videos were coded using 10.000 Kbps (10 Mbps) for Average Bitrate (ABR) in High Profile at 3.2 level. It was used three reference frames, CABAC and Trellis activated and a 4:2:0 YCbCr sub sampling. The video frame rate was converted to 30 FPS to be ISDTV compatible.

The audio was coded in Constant Bitrate (CBR) mode with 192 Kbps using the Low Complexity (known as AAC-LC) in version 4. It was used 48 kHz for output frequency.

An Intel Core i5 760 with 2.8 GHz and 4 GB of RAM was the basic platform for the experiment. Three video cards configurations were used:

- The first configuration was named Computer A, without any dedicated video card and using only the software-based decoding of Media Processing.
- The second, named Computer B, using a XFX Geforce 9500GT with 512 MB of DDR2 RAM memory and the Media Processing hardware accelerated decoding activated.
- The last one, named Computer C, using a Zotac Geforce GTX 470 with 1280 MB of DDR5 RAM memory and the Media Processing hardware accelerated decoding activated.

3.1. Results

The processor usage, in percentage, and memory cost, in MegaBytes, results for the performance tests are presented in Fig. 2 and Fig. 3 respectively.

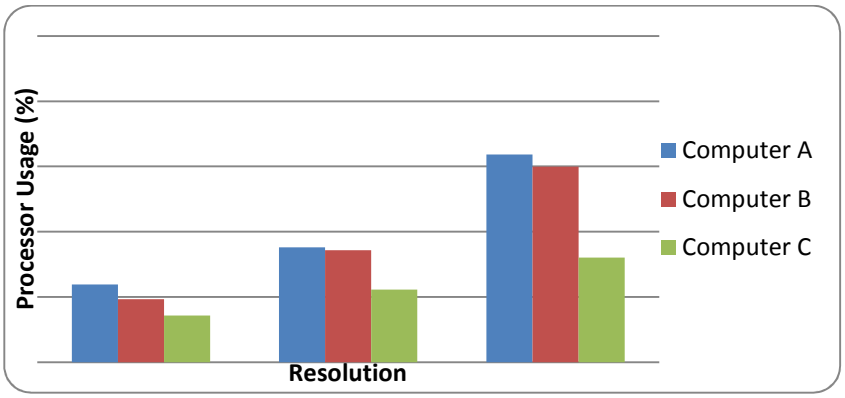


Fig. 2 - Processor usage (%) of Media Processing.

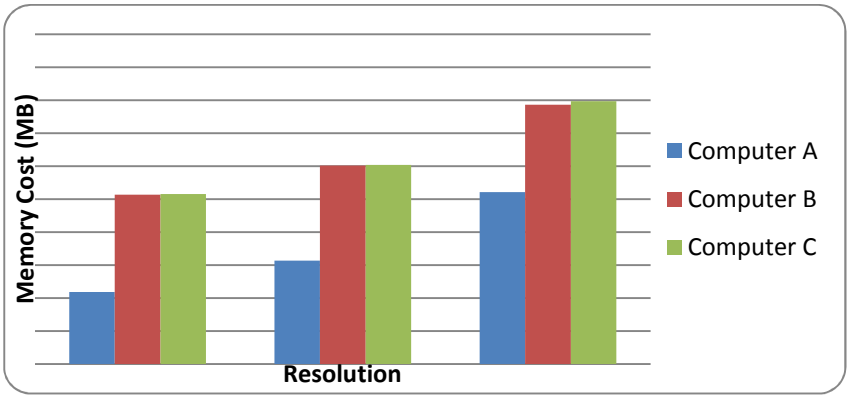


Fig. 3 - Memory cost (MB) of Media Processing.

The results show a processor usage increment when increasing the video resolution, which can be explained with the increase of the pixels which needs to be processed. The Computer A and Computer B presented an equivalent performance in terms of processor usage. However, the Computer C presents a better performance. The processing power of Geforce GTX 470 when compared to Geforce 9500GT suggests that the reduction of the main processor usage is directly related to the increase in video card processing power. Also, the equivalency between the Computer A and Computer B test results suggest that the Computer B video card is not able to reduce the load in the main processor in a satisfactory manner when decoding H.264/AVC videos.

In terms of memory, the implementations using the hardware acceleration in video card presented a larger memory cost. This behavior can be explained by the decoding methodology that libVLC adopts. Basically, the coded frame is sent to the video card and the decoded frame is kept in memory while not displayed. The

memory cost increases even more because several frames are kept in the memory to prevent synchronization problems. As with processor usage tests, the memory cost has risen as the increase in video resolution.

4. Conclusions and Future Works

This paper presented a decoding module, the Media Processing, for the Ginga ISDTV middleware. A performance test was performed to evaluate the component when using some of the available methods.

The ISDTV plays an important role in South America, where the standard was adopted in others countries like Argentina, Chile and Venezuela. Besides, Africa countries like South Africa and Democratic Republic of Congo are currently testing the ISDTV for possible use as their Digital TV standard. The Media Processing presented in this paper can be used in all these countries to achieve video, audio and subtitle decoding features.

Our Media Processing performance tests demonstrates that better results can be achieved in terms of processor usage when using a video card with high processing power with hardware acceleration activated. When using a video card with lower processing power, the advantage becomes negligible.

In terms of memory, the Media Processing hardware acceleration increases more than two times the memory needs when compared with the full software solution. The Computer B, without a significant better processor usage when compared with Computer A and with two times the memory cost, becomes the worst solution analyzed.

If the device presents a compatible video card and enough memory, the best solution consists in to activate the Media Processing video card hardware acceleration. On the other hand, if memory is limited, the full software decoding is better, but with a significant increase in processor usage.

Finally, it is important to mention that these results also point to the fact that an Application Specific Integrated Circuit (ASIC) dedicated to decode High Definition videos for the Brazilian Digital TV standard could be investigated. Powerful CPU, GPU and video cards are not feasible to be embedded in affordable Digital TV equipments.

For future works it is intended to do a wider performance tests using some other optimizations based in the processor architecture. It is also intended to use the Media Processing in a portable device to analyze its behavior when running streams with different video resolutions. Finally, another version of Media Processing is being implemented using the NVIDIA CUDA framework to achieve even better performance with a computer equipped with a dedicated video card.

5. References

[1] G. Sullivan, P. N. Topiwala and A. Luthra. “The H.264/AVC Advanced Video Coding standard: overview and introduction to the fidelity range extensions”, 2004. XXVII Conference on Applications of Digital Image Processing. p. 454-474.

[2] VideoLAN. “libVLC – VideoLAN Wiki”, 2010. Available: wiki.videolan.org/LibVLC

[3] M. Weinberger, G. Seroussi, and G. Sapiro, “The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS,” IEEE Trans. Image Processing, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[4] VideoLAN. “VLC Release 1.1.5”, 2010. Available: www.videolan.org/vlc/releases/1.1.5.html

[5] T. H. Trojahn, J. L. Gonçalves, J. C. B. Mattos, L. S Da Rosa Junior, L. V. Agostini. “A Media Processing Implementation Using Libvlc for the Ginga Middleware”. 5th FUTURETECH, pp.1-8, May 2010.

[6] Blender Institute. “Big Buck Bunny”, 2008. Available: www.bigbuckbunny.org

[7] Orange Open Movie Project. “Elephants Dream”, 2006. Available: www.elephantsdream.org

[8] Blender Institute. “Sintel”, 2010. Available: www.sintel.org

[9] X264. “VideoLAN - x264”, 2011. Available: www.videolan.org/developers/x264.html

[10] Nero AG. “Nero AAC Codec”, 2011. Available: www.nero.com/eng/technologies-aac-codec.html.

Random Search Motion Estimation Algorithm for High Definition Videos

[1] Cássio Cristani, [1] Pargles Dall'Oglio, [2]Diego Noble, [1,2] Marcelo Porto, [1] Luciano Agostini, [2] Sérgio Bampi.
{crcristani,pdalloglio,lvagostini,dvnoble}@inf.ufpel.edu.br, {msporto, bampi}@inf.ufrgs.br

[1]GACI, Federal University of Pelotas – UFPel
[2]Institute of Informatics, Federal University of Rio Grande do Sul - UFRGS

Abstract

This paper presents a new algorithm focusing in high quality fast motion estimation for high definition video coding. This algorithm provides more efficiency to avoid local minima falls in fast motion estimation due to random search exploration. The developed algorithm is called Random Search with Iterative Center Evaluation and Iterative Final Refinement (RSICIFR). It evaluates the central region of the search area, and also chooses, randomly, many other candidate blocks in the search area. This approach gives to the algorithm the possibility to avoid local minima falls, increasing the quality results, especially for high definition videos. The RSICIFR algorithm was implemented and evaluated to ten HD 1080p test video sequences. The evaluation results shows that a quality gain can be achieved in comparison of many fast algorithms.

1. Introduction

Motion Estimation (ME) presents the highest computational complexity among all steps of the current standards for video coding. In fact, ME represents 80% of the total computational complexity of current video coders [1]. The search for best vectors is known to be very expensive in terms of calculations and, consequentially, in terms of processing time. The Full Search (FS) [2] algorithm must explore all possibilities in a given search window, which implies in a very high computational cost, especially for high resolution videos. Based on this fact, it is important to explore new solutions which bring a good tradeoff between objective quality (PSNR) and computational complexity.

There are many fast algorithms and techniques in scientific literature which handle with this complexity at different levels of impact in objective quality (PSNR). These algorithms exploit the information redundancy to reduce the computational complexity. However, fast algorithms have a weak point which is the increase in local minima falls with the increase of the video resolutions, especially for high activity video sequences. This undesired characteristic leads the algorithm to choose not good motion vectors, consequentially, perceivable losses in the visual quality could be achieved in comparison with FS algorithm. In the H.264/AVC standard [3], the current most efficient video coding standard, there is no restriction about how the block matching is done in the motion estimation process, so there is a lot of space to explore new ideas. Hardware solutions are very important for real time ME in high definition videos applications. However, fast algorithms have not easy hardware implementation, some characteristics as data dependencies and not regular memory access can be very complicated to deal with. In other hand, FS algorithm requires lots of hardware resources to achieve high performance. In this context, a fast algorithm, that can be easily implemented in hardware, is very important for real time ME in high definition videos.

In this paper, we propose a new algorithm for fast motion estimation targeting high quality when processing high definition videos. This algorithm provides an efficient way to avoid local minima falls in the fast ME for high definition videos, due to the random search exploration. This algorithm is not focused on a specific standard and it can be used with all current standards. The developed algorithm is called Random Search with Iterative Center Evaluation and Iterative Final Refinement (RSICIFR). This paper presents the new algorithm, its results and comparisons with other well known motion estimation algorithms. The RSICIFR was applied to ten HD 1080p test video sequences and the results shows that an average gain of 1.49dBcan be achieved in comparison with the Diamond Search (DS) algorithm [4]. The computational cost is increased in comparison with the original DS, however it is more than 26.5 times lower when compared with FS algorithm for the same search area.

The rest of this paper is organized a follows: Section 2 describes the main concepts related to the ME process and also some concepts for common fast ME algorithms. The proposed algorithm is showed in Section 3 and the results at Section 4. The comparative results are presented in the Section 5. Conclusions are given at the section 6.

2. Motion Estimation

A digital video consists of a sequence of independent images, captured during the time. To obtain continuous motion (real time) a capture rate of 24 to 30 pictures (frames) per second is required [5]. Video compression is based on the elimination of data redundancies. Data is considered redundant if its value does not represent a new relevant information to the representation of the frame. There is three different types of redundancies exploited in video compression: spatial redundancy, temporal redundancy and entropy redundancy. The motion estimation handles with the redundancy between neighboring frames, called temporal redundancy. The frame is divided into blocks to be processed. These blocks are compared with blocks from previews, and already processed frames, called reference frames. The motion estimation process tries to identify and reduce the temporal redundancy between adjacent frames of a scene, mapping this information using motion vectors of two dimensions (x, y). The motion vector indicates the position where the most similar block (in comparison with current block) was found in the reference frame.

The search algorithms define how the search for the most similar block (best matching) must occur inside the search area in the reference frame. Full Search (FS) algorithm is the one that generates the optimum motion vector for a given search area. It evaluates all the candidate blocks in search area, so it can always find the most similar candidate block, however, its computational costs is huge. The fast algorithms (suboptimal) did not compare all possible candidate blocks, so the computational cost in relation to the FS is much lower. There are also quality losses, mainly due to the occurrences of local minima falls.

A local minima is a candidate block that results a small residue (in a given region), however, this residue does not represent the global minimum, which is smaller and can be located in another region. There are several fast algorithms in the literature that have already been consolidated. The comparisons in this work will be done with the Diamond Search (DS), Three Step Search (TSS)[6], Four Step Search (FSS)[4].The RSICIFR algorithm

3. The RSICIFR algorithm

The Random Search with Iterative Center Evaluation and Iterative Final Refinement (RSICIFR) do the search for the best result in four steps. In the first step, the central block of the search area and their neighbors left, right, top and bottom are calculated. The second step is an iterative search, made when the best result is not found at the central block. The third step is to randomly choose N candidate blocks inside the search area, this step can be done in parallel with steps one and two. The fourth step is an iterative search using the best candidate block from the step three as a new center. The final result is obtained due to the comparison with the best result from the central iterative search and the iterative search done in the best random candidate block.

Figure 1 shows a search example of the RSICIFR algorithm, the first step is illustrated with the number one. In this example, the best match is not found at the center, the right neighbors is chosen as the best candidate block, so the algorithm starts the iterative search, comparing three more candidate blocks (the neighbors), illustrated as number two in Figure 1. The search continues still the best result is found at the central block. In best case, the best result will be found at the center and only the five center blocks (number one in Figure 1) are calculated, and the iterative search will not be done. In the third stage N candidate blocks are randomly chosen inside the search area (number three in Figure 1). The N candidate blocks are evaluated and the best candidate block is found. The same iterative process of the step two is applied to this best candidate block , where its four neighbors is evaluated, and the iteration is applied when the best result is not found at the central candidate block. The blocks resulting from the stages two and four are compared, and the motion vector will be generated for the block that presents the best result for similarity.

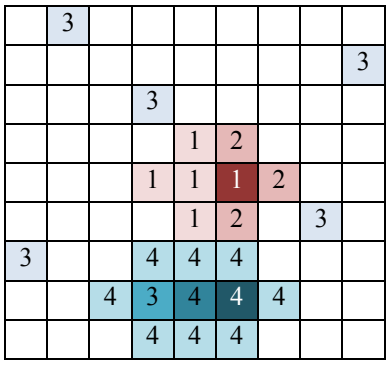


Fig.1 – The function RSICIFR with N=6

4. Software Evaluation Results

The Proposed algorithm was described in C language and it was evaluated with ten HD 1080p test video sequences [7]. The used search window for all evaluations is a range of [-16, +16] pixels, and the used distortion criteria is the Sum of Absolute Differences (SAD) [4]. The synthesis was made for 8x8, 16x16 and 32x32 block sizes. The block size 32x32 was not supported by any current coding standard, this is one of the new proposes for the HEVC, the new coding standard, still in development stage[8]. The number of randomly chosen blocks (N) was 16. The experiments consider the first two hundreds frames of all the ten HD 1080p sequences. The simulations do not explore the possibility of parallelism of the stages, since the time of simulation was not evaluated.

The Table 1 presents the average results (for the 10 test video sequences) of RSICIFR algorithm for different block sizes. The quality results are presented in PSNR gain and percentage of residue reduction (PRR). The PRR is the differences between the differential residue (without ME) and the residue achieved after the ME process. The computational complexity results are presented in the number of evaluated candidate blocks (ECB) and the number of used SAD operations. The evaluation was done with a small search area, it increases the probability of finding the global minimum. It also an interesting strategy for hardware implementation, since decreases the size of necessary internal memory.

Tab.1 - Evaluation results for different size block

Block Size	PSNR (dB)	PRR (%)	# ECB ($x10^6$)	SAD($x10^9$)
8x8	34.46	62.34	274.01	16.88
16x16	33.60	57.57	73.32	18.77
32x32	32.67	53.33	19.62	20.09

The results presented in the Table 1 shows that the growing in the block size implies directly in quality losses. For 8x8 blocks, the RSICIFR algorithm achieves 34.46dB in PSNR, and more than 62% in residue reduction. Using larger blocks, the PSNR gain is decreased, 0.86 dB for 16x6 blocks, and 0.79 dB for 32x32 blocks. The number of evaluated candidate blocks is decreased for higher block sizes, however, this decreasing is not proportional as the blocks growing. It indicates that for higher blocks the iterative steps are more intense, it causes a small increasing in the number of SAD operations for higher block sizes. Even using for a small search area, the quality results obtained by the RSICIFR algorithm are very interesting. It demonstrated that the algorithm can efficiently explores the search area. It also an interesting result for a hardware implementation of the RSICIFR algorithm, since a small internal memory will be necessary.

5. Comparative Results:

The RSICIFR was compared with the FS, DS, TSS and 4SS algorithms. In Table 2, the average results of PSNR, PRR and the number of SAD operations are presented. Ten HD 1080p test video sequences was used, each one has different content characteristics, so an average result can be achieved with this set of test video sequence. Thus, is possible to show that the developed algorithm has good results for any kind of video. The Figure 2 presents a graph with the quality results for all evaluated algorithms. The RSICIFR algorithm has the best result in comparison with all fast algorithms. The RSICIFR algorithm presents a PSNR gain of 1.02dB, 1.49dB and 2.57dB, in comparison with 4SS, DS and TSS algorithms, respectively. Comparing with FS algorithms, the losses in PSNR is only about 1.43dB. However, the reduction in the number of SAD operations is about 26 times.

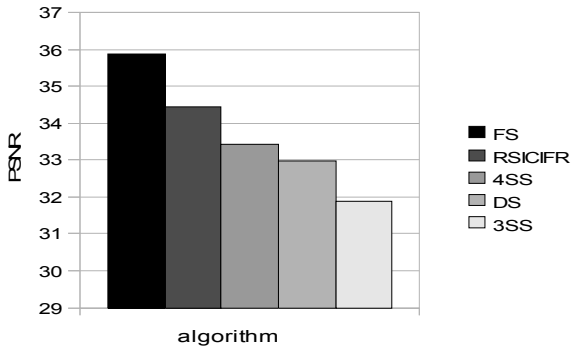


Fig.2 – PSNR comparison

The quality losses of the fast algorithms are generally caused by the local minima falls. The RSICIFR algorithm can reduce the local minima fall due to the randomly exploration of the search area. The quality

results, presented by the RSICIFR algorithm, shows that this approach is an efficient strategy to avoid local minima falls in high definition videos.

The RSICIFR algorithm presents the best tradeoff between quality and computational cost, since it can increase the quality in comparison with fast algorithms with an acceptable increasing in the number of SAD operations.

Tab.2 – Comparative results for RSICIFR, FS, DS, 3SS and 4SS.

Algorithms	PSNR (dB)	PRR(%)	SAD (x. 10^9)
FS	35,89	72,91	447,381
RSICIFR	34,46	62,34	16,897
4SS	33,44	65,60	14,922
DS	32,97	62,66	10,795
3SS	31,89	56,60	11,211

6. Conclusions and Future Works

This paper presented a new algorithm for motion estimation, named Random Search with Iterative Center Evaluation and Iterative Final Refinement (RSICIFR). This algorithm can reduce the local minima fall, increasing the quality results. The RSICIFR algorithm can achieved a gain of 1.49dB in comparison with DS algorithm, 1.02 dB in comparison with 4SS, and 2,57dB in comparison with TSS algorithm. In comparison with the FS algorithm, a reduction more of 26 times in the number of SAD operations can be achieved, with quality losses about 1.43dB. The results for the RSICIFR algorithm shows that it presents the best tradeoff between quality and computational cost among all evaluated fast algorithms This algorithm is also a good option for hardware implementation, since it can achieve good quality results with a small search area, which implies in lower requirements for internal memory.

As future work is intended to generate a new version of RSICIFR with multiplies iterative searches. This algorithm will present an increase in the computational cost, however, it can significantly increase the quality results.

7. References

[1] PURI, A. et al. Video Coding Using the H.264/MPEG-4 AVC Compression Standard. Elsevier Signal Processing: Image Communication, [S.l.], n. 19, p.793–849, 2004.

[2] BHASKARAN, V.; KONSTANTINIDES, K. Image and Video Compression Standards: Algorithms and Architectures. 2nd ed. Boston: Kluwer Academic Publishers, 1999.

[3] 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), JVT-G050r1, Geneva, May 2003.

[4] KUHN, P., Algorithms, Complexity, Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Springer, June 1999.

[5] GONZALEZ, R.; WOODS, R. Processamento de Imagens Digitais. São Paulo: Edgard Blücher, 2003.

[6] JING, X.; CHAU, L. An efficient three-step search algorithm for Block motion estimation. IEEE Transactions on Multimedia, [S.l.], v. 6, n. 3, p. 435-438, June 2004.

[7] Xiph.org: Test media, available at <http://media.xiph.org/video/derf/>, December, 2009.

[8] JOINT COLLABORATIVE TEAM ON VIDEO CODING - JCT-VT: Documents of meetings of the group in: <http://wftp3.itu.int/av-arch/jctvc-site/>. Acess in: Jan. 2011.

CABAC Integration Into an H.264/AVC Intra-only Hardware Video Decoder

[1]Alonso A. de A. Schmidt, [1]Altamiro A. Susin
{alonso.schmidt, altamiro.susin}@ufrgs.br

[1]Universidade Federal do Rio Grande do Sul

Abstract

This work presents the integration of CABAC entropy decoder into an H.264 / AVC intra-only Video Decoder. Two previously developed Intellectual Properties were integrated with the development of a control and synchronization module. Also, since CABAC entropy decoder uses previously decoded syntactic elements values to select context models to estimate symbols occurrence probabilities, it was also developed a module to organize those syntactic elements values in memories and provide them as neighbor values to the current syntactic element being decoded. The final architecture reaches the frequency necessary to decode HD videos.

1. Introduction

The project of an H.264/AVC hardware video decoder is a goal of the Rede H.264 project [1]. The H.264/AVC is video compression standard selected for the SBTVD (*Sistema Brasileiro de Televisão Digital*) [2]. The Rede H.264 project is a consortium of several Brazilian universities to develop national products to deal with the source-signals coding for the SBTVD [1].

The development of the H.264/AVC video decoder is done by incremental hardware and modular software co-design in C / VHDL with FPGA prototyping. A software, called PRH.264 [3] was made to decode H.264/AVC progressive video bitstreams up to the main profile. This software easily provides intermediary decoding data for validating every hardware module.

There are three profiles in H.264/AVC, adopted by the SBTVD, in increasing level of complexity, which are *Baseline, Main* and *High*. At the time this work was being done, there was available an intra-only baseline H.264/AVC video decoder and a parser with support for new modules [4]. Since the development is done in an incremental approach, the team was working on the integration of the motion compensation module to complete the video decoder for the baseline profile, and the integration of the CABAC entropy decoder. The latter is the topic of this paper.

Fundamental H.264/AVC features, considered in this work, are explained in the section 2, focusing in specific knowledge needed to integrate and debug the CABAC entropy decoder. Section 3 explains functionalities of the CABAC decoder, the parser and propose some modules needed to make the integration possible. In section 4, simulation and FPGA synthesis results are presented. Finally, in section 5 concluding remarks and future works are indicated.

2. H.264/AVC Standard in the SBTVD

H.264/AVC is the video compression standard adopted by the SBTVD. Developed to surpass MPEG-2, H.264/AVC reaches higher compression rates taking advantage of new coding techniques. However, its coding algorithm leads to a higher computational complexity.

In the fig. 1, which shows a simplified block diagram of the H.264/AVC decoder, a video bitstream is processed by the parser and entropy decoder to supply parameters to the prediction and residual blocks. Images are reconstructed by adding residual data to predicted data. Residual data is obtained by the entropy decoder in the form of coefficents that must pass through the processes of inverse quantization and inverse transform. After the whole image has been processed, it is filtered to reduce block edges and it is stored for exibition and motion compensation reference.

To represent images, the color space YcbCr is used, with subsampling 4:2:0. Video images can have the structure of frames or fields (for interlaced video), and can have one or more slices, which are decoded independently. Each slice is divided into 16x16 samples regions called macroblocks, that are decoded in raster-scan order, except in the case of adaptive frames. Adaptive frames are processed in raster-scan order of pairs of macroblocks, that can be coded either as frames or fields. In each pair of macroblock, the top macroblock is first decoded, then the bottom one

In the intra prediction, a macroblock is reconstructed through a directional copy or mean DC value of border samples of the neighbor samples. The prediction can be done over a whole macroblock in the intra 16x16 prediction mode, or over blocks of 4x4 samples. In the first case, there are four possible intra 16x16 modes that are derived directly from the intra macroblock ID. In the case of the intra 4x4, there are nine possible prediction modes that are each derived separately.

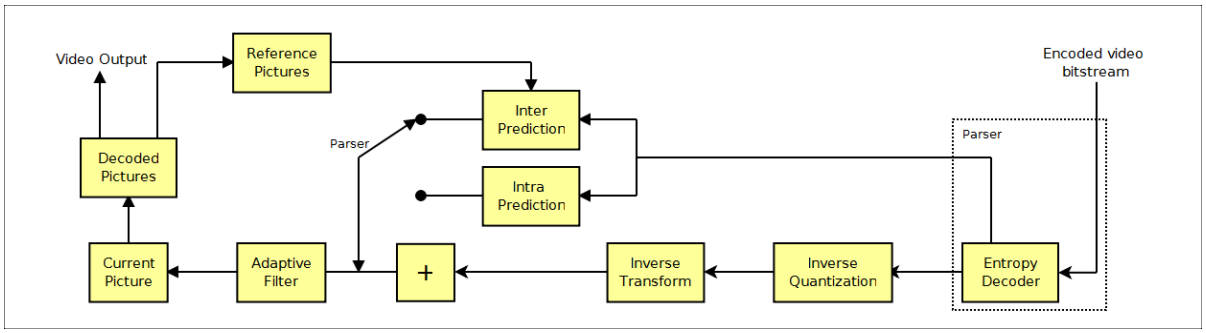


Fig. 1 – H.264/AVC video decoder block diagram.

Residual data is obtained by the entropy decoder in the form of coefficient blocks of an integer transform, based on DCT (*Discrete Cosine Transform).* There are arrays of 4x4 coefficients for Luminance AC, Chrominance AC, or Luminance DC. For chrominance DC the block is an array of 2x2 coefficients. The coefficient blocks are processed by an inverse quantization and then an inverse transform, employing only adders and shifters, to build the residual block data to sum with the predicted data.

2.1. CABAC decoding

CABAC (Context-based adaptive Binary Arithmetic Coding) is entropy coding employed to obtain the value of each syntactic element for the reconstruction of the macroblocks of the image. It supplies residual coefficents and parameters for the spatial and temporal prediction. Data compression is reached applying principles of arithmetic coding, and it depends on an accurate estimation of symbol occurrence probabilities.

Every syntactic element is obtained by decoding *bins* of a *binstring*. The binstring is converted to a syntactic element value by a de-binarization process.

For each bin to be decoded, a context model is selected. The context model stores the probability of a *bin* being '0' or '1'. The selection of the context model is based on values of previously decoded syntactic elements, at the neighborhood of the current macroblock or partition. There are 460 context indexes specified in the H.264/ AVC Standard [5] for videos in the *main* or *high* profile.

3. CABAC entropy decoder integration

The CABAC entropy decoder (denominated here as CABAC entropy decoder core) adopted for integration [6] implements no control over the sequence of the syntactic elements to be decoded by the video decoder. This task is supposed to be executed by an external control, done by the parser. The CABAC entropy decoder core also doesn't perform contexts initialization, bitstream handling, coefficient blocks building nor syntactic elements values storage and neighborhood managing necessary for context selection.

An already developed module [7] was adopted for contexts initialization. It implements four-stage pipeline to write in a RAM port the value of 460 context variables consuming 464 cycles. Other modules presented here had to be developed to complete the CABAC entropy decoder.

3.1. General control and communication with parser

A module was created to command contexts models initialization, environment values reading from the bitstream and syntactic elements decoding. This module is responsible for receiving syntactic elements decoding request by the parser and coordinating every module in the cabac decoder, as well as informing the parser when the task is finished. For this purpose, a *finite state machine* as shown in fig. 2 was designed.

3.2. Bitstream Handling

For this purpose, the CABAC environment variables are initialized consuming 9 bits from the bitstream in the *init_env_vars* state in the fig. 2. Then, in state *init_bs_handler*, 24 bits are read from bitstream without consumption, to feed three byte buffers for CABAC entropy decoder core.

While CABAC entropy decoder core is operating in state *decod_se*, it requests shifts in the buffers that must be read from the bitstream with the signal *incr_decod_stream_lenght* and marks current bit position in the first byte with the signal *bits_to_go*. The module *bs_handler* read this information and mark bits consumption to update the buffers, by comparing those two signals with copies stored in bs_handler itself and performing simple arithmetic operations.

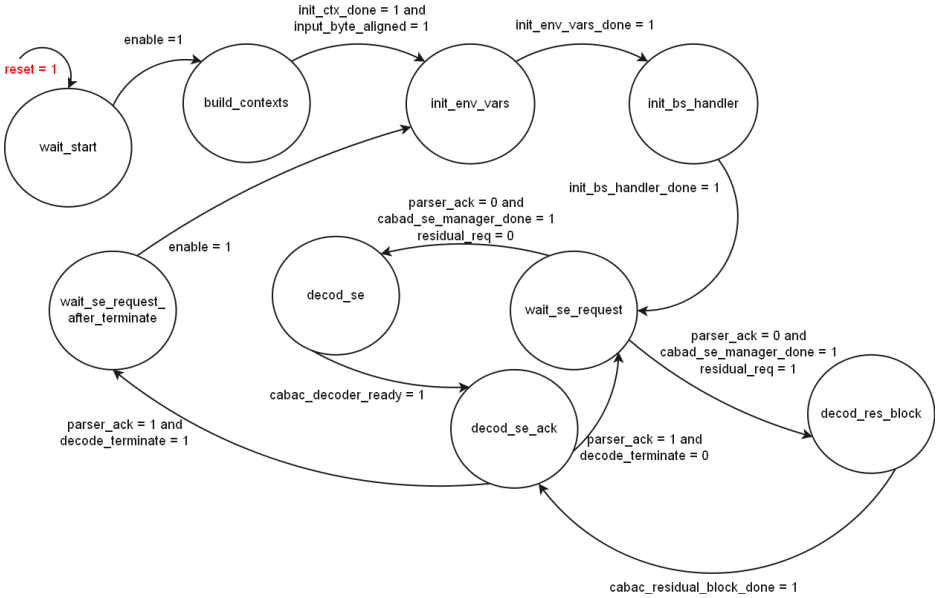


Fig. 2 – General control for CABAC entropy decoder modules

3.3. Decoded syntactic elements managing

CABAC entropy decoder core [6] uses a 41 bits input to calculate the context model index to decode a syntactic element (SE). The value that must be present in this interface depends on the syntactic element type to be decoded. For most of the syntactic elements, values of top and left neighbor syntactic elements must be provided for context selection. In an adaptive frame, the top or left macroblock may be the top or the bottom macroblock of a pair, as show in fig. 3. The occurrence of syntactic elements depends on its type and may be: one per macroblock (16x16 samples), one per sub-macroblock (8x8 samples) or one per block (4x4 samples), also shown in fig. 3. Thus, there must be syntactic elements values stored for the previously decoded neighbor macroblocks and inside the current macroblock being decoded.

The proposed solution involves grouping syntactic element types into four memories with different data width, where they are to be stored, as shown in tab. 1. Two memories are reserved for inter prediction syntactic elements. The memories are used for top SEs only, except for mb_se_info that can be used to read the left and the top neighbor at the same time in a dual port. The memory mb_se_info is used to read at once all SEs that occur once per MB. For others SEs, the left neighbors are stored in registers.

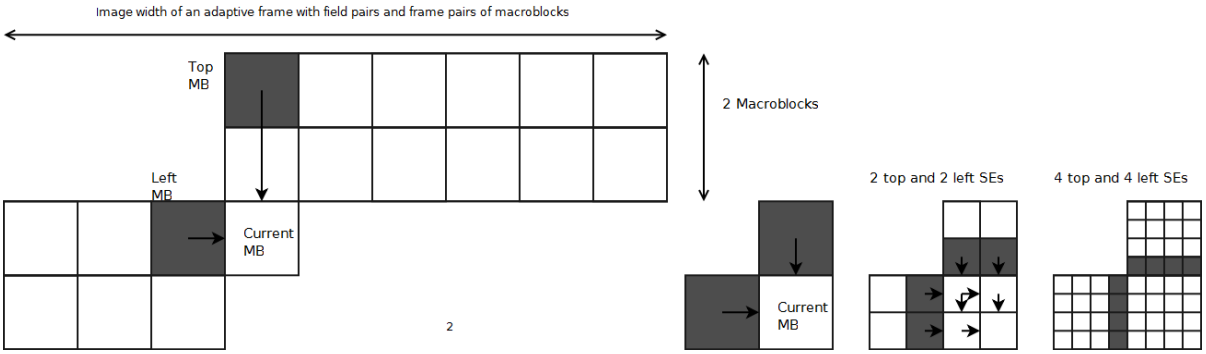


Fig. 3 – General control for CABAC entropy decoder modules

Tab.1 – Memories designed for storing previously decoded syntactic element values

Memory	Data width	Granularity	SEs per MB	Dual port
mb_se_info	20	16x16	1	yes
cbf_ac_info	8	4x4	8	no
Ref_idx_info	20	8x8	4	no
mvd_info	128	4x4	1	no

4. Simulation and synthesis results

To verify the proposed architecture, a test bench for the parser was executed. It reads a H.264/AVC video bitstream encoded with JM [8], and saves residual coefficient blocks for comparison with data generated by PRH.264 [3]. The results were proved correct for the first image frame, which contains only intra macroblocks.

CABAC entropy decoder (CABAD) and it's most expressive modules separately were synthesized for Xilinx Virtex 5 FPGA, as shown in tab. 2. Synthesis of the parser prior to the integration and then including the CABAC entropy decoder were done and the results are shown in tab. 3. The max frequency of 56 MHz is higher than the frequency established in [9] and enough for the CABAC entropy decoder to process HD video compressed data.

Tab.1 – Virtex 5 FPGA Synthesis summary for CABAC entropy decoder modules

Module	Flip Flops	LUTS	BRAMS	Max frequency
CABAD	861	4194	198 KB	56 MHz
CABAD CORE [6]	441	3035	90 KB	57 MHz
SE MANAGER	221	643	36 KB	176 MHz
RES. BLOCK	75	244	72 KB	292 MHz
BS HANDLER	19	186	0	318 MHz

Tab.2 – Virtex 5 FPGA Synthesis summary for parser without CABAC and then after its integration

Integration	Flip Flops	LUTS	BRAMS	Max frequency
Before	1339	3762	18 KB	103.4 MHz
After	2750	8993	216 KB	56 MHz

5. Conclusion and future works

This paper shows adequate solutions to the main problems that arise with the integration of CABAC into an H.264/AVC video decoder, reaching the frequency of 50MHz necessary for HD as shown in [9]. As future work, support for inter prediction must be integrated, with the development of arbitrarily sized macroblock and sub-macroblocks partitions neighborhood managing. Also, the parser with CABAC entropy decoder must go through a rigorous validation process, with a wider variety of video bitstreams.

6. References

[1] "Rede H.264 SBTVD Wiki", www.lapsi.eletro.ufrgs.br/h264/wiki, 2011.

[2] "Televisão Digital Terrestre – Codificação de video, audio e multiplexação", ABNT, Rio de Janeiro-RJ, 2007.

[3] A. A. Schmidt, F. F. Vidor, M. A. Lorencetti and A. A. Susin, "Development of a Software Model for an H.264/AVC Progressive Main Profile Hardware Video Decoder", 10th Students Forum on Microelectronics SForum'10, São Paulo-SP, 2010.

[4] M. A. Lorencetti, "Parser em VHDL para Decodificador de Vídeo H.264 para SBTVD", UFRGS, Porto Alegre, 2010.

[5] Video Coding Experts Group, "ITU-T Recommendation H.264 (03/05): Advanced video coding for generic audiovisual services", *International Telecommunication Union,* 2005.

[6] J. P. A. Carvalho, "Arquitetura Dedicada para Decodificação CABAC H.264/AVC em Sistemas em Silício", master's thesis, UnB, Brasília, 2009.

[7] D. A. Deprá, "Algoritmos e Desenvolvimento de Arquitetura para a Codificação Binária Adaptativa ao Contexto para o Decodificador H.264/AVC", master's thesis, Instituto de Informática, UFRGS, Porto Alegre, 2009.

[8] "H.264 Reference Software", http://iphom.hhi.de/suehring/tml/, 2010.

[9] A. C. Bonatto, A. B. Soares, A. Renner, A. A. Susin, L. Silva, S. Bampi, "A 720p H.264/AVC Decoder ASIC Implementation for Digital Television Set-top Boxes", SBCCI, São Paulo-SP, 2010.

A High Throughput Hardware Solution for the H.264/AVC Quarter-Pixel Motion Estimation Refinement

Marcel Moscarelli Corrêa, Mateus Thurow Schoenknecht, Luciano Volcan Agostini
{mmcorrea, mtschoenknecht, agostini}@inf.ufpel.edu.br

UFPel – Federal University of Pelotas

Abstract

This work proposes a hardware solution for the H.264/AVC Quarter-Pixel Motion Estimation Refinement, ready to be used in a complete Fractional Motion Estimation architecture. The architecture was optimized to reach a high throughput through a balanced pipeline and parallelism exploration. The design was described in VHDL and synthesized to an Altera Stratix III FPGA device, achieving an operation frequency of 245 MHz and processing up to 39 QHDTV frames (3840x2048 pixels) per second. This architecture is also able to reach real time when processing lower resolutions, like HD 1080p (1920x1080 pixels) with lower operation frequencies. The final results are very competitive when compared to related works.

1. Introduction

The H.264/AVC (Advanced Video Coding) standard [1] is the state of art in video coding. It was developed by experts from ITU-T and ISO/IEC intending to achieve the highest compression rates when compared to older standards, like the MPEG-2. The current video coding standards primarily define two things: (1) a coded representation (or syntax) which describes the visual data in a compressed form and (2) a method to decode the syntax to reconstruct the visual information [2].

The H.264/AVC has a very high computational complexity, which makes difficult for software solutions to encode high definition videos in real time when using the H.264/AVC most complex features. The high complexity of the H.264/AVC encoding tools are also a restriction for embedded and mobile systems, since software implementations in this scenario demand high performance embedded CPUs, consuming a lot of energy. For these reasons, hardware architectures are the best way to encode and decode high definitions videos in real time processing, e.g. HD 1080p (1920x1080 pixels) at 30 frames per second.

The Motion Estimation (ME) is most complex module of the H.264/AVC encoding process. It explores and reduces the temporal redundancy, which is the similarity between sequential frames and works by splitting the current frame into several macroblocks (16x16 pixels) and then searching in the previous coded frames (reference frames) for the macroblock that is most similar to the current one. After this search, an integer motion vector (IMV) is generated to indicate the offset of the selected macroblock inside the reference frame. The goal of the ME is to find the best IMV whilst keeping the computational complexity between acceptable limits [2].

This work proposes a high throughput and low cost hardware architecture to perform the interpolation and refinement search of quarter-pixel samples used in the fractional motion estimation (FME). This architecture will be integrated to a Half-Pixel ME Refinement architecture and to an Integer ME (IME) architecture (which are being designed in parallel works), generating a complete FME solution able to process high resolution videos in real time.

This paper is structured as follows: Section 2 presents the FME and the quarter-pixel ME refinement process, Section 3 discusses software experiments, Section 4 presents the proposed hardware architecture, Section 5 shows the synthesis results and discusses related works, and finally, Section 6 concludes this work.

2. Fractional Motion Estimation

A characteristic that contributes to the high compression rates achieved by the Motion Estimation of the H.264/AVC standard is the possibility to generate fractional motion vectors [1]. In other words, a movement that happens from a frame to another is not restricted to integer pixel positions only.

The H.264/AVC includes both half-pixel and quarter-pixel ME refinements. In this case, the process usually follows three steps: (1) Integer Motion Estimation, (2) Half-Pixel ME Refinement, and finally, (3) Quarter-Pixel ME Refinement. These refinement processes increase significantly the computational complexity of the ME.

2.1. Quarter-Pixel Interpolation Process

The quarter-pixel accuracy in the FME is a H.264/AVC innovation. The interpolation of quarter-pixels happens after the interpolation of half-pixels and the half-pixel ME search. It uses the best match block

composed by integer position samples and half-pixels to generate a new search area. Any quarter-pixel can be obtained through a bilinear filter, defined in (1), where A and B are 8-bit luminance samples.

$$y = (A + B + 1) >> 1 \qquad (1)$$

There are three types of quarter-pixel samples: (1) **H Type**, which is calculated using the two closest horizontal samples; (2) **V Type**, calculated using the two closest vertical samples; and **D Type**, calculated using the two closest diagonal samples in each direction. Fig. 1 shows a set of interpolated half-pixels between the integer position samples. The integer samples and the half-pixel interpolated samples are used to generate the quarter-pixel samples. Fig. 2a shows a set of H Type quarter-pixels, Fig. 2b shows a set of V Type quarter-pixels, and finally, the Fig. 2c shows a set of D Type quarter-pixels.

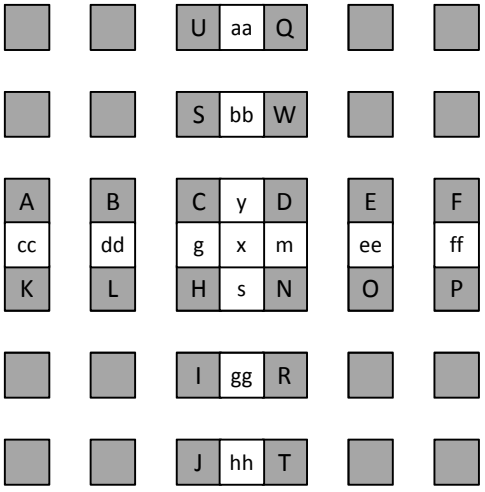


Fig 1 – Half-pixel positions (white squares) between integer position samples (gray squares).

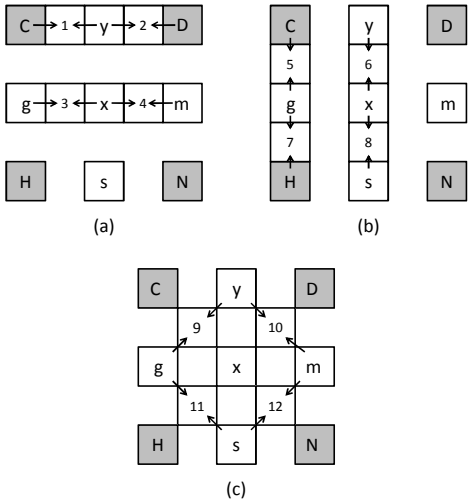


Fig 2 – Quarter-pixel positions (labeled with numbers) between half-pixels and integer position samples.

2.2. Quarter-Pixel Motion Estimation Search

Using the interpolated search area, the search process tests all the eight possible matches inside this area to check if there is a block composed by quarter-pixels that is more similar to the original block than the best match found by the previous refinement step. The search is done using a block-matching algorithm, which uses a distortion criterion to determine the most similar block. This criterion can be a simple arithmetic difference between blocks or more complex calculations. Among the most used distortion criterion are the Mean Square Error (MSE), Sum of Squared Differences (SSD) and the Sum of Absolute Differences (SAD) [1].

The SAD is defined in equation (2) and it is probably the most widely used criteria for reasons of computational simplicity [1]. The SAD takes the absolute value of the difference between each sample in the original block and the corresponding sample in the candidate block. These differences are added to create a simple metric of block similarity.

$$SAD(P,O) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left| P_{i,j} - O_{i,j} \right| \qquad (2)$$

Once the SAD values for all the eight possible matches are calculated, search module will check if there is a better match by comparing the SAD values of the quarter-pixel candidate blocks with the best match of the half-pixel search. If there is a better match, the motion vector must be modified through the addition of the corresponding quarter sample precision offset.

3. Experimental Evaluation

Using the H.264/AVC reference software [3], an evaluation was done to check the impact of simplifying the ME excluding the variable block size support (VBS) and maintaining the fractional motion estimation. Five QCIF video sequences commonly used by the video coding community were coded using two different ME block size configurations. Two main results were measured: PSNR and bit-rate.

Table 1 considered the use of all possible partitions and sub partitions sizes, and Table 2 presents the results when only the 8x8 block size is used. The use of only 8x8 blocks increased the bit-rate by 3.84% and reduced the PSNR by 0.32 dB on average when compared to the optimal case (Table 1). Considering the presented results, we decided to design a complete fractional motion estimation architecture that supports only the 8x8 block size, since this decision simplifies a lot the hardware design without expressive quality and compression

rate losses. The use of a single block size reduces a lot the motion estimation complexity and avoids the necessity of a decision mode to choose the best block size. The decision mode is a big bottleneck in the H.264/AVC encoder, since the adopted rate-distortion criterion (RDO) is very complex [4].

Results for the complex variable block size configuration.

Video Sequence	PSNR (dB)	Bit-rate (Kbps)
Akiyo	40.82	114.5
Bridge Close	37.38	155.33
Coastguard	34.12	272.33
Foreman	36.47	207.87
Mobile	32.59	391.81
Average	**36.28**	**228.37**

Results for the simple block size configuration.

Video Sequence	PSNR (dB)	Bit-rate (Kbps)
Akiyo	40.47	116.53
Bridge Close	37.16	154.56
Coastguard	33.80	277.21
Foreman	36.01	225.24
Mobile	32.34	412.16
Average	**35.96**	**237.14**

** ME configuration: FullSearch algorithm, 32x32 search area, one reference frame, quarter-pixel accuracy, QP=28*

4. Quarter-Pixel Motion Estimation Refinement Architecture

This paper presents an architecture that increases the accuracy of the ME process. The architecture is divided in two main parts: the quarter-pixel interpolation and the motion estimation search. These two operations were interleaved to reduce the use of clock cycles.

4.1. Proposed Interpolation Unit

The Quarter-Pixel Interpolation Unit for 8x8 blocks proposed in this section has two main parts: the Filter Line and the Buffers.

The Filter Line is a module that interpolates an entire line of two possible matches in a single step. Since there are 8 quarter-pixels per line, the Filter Line must have 16 Processing Units (PU). Each PU is a basic operative unit used to apply the bilinear filter defined in the equation (1).

The buffers are necessary to store and shift both the input and output samples. There is a main buffer to store the 17x17 area composed by integer position samples and half-pixels, and three buffers to store the interpolated V, H and D Type quarter-pixels.

The interpolation architecture needs 17 clock cycles to fill the main buffer, one line per clock cycle. Then, it needs another 32 clock cycles to interpolate and store all quarter-pixel samples. A total of 49 clock cycles are necessary to interpolate a new search area composed by quarter-pixels around the best match chosen by the previous refinement process.

4.2. Proposed Search Module

The quarter-pixel search process is very similar to the half-pixel search, and it has two steps: the SAD calculation and the SAD comparison. Since the interpolation unit can interpolate an entire line or column of two blocks composed by quarter-pixels in a single step, the SAD calculation can be done in parallel with the interpolation.

The SAD calculation is done using two SAD Trees operating in parallel. The SAD Tree (ST) is a module that calculates the SAD value for a candidate block line by line. The SAD value of a single line or column is calculated and stored in an accumulator register. Each ST use a 2-stages pipeline configuration, taking 9 clock cycles to calculate the SAD value for the entire block.

The SAD Comparator (SC) can compare four SAD values in parallel and it uses a two-stage pipeline configuration. As first step, the comparator stores the smallest SAD value and its corresponding FMV among the V and H Type possible matches and then compares it to the best SAD found by the previous refinement process. Extra clock cycles are not necessary since this comparison occurs in parallel with the D Type interpolation and SAD calculation. As second step, the comparator stores the smallest value among the D Type possible matches and compares it to the smallest SAD obtained in the first step. Two extra clock cycles are necessary to obtain the smallest SAD value and its corresponding MV.

5. Synthesis Results and Related Works

The proposed design was fully described in VHDL and synthesized to an Altera Stratix III ep3sl50f484c2 FPGA device using the Quartus II synthesis tool [5].

When synthesized to the ep3sl50f484c2 device the architecture achieves a maximum frequency of 245 MHz in the worst case (when the slow 1100mv 85c mode is considered) and 252 MHz in the best case (when the slow 1100mv 0c mode is considered). At the worst-case scenario, our architecture can process up to 39 QHDTV frames (3840x2048 pixels) per second. However, since the quarter-pixel ME refinement is the less

complex process in the FME, it might be limited by a bottleneck. The minimum frequencies to process videos at 30 frames per second are: 188.01 MHz for QHDTV, 49.97 MHz for HD 1080p (1920x1080), and 22.03 MHz for HD 720p (1280x720).

The operative module (Filter Line, two SAD Trees, and SAD Comparator) of the architecture consumed 659 LUTs (1.7% of the target device resources) and 132 dedicated logic registers (0.3% of the target device resources). The buffers were mapped as registers banks and consumed 9 registers (13-bits), 801 registers (8-bits), 289 4:1-multiplexers (8-bit) and 512 2:1-multiplexers (8-bit).

Works focusing on hardware design for a quarter-pixel ME refinement that works only with 8x8 blocks were not found in the literature. For comparison, two works about FME are considered, however these works target different technologies.

Performance comparison among related works.

	Oktem [6]	**Kao [7]**	***This Work***
Technology	FPGA Virtex2	TSMC 0.13 μm	FPGA Stratix III
Support to VBS	Yes	Yes	No
Quality Drop (PSNR)	-	~0.27 dB	~0.32 dB
Maximum Frequency	60 MHz	100 MHz	245 MHz
Throughput (QHDTV frames/s)	1.3	29.3	39

The FME architecture proposed by [6] gives support to VBS by splitting a macroblock into smaller 4x4, 8x4 or 4x8 blocks. Since [6] takes 44 clock cycles to perform the interpolation and search of a 4x4 block, it needs 176 clock cycles to interpolate an 8x8 block. Our design needs 61% less clock cycles to process a 8x8 block, making it more suitable for high quality video coding, with a quality loss of only 0.32 dB.

The FME architecture proposed by [7] gives support to VBS too, and it uses a heuristic based mathematical model to make the quarter-pixel interpolation less expensive, however it degrades the resultant quality of the video. When compared with our approach, the solution proposed in [7] causes an inexpressive gain of 0.05 dB in terms of quality. Even using a heuristic to reduce the ME complexity the solution [7] is more complex, since it supports VBSME and makes necessary the use of a decision mode.

6. Conclusions and Future Works

In this paper, a high performance and low cost hardware architecture for the H.264/AVC quarter-pixel motion estimation refinement process was presented. This architecture supports only 8x8 block sizes instead of the complete VBS proposed by the H.264/AVC standard. This simplification reduces a lot the ME complexity with a little loss of 0.32 dB in PSNR.

When synthesized to an Altera Stratix III FPGA, our architecture achieves a very high processing rate. It can process high definition videos like HD 1080p in real time (30 frames per second) when running at lower frequencies, such as 50 MHz, which is very important when power and energy consumption restrictions are being considered. Our solution is also able to process 39 QHDTV frames per second surpassing all related works in terms of throughput. A high throughput is very important to not degrade the performance of a complete FME architecture.

Our solution presents a very good tradeoff between throughput and hardware cost, since it requires a low frequency to encode high definition videos and it consumes a small amount of hardware resources.

As future works, we plan to design a fully functional Fractional Motion Estimation architecture with fast search algorithm and quarter pixel accuracy.

7. References

[1] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, 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.

[2] I. G. Richardson, H.264 and MPEG-4 Video Compression, Wiley, 2003.

[3] JM17. “H.264/AVC JM Reference Software.” Mar, 2011; http://iphome.hhi.de/suehring/tml

[4] Sullivan, G; Wiegand, T. "Rate-Distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74-90, 1998.

[5] ALTERA. “FPGA CPLD and ASIC from Altera.” Mar, 2011; http://www.altera.com

[6] Oktem, S; Hamzaoglu, I, “An efficient hardware architecture for quarter-pixel accurate H.264 motion estimation,” 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, August 2007.

[7] Kao, C; Kuo, H; Lin, Y, “High performance fractional motion estimation and mode decision for the H.264/AVC,” IEEE International Conference on Multimedia and Expo, July 2006.

A Rate-Distortion Metric Targeting Perceptual Video Coding

Bruno George de Moraes, Ismael Seidel, José Luís A. Güntzel
{brunogm,ismaelseidel,guntzel}@inf.ufsc.br

Embedded Computing Lab. (LCE)
Federal University of Santa Catarina, Florianópolis, Brazil

Abstract

This paper presents a new metric, called Quantization Distortion Energy (QDE), based on perceptual video coding, proposed for block matching considering the losses in quantization, and its evaluation in an H.264/AVC encoder. Three tests were elaborated to compare the metric using rate–distortion optimization (RDO) with Sum of absolute transformed differences (SATD) metric. The first test shows the curve of structural dissimilarity (DSSIM) and peak signal to noise ratio (PSNR) at various bit-rates. The second test shows the size reduction with the quality normalized by SSIM, resulting in a variable constant ratefactor (CRF) for the proposed encoder and a fixed CRF for the original encoder. In the third test an algorithm was used to perform the second test automatically spanning CRF range. The results show that the proposed metric, at the same quality in SSIM, provides gains in bit-rate ranging from 2% to 60%. Furthermore, in most cases considering the same bit-rate, gains were obtained in both quality metrics: SSIM and PSNR.

1. Introduction

The H.264/AVC standard provides at least 50% of bit rate improvement over MPEG2, which has guaranteed its widespread adoption for transmission and storage of high definition video in applications such as some digital television standards (e.g., SBTVD, DVB-S2), recent movie formats (e.g., Blu-ray and AVCHD), etc. Despite being the state-of-art for its efficiency, the bandwidth and costs associated with, per example, a 1080p IPTV (Internet Protocol television) application has limits in user-base by definition of broadband penetration on the market. Therefore, improving the analysis step of an H.264/AVC encoder contributes to expand its use to other applications.

Motion pictures or videos have lots of data. To compress them, each frame is divided into macroblocks and further into smaller blocks that are submitted to the Inter and Intra Frame Prediction techniques. Those techniques explore temporal and spatial redundancy (i.e., nearby video blocks exhibiting high correlation) by coding only the differences between similar blocks. A search algorithm will look for candidate blocks, and the best candidate block is selected in a simple view with a specific distortion metric [1]. Thus, by exploring the encoder metrics used to find matching blocks with minimum residues and distortion, it is possible to achieve better compression rates, while keeping the video quality or, conversely, increasing quality for the same bit-rate.

Distortion metrics are divided into objective and subjective. Objective metrics are purely mathematical models derived from signal theory and do not consider human visual system (HVS) characteristics. However, for performance reasons, they are the most used ones. Subjective metrics exploit the HVS models. The few existing ones suffer from poor performance and explore only a small set of features. For that reason, they are neither widely used nor well accepted [2]. Examples of the common used metrics are the SAD (Sum of Absolute Differences) [3], the SSD (Sum of Squared Differences) [4] and the SATD (Sum of Absolute Transformed Differences) [3]. This paper presents, as contribution, the development and evaluation of a subjective metric, named Quantization Distortion Energy (QDE) which is explained in Section 2.

The rest of this paper is organized as follows. Section 3 shows the evaluation process for the metric, whereas Section 4 comments the evaluation results. Conclusions and perspective work are outlined in Section 5.

2. Proposed Metric

Most of the metrics used for matching block does not directly consider the quantization step, which occurs after the discrete cosine transform (DCT) [5], for the current block. Typically, only the reconstructed data is considered containing losses in quantization, inverse quantization and inverse transform. The DCT coefficients represent frequency bands, for which it is known that the HVS has different sensitivity, as defined by the contrast sensitivity function (CSF) [6]. The quantization eliminates some of those frequency bands in order to achieve high compression rates. However, as drawback, the quantization causes a great amount of image distortion, either by rounding errors or by frequency band elimination [4].

By considering the errors caused by the quantization step, it is possible to predict what block match candidate will cause less distortion after the inverses of quantization and transform steps, for frame re-assembly. The worst errors occur when some transformed coefficients are eliminated. Based on that fact, we propose to apply a greater weight for the coefficients that will cause the most distortions. This is done by

predicting the eliminated coefficients before the quantization. Thus, for all the 52 quantization parameters (QP), the thresholds of DCT coefficients that will be eliminated shall be calculated, for the multiplicative factors (MF) table of H.264/AVC [3].

$$0 = round(Coeff * MF[QP\%6, n] / 2^{floor(QP/6)}) * \frac{1}{2^{15}} \text{ [3] (1)}$$

In order to calculate the metric some threshold values are obtained from the quantization equation (1) which was solved for "Coeff", obtaining the root value for all MF, as mentioned before. The thresholds were calculated for 4x4 and 8x8 DCT of H.264/AVC. Only the QP 51 thresholds were stored, whose values can be seen in Tab.1, for 4x4 blocks. The others thresholds can be obtained by doing a simple shift operation by *[8 - floor(QP/6)]*. By comparison of the DCT coefficients with the thresholds values, the coefficients can be classified in two groups, eliminated and encoded. The encoded will be handled as in SATD, by gathering absolute sums. The eliminated coefficients will be quadratically summed as in SSD. This ensures that the encoded group will be representative for the bit-rate and the eliminated group will estimate the distortion caused by quantization. Such proposed metric will be called Quantization Distortion Energy (QDE).

Tab. 1 - The QP 51 thresholds coefficients for 4x4 block

640	1039	1599
703	1119	1800
832	1279	2000
896	1440	2300
1023	1599	2500
1151	1840	2899

3. Testing QDE

As test platform for QDE the H.264/AVC encoder of the x264 project [7] was used. In the x264 there are three modes of rate control: Constant Quantizer, Bit-rate and Constant Ratefactor, which is similar to constant quantizer, but acts on reducing/improving quality according to frame importance, maximizing perceptual quality of the encoded video. To compare to the original version of the x264, which uses SATD, appropriate modifications in the encoder were accomplished, in order to use QDE thresholds to perform coefficient classification. Residues were summed in absolute and squared parts respectively, constituting the modified x264 version. It is important to notice that in x264, after quantization there is one step called decimation which eliminates some coefficients. This step is present in both encoders, and hence does not influence the comparison itself. Those two versions of x264 encoder were compared against each other, to show the possible gains or losses of the new metric. For the tests, nine random video samples were selected from the Xiph.org Test Media Repository [8] to form a miscellaneous resolution and complexity set. The tests were run on a 2.83GHz Intel Core 2 Quad Q9550 machine with 12M Cache and 4GB of RAM with Ubuntu Linux 10.04 SMP 64bit 2.6.32-23-generic operating system. The three performed tests are referred to as "A", "B" and "C".

In the experiment "A", the encoder is parameterized to compress the video by restricting bit-rates at the values of 400, 700, 1000, 2000, 4000 and 8000 Kbit/s, using the multi pass controller, which guarantees the specified rates. A script iterated all bit-rate values for both encoders, and then two charts were generated from the statistics file, one chart showing the curves for the bit-rates against PSNR [4], and another one with bit-rates against DSSIM [4], which is a distance metric derived from SSIM [9].

The SSIM measures the structural similarity between two images: an original (distortion-free) to which the distorted image will be compared. This kind of approach is known as *full-reference* (FR). The other two approaches are *no-reference* (NR), in which the reference image is not available, and *reduced-reference* (RR), in which the reference image is only partially available. The most traditional image/video quality evaluation metric is the PSNR. However, the PSNR values do not correspond to the perceived visual quality due to the non-linear behavior of the HVS. In principle, the SSIM assumes that the HVS perception is highly adapted to extract structural information from a scene. Hence, SSIM is one of the most effective perceptual quality evaluation metrics and that justifies its adoption in this work to measure quality.

For experiment "B", the quality measured in SSIM was restricted by encoding the video with CRF [7] equals to 20.0 (23.0 by default at x264 encoder, used 20.0 to have high quality samples). For each test sample, the same quality was sought by doing manual changes in CRF on the modified version of the encoder until the quality was reached. By doing so, it was possible to measure the differences in video size for the two encoders.

Experiment "C" can be considered as the most important one because it allows the comparison of gains (in bit-rate) for the same SSIM on both encoded videos. As long as in x264 there was not a way to choose final SSIM, an algorithm was created to iterate over several test samples spanning CRF from 10 to 30, looking for convergence between the original SSIM and the modified one by adjusting its CRF.

.

4. Results

The following results present the QDE performance in quality against the SATD as defined in the RDO of the x264, both metrics act as D metrics (part of RDO) in such case. Charts showed in Fig.1 indicates, for the sequences "Riverbed" and "Sunflower, the behavior of QDE and SATD in quality, DSSIM (upside charts) and PSNR (downside charts) versus the bit-rate of the final encoded sample, as defined in experiment "A". The charts of "Riverbed" show that, for all samples, QDE had better DSSIM and PSNR over SATD. "Sunflower" is a video sample that achieved gains up to 60% in compression at experiment "C" (see Fig. 3), for the same quality in original encoder. This sample in lower bit-rates has losses in PSNR and DSSIM. However, as its bit-rate increases, the QDE shows better results than SATD. Thus, experiment "A" shows an apparent tendency of QDE superiority over SATD, as the bit-rate increases. On the other hand, experiment "C" shows that there are cases, for the same quality, where higher bit-rates result in a worse compression with QDE.

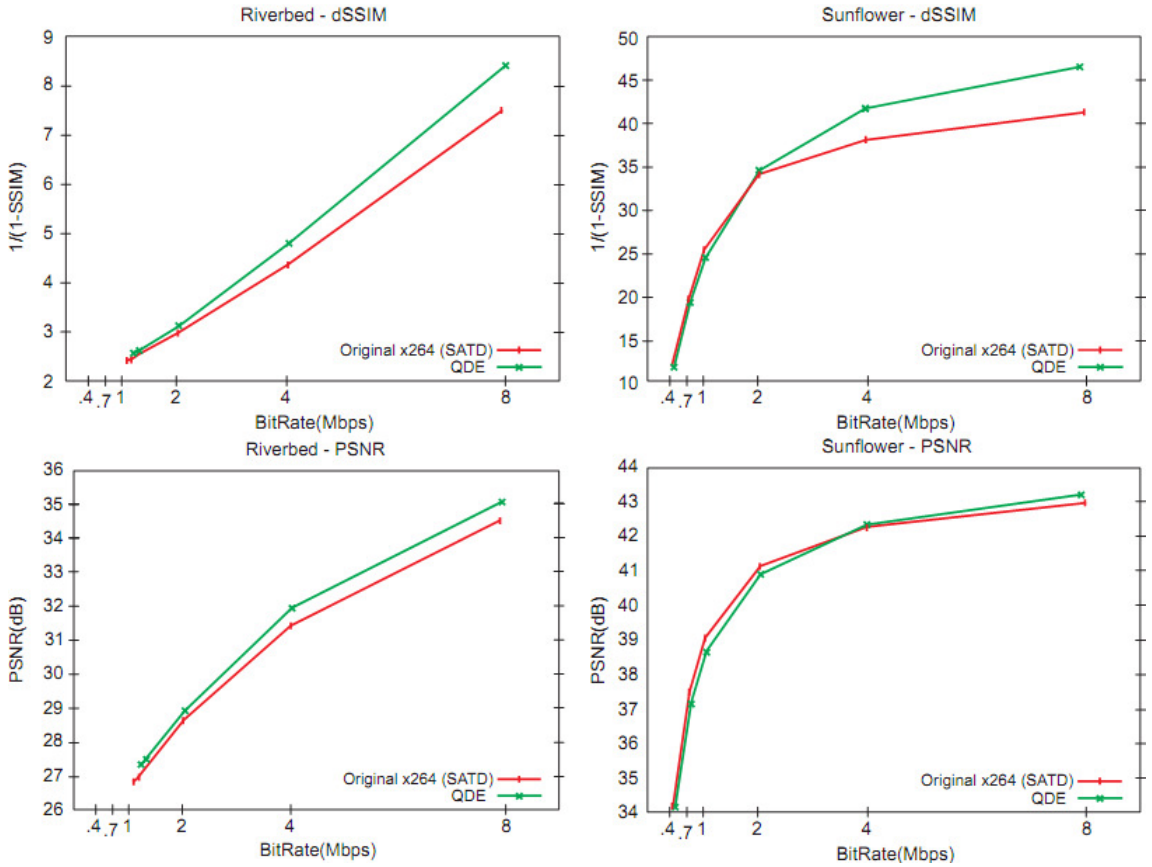


Fig. 1 - Graphics generated for PSNR and DSSIM for sequences "Sunflower" and "Riverbed", as part of the experiment "A"

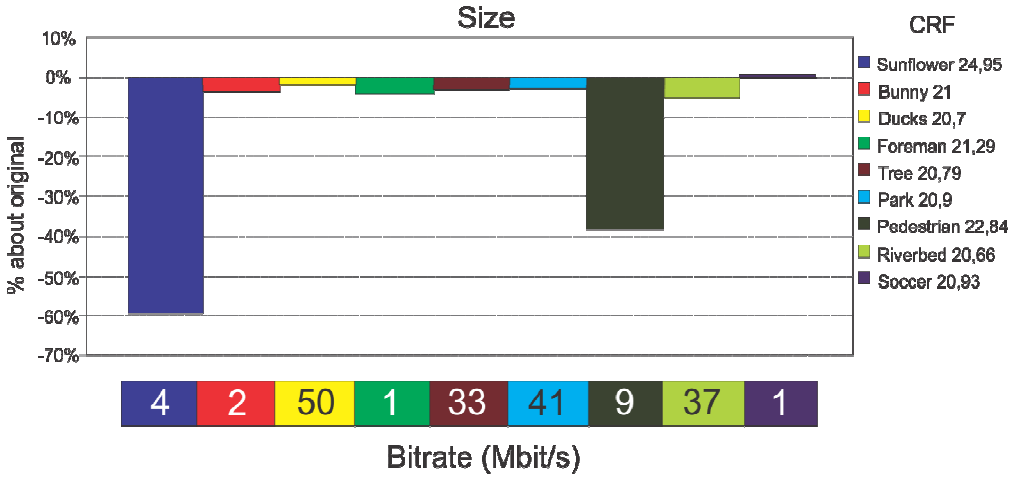


Fig. 2 - Experiment "B": results at size of QDE relative to the original

The simplest experiment, "B" (Fig. 2), is a short overview of the metric at CRF 20 for all nine tested video samples. In Fig. 3 it is possible to see the results for experiment "C". In this figure each marked point at

the curves represents, from right to left, a CRF value of the original coded video, ranging from 10 to 30. The horizontal axis informs the quality in SSIM achieved for both encoders at different CRFs. The vertical axis shows the gain in size (kb) relative to the original encoded sample. In the case of "Ducks take off" there is a steady improvement at CRF range from 10 to 19 of 0.5% to 2.5%. Above that, from CRF 20 to 27, there are higher gains until a peak of ~5%. From CRF 28 to 30 diminishing gains occur due to the limits imposed by lower qualities. Yet for the experiment "C", the results for "Sunflower" samples show a minor improvement range until CRF 15 (in practice, lossless). Above that, a substantial gain, up to a peak of ~60% in CRF 20. Above such CRF the gains diminish down to some penalty at CRF 28 (with a bottom of ~ -5%). These effects in CRFs greater than 28 are felt even at CRF 51 in which QDE presents better SSIM quality than attainable, because there is no higher CRF. Thus, the problem relies on achieving higher quality for the entire quantizer range allowed by H.264/AVC. For CRF after ~28, the encoder tries to match the available space in the encoding buffer, for which a low bit-rate is needed.

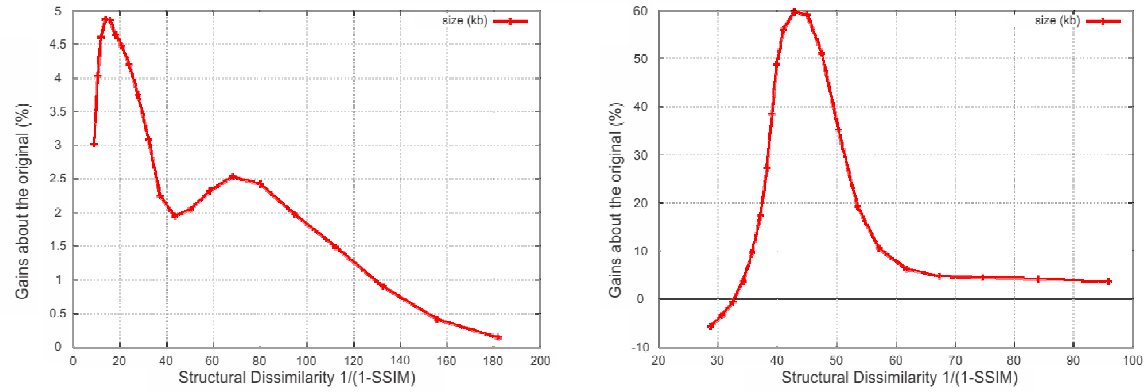


Fig. 3 - Experiment "C" shows the result of DSSIM vs. size gain over the original using the proposed metric for samples "Ducks Take Off" and "Sunflower", respectively

5. Conclusion and perspective work

The evaluation of the Quantization Distortion Energy (QDE) metric, based on the prediction of eliminated coefficients caused by quantization, showed that there is much information available during the encoding process which could be used for better block-matching, and this metric successfully utilizes some of this information. The generated graphics show gains in videos coded with QDE. This efficacy has been presented with the three conducted experiments for most of the tested video samples. Regarding experiment "B", only one among the nine tested videos had a loss in bit-rate, for the same quality. Experiment "A" shows that, switching the bit-rate, the quality can be increased not only by increasing the bit-rate itself, but also by using new block match metrics, such as QDE. Experiment "C" shows that there are certain CRF ranges where, at the same quality, compression rates are better for QDE than for SATD.

Ongoing work includes the implementation and comparison of the QDE within the JM reference software of H.264/AVC [10] to also investigate how this proposed metric behaves. In addition, a hardware implementation of QDE is currently under development.

6. References

[1] R. Dornelles, L.Agostini. "A Low Cost Real Time Motion Estimation/Compensation Architecture for the H.264/AVC Video Coding Standard". *Proc. 25th South Symposium on Microelectronics* (SIM'10), 2010.
[2] H. R. Wu, K. R. Rao. *Digital Image Video Quality and Perceptual Coding*. CRC, Boca Raton, FL, 2005.
[3] I. Richardson, *H.264 and MPEG-4 video compression – Video Coding for Next-Generation Multimedia*. John Wiley&Sons, Chichester, 2003.
[4] B. G. Moraes. "Uma métrica para Taxa de Distorção voltada codificação de vídeo perceptiva". Undergrad dissertation, UFSC, Florianópolis, 2010.
[5] E.T.M. Manoel. "Codificação de Vídeo H.264 - Estudo de Codificação Mista de Macroblocos". M.Sc. dissertation, UFSC, Florianópolis, 2007.
[6] R. L. D. Valois e K. K. D. Valois, *Spatial vision*. Oxford University Press US, 1990.
[7] L. Merritt, et al. x264 Project. Available at: <http://developers.videolan.org/x264.html>.
[8] Xiph.org. Test media. Available at: <http://media.xiph.org/>.
[9] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.
[10] JM Joint Video Team Reference Software. Available at: <http://iphome.hhi.de/suehring/tml/>.

Processor and Demux Integration for the SoC-SBTVD

[1]Jeffrei Moreira, [1]Jônatas Rech, [1]Henrique Klein, [1]Altamiro Susin
jeffrei.moreira@ufrgs.br, {jrrech,haklein}@inf.ufrgs.br, altamiro.susin@ufrgs.br

[1]Universidade Federal do Rio Grande do Sul

Abstract

The main goal of this paper is the integration of a transport stream *demultiplexer (Demux) and a closed caption decoder to the main processor of a Set-top Box (STB) for the Brazilian Digital Television System (SBTVD). The modules were presented in a previous paper at the 25th SIM ([1]), and now the integration to the STB architecture is reported. The* AMBA *bus was used to connect the main processor core to the demultiplexer and other peripherals. The attachment of the decoders is briefly addressed, with an emphasis to the coupling of a* closed caption *decoder to the corresponding demultiplexer output and its interface with the digital video output through the system main memory. An* Aeroflex Gaisler *Leon3 core was used as main processor.*

1. Introduction

The project in the framework of which this work has been developed is called System on Chip for the Brazilian Digital Television System (Soc – SBTVD), and comprehends a national academic cooperation in order to develop an encoding and decoding specific hardware complying with the ISDB-T standard for digital television broadcasting. This work follows the path started in the 2010' SIM with [1], describing the implantation of a development environment for an open-source soft-core processor. Although further work showed that the Plasma processor performed faster than the Leon3 core in data demultiplexing tasks, the latter has been chosen due to greater portability, support for operational systems and documentation, leaving raw data demultiplexing to a dedicated hardware module.

The work flow begins with the compilation of a Linux kernel for the Leon3 core, followed by a description and comparison of data input methods used in the initial integration tests. After that, the integration of the CPU, hardware demux peripheral and closed-caption decoder will be reported.

2. Leon3 processor and Transport Stream input

2.1. Operational system compilation

The Leon3 processor is a soft-core CPU developed by Aeroflex Gaisler [2] (former Gaisler Research), totally described in open-source VHDL (except for optional IP cores not needed for this project). It features, among other IP cores, a memory management unit (MMU), that allows us, combined with an external RAM, to run virtual memory-based operational systems (i.e. Linux 2.6) on the processor. Fig. 1 shows a diagram of the possible configurations.

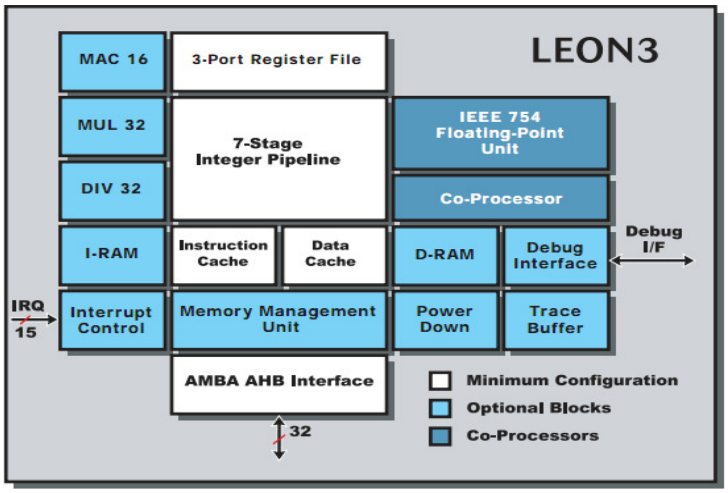


Fig. 1 - block diagram of the Leon3 CPU;

The Leon3 hardware is being currently prototyped in a Xilinx Virtex 5 LX110T FPGA, hosted in a XUPV5-ML509 board. The Virtex 5 device was preferred to the Virtex 2 for being able to host the whole system when the integration reaches its final stage, since the processor itself occupied nearly 90% of the Virtex 2 FPGA, while in the Virtex 5 approximately 40% of the gates were used.

The intention in running a fully-featured OS comes from the future need of providing and managing a graphical user interface for the SBTVD, along with the wish of having a more powerful platform for software than that provided by the Plasma processor (both in terms of performance and library availability). It also simplifies the task of data input, since it saves the programmer the work of tweaking the lower network layers in case of an Ethernet-based transmission, or the abstraction of a file system in a mass storage device (local or remote).

A previously running version of Linux (kernel 2.6.11.1) was taken from a previous work as a start off. The migration to kernel 2.6.36 occurred as Gaisler Research announced official support and additional driver packages, and required minor adjustments to match the synthesized hardware current settings.

2.2. Transport Stream data input

At the earliest stage of research on the data input and demultiplexing, the Plasma processor was used to run a software demux fed by Transport Stream (TS) chunks previously stored in internal RAM. Such test was for functional software and processor validation only, since it did not represent a continuous and "infinite" source of data, as does real digital TV broadcasting. With support from an OS, the implementation of a data source using an Ethernet interface seemed reasonable, since it supports high speed data transfers (up to 100Mbps, much more than the approx. 2.3 MByte/s needed for FullHD video) and was readily available for testing (an Ethernet MAC IP core is available as a peripheral for Leon3). The utilization of a CompactFlash card as source of data was also considered, but it was no longer supported by Linux newer kernels, being eventually discarded.

In the current system, two data sources are available, one through an NFS share and the other through a UNIX sockets interface, managed by two pieces of software that work under the client-server model. The NFS share consists on a remote storage folder, being transparent to the client. A client running on Leon3 can access the files and folders as they were stored locally, and the OS does the job of fetching the data on the server through the NFS protocol. Further testing must be done in order to measure the impact of NFS in memory and bandwidth utilization.

The data input through UNIX sockets interface is performed, as previously stated, by two pieces of software. The communication occurs in a lower level than that of the NFS share, which means we have more control over the data transfer. The main advantage of this method is that it is possible to capture TS data using a "PenTV" USB receiver in the server and continuously send it through the socket as data is received.

3. System description

In such a complex project, there is significant interaction and dependency between the different modules, so the information flow between them is also relevant. That is why a dedicated bus is necessary. As previously stated in [1], the AMBA bus open standard has been chosen, and a demultiplexer has been attached to it as a new peripheral to the Leon3 processor.

3.1. Amba Bus

The AMBA bus specification, supplied by ARM Holdings, is implemented in the IP library GRLIB, distributed for free by Aeroflex Gaisler. Although several variations of AMBA are available, only AMBA 2.0 – AHB is used. AHB is a high performance, multi-slave bus that can withstand up to 16 masters. It is structured around two main elements: the arbiter and the address decoder. The former receives access calls from the master modules and must then decide which one of them the access priority will be given to; the latter is responsible for finding the selected slave in the memory map and routing the master's instruction to this slave.

The bus specific signals are exchanged inside dedicated records, which conceal the granting and response signals among other system configurations. This AHB specification also features a pipelined mode of operation with independent address and data phases and allows one data transfer per cycle.

- **The demultiplexer**

The demultiplexer receives from the microprocessor chunk of TS that contains an Elementary Stream (ES) of audio, video or data, as well as other network configuration information. It must then parse the input packet header, extract the ES and identify its nature in order to send it to the proper decoder.

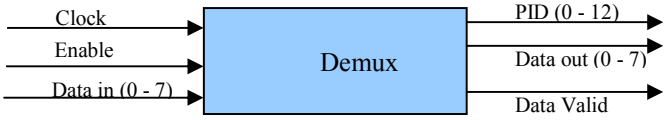


Fig. 3 – Demultiplexer interface

The demultiplexer designed for the Set-top Box, which is currently under development, has a typical DVB-SPI input, depicted in Fig. 3. Its output includes a Packet Identifier (PID) that indicates the nature of the ES (video, audio or data) as well as a serial byte stream with its respective validation signal.

4. System integration

The integration of the closed-caption decoder is a good example of the solutions developed so far. A new entity which performs the transition between the AMBA protocol and the input format of the demultiplexer has been created. A basic schematic of the signals that the interface handles is shown in Fig. 4.

The Amba2Demux interface reads all the signals from the bus output, including the slave selection signal during the address phase, indicating that the next data phase is meant for this slave (for the moment, the slave only handles writing operations, returning an error when a master tries to read from it. Although, a returning path might be added for monitoring purposes.). The interface can signal the bus when the transfer is completed or when an error has occurred**,** and is designed to occupy the bus as little as possible (typically for one cycle). A specially designed FIFO holds the 32-bit data, and a buffer allows it to output one byte at a time. This peripheral occupies only 256 bytes in the bus memory map, the minimal amount of space required by the processor.

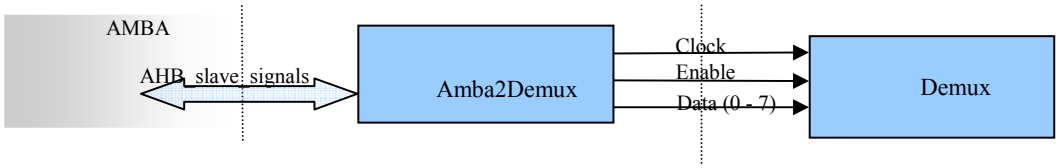


Fig. 4 – Data path from the processor bus to the demultiplexer

For the interface between the demultiplexer and the closed-caption decoder, a simple PID interpreter has been written, which receives the PID from the demultiplexer and, according to a previously defined table, indicates the packets intended for the closed-caption decoder. In the version under development, the necessary step of interpreting the PID signal will be taken inside the demux unit, making a selection signal directly available for each decoder.

The chain datapath of Fig. 4 has been written, integrated to the processor and simulated, with satisfactory results in the transmission of the incoming data, without losses. It was observed that the Amba2Demux internal FIFO may have an optimal depth of 4 words, meaning that the proposed interfaces do not represent any kind of bottleneck for the data flow, and respond properly to every possible situation that may occur during the different bus states.

5. Closed Caption manipulation

The demultiplexing tasks are followed be the audio, video and data decoders. The operation of the latter consists on decoding the information regarding the subtitles sent by the network, a work initiated by F. Caetano in [3], who built the filter and parser modules. The subtitles are divided mainly in control bytes and data bytes, as shown in [4]. So, it was built a module that first separate them and then translate the data bytes with the parameters set by the control bytes into a matrix, that indicates positioning in the TV screen, and a 16-bit data, that consists of information on which symbol and color that particular character indicates. The 16-bit data is composed by 8 bits indicating color from both background and caption and an 8-bit pointer to a ROM containing the symbols in pixels. The input stream and its decoded data are shown in Fig. 5 and 6.

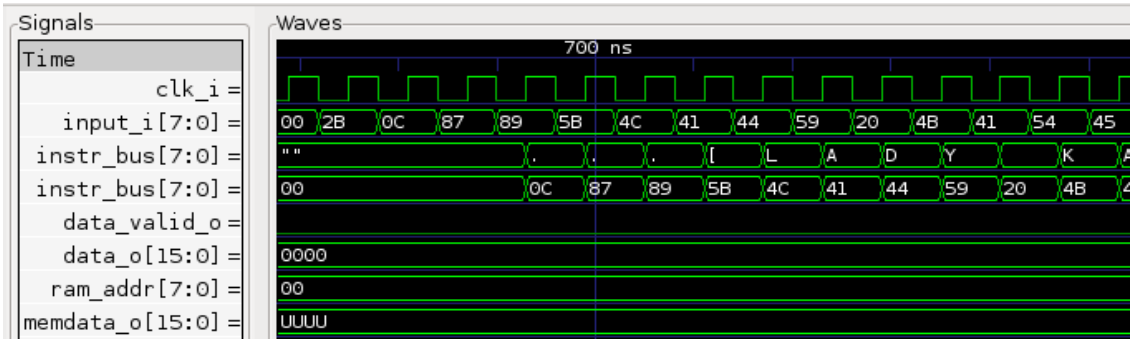


Fig. 5 – Separation between data and control bytes: “input_i” is the input stream and “instr_bus” (shown in both hexadecimal and ASCII) indicates the data bytes.

In Fig. 6 it is also shown that the caption data is stored into a RAM memory when the flag “data_valid_o” is set. The data is written in the position indexed by the matrix's row and column values (concatenated in the signal “ram_addr”), so that it stores in the right position for showing on the TV screen. Later on the project, the video output module will read from the RAM the parameters of color and the pointer to the symbol ROM

memory so it can display the video with the subtitles in it. An example of the reading of data is depicted in Fig. 7.

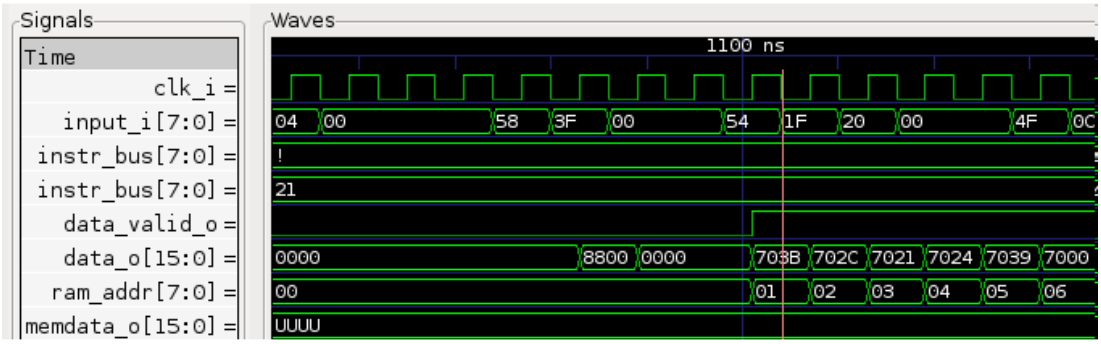


Fig. 6 – Decoding of the data in picture 1: "data_o" represents the 16-bit output and "ram_addr" consists of the positioning in the TV screen.

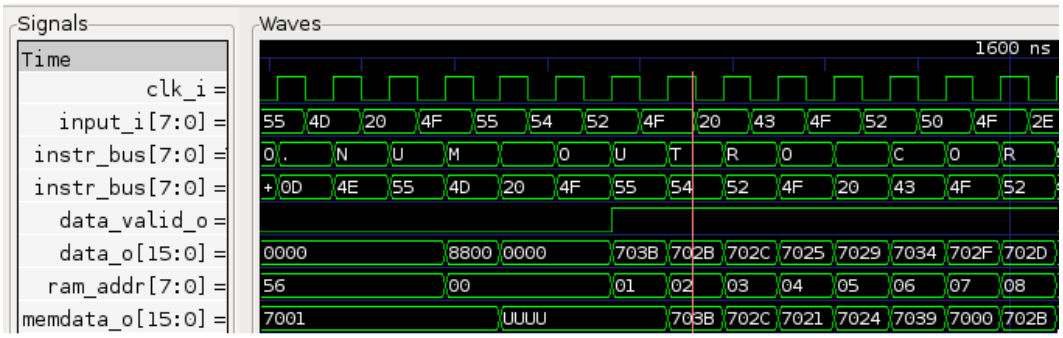


Fig. 7 – Reading of the RAM memory: "data_o" in picture 2 is read from the "ram_addr" position.

As for the video, the data decoded in the H.264 video decoder is stored in the external DDR memory, and the video output reads from the DDR in a line of pixels format, instead of the written format which is in macroblocks. Furthermore, in order for the video output from the set-top-box to successfully show the video multiplexed with the subtitles, it is necessary to access the external DDR memory containing the video data and to read both the RAM and ROM memories simultaneously. So, when the control indicates that a caption is required, the position of the caption in the video output is set to be the data from the memories: the color (which comes from the RAM) and character in pixels (which comes from the ROM indexed by the RAM less significant 8 bits). The video output and closed caption modules were tested independently and need yet to be fully integrated with the symbol ROM memory.

6. Conclusion

This paper described the efforts made towards the achievement of a full functional set-top box system for the Brazilian digital television system. Focusing on the system integration, an example of the complete path for the reception, demultiplexing and decoding of closed caption was also presented, with significant results on the system performance. The next steps include the integration of the other decoders in order to have the complete system ready for testing and further improvements and optimizations.

7. References

[1] J. Rech, L. Faganello and A. Susin, "A front-end development environment for the Brazilian digital television system", XXV South Symposium on Microelectronics (SIM 2010). SIM, 2010.

[2] Aeroflex Gaisler/ GRLIB, accessed in March 2011; http://www.gaisler.com/cms/

[3] F. Caetano, "Graphic Processor", dez. 2010; http://hdl.handle.net/10183/27976

[4] ABNT NBR 15606-1, "Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital, Parte 1: Codificação de dados", 2010. pp. 20-23.

Design Automation Tools 3

On Placement Coloring

Guilherme Flach, Marcelo Johann, Lucas Nunes, Ricardo Reis
{gaflach, johann, lpnunes, reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul (UFRGS)

Abstract

We present a new technique for visually comparing placement results through component coloring using partitioning and the Fiedler eigenvector. By coloring a placement result, we can quickly get insights on how to improve a placement tool and understand what matters and what does not to the placement quality. As a side effect, our coloring scheme may generate beautiful images, making this paper about both artistic and scientific experiments.

1. Introduction

Placement is the task responsible to place components evenly without overlap over the circuit area so that wirelength and other parameters are targeted. Put in those words, placement seems to be a simple task, however, due to the huge number of components that should be processed at same time, placement is challenging. Indeed placement is a NP-Complete problem [1] and hence practical techniques do not try to solve it optimality.

The large number of components also imposes a difficulty when comparing placement results from different placement tools. Although other metrics as congestion and routability are being introduced, half-perimeter wirelength (HPWL) still being the one most used to compare placement results. However HPWL only can indicates that one placement result may be assembled with more or less wirelength, but does not answer an important question for research: why one placement result is better than other?

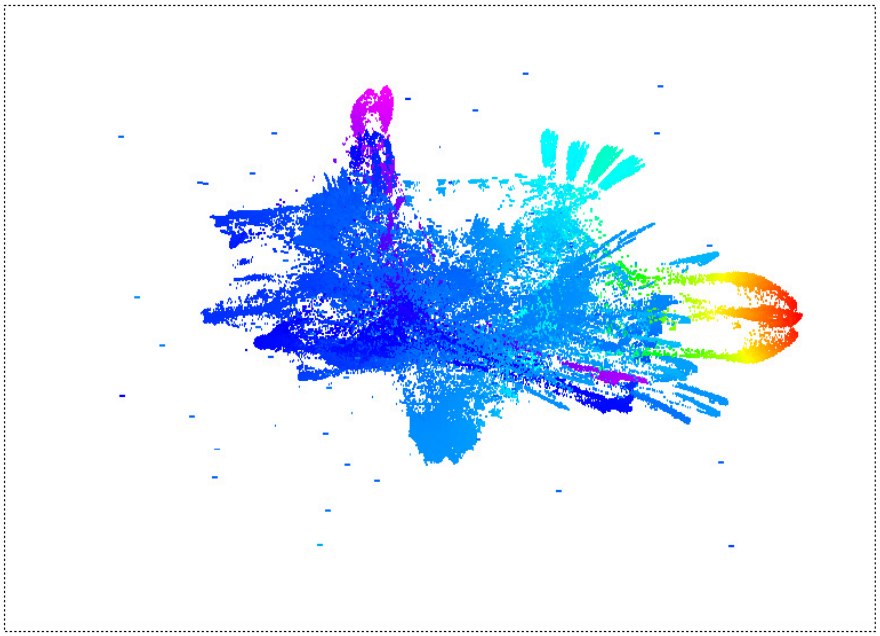


Fig. 1 – An example of placement coloring using Fiedler eigenvector.

To try to answer that question we developed two schemes for component coloring using partitioning [2] and the Fiedler eigenvector [3]. Wisely coloring components allows to compare visually different placement results and get insight on why a placement results is better than other.

2. Placement Coloring

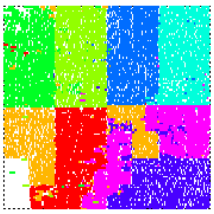
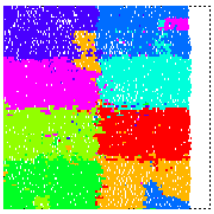
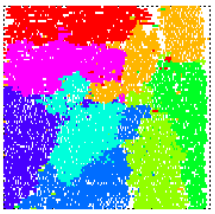
It is known that placement and partitioning are in some extent correlated [2]. This correlation suggests that cells in a same partition tend to be placed in such a way they are clustered. This is the main idea and motivation behind our coloring schemes. With that in mind, we then have selected two algorithms to color components: (1) min-cut partitioning and (2) Fiedler eigenvector.

2.1. Coloring using Min-Cut Partitioning

A min-cut partitioner attempts to minimize the number of hyperedges which nodes belong to different partitions keeping partitions balanced.

A circuit netlist is directly mapped to a hypergraph and so any partitioner that supports hypergraphs can be used. In this work, we use hMetis [4]. Coloring a placement using a min-cut partitioning results is straightforward: to each partition a different color is set and all cells belonging to a partition are painted with the partition color. An example of min-cut partitioning coloring is presented in Table 1 where the netlist was partitioned in 8 balanced partitions.

Tab.1 - Min-Cut coloring scheme. Images for ibm01 benchmark using 8 partitions.

Capo	Dragon	FastPlace 3	mPL
HPWL: 1.90689e+06	HPWL: 1.87986e+06	HPWL: 1.72773e+06	HPWL: 1.6185e+06
			

2.1.1. Experimental Results

This coloring scheme was then used to colorize ISPD 2002 benchmarks set [10] using the placement results from four placers: Capo [6], Dragon [7], FastPlace 3 [8] and mPL [9]. Each benchmarks-result pair was colored using 2, 4, 8, 16, 32, 64 and 128 partitions.

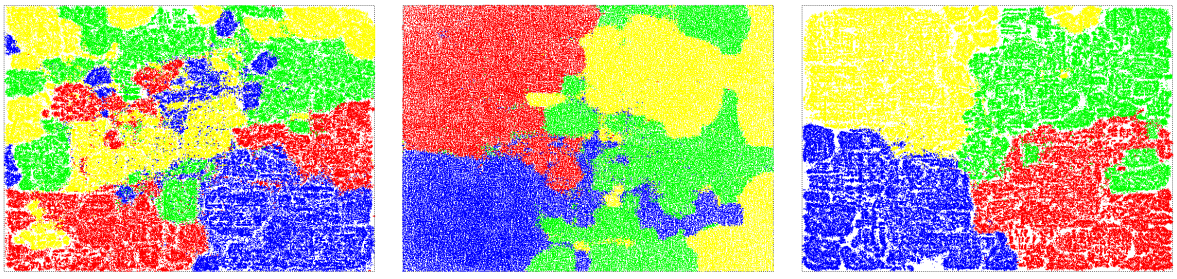
Table 1 presents the bechmark-result pair for *ibm01* benchmark colored using 8 partitions. We can notice clearly that Capo and Dragon use partitioning based techniques in their placement flow as the clusters are well-shaped. Analytical placers do not have this pattern and tend to form amorphous clusters. This patterns of well-shaped and amorphous clusters remains through all experiments using these placers.

Another tendency in the experiments indicates that amorphous clusters implies in better HPWL results. For this, we may conclude that, although placement and partitioning are related, imposing a shape to the partition cluster may harm the result quality. Both FastPlace and mPL do not use partitioning, but as the experiments indicate the partitioning is indirectly take into account.

2.1.2. Case Study: Improving Our Placement Tool Quality using Partitioning

When developing our placement tool, we were able to beat most of state-of-the-art placer runtimes, but were unable to match the same HPWL quality. Using the min-cut partitioning coloring scheme we realized that our placer failed to capture global information on how cluster must be placed. This problem can be noticed in Figure 2a where clusters formed by our prior placer are less concentrated than those formed by FastPlace 3 as shown in Figure 2b.

To insert global view in our placement tool, we used partitioning itself. As we have noticed previously, imposing a hard shape for partition clusters could harm the result quality, so we use the partitioning of cells only to impose an initial position for each cell.



(a) (b) (c)

Fig. 2 – Placement coloring. (a) Our prior placer result; (b) FastPlace 3 result; (c) Our final placer result using partitioning for initial placement.

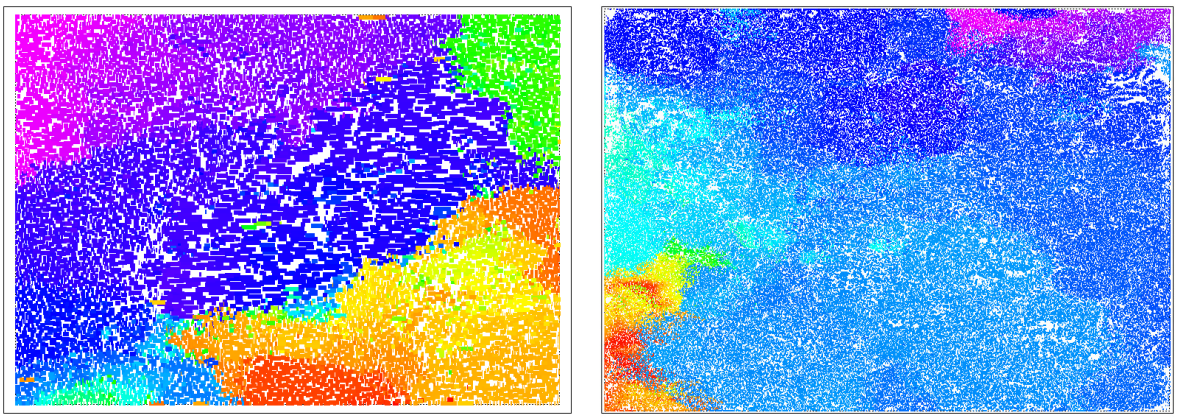
The initial placement is done placing cell in the same partitioning in one of the four placement area quadrants instead of putting all cells in the middle of the circuit. This increased the total run-time of the

placement, but allowed our placer to provide state-of-the-art results. The result after partitioning is presented in Figure 2c.

2.2. Coloring using Fiedler Eigenvector

The Fiedler eigenvector provides a ordering of nodes of a connected graph and can be used for partitioning [3]. The great advantage over min-cut partitioner is that Fiedler method provides a way to measure how close and how far in connectivity terms a cell is from other creating a *continuous* partitioning. The disadvantage is that Fiedler method only works on graphs so the netlist, a hypergraph, must be casted to become a graph.

The Fiedler eigenvector is the eigenvector associated to the second smallest eigenvalue of the Laplacian matrix of a graph. The Laplacian matrix represents the connectivity of the graph, hence of the circuit. Each cell is associated to an element in the eigenvector. In this work, we use the method described in [5] for computing the Fiedler eigenvector.



(a) (b)

Fig. 3 – Fiedler coloring scheme: (a) ibm01; (b) ibm18.

To color the cells we first normalize the Fiedler eigenvector such that the smallest value in the eigenvector is mapped to zero and the largest value is mapped to one. After we simply paint the cell picking up the color in a color temperature pallet respective to the element value in the vector associated to that cell. This way, cell with values bellow 0.5 are mapped to cold colors and the remain ones to hot colors.

2.2.1. Experimental Results

Again, this coloring scheme confirmed the fact that placement and partitioning are correlated problems. All images generated from results provided by good placer show same-colored cells clustered together. Here another interesting effect is noticed hot colors and cold colors tend to appear clustered as can be seen in Figure 3. This relates with the fact that Fiedler eigenvector can be used as a approximation for min-cut bi-partitioning.

Fiedler coloring brings another very useful property for analytical placer. These placers start with cells concentrated in the center of the placement area and iteratively spread cells until they are evenly spread over the circuit. In the spreading process, many placers create tentacles as the mass of cell is spread from the very concentrated initial state. This effect can be seen in Figure 1 where tentacles are formed by cluster of same-colored cells. Therefore Fiedler coloring scheme can predict cell clusters which are more easily expelled from the mass of cell. Such property is still in study and it was not used in our placer till now.

3. Conclusions and Future Work

In this paper we developed and presented some experimental results that supported placement coloring as a technique to compare visually placement results as well as a tool to get insights on how to improve placement quality. Our coloring schemes are created using the min-cut partitioning and the Fiedler eigenvector. Both coloring schemes confirmed that placement and partitioning are indeed correlated problems, since same-colored components tend to be placed together. However it is shown that imposing a hard shape for cells of a partition may harm the placement quality. This indicates that partitioning can be used as a guide for placement, but not as a sole solution.

Besides technical contributions presented by coloring schemes, as a side effect, beautiful images are generated. This make the coloring technique useful for both science and, in some extent, for arts.

Undoubtedly, this is an underdevelopment work and many more experiments can be performed. A interesting experiment to be done is to study the relation of hierarchical partitioning and the min-cut partitioning. Also new coloring schemes can be explored.

4. References

[1] Donath, W. E. 1980. Complexity theory and design automation. In Proceedings of the 17th Design Automation Conference. pp. 412-419.

[2] M. A. Brever,"Min-Cut Placement", Journal of Design Automation and Fault Tolerant Computing, Oct., 1977, pp. 343-362.

[3] A. J. Seary and W. D. Richards. Partitioning networks by eigenvectors. In Proceedings of the International Conference on Social Networks, volume 1, 1995.

[4] G. Karypis and V. Kumar. "hMETIS 1.5: A hypergraph partitioning package". Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the Web at URL http://www.cs.umn.edu/~metis.

[5] N. P. Kruyt. "A conjugate gradient method for the spectral partitioning of graphs". Parallel Computing, 22(11):1493–1502, 1996.

[6] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov. Capo: Robust and scalable open-source min-cut floorplacer. In ACM/SIGDA International Symposium on Physical Design (ISPD), pages 224–226, 2005.

[7] T. Taghavi, X. Yang, and B.-K. Choi. "Dragon2005: Large-scale mixed-size placement tool." In ACM/SIGDA International Symposium on Physical Design (ISPD), pages 245–247, 2005.

[8] N. Viswanathan , Min Pan , C. Chu. "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control". Proceedings of the 2007 Asia and South Pacific Design Automation Conference, p.135-140, January 23-26, 2007.

[9] T. Chan, J. Cong, and K. Sze. Multilevel generalized force-directed method for circuit placement. In ACM/SIGDA International Symposium on Physical Design (ISPD), pages 185–192, 2005.

[10] N. Viswanathan , C. C. Chu, "FastPlace: efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model", Proceedings of the 2004 international symposium on Physical design, April 18-21, 2004, Phoenix, Arizona, USA.

A Test Environment for Validation of Subthreshold and Leakage Current Estimation Method in CMOS Logic Gates

[1]Kim A. Escobar, [2]Paulo F. Butzen, [1]André I. Reis, [1]Renato P. Ribas
{kaescobar,pbutzen,andreis,rpribas}@inf.ufrgs.br

[1]Institute of Informatics, UFRGS, Porto Alegre, Brazil.
[2]Center for Computational Science, FURG, Rio Grande, Brazil

Abstract

Several analytical methods for estimating the subthreshold and gate oxide leakage currents in CMOS circuits have been presented in the literature. However, such currents are strongly dependent on the fabrication process targeted, and the methods present different prediction data accuracy according to the profile of logic gates, in terms of transistor arrangements, stacked devices, and gate sizing. In this work is presented a test environment developed to evaluate and validate the method of subthreshold and gate leakage current estimation for general CMOS logic gates proposed in [1]. PTM 32 nm CMOS parameters were considered to demonstrate the proposed test environment.

1. Introduction

Computer-aided design (CAD) tools are quite necessary on integrated circuit (IC) digital design [4] due to the high complexity of current VLSI systems. These tools are used for different purposes, from the analysis processes yield to the automatic construction of circuit layout. Two general types of CAD tools for circuit analysis could be identified: (a) the ones that applies numerical methods and (b) the other ones based on analytical models. Nowadays, an important circuit evaluation task to be embedded in the IC design flow is related to the static power consumption estimation, including the subthreshold and gate oxide leakage currents.

The reduced dimensions of advanced MOS transistors at nanometer scale have increased the static consumption. Such increasing in the static consumption is caused by static currents like subthreshold and leakage currents [2]. Because of these, more accurate methods to estimate such static currents in CMOS logic gates are needed. Several methods for subthreshold and leakage currents estimation have been proposed in the literature [1]-[7]. One of the main challenge in developing these kind of methods and algorithms is the wide range of existing manufacturing processes, which are somewhat different from each other, presenting many particular effects that can affect the static currents behavior, like the 'inverter narrow width effect' (INWE) [8]. Thus, it is necessary to have available a test environment to verify, evaluate and validate the estimator for each new process targeted.

This paper describes such a kind of test environment developed to verify the accuracy of the subthreshold and gate leakage current estimation method proposed in [1]. This method has been included in the SwitchCraft tool [9]. The method presented in [1] intends to be independent of technology. It uses the configuration file containing specific parameters related to a pre-characterization of the addressed technology. This paper focuses on showing the wealth of data that can be obtained from the environment. The proposed environment presents specific pre-defined sets of gate netlists choose appropriately for different type of analysis.

Initially, in Section 2 is presented the technical background in order to understand the main subject of this paper. The methodology used in the environment to validate the estimator is described in Section 3. Also, in Section 3, is presented briefly the execution flow of the enviroment, inlcuding all test steps. Some results and analysis are inferred using some output examples in Section 4. The main conclusions are outlined in Section 5.

2. Static Current Modeling (Technical Background)

With the scaling of transistors, the static currents are growing in relation with the total current. These static currents comprise mainly the subthreshold current, gate-oxide tunneling current and reverse-bias pn-junction [1], as illustrated in Fig. 1. Also, with less significance to normal mode of operation, there are the gate induced drain leakage (GIDL) and punchthrough current [10]. The subthreshold current occurs when the transistor is operating in weak inversion region, being that this current component increases exponentially with the scaling of transistor threshold voltage [1]. The gate-oxide tunneling current happens because of the scaling of oxide thickness, even with some techniques to reduce this current, like high-K dielectrics [11]. It is important to consider such current in the calculation of the total static current in logic gates. The other currents are not taken into account in this work.

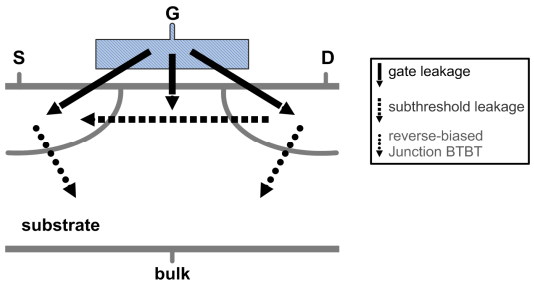


Figure 1 – Major leakage mechanisms in MOS transistor.

There are several analytical models for estimation of these static currents proposed in the literature. There are some models that take into account only the subthreshold current component, like the models proposed in [2], [3] and [4]. Moreover, a method to estimate the gate-oxide tunneling current is presented in [5]. Furthermore, there are some models that predict both currents and their interaction given in [1] and [6]. The model proposed in [7] takes into account also the BTBT leakage.

The environment proposed in this paper is intended to validate de estimating method proposed in [1]. One of the main features of this model is to be process independent. Another feature is that DIBL and body effects are considered in this model, which gives it a good precision on the leakage estimation task. In spite of this features, there are some effects not actually covered by the model due to the large variety of existing processes, as the inverted narrow width effect (INWE) that happens only in some kind of technologies, like SOI-based processes. In order to evaluate how much these particular effects can impact the estimation accuracy, this test environment is proposed herein.

3. Test Environment

This proposed environment is modular and can be divided in five parts, as illustrated in the flow diagram in Fig. 2: (1) process analysis, (2) transistor scaling analysis, (3) transistor stack analysis, (4) general analysis of logic gates composed only by series-parallel (SP) networks; (5) analysis of logic gates with non-series-parallel (NSP) transistor arrangements. The main tasks (or steps) executed in the analysis flow are: extraction of process parameters for specific estimator configuration; leakage estimation through Java program; electrical simulations using spice decks; and generation of results data (tables and graphs).

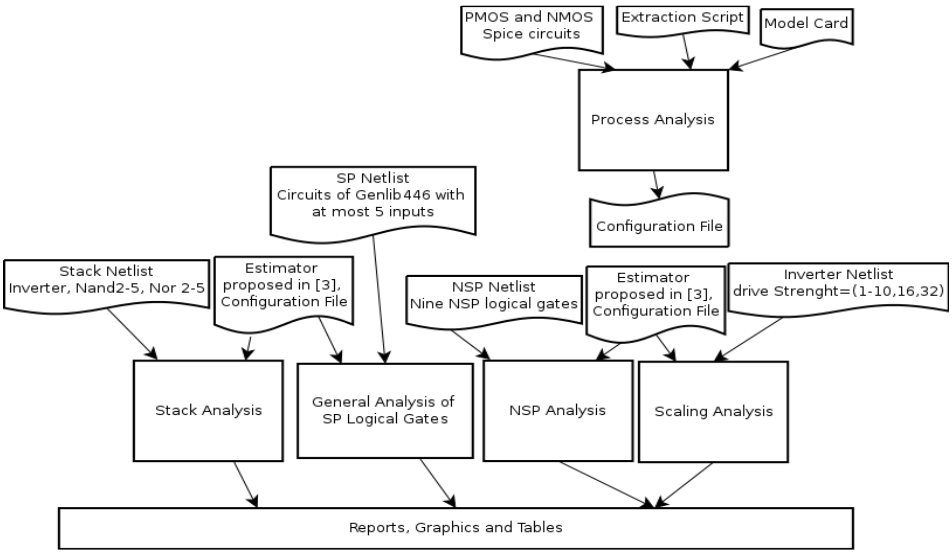


Figure 2 – Test environment flow and analysis available.

The first evaluation part corresponds to the process parameter extraction task. The input data is only the model card of the target process under evaluation. The extractor is basically a shell script program that works with regular expression and some text editors, like ‘sed’ and ‘cut’, to create the configuration file required by the leakage estimator. There are two additional programs that generate the configuration file to a range of device operating temperatures and power supply. These ones provide data tables to be used by gnuplot script in order to generate plots for analysis of power supply and temperature variations versus a specific desired parameter for analysis.

The other analysis provided by the environment presents similar tasks and steps. First of all, the static currents are estimated using the Java program developed for this purpose. As mentioned, it represents the execution of the method proposed in [1]. The input data is the circuit netlist described in Spice format. The output data is the current estimation of all possible input vectors and the computation time for that. In the next

step, electrical simulations (HSPICE) are executed using the same set of circuits. The output results are comparative tables with several information, such as average, minimum and maximum leakage currents (predicted analytically and obtained from simulations), as well as the error involved.

The different evaluations provided by the test environment are:

Process analysis – Among the values obtained in the process parameters extraction are represented in the configuration file are the subthreshold and gate currents related to the minimum transistors width allowed by the target process. This values are called, respectively, 'is0(N/P)' and 'ig(on/off)(N/P)'. Moreover, DIBL and body effect are represented by the parameters 'n(N | P)' and 'y(N | P)', respectively.

Transistor scaling analysis – The objective is to evaluate the behavior of the model with respect to the scaling of inverter gates, i.e., by increasing the drive strength of this logic gate. It is expected that the minimum inverter size electrical behavior fits well with the predicted leakage value, since the parameters used in the method configuration is based on these PMOS and NMOS transistors sizing. The differences observed in this analysis correspond to the general inaccuracy of the estimation method.

Transistor stack analysis – The behavior of the model is evaluated with respect to the transistor stack effect. The stack effect occurs when two or more transistors are placed in series and it presents a particular impact in the subthreshold current decreasing, affecting the total static current value. It is expected that, even with an increasing inaccuracy of the method, the total estimation error reduces significantly due to the reduction of absolute values of currents involved. The circuits used for this analysis are the minimum inverter (reference), NAND and NOR gates with up to 5 inputs.

General analysis of SP logic gates – In order to evaluate the general behavior of the leakage estimation method, a set of series-parallel complex CMOS gates have been considered. They were taken from the 'genlib_44-6' list for functions with up to 5-inputs [12]. This kind of CMOS logic gate is also covered by other estimation methods and they are quite representative in terms of the cells usually instantiated in digital circuits.

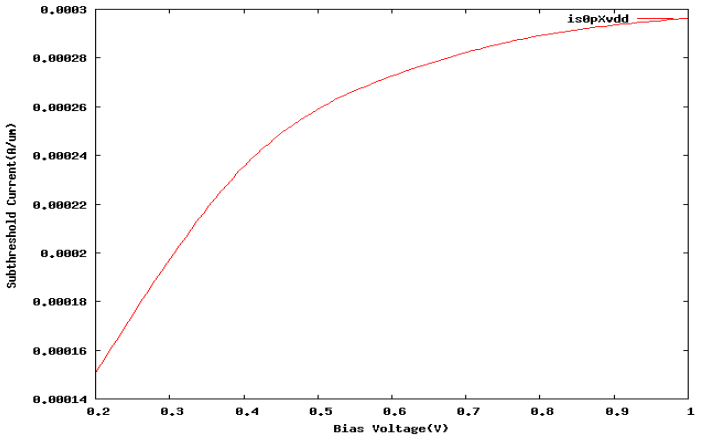
Analysis of NSP logic gates – Differently from others, the methods proposed in [1] is the only one that is suitable for this kind of logic gate. It was created a set of nine NSP logic gates, with up 5-inputs, whose SP counterparts represent gates with more transistor count.

4. Experimental Results (Environment Validation)

Some experimental results are shown here to illustrate and validate the proposed test environment. The PTM 32 nm CMOS parameters were considered to demonstrate these results. In Fig. 3a is shown the principal parameters of the configuration file of the estimator. As mentioned before, the process analysis could be exploited to evaluate the behavior of some parameters in relation to power supply and operating temperature. An example if illustrated in Fig. 3b.

```
************************************
*** Arquivo de configuração para ***
***     tecnologia de ptm_32n     **
************************************
Vdd= 1
Vinc= 2.6650E-01
OutVdrop= 7.3692E-01
VT= 0.0258
VBp= 1
VBn= 0
n= 1.45
* ---------------------------------
* parametros do pmos
np= 2.6361E-01
yp= 1.5424E-01
is0p= 2.9635E-04
igon0p= 1.2604E-02
igoff0p= 6.1541E-03
konp= 4.5671E+00
koffp= 4.5460E+00
* parametros do nmos
nn= 2.0305E-01
yn= 1.8533E-01
is0n= 1.4862E-03
igon0n= 6.6845E+00
igoff0n= 1.4095E+00
konn= 5.8911E+00
koffn= 5.5355E+00
* ************************************
```

(a)



(b)

Figure 3 – Process analysis: (a) configuration parameters, and (b) supply voltage variation.

Another interesting result demonstrated here is the second type of analysis, described above, related to the transistor scaling observed in inverter gates with different drive strengths. As show in Fig. 4, the inaccuracy of the method increases when the transistor widths become larger. It suggests that there is a room for future investigations, and further development must be done to improve the leakage method proposed in [1].

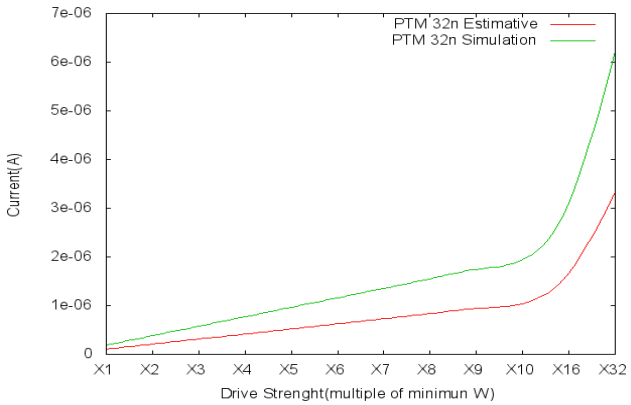


Figure 4 – Behavior of total static current in CMOS inverter.

5. Conclusions

A test environment for evaluation and validation of the subthreshold and gate oxide estimation method proposed in [1] has been described in this paper. It is quite useful to verify the validity of the leakage method in different CMOS processes, as well as in different logic gate topologies. This automatic environment provides a fast and useful analysis before applying the target estimation method in new processes. The results suggest that the evaluated leakage method can be further improved. Other SMOC process are being evaluated identify general inaccuracies and the ones specific related to particular technologies.

6. Acknowledgments

Research partially funded by Nangate Inc. under a Nangate/UFRGS research agreement, by CNPq and CAPES Brazilian funding agencies, and by the European Community's Seventh Framework Programme under grant 248538 - Synaptic.

7. References

[1] BUTZEN, P. F.; da ROSA Jr., L. S.; CHIAPPETTA FILHO, E. J. D.; REIS, A. I.; RIBAS, R. P., Standby power consumption estimation by interacting leakage current mechanisms in nanoscaled CMOS digital circuits. Microelectronics Journal, v.4, n.4, p.247-255, Apr. 2010.

[2] GU, R. X.; ELMASRY, M. I., Power dissipation analysis and optimization of deep submicron CMOS digital circuits. IEEE J. Solid State Circuits, v.31, n.5, p.707-713, May 1996.

[3] NARENDRA, S. G.; CHANDRASKASAN, A., Leakage in Nanometer CMOS Technologies. New York: Springer, 2006.

[5] LEE, D.; KWONG, W.; BLAAUW, D.; SYLVESTER, D., Analysis and minimization techniques for total leakage considering gate oxide leakage. Proc. DAC, 2003, p.615-621.

[6] YANG, S. *et al.* Accurate stacking effect macro-modeling of leakage power in sub-100nm circuits. Proc. Int'l Conf. of VLSI Design, 2005, p.165-170.

[7] MUKHOPADHYAY, S. *et al.* Gate leakage reduction for scaled device using transistor stacking. IEEE Trans. on VLSI Systems, v.11, n.4, p.716-730, Aug. 2003.

[8] PACHA, C *et al.* Impact of STI-induced stress, inverse narrow width effect, and statistical Vth variations on leakage currents in 120 nm CMOS. Proc. ESSDERC, 2004, p.397-400.

[9] CALLEGARO, V.; MARQUES, F. S.; KLOCK, C. E.; da ROSA Jr., L. S.; RIBAS, R. P., REIS, A. I., SwitchCraft: a framework for transistor network design. Proc. SBCCI, 2010.

[10] AGARWAL, A *et al.,* Leakage power analysis and reduction: models, estimation and tools. IEE Proceeding, v.152, n.3, p353-368, May 2005.

[11] GUSEV, E. *et al.,* Ultrathin high-K gate stacks for advanced CMOS devices. IEDM Technical Digest, p.451-454, 2001.

[12] Sentovich, E.M. *et al.*, "SIS: A system for sequential circuit synthesis." Tech. Rep. UCB/ERL M92/41. UC Berkeley, 1992.

CAD Tool for Switch Network Profiling

Carlos E. Klock, Vinicius Callegaro, André I. Reis, Renato P. Ribas
{ceklock,vcallegaro,andreis,rpribas}@inf.ufrgs.br

PPGC, UFRGS, Porto Alegre, Brazil.

Abstract

This paper presents a CAD tool to analyze switch networks and generate profiles to detect possible issues that may affect the design of integrated circuits, especially those related to routing congestion. The analysis of a switch network is useful to find out information about the general structure of the network and its layout, for instance, to predict the behavior in terms of physical area and signal routing congestion. The experimental results show that the proposed tool can identify undesired diffusion separations and possible routing congestion. Specific applications include RTL (regular transistor layout) and standard cells.

1. Introduction

The automation of layout generation is playing an important role in cell design. The complexity of making well elaborated layouts with a high density of transistors is a motivation to design CAD tools to evaluate and automate the process of layout generation [1]-[5].

As the number of transistors increase, more wires are needed to connect the signals in the circuit, resulting in more density and, consequently, more signal routing congestion [3][4]. Furthermore, the size of the transistors has decreased but the dimensions of the wires are not actually decreasing at the same rate [3]. The density of the wires in an integrated circuit (IC) defines the complexity of the signal routing task. This represents an important issue in IC design because wires occupy a significant area in the final physical layout. In general, the use of metal layers and contacts is optimized by reducing the number of signals to connect. The number of contacts is of special concern because they can represent a significant penalty related to area. To minimize these problems some authors [1][6] proposed methods for minimum-width transistor placement. These methods try to avoid the implications involved with diffusion gaps (separations) by pairing N and P transistors by the common gate input signal. The idea is that less diffusion gaps lead to fewer contacts and, consequently, fewer wires.

In this paper, we propose a CAD tool to identify the aforementioned issues. To do so, the tool analyzes the graphs and the Euler paths of logic gates. Logic gates are composed of switch networks. Thus, the goal of this tool is to profile switch networks and to estimate how many connections and switches (transistors) exist in an IC. This is important to realize if a specific network needs special attention related to wiring congestion. A switch network here means that there is no concern about physical aspects in a first moment, because the profile is generated from the logical data structure represented by a graph model, where the edges are the switches and the vertices are the nodes (nets) of the network [6]. Hence, switch networks and logic gates are treated in a high level of abstraction by the proposed CAD tool. The tool presented here is going to be incorporated to the SwitchCraft environment [5], which provides a set of modules for automatic generation and analysis of switch networks and logic gates.

The next sections are organized as follows. First of all, in Section 2, we describe the technical background involved with this work. In Section 3 are described the methods that are employed by the tool to analyze the switch networks and generate the profiles. The next sections present applications and experimental results. Finally, the conclusions of this work are outlined.

2. Technical Background

Graph models, as stated in Section 1, are high level abstract representations of transistor (switch) networks [6]. Switches are the edges of the graph. Nodes (nets) are the vertices of the graph, as seen in Fig. 1. From the graph model we can extract much information. The most important one is the Euler paths.

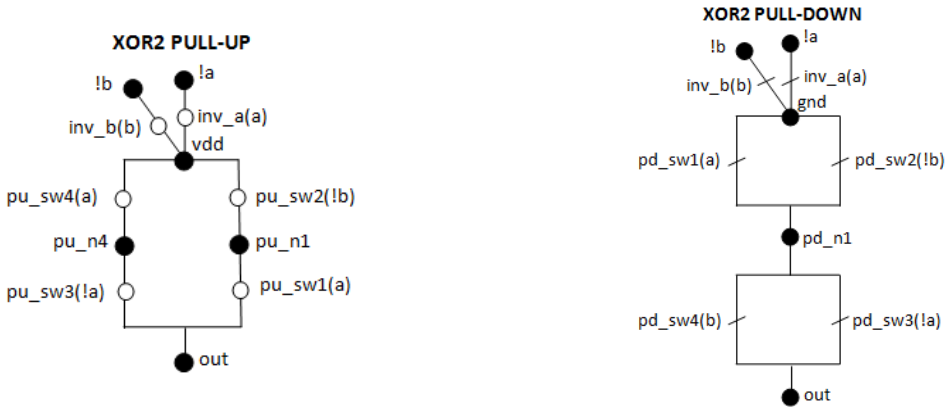
An Euler path, or Eulerian path, is a walk on the edges of a graph which uses each graph edge exactly once. An undirected graph contains an Eulerian cycle if and only if it is connected and all vertices are of even degree. A graph that contains an Eulerian cycle is called Eulerian graph. An undirected graph contains an Eulerian path if and only if it is connected, and all but two vertices are of even degree. In this case, these two vertices represent the beginning and ending points of any path [7][8]. A graph that contains an Eulerian path is called semi-Eulerian graph.

A connection is defined by repeated nodes in the Euler path. If a node appears more than once in the Euler path then there is one connection (net), i.e., they must be connected in the physical layout. For example, two nodes *n1* and three nodes *n2* equals 2 connections. No repeated nodes equal zero connections.

An external node is a node that does not belong to the network, such as power supply (vdd), ground reference (gnd) and output node in logic gates. Internal nodes, by the way, are the nodes that belong to the

network. It is understood here that a CMOS logic gate is composed by two logic planes (i.e., two networks), as illustrated in Fig. 1. They are the pull-up plane that connects the value '1' source to the output node of the gate, and the pull-down plane that connects the value '0' source to the output node of the gate. An unconnected node is a node that is external or is not repeated in the Euler path.

A break is a special edge that represents a permanent off-switch, added to *eulerize* a graph when it does not have an Euler path. A break may also be a separation of the diffusion in the layout of a circuit.



(a) **(b)**

Figure 1 – Example of pull-up (a) and pull-down (b) switch networks of an exclusive-OR gate.

3. Proposed Methods

The proposed profiling tool aims to generate information about a switch network that includes:

- number of variables: the total number of positive polarity variables in the networks (in the logic gate);
- input inverters: the number of input inverters in the logic gate, related to the number of negated inputs;
- output inverter: indicates if there is an inverter at the output of the logic gate;
- size of the Euler paths: the number of edges of the Euler paths at each plane (pull-up and pull-down);
- number of connected nodes: number of internal nodes with connection at each plane;
- number of unconnected nodes: nodes that are not repeated in the Euler path or external nodes;
- path beginning at power node: true if starts at vdd or gnd (otherwise false);
- number of breaks: the number of diffusion gaps or disabled switches.

To build the graphs that are used in this paper, we start from a switch network description or from a Boolean expression. The SwitchCraft environment [5] is employed to generate the graphs. The graphs contain all the required information to generate the Euler paths with a path finding algorithm.

Note that to find Euler paths of a given graph it must be Eulerian or semi-Eulerian. If is not possible to find an Eulerian path or Eulerian cycle then the graph must be *eulerized*. The profiling tool checks the condition of the graph to find out if is possible to find an Eulerian path or cycle. If not possible, one or more breaks must be added to the graph, until only two vertices have odd degree. In the case of switch networks, it is desired to add the minimum number of breaks to the graph, what is done in our algorithm of *eulerization*. To calculate the minimum number of breaks we must keep in mind that we want to have two vertices of odd degree after the process of *eulerization*. Thus, the final number of breaks is equal to the number of odd degree vertices divided by 2 and minus 1.

When the graph is *eulerized* we must find the Eulerian paths. There are many algorithms described in the literature to find Euler paths [7][8]. The profiling tool uses one of these algorithms to find the Euler paths and selects one of the Euler paths that were found. The Euler path to be selected will be the one with the less number of connections and, if possible, starting at a power node and ending at an output node (or vice-versa).

3.1. Single Switch Network and Internal Connections

Looking for a single switch network the tool extracts the Euler paths and selects one of them to start the profiling mechanism. When evaluating the Euler path alone, the profiler extracts information like the number of connected and unconnected nodes, size of the Euler paths and number of breaks. We want to find the path with the smallest amount of connections to achieve a circuit with less wire routing congestion [3].

With this information is possible to detect the wire behavior and possible crossings that may happen if there are interchanging repeated nodes, and thus estimate the height of the cell. For example, suppose that [X1, X2, X3, X2, X3, X4] is an Euler path, where each Xn is a node. The two X2 nodes are connected, as well as the X3 nodes. Considering this Euler path, the minimum height (or number of rows) of the cell represented by this network will be 2.

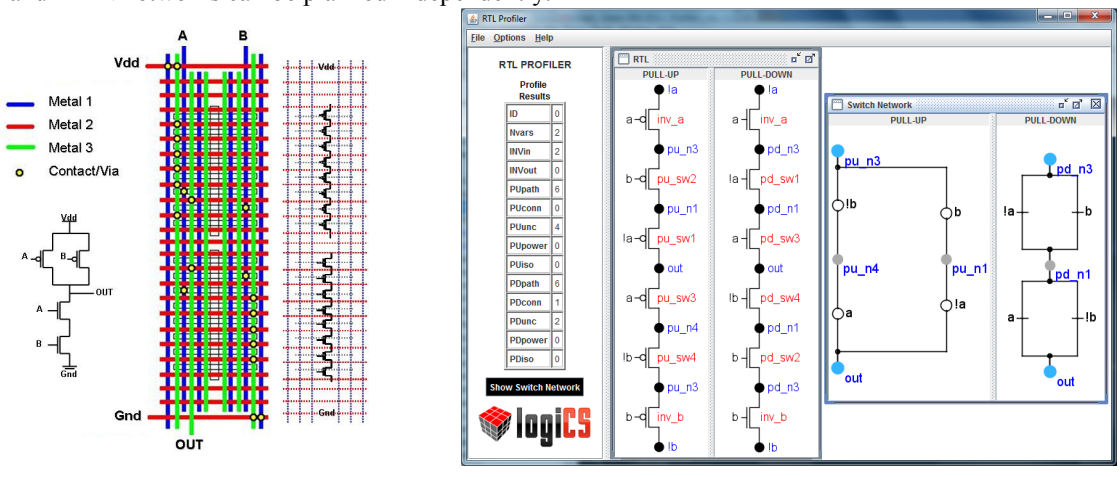
3.2. Alignment of Two Switch Networks

In [1] and [6] the authors define methods for the alignment of switch networks by matching the Euler paths of pull-up and pull-down planes. This avoids the usage of extra metal wires to interconnect the two planes. In fact, this is a desirable solution for CMOS styles that use topologically complementary networks, where the number of transistors in the pull-up is the same of the pull-down. But there are other types of networks that are not complementary or cannot be aligned because they are not placed horizontally. This is the case of VCTA networks [9], for example that is presented in the next section. For such non-complementary networks, the tool proposed in this paper is of special interest because the routing congestion will be more critical.

The Euler paths of both pull-up and pull-down networks are very important in the process of layout generation because the number of breaks (separations of diffusion areas), the number of connections and the proper alignment of pull-up and pull down are directly related to the quality of the Euler paths that are selected. With good paths and good alignment of both pull-up and pull-down less wires and contacts are needed [1][6], but this subject is beyond the scope of this paper and may be implemented as a future work.

4. Applications

The proposed tool is quite useful for logic gate design. One interesting application is the new regular transistor layout (RTL) strategy applied to improve the parametric yield of circuits in advanced nanometric CMOS processes. A particular tool version has been provided to evaluate the implementation of logic gates over the via-configurable transistor array (VCTA) approach [9]. The VCTA presents a basic structure composed by two transistor regions: a PMOS region for building the pull-up network and a NMOS region for pull-down plane implementation, as illustrated in Fig. 2a. The transistor regions are physically disposed in line (in a single column), differently from conventional standard cell and mask-programmable gate array linear matrix technique, in which both pull-up PMOS (PU-P) and pull-down NMOS (PD-N) are organized as rows that run in parallel. As a result, while in standard cells and gate arrays the polysilicon wires of two transistors controlled by the same signal should be aligned to optimize the layout construction, in a VCTA structure both PU-P and PD-N networks can be planned independently.



(a) **(b)**

Figure 2 –Regular transistor layout (RTL): (a) VCTA approach [9], and (b) RTL CAD profiler.

In order to build the PU-P and PD-N logic planes on the VCTA structure, the Euler path of each plane must be obtained according to the transistor network. In a second step, the found Euler paths have to be assigned (or physically mapped) to the transistor columns. In the case where the Euler path in a plane has more transistors than the number of available devices in a column of the array, the Euler path must be broken in two or more sub-paths to be built in more than one VCTA unit. Conversely, when the Euler paths of a logic gate are shorter than the number of transistors available in the VCTA cell, the remaining devices can be isolated by connecting PMOS gate terminals to 'vdd' and NMOS gate terminals to 'gnd'. Note that Euler paths may require breaks in the transistor networks. Specific CAD tool were developed for this application, as shown in Fig. 2b.

5. Experimental Results

Assuming that we have a 2-input exclusive-OR (XOR2) network, like the one illustrated in Fig. 1, we may achieve the following Euler paths:

- Pull-up: !a, vdd, pu_n4, out, pu_n1, vdd, !b (nodes)
  inv_a, pu_sw4, pu_sw3, pu_sw1, pu_sw2, inv_b (edges)
- Pull-down: !a, gnd, pd_n1, out, pd_n1, gnd, !b (nodes)
  inv_a, pd_sw1, pd_sw3, pd_sw4, pd_sw2, inv_b (edges)

Table 1 below presents some information that can be extracted from the Euler paths, and from the XOR2 graph, depicted in Fig.1, we have:

- number of variables: 2
- input inverters: 2
- output inverter: false

From these results is possible to see that the implementation of a XOR2 using this methodology will have no separations in the diffusion area of pull-up and pull-down networks. It is also possible to observe that the pull-up has no repeated internal nodes, so no internal connections are needed. The pull-down network needs one connection because the node *pd_n1* appears twice in the Euler path.

Experiments to evaluate CMOS logic gate implementation in a single VCTA unit [9] also have been performed. A set of 222 cells including the 4-input NPN representative functions has been analyzed. The main conclusions of this analysis is that few logic gates can be built in a single VCTA cell if this cell has only 6 PMOS and 6 NMOS transistors and 4 inputs, as was the current configuration. With this cell it becomes necessary to spread the (transistor network of the) logic gates in two or more cells or to remap complex gates into combinations of simpler (basic) ones. The optimal number of pre-designed column heights in each logic plane is around 12 PMOS and 12 NMOS transistors for this specific set of cells. In this case, half of the cells from the set used in the experiment would fit in a single unit, while two units should be enough for building the rest of them.

Table 1 – Evaluation of exclusive-OR Euler paths, from Fig. 1.

Network	Number of connections	Not repeated nodes (nodes that do not need a connection)	Breaks	Euler path size
Pull-up	0	4 (!a, pu_n4, pu_n1, !b)	0	6 edges
Pull-down	1	2 (!a, !b)	0	6 edges

6. Conclusions

This paper presented a useful tool to profile switch networks. It is able to extract information from the graph and Euler paths of CMOS logic gates, comprising the pull-up and pull-down logic planes. With the extracted information is possible to do a better analysis of the network with a focus in the number of connections that will be needed to generate the layout of the network, thus trying to avoid routing congestion and the need of more metal layers. As future work, the order of the Euler paths may be implemented to match pull-up and pull-down polysilicon stripes, trying to avoid the need of extra metal wires to connect pull-up and pull-down planes.

7. Acknowledgements

Research partially funded by Nangate Inc. under a Nangate/UFRGS research agreement, by CNPq and CAPES Brazilian funding agencies, and by the European Community's Seventh Framework Programme under grant 248538 - Synaptic.

8. References

[1] Iizuka, T.; Ikeda, M.; Kunihiro, A.; “Exact minimum-width transistor placement without dual constraint for CMOS cells”, in Proceedings of the 15th ACM Great Lakes symposium on VLSI, pp. 74-77, 2005.

[2] Guruswamy, M. *et al*. “CELLERITY: A fully automatic layout synthesis system for standard cell libraries”, in Proc. ACM/IEEE 34th Design Automation Conference, pp. 327-332, 1997.

[3] Saxena, P.; Shelar, R. S.; Sapatnekar, S.; "Routing congestion in VLSI circuits", Springer, 2007.

[4] Wang, M.; Sarrafzadeh, M.; “Modeling and minimization of routing congestion”, in Proceedings of the 2000 Asia and South Pacific Design Automation Conference, pp. 185-190, 2000.

[5] Callegaro, V.; Marques, F. S.; Klock, C. E.; Rosa Junior, L. S.; Ribas, R. P.; Reis, A. I.; “SwitchCraft: a framework for transistor network design”, in: SBCCI 2010, São Paulo, p. 49-53.

[6] Uehara, T.; VanCleemput W.; “Optimal layout of CMOS functional arrays”, in IEEE Transactions on Computers, vol. c-30, n.5, May 1981.

[7] Skiena, S.; “The Algorithm Design Manual”, Second Edition, Springer, 2008, pp. 502-504.

[8] Harju, T.; “Lecture Notes on Graph Theory”, Department of Mathematics, University of Turku, Finland, 2011, pp. 29-31

[9] Pons, M.; Moll, F.; Rubio, A.; Abella, J.; Vera, X.; González, A.; “VCTA: A via-configurable transistor array regular fabric”, in IEEE/IFIP VLSI System on Chip Conference (VLSI-SoC), 2010.

A Lookup Table Method for Optimal Transistor Network Synthesis

[1]Anderson Santos da Silva, [2]Vinicius Callegaro, [1,2]Renato P. Ribas, [1,2]André I. Reis
{assilva, vcallegaro, rpribas, andreis}@inf.ufrgs.br

[1]Institute of Informatics, UFRGS, Porto Alegre, Brazil
[2]PPGC, UFRGS, Porto Alegre, Brazil

Abstract

This paper presents a lookup table based method for switch networks and logic gates implementation. The proposed approach is able to deliver optimal transistor networks for all Boolean functions up to four inputs. Results show a new transistor count lower bound for P-class and NPN-class for four inputs logic gates.

1. Introduction

Nowadays, technology-scaling problem have been addressed by several approaches. One of the most important design strategies is the layout regularity [1], which is able to produce lithography-aware circuits up to 20 nm of transistor channel length. However, the regular transistor layout (RTL) approach has an important drawback related to the increased circuit area when compared to the most popular standard cells methodology. In order to overcome that, algorithms for generating optimized transistor networks can be applied for reducing the total transistor count in logic gates.

There are several algorithms for generating transistor networks. These algorithms could be divided into two main groups: factorization [2][3] and graph-based [4]-[6]. Factorization methods always deliver transistor networks in series-parallel arrangements. It is known that series-parallel arrangements may not reach the optimal design regarding transistor count. Graph-based algorithms are able to generate not only series-parallel networks, but also non-series-parallel arrangements, i.e., presenting Wheatstone-bridge type connections. These kinds of arrangements consist of transistors that cannot be classified as series or parallel switches in the network arrangement. The non-series-parallel networks have presented promising results in the transistor reduction count [6], being of great interest to reduce the circuit area when RTL approaches are targeted.

Some algorithms for generating non-series-parallel arrangements have as main goal the minimization of the total number of transistors in networks and consequently in logic gates [4][5]. Other ones reach transistor arrangements that respect the minimum theoretical transistor stack for a specific Boolean function [6]. Note that there is a large variety of transistor network arrangements that represent the same logical functionality. Such richness on arrangements could be used on library-free technology mapping concept, by exploiting automatic on-the-fly cell generation [7].

This paper presents a lookup table method for delivering optimal transistor networks for all functions up to four input variables. These optimal transistor networks also respect the minimum theoretical stack for a target Boolean function, while using non-series-parallel arrangements to reduce the total transistor count in networks. Results show a new lower bound on transistor count for 4-input P-class and NPN-class functions [8] when compared with previous approaches [5][6]. A set of all functions up to 4 inputs was performed, giving the result in transistor count through the implementation of all these functions as logic gates.

2. Technical Background

The basic element to implement a switch network is the logic switch. This element can be called ‘direct switch’ if it is turned on by applying the logic value ‘1’ in the control terminal, and ‘complementary switch’ if it is turned on by applying the logic value ‘0’ in the control terminal. By composing switches, it is possible to build arrangements, known as logic networks or switch networks, in order to provide the interconnection between two different terminals according to a given logic function behavior.

Depending on the technology used, these switches can be implemented as physical devices. In the current CMOS technology, they are represented by the NMOS transistor (direct switch) and the PMOS transistor (complementary switch).

When looking at a single two terminal network, it may present the following properties [6]:

- **Planar network** – Networks corresponding to a planar graph. This kind of graph can be drawn in the plane without crossing lines. In the case of networks, it is additionally required that the terminals be externally connected without crossing any lines. Planar networks can provide a dual graph, which has the interesting property of being logically and topologically complementary.
- **Series-parallel network** – When all switches in the network are connected in series or in parallel arrangements recursively. A network is series-parallel if and only if there is no embedded network presenting a Wheatstone-bridge configuration.
- **Bridge network** – A network with an embedded network containing at least one Wheatstone-bridge configuration. A bridge network may or may not be planar, and it is never a series-parallel network.

- **Bidirectional transistors** – A bridge network where transistors may conduct current from drain to source or from source to drain device terminals according to the input vector. That means the bidirectional transistors are activated by different logic vectors in both directions.

Additionally, switch networks can present two-terminals, three-terminals, or multiple-terminals. Two-terminals networks provide the connection between two nodes, and are usually applied to built single-rail logic gates. Three-terminal networks, in turn, are capable of attaching one node to other two terminals, which are frequently one for the direct polarity signal (or direct path) and the other for the inverted polarity signal (or complementary path). These ones are exploited to design dual-rail CMOS logic families, like DCVSL, DSL and ECDL [9]. Multiple-terminal networks are useful to build multiple-output gates like the Manchester chain used in carry lookahead adders [10].

These characteristics are very important once it is possible to build logic gates with similar functionality, and distinct electrical behavior (timing and power consumption) to compose digital circuits by exploiting these different switch arrangements.

3. Proposed Approach

The proposed work focuses only on Boolean functions that have up to four input variables. This set represents the great majority of logic gates available in standard cell libraries. The transistor arrangements returned by our approach intends to be the optimal solution regarding the transistor count. In order to make the lookup table, the following tasks are performed:

- implementation of a NPN-matching algorithm;
- manual transcription of catalog by Moore for a digital format (Spice netlist like);
- lookup table synthesis.

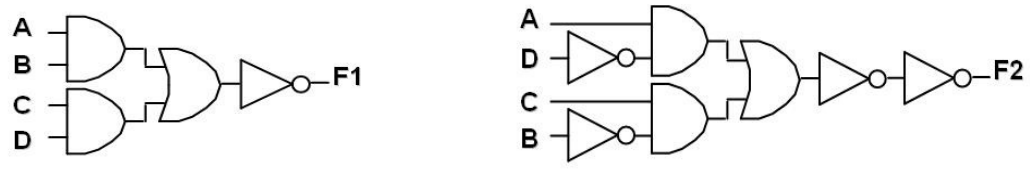
3.1. NPN Matching

Determining when a logic gate could implement an arbitrary logic function is a common problem on the logic synthesis. This matching could be done by permutation of the inputs and/or inserting inverters on the inputs/output of the logic function.

The circuit illustrated in Fig. 1a implements the logic function described by Equation (1). To perform the different logic function described by Equation (2), it is possible by just swapping the inputs B and D, and inserting additional inverters on such inputs as well as in the circuit output node. The new circuit is depicted in Fig. 1b.

$$F1 = !(A*B + C*D) \quad (1)$$

$$F2 = (A*!D + !B*C) \quad (2)$$



(a) **(b)**

Figure 1 – Circuit implementation of logic functions described in: (a) Equation 1 and (b) Equation 2.

The rearrangement made in Fig. 1b was possible because the Equations (1) and (2) belong to the same NPN-class. In this way, each function belongs to an NPN-class. The algorithm that shows how to rearrange the logic gate to implement an arbitrary logic function is the NPN matching. Several approaches for NPN matching have been proposed in the literature [11][12]. In this work, the approach proposed by Sasao [12] was considered.

3.2. Catalog by Moore

The main resource of this work is a catalog, published by Edward Moore [13], in 1958. This catalog has the optimal switches arrangements for all NPN-class up to four inputs functions set. As it is possible to see in Fig. 2, each NPN class has a main switch network. The function and its complement always belong to the same NPN-class, and consequently the same related network on the catalog.

For a standard CMOS logic gate design, it is necessary to obtain both pull-up and pull-down planes. Pull-up plane is responsible for implementing the direct function and the pull-down plane the complementary function. In order to achieve that, it is necessary to find the complementary network on the catalog. However, when the network is planar, the complement is absent from the catalog. In order to achieve the topological and logical complementary network, the dual graph of the original network is obtained [14]. For functions which

the best arrangement is a non-planar-bridge network, the complementary network is always present on the catalog.

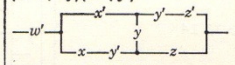
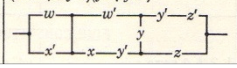
Harvard serial No.	Complement	Category class	Constituents (Elementary circuits)	Algebraic expression for series parallel Schematic for others	Notes	Number of contacts	Number of springs	Number of circuits
		I	ONE CONSTITUENT					
1	400	1	0 0/4 4/4 (1234/4 0123/4)	$w'x'y'z'$	J	4	8	16
		II	TWO CONSTITUENTS					
2	396	1	0 1 same cube	$w'x'y'$	J	3	6	32
3	397	2	0 3 same cube	$w'x'(y'z'+yz)$	J	6	10	48
4	398	3	0 7 same cube	$w'(x'y'+xz)(y+z')$	J	7	11	32
5	399	4	0 15 04/4 (123/4)	$(w'+x)(wy+x'z')(y'+z)$	J	8	12	8
		III	THREE CONSTITUENTS					
6	390	1	0 1 2 same cube	$w'x'(y'+z')$	J	4	8	96
7	391	2	0 1 6 same cube	$w'(x'y'+xz'y)$	J	6	10	192
8	392	3	0 1 14	$(w'x'+wy)(xz'+y')$	J	7	11	64
9	393	4	0 3 5 same cube		Q	8	13	64
10	394	5	0 3 12	$(w'x'+wy'x)(yz+y'z')$	W	9	14	48
11	395	6	0 3 13		X	9	14	96

Figure 2 - Catalog switch networks by Moore [13].

3.3. Proposed Lookup Table

The proposed lookup table (LUT) consists on all functions up to four inputs, and for each function the following information is stored:

- which NPN-class it belongs;
- transistor count to implement the logic function;
- inputs permutations;
- inputs and output inversion.

In terms of memory usage, the following cost of bits was used:

- NPN-class: 9 bits;
- transistor count: 4 bits;
- input order: (2 bits each) times four inputs: 8 bits;
- input inversion: (1 bit each) times four inputs: 4 bits;
- output inversion: 1 bit.

The set of all functions up to four inputs contains 65,536 elements (2^{2^4}). In this sense, for each function we have used 26 bits of information. Therefore, we will create a primitive *int* array with 65,536 elements. As a primitive *int* is represented with 32 bits in almost all program languages, it is enough for us. In this way, the total memory used by our lookup table is only 256 Kbytes ($2^5 \times 2^{16} - 2^3$). In order to access the information element on array, the function is itself the key.

4. Experimental Results

In Table 1 are shown total switch count and the execution time required to provide it considering three sets of Boolean functions: NPN-class, P-class and all 65,536 functions mentioned in the previous section. As shown in Table 2, when comparing the total number of transistors used to implement the 4-input P-class functions obtaining by applying other transistor network generation methods, the proposed approach demonstrates that there is room for optimization in the existing methods presented by Kagaris *et al.* [5] and Da Rosa Jr. [6].

Table 1 – Transistor count to implement the different sets of functions with 4 variables.

Class	Switch Count	Time
NPN-class	3,563	657 µs
P-class	64,530	11 ms
All functions (65,536)	1,050,880	3 min

Table 2 – Comparison between switch network generation methods to implement the P-class.

P-class functions	Transistor Count
Kagaris [5]	97,174
Da Rosa Jr. [6]	97,098
LUT approach	91,177

5. Conclusion

This paper presented a lookup table (LUT) method for delivering transistor networks for all functions up to four input variables. As our approach always reach an optimal transistor arrangement, it could be used to find out the lower bound transistor count for every benchmark function set with up to four input variables. As future work, a method for generating circuits with more than 4 inputs will be performed. The main idea is generate cofactors and cube-cofactor of a target function and compose the circuit using the proposed lookup table.

6. Acknowledgments

Research partially funded by Nangate Inc. under a Nangate/UFRGS research agreement, by CNPq and CAPES Brazilian funding agencies, and by the European Community's Seventh Framework Programme under grant 248538-Synaptic.

7. References

[1] B. H. Calhoun, Yu Cao, Xin Li, Ken Mai, L. T. Pileggi, R. A. Rutenbar, and K. L. Shepard, "Digital circuit design challenges and opportunities in the era of nanoscale CMOS", IEE Proccedings, vol. 96, no. 2, pp. 343-365, Feb. 2008.

[2] M. G. A. Martins, L. S. Rosa Jr., A. B. R. Rasmussen, R. P. Ribas, and A. I. Reis, "Boolean factoring with multiple objective goals". ICCD 2010, pp. 229-234.

[3] V. Callegaro, L. S. Da Rosa Jr., A. I. Reis, and R. P. Ribas, "A Kernel-based approach for factoring logic functions," Microelectronics Students Forum, 2009.

[4] J. Zhu and M. Abd-El-Barr, "On the optimization of MOS circuits," IEEE Trans. on Circuits and Systems, vol. 40, no. 6, pp. 412-422, June 1993.

[5] D. Kagaris and T. Haniotakis, "A methodology for transistor-efficient supergate design," IEEE Trans. on VLSI Systems, vol. 15, no. 4, pp. 488-492. Apr. 2007.

[6] L. S. Da Rosa Junior, "Automatic generation and evaluation of transistor networks in different logic styles," PhD Thesis, PGMicro / UFRGS, 2008.

[7] J Xue, D Al-Khalili, and C. N Rozon, "Technology mapping in library-free logic synthesis," Proceedings of SPIE, 2005.

[8] T. Sasao, Switching Theory for Logic Synthesis. Norwell, Kluwer Academic Publishers, 1999.

[9] J. M, Rabaey, A. Chandrakasan and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd ed., Prentice Hall, 2005.

[10] N. H. E. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 3rd ed., Pearson/Addison Wesley, 2005.

[11] U. Hinsberger and R. Kolla, "Boolean matching for large libraries," Proc. Design Automation Conference (DAC), pp. 206–211, 1998.

[12] D. Debnath and T. Sasao, "Efficient computation of canonical form for Boolean matching in large libraries," Proc. Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 591-596, 2004.

[13] E. F. Moore, "Table of four-relay contact networks", In: Logical Design of Electrical Circuits, by R. A. Higonnet and R. A. Grea, McGraw-Hill, 1958.

[14] V. Callegaro, L. S. Da Rosa Jr., A. I. Reis, and R. P. Ribas, "A Graph-based solution for dual transistor network generation," Student Forum on Microelectronics, 2008.

NoCs, MPSoCs and Analog Design

A Self-adaptable Distributed DFS Scheme for NoC-based MPSoCs

[1]Thiago Raupp da Rosa, [1]Douglas Cardoso, [1]Fernando Moraes
{thiago.raupp, douglas.cardoso}@acad.pucrs.br, fernando.moraes@pucrs.br

[1]Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS

Abstract

A clear trend in high-end embedded systems is the use of multiprocessor systems on chip (MPSoCs). As processor count in these devices increases, the use of NoCs becomes relevant, if not mandatory. However, power and energy restrictions, especially in mobile applications may render the design of NoC-based MPSoCs over-constrained. The use of traditional dynamic voltage and frequency scaling (DVFS) techniques proved useful in several scenarios to save energy/power, but it presents scaling problems and slow response times. This work proposes a self-adaptable distributed dynamic frequency scaling (DFS) scheme for NoC-based MPSoCs. It takes into account the communication load and the utilization level of each processor to dynamically change its operating frequency. Frequency change decisions and clock generation are executed locally to each processor. Clock generation is simple, based on clock gating of a single global clock. The overhead of the scheme in terms of area is minimum, the range of generated clocks frequencies is wide and the response time of frequency switching is negligible. Experimental results in an actual MPSoC running a real application show that the proposed scheme has an average execution time overhead below 14%, and may lead to considerable power and energy savings, since it allows an average reduction of 64% on the total number of executed instructions.

1. Introduction

MPSoCs increase system performance by employing multiple processors to execute system tasks, which are interconnected by a communication infrastructure. NoC-based MPSoCs provide massive computing power on a single chip.

Energy consumption in CMOS circuits can be reduced by controlling two main variables: the supplied voltage and the operating frequency. Controlling these two variables at runtime is the basis of Dynamic Voltage and Frequency Scaling (DVFS) techniques.

This paper proposes and evaluates a new technique for Dynamic Frequency Scaling (DFS) with fixed system voltage. The main goal of this technique is to enable fast frequency switching according to each processor's workload and communication load. The proposed DFS scheme is evaluated in a synthesizable NoC-based MPSoC.

The rest of this paper is organized as follows. Section 2 reviews the related work, comparing it to the proposed technique. Section 3 presents the clock generation module. Section 4 describes the MPSoC architecture and the required modifications to enable the DFS scheme. The proposed DFS controller is presented in Section 5. Section 6 presents the experimental setup and results. Finally, conclusions and future works are drawn in Section 7.

2. Related Work

Tab. 1 summarizes the state of the art according to three criteria: target architecture, monitoring parameter, and implementation. Just a few works address NoC-based MPSoCs [1], [10]-[13]. Most works target only one CPU or bus-based architectures, using a centralized controller . DVFS schemes may use hardware or software control parameters. Hardware parameters for controlling DVFS include temperature [7], [10], process variation [1], current [2] and load in communication buffers [4]-[6]. Software parameters include application profile [8], [13] and scheduling tasks [3], [9], [12]. In terms of implementation, most proposals employ software parameters, releasing to the hardware the monitoring process (when a hardware parameter is monitored). The approach proposed here aims to control DFS scheme through hardware and software mechanisms. The hardware mechanism obtains data from the Network Interface (NI) and from the processor to parameterize the clock generation module, setting the correct frequency for the NoC and PEs. Software mechanisms are responsible for monitoring a set of parameters, making them available to the hardware mechanism.

3. Clock Generation

This Section presents the principles and design of a clock generator to enable the proposed DFS scheme. This module uses as input a reference clock, which consists in the highest frequency usable in the system as a local clock. The principle of the clock generation process is to achieve clock division by simply omitting selected cycles of the reference clock, as Fig. 1 illustrates: initially, inputs num_i and den_i are natural numbers 2 and 5, respectively. This corresponds to set the frequency of the clock generator to two-fifths (40%) of the

Tab. 1 - DVFS state-of-the art comparison.

Author	Architecture	Monitoring Parameter	Implementation
Herbert [1] 2009	NoC-Based MPsoC	Process Variation	Off-line Calibration (Design Variability), Algorithms in Software
Pourshaghaghi [2] 2009	Single CPU	CPU Supply Current	Fuzzy Logic Controller in Hardware
Chabloz [3] 2010	Synchronous Islands	Tasks Deadlines	GRLS Scheme, Local clock generation
Yin [4] 2009	NoC	Queues Load	Voltage Selection via Transistors
Alimonda [5] [6] 2006/2009	Bus-Based MPSoC	Queues Load	Central Controller Hardware
Shu [7] 2010	Single CPU	CPU Temperature	Temperature Sensors and Software Algorithm
Salehi [8] 2010	Single CPU	Application History	Software Tracking Application Workload
Liu [9] 2009	2 CPUs, Bus-based Interconnect	Tasks Slacks	Task Graph Unrolling Software
Puschini [10] [11] 2008/2009	NoC-Based MPSoC	Temperature and Task Synchronization/Latency	Parameter Modeling, Game Theory Algorithm
Goossens [12] 2010	NoC-Based MPSoC	Tasks Slacks	Voltage and Frequency Scaling Hardware, Software to adjust the Controller
Kong [13] 2008	Bus-Based MPSoC	Application Profile	Software computes Suitable DVFS Level and Informs Controller Hardware
This Work	NoC-Based MPSoC	Communication and CPU Load	Local Clock Generation, Controller sets Correct Frequency Level. Software updates Controller with Current CPU state.

reference clock. In other words, for each *den_i* reference clock cycles, *num_i* cycles are propagated to the output clock.

Any frequency obtained by changing the *num_i* and *den_i* values can be generated with the obvious exceptions (*den_i*=0 is not an acceptable value, *num_i*=0 corresponds to a clock gating action and the constraint $num\_i \leq den\_i$ must be respected). Before changing the *num_i* and *den_i* values, the *restart_i* signal must be asserted to momentarily stop the output clock and reinitialize internal registers. After releasing *restart_i*, the new frequency, defined by the modified values of *num_i* and *den_i* appears at the output.

The main advantages of this clock generation module are the low area overhead and a large set of generated frequencies. In addition, the clock output is always stable, contrary to what happens in standard DFS methods, where the time required to stabilize a new frequency can be significant. The proposed module is also glitch free by construction. Such features make the use of the proposed clock generator module appropriate for distributed DFS in MPSoCs, where each PE may have its own frequency according to its load. The drawbacks of the approach are how to couple it with voltage scaling and the need to design the critical paths of all modules in the system to support the reference clock frequency.

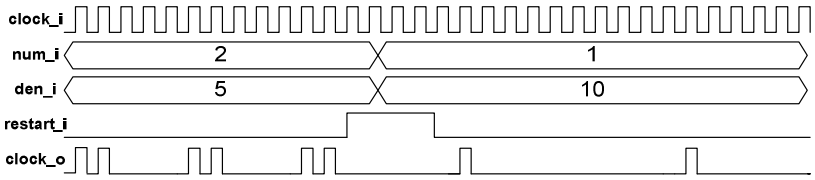


Fig. 1 - Example of the proposed clock generation process. Signal clock_i is the reference clock and clock_o is the output of some clock generator.

4. System Architecture

The reference MPSoC [14] is a homogeneous multiprocessing NoC-Based MPSoC. Fig. 2 shows an instance of this MPSoC. The 2-D mesh NoC used in the reference MPSoC has the following features: wormhole packet switching, flit width equal to 16 bits, XY routing algorithm, round-robin arbitration, input buffers with 8-flits depth.

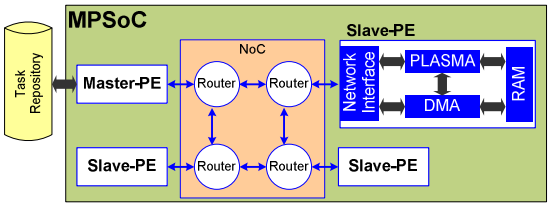


Fig. 2 - Block diagram of the reference MPSoC architecture.

Each PE includes the following modules: (i) a 32-bit Plasma processor (a MIPS-like architecture); (ii) a local memory (RAM); (iii) a DMA module, responsible for transferring the task object code to the memory and messages to/from the NoC to the local memory; (iv) a network interface (NI). Two types of PEs are used: slave and master. Slave-PEs are responsible for executing application tasks, while the Master-PE is responsible for managing task mapping and system debug. The task repository is an external memory, responsible to store all object codes of applications that will eventually be executed. Each slave processor runs a multitask *microkernel*

that enables the communication between tasks through *send* and *receive* primitives, respectively called *WritePipe()* and *ReadPipe()*. Each *microkernel* contains a vector, named *pipe*, which contains messages to be exchanged between tasks. The most relevant features of the system for the DFS controller are task scheduling and the inter-task communication process. Monitoring the scheduler it is possible to evaluate the CPU utilization and monitoring the *pipe* occupation it is possible to evaluate the communication load.

4.1. Router-PE GALS interface

The present work assumes the NoC operates at a fixed frequency (the reference frequency divided by two) and only processors change their frequency. Therefore, the router-PE interface needs to be modified to adapt to the GALS paradigm, since processor and router may operate at different frequencies due to the DFS scheme. This is achieved by substituting buffers in the NoC and network interface by bisynchronous FIFOs, and introducing two-flop synchronizers in control signals. The router-PE interface contains two buffers (GALS FIFOs), one at the router to receive data from the PE, and the other at the NI, to receive data from the NoC. Besides these hardware modifications, the *microkernel* was changed to monitor CPU utilization and communication *pipe* occupancy, storing them in new memory-mapped registers. Based on this information, the controller can take decisions and dynamically change the processor frequency.

5. The DFS Controller Structure

The DFS controller computes the communication load and CPU utilization level according to values provided by the *microkernel*. Such values are used by the controller to define each PE frequency. The controller always operates at the *reference frequency* (the highest frequency in the system, used as input to the clock generation module). As the DFS controller works at the reference frequency and the processor at a different frequency, a synchronization scheme between them is necessary. The controller uses the clock generation module, detailed in Section 3, to provide the output clocks. The role of the DFS controller is to choose the correct PE frequency, by evaluating the following parameters:

- Pending message requests from other tasks. This situation takes place when the processor is not producing data to the consumer task.
- Occupancy of the *pipe*. If the pipe has a high occupancy, the processor is producing messages at a higher rate than the consumer tasks can consume, while the inverse scenario means a lack of produced messages. Upper and lower parameterizable thresholds define the *high* and *low* occupancy states, respectively. Occupancy between these values defines an *operational* state.
- CPU utilization. When the utilization is low the CPU is not executing any task or tasks are blocked, e.g., waiting message(s) from other tasks. When the utilization is high, tasks are using the processor at the maximum rate. Two parameterizable thresholds define *high*, *low* and *operational* CPU utilization states.

Frequency decreases in three situations: (*i*) the *pipe* is almost full; (*ii*) the *pipe* occupation is increasing, i.e. in the previous evaluation its state was *low* and the present state is *operational*; (*iii*) the *pipe* occupation is almost empty and the CPU usage is *low*, meaning that even at a lower frequency the data in the *pipe* is being consumed. Frequency increases in three situations: (*i*) existence of pending messages with operational or high CPU utilization; (*ii*) the *pipe* is almost empty and the CPU has high utilization; (*iii*) the *pipe* occupation is dropping, i.e. in the previous evaluation its state was *high* and the present state it is *operational*.

Lastly, when a given PE receives a message request, and it has data to transmit, this PE goes to the reference frequency during the message transmission. This action avoids stalling consumer PEs operating at higher frequencies than the producer PEs.

6. Experimental Results

This section employs an instance of the reference MPSoC with 6 processors (1 Master-PE and 5 Slave-PEs) and a 3x2 NoC to demonstrate the characteristics and advantages of the proposed DFS scheme. A *Partial MPEG filter* application was used to evaluate the performance of the proposed DFS controller. The partial MPEG filter is composed by five tasks, modeled as a pipeline. The DFS controller was parameterized to generate 9 different frequencies: 5, 10, 25, 40, 50, 60, 75, 90 and 100% of the reference frequency. In the graphic presented in this Section, these frequencies are plotted in the y axis, with values ranging from 0 to 8.

The result for the partial MPEG decoder is shown in Fig. 3. In this application *iVLC* is a CPU-intensive task. Tasks *iQuant* and *IDCT* are simpler than *iVLC*. Tasks *Start* and *Print* are used to initialize the system and to print the results, respectively. In this test case, 200 frames were transmitted.

The graphic shows that only the task executing a high amount of computation had its frequency increased to the reference frequency, while *Print* and *Start* tasks had their frequency decreased to the lowest frequency level. The execution time overhead, compared to the execution with the whole system operating at reference frequency was 13%. The number of executed instruction is reduced in 64%.

When the whole system executes with no DFS scheme, the six processors and the NoC operate at the reference frequency. On the other hand, using the proposed DFS scheme, only one processor operates at the

reference frequency, while three other processors and the NoC operate, in average, at half of the reference frequency (including the Master-PE) and two processors operate at the lowest frequency level.

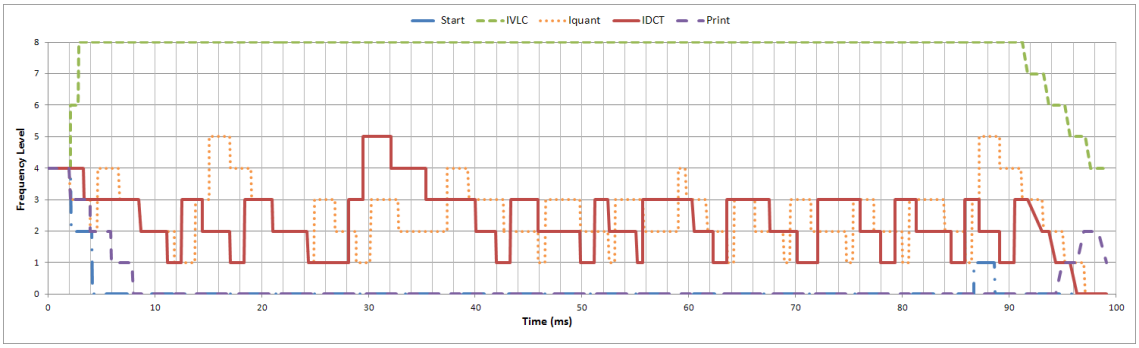


Fig. 3 - Partial MPEG filter execution for 200 frames.

7. Conclusion

This work proposes a new DFS scheme and evaluates it in a real MPSoC platform. The frequency scaling scheme is based in the communication load and CPU utilization of each MPSoC PE. The DFS controller uses information provided by the *microkernel*. A clock generation module was designed to enable frequency changing. This module presents a low-area overhead and requires no stabilization time at each frequency switching. Results shows that the DFS scheme adjust the processor frequency according to the load injected into the network. As shown in the MPEG benchmark, the CPU-intensive task has its frequency increased to generate more data to the other tasks. Once the tasks with lower injection rate reach the reference frequency, the system stabilizes, reducing the frequency of other tasks. The proposed DFS method has a small impact in the total execution time. Therefore, an important energy reduction is expected, since few processors of the MPSoC operate at the reference frequency, drastically reducing the number of executed instructions.

8. References

[1] Herbert, S.; Marculescu, D. "Variation-aware dynamic voltage/ frequency scaling". In: HPCA, pp. 301-312, 2009.

[2] Pourshaghaghi, H.R.; de Gyvez, J.P. "Dynamic voltage scaling based on supply current tracking using fuzzy Logic controller". In: ICECS, pp.779-782, 2009.

[3] Chabloz, J. M.; Hemani, A. "Distributed DVFS using rationally-related frequencies and discrete voltage levels". In: ISLPED, pp.247-252, 2010.

[4] Yin, A. et al. "Architectural Exploration of Per-Core DVFS for Energy-Constrained On-Chip Networks". In: DSD, pp.141-146, 2009.

[5] Alimonda, A. et al. "Non-Linear Feedback Control for Energy Efficient On-Chip Streaming Computation". In: IES, pp.1-8, 2006.

[6] Alimonda, A. et al. "A Feedback-Based Approach to DVFS in Data-Flow Applcations". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 11, pp. 1691-1704, 2009.

[7] Shu, L.; Li, X. "Temperature-aware energy minimization technique through dynamic voltage frequency scaling for embedded systems". In: ICETC, pp. 515-519, 2010.

[8] Salehi, M. E. et al. "Dynamic Voltage and Frequency Scheduling for Embedded Processors Considering Power/Performance Tradeoffs." IEEE Transactions on Very Large Scale Integration Systems, in press, 2010.

[9] Liu, S.; Qiu, M. "A Discrete Dynamic Voltage and Frequency Scaling Algorithm Based on Task Graph Unrolling for Multiprocessor System". In: SCALCOM-EMBEDDEDCOM, pp.3-8, 2009.

[10] Puschini, D. et al. "Temperature-Aware Distributed Run-Time Optimization on MP-SoC Using Game Theory". In: ISVLSI, pp.375-380, 2008.

[11] Puschini, D. et al. "Adaptive energy-aware latency-constrained DVFS policy for MPSoC". In: SOCC, pp.89-92, 2009.

[12] Goossens, K. et al. "Composable Dynamic Voltage and Frequency Scaling and Power Management for Dataflow Applications". In: DSD, pp.107-114. 2010.

[13] Kong, J. et al. "Low-Cost Application-Aware DVFS for Multi-core Architecture". In: ICCIT, pp.106-111. 2008.

[14] Carara, E.A. et al. "HeMPS - a framework for NoC-based MPSoC generation". In: ISCAS, 2009, pp. 1345-1348.

Analog Design Methodology adopted in Training Center 1

[1]Sandro Ferreira, [1]Everton Ghignatti, [1]Alcides Costa, [2]Eric Fabris
{sandro,everton,alcides}@nscad.org.br, fabris@inf.ufrgs.br

[1] NSCAD, [2] UFRGS

Abstract

This article presents the analog design flow, management structure and design architecture adopted in the Brazilian IC design training program during the analog IC design phase, developed at Training Center #1 in Porto Alegre. The architecture selected for implementation is an IEEE 802.15.4 PHY transceiver.

1. Introduction

The Brazilian IC design training center #1 (CT1) was created in March 2008 as a government effort to develop qualified human resources to the recent country semiconductor sector. Sponsored by the MCT (Brazilian Ministry of Science and Technology) and part of the CI Brasil Program [1], CT1 is located at Instituto de Informática da UFRGS, in Porto Alegre. It is supported by the Nucleo de Suporte ao CAD (NSCAD) Team [2]. The training center #2 (CT2) is also part of the CI Brasil Program and it is located at Centro de Pesquisas Renato Archer (CTI), in Campinas.

Since its beginning, CT1 and CT2 trained more than 300 professionals on digital, analog and mixed-signal (AMS) and radio frequency (RF) IC design areas to the country. The CI Brasil training program is structured in the following way: an initial five months long phase, focused on theoretical lectures and EDA tool training using Cadence Design Systems software, followed by a second phase (design phase), seven months long, where the students are immersed in a carefully prepared simulated project environment, challenging a real IC design.

This article presents the structure developed by CT1 to support the AMS and RF design phase of the CI Brasil Program and the proposed design itself. It focuses on the AMS and RF front end transceiver that implements the protocol IEEE 802.15.4 PHY Layer. The proposed design was chosen to give the students a real experience in design with all the challenges pertaining to real RF implementations being at the same time feasible in the design period of time. The next sections of the article present the design phase, including its main steps, major milestones, management structure adopted, followed by an overview of the transceiver design scope and its top-level design. Finally, some conclusions are presented.

2. Design Phase

The second phase of the training takes students to a simulated environment where the student participates in a complete analog design flow from preliminary specifications to tape out.

During the phase, students exercise best practices in design and management, working as a design team. Project management concepts, such as tight schedule control, deadline dependencies and communication skills are exercised according to a predefined Communication Plan used throughout the project.

The Analog Team is composed by NSCAD Team and students according to the following roles: Technology Manager, Technical Leader, Field Application Engineer, Team Leader, Block Leader and Designer. First three roles are played by NSCAD Team and the other roles are played by students.

The project is divided in five steps, according to the Project Management Body of Knowledge [3]:

- Initiation – when NSCAD Team establishes main project objectives.
- Planning – NSCAD Team defines product specifications and technology to be adopted. Project environment is prepared as well as templates, standards and schedule to be adopted during Execution.
- Execution – performed by the complete Analog Team, is the product development phase itself, which is divided in six design phases, as presented in the next section of the article (section 2.1).
- Monitoring and Control – performed by NSCAD Team, involves all management and quality aspects that run in parallel with the Execution phase. For instance, schedule control, meetings, milestone checking, and technical support to the design team and version control.
- Closing – NSCAD Team performs necessary adjustments to close the project.

2.1. Execution Phases

In order to synchronize efforts, organize and facilitate tracking, the project execution is divided into six phases that implement the analog design flow. The design flow is slightly different when it refers to block-level design (Fig. 1) or top-level (Fig. 2). The steps that are part of the design flows are described during the execution phases presented below. The top-level design (Fig. 2) is in charge of integration of the blocks, power distribution, ESD protection and routing of the connections at top level. The block-level design (Fig. 1) focus on the individual blocks that are part of the top-level architecture.

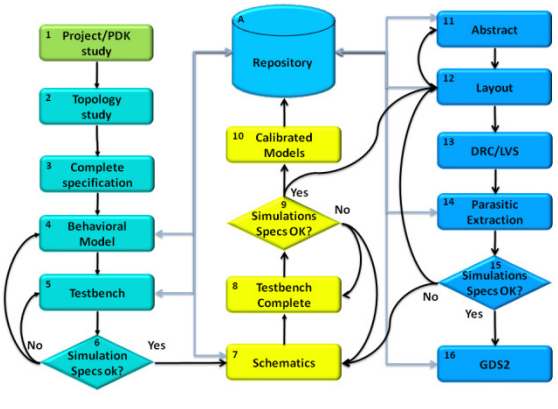


Fig. 1 – Block-Level Analog Design Flow

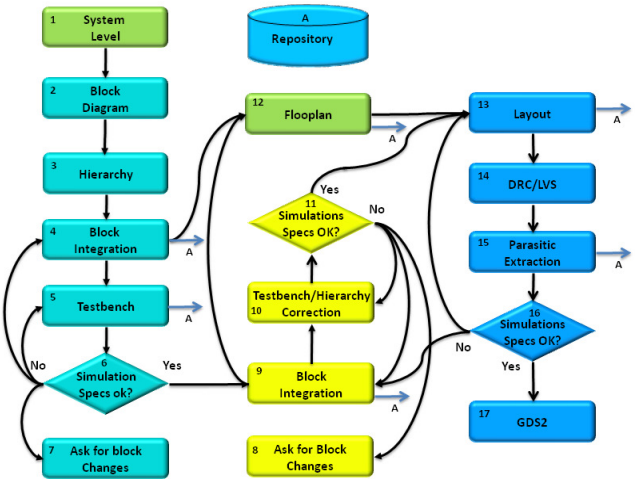


Fig. 2 – Top-Level Analog Design Flow

During the entire design process all design files, like models, testbenches, simulation states, schematics, layouts, abstracts and extracted layouts must be saved in a specific repository respecting naming conventions.

Fig. 3 presents the preliminary schedule adopted to give a better understanding of the design effort in time.

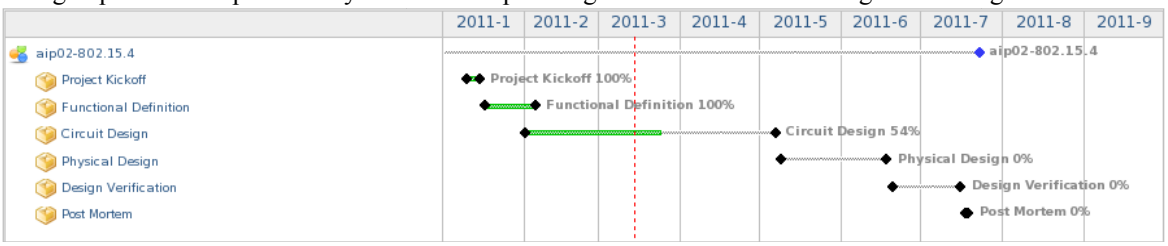


Fig. 3 – Preliminary Schedule

2.1.1. Project Kickoff

Project Kickoff is the first phase of the Execution when designers are prepared to start the project. Design Team is divided according to the blocks in order to develop circuit hierarchy (step 1 in Fig. 2). Team Leader is also selected among the students according to his technical background, leadership and communication skills.

During this phase, designers should understand project structure (standards, templates, conventions, communication lines, directory structure, repositories, etc) and study the adopted Process Design Kit (PADs, devices, design rules, ESD protections, etc).

2.1.2. Functional Definition

During the Functional Definition phase, designers begin to understand their assigned circuits and research circuit topologies to reach specifications. Preliminary floorplans, preliminary verification and testability approach, as well as noise minimization and ESD protection strategies are also developed (steps 2 to 6 in Fig. 1 and Fig. 2). Main objectives of the phase are listed below:

- Understand product specifications, system architecture and identify top-level operating modes;
- Perform system level simulations to verify system functionality and establish block specifications using spreadsheets, behavioral models or system-level simulation tools. (step 1 in Fig. 2);
- Develop and test detailed behavioral models and testbenches to simulate block specifications and to be used in mixed-level simulations during schematic developments.

2.1.3. Circuit Design

In the Circuit Design phase, designers must complete specifications at transistor level and update floorplans. At the end of the phase, block and top-level should pass specifications over process, voltage and temperature (PVT) corners and Monte Carlo process mismatch analysis whenever proved necessary (steps 7 to 10 in Fig. 1). Due to the difficult to achieve circuit specifications this is the longest phase in the Execution Process. During this time designers should:

- Develop transistor-level block schematics considering mismatch, mainly in differential circuits;
- Update testbenches to verify block level specifications and top-level functionality (steps 8 to 11 in Fig. 2). Simulate all specifications over corners and Monte Carlo and fix circuits when necessary;
- Calibrate behavioral models to match schematics behavior.

2.1.4. Physical Design

Physical implementation is performed in this phase according to foundry design rules. Parasitic components are extracted and circuits are simulated again using the same testbenches (item 11 to 15 in Fig. 1). The main phase objectives are presented below.

- Develop circuit physical implementation and generate abstract views with layout dimensions and pin positions to be used at top-level floorplan;
- Verify that circuit functionality is achieved over PVT corners is still valid after layout using preferably the same testbenches used in Circuit Design Phase;
- Perform block-level physical verification: Design Rules Check (DRC), Layout Versus Schematic (LVS), electromigration (EM), IR drop, latch-up, etc.

2.1.5. Design Verification

This is the time reserved for necessary block adjustments and top-level functional and physical verification (steps 13 to 16 in Fig. 2). At the end of the phase GDS2 standard format file is created. GDS2 is the final output of the design flow and is delivered to IC foundries for fabrication.

Physical verification (DRC/LVS/Parasitic Extraction) and Layout Versus Layout (LVL) is also performed on top-level, therefore on the complete IC design.

2.1.6. Post Mortem

During Post Mortem, technical meetings are realized to collect lessons learned during project and to better understand future improvements necessary in the design flow and management or IT infrastructure.

3. Technical Characteristics of the Design

The design developed in Phase 2 is the analog part of the PHY interface of IEEE 802.15.4 wireless protocol designed to support a future Zigbee implementation.

IEEE 802.15.4 standard addresses easy, low-cost power-friendly and flexible implementations of a virtually unlimited number of wireless low data rate monitoring and control applications. It is targeted towards wireless personal area network (WPAN) technology ranges and support industrial monitoring and control, home automation, sensor networks for gaming, medical and automotive solutions [4].

This design focuses on the 2.4 GHz PHY implementation since it is the frequency designated for Zigbee in Brazil. The basic characteristics of the selected IEEE 802.15.4 PHY implementation are presented in tab. 1.

Tab.1 – IEEE 802.15.4 Standard Characteristics [4] [5]

Parameter	2.4 GHz PHY
Sensitivity @ 1% PER	-85 dBm
Receiver Maximum Input Level	-20 dBm
Signal Bandwith	3 MHz
Output Power (Lowest Maximum)	-3 dBm
Operating channels	12 channels, from 2405 MHz to 2480 MHz
Data Rate	250 kb/s
Chip Rate (DSSS operation)	2 Mchip/s
Chip Modulation	O-QPSK with half-sine pulse shaping

As presented in tab. 1, PHY standard uses Direct Sequence Spread Spectrum (DSSS) with a chip rate of 2 Mchip/s and a data rate of 250 kb/s. Chip modulation is O-QPSK with half-sine pulse shaping (MSK), so quadrature modulation is used in the transceiver as can be directly observed in fig. 4.

Transceiver operates in a Time Division Duplex (TDD) basis, receiver and transmitter are alternatively turned on and off during communication using one of the programmed channels. Transceiver architecture is divided in top-level subsections to simplify design management and assigned to Block Leader Designers.

- Receiver Section – Receiver is formed basically by LNA, Receiver Downconverter and Receiver Buffer with offset correction and programmable gain. It implements an IF quadrature architecture with intermediate frequency equals to 2 MHz. Synthesizer clock is divided by two using CML logic for quadrature generation and phase noise reduction.
- Transmitter Section – composed by Power Amplifier, Modulator using quadrature architecture and Buffer. Direct conversion topology is adopted in order to avoid external filtering.
- Synthesizer Section – implements an integer programmable second order – type II PLL. Synthesizer is composed by the following basic blocks: Phase/Frequency Detector, Charge Pump, Loop Filter, Voltage Control Oscillator, VCO Buffer, Programmable Divider and Voltage Regulator. Reference clock comes from on-chip crystal oscillator. Due to the high frequency generated by the synthesizer, a hybrid CML/CMOS programmable divider is used. Frequency output from VCO Block is provided to

receiver and transmitter section through buffers to reduce LO pulling. VCO circuit operates with a separate regulation to improve power supply noise immunity.

- AMS Section – is composed by sub-blocks identified as ADC, DAC and PLL which are used in the analog baseband before signals are delivered to digital baseband. The Dual ADC is a two 8-bit 20 MHz Analog-to-digital converter with differential input stage and parallel output. The dual DAC consists of two 8-bit 20 MHz digital-to-analog converters (DAC). A Phase-locked Loop (PLL) is in charge of generating low jitter clocks for the ADCs, DACs and digital IF processing. Stable reference currents are provided by the Bandgap/Bias block using an external resistor.

The control block is in charge of VCO calibration and Analog Test Bus selection during test mode. Analog Test Bus provides testability function to each one of the blocks, allowing for the observation of DC Bias points and verification of block-level functionality. Control block also turns on and off blocks individually using digital registers during transmit, receive and testing operation to reduce power consumption in the transceiver.

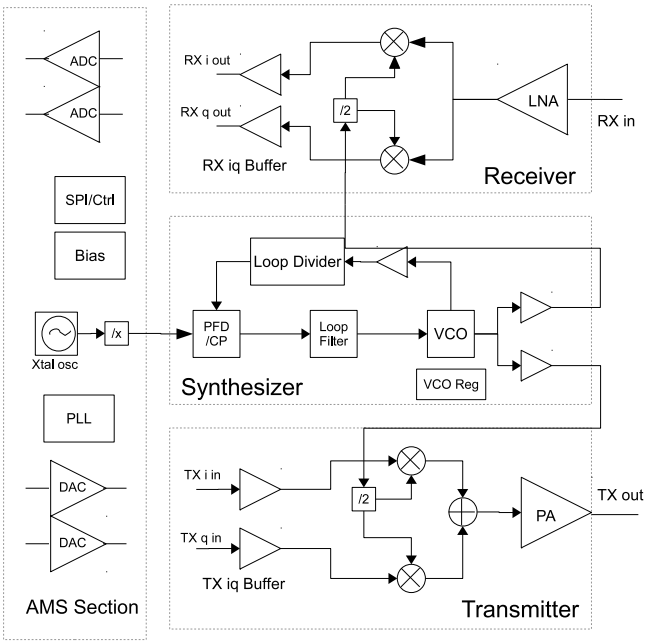


Fig. 4 – AMS-RF Proposed Architecture

4. Conclusions

The article presented the methodology and organizational structure developed to support the analog design phase in the Brazilian IC design training program at CT1. The methodology provides students with a real design experience and emphasizes project management processes. Design scope was also briefly presented as well as top-level characteristics.

5. References

[1] http://www.mct.gov.br/index.php/content/view/24595.html, accessed on March 20, 2011.

[2] http://www.nscad.org.br, accessed on March 20, 2011.

[3] "A Guide to the Project Management Body of Knowledge: (Pmbok Guide)", 4th Ed, Project Management Institute, 2008.

[4] "IEEE std. 802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)", available at <http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf>, accessed on January 20, 2011.

[5] Khanh Tuan Le, "Designing a ZigBee-ready IEEE 802.15.4- compliant radio transceiver", available at <http://mobiledevdesign.com/hardware_design/radio_designing_zigbeeready_ieee/>, accessed on January 20, 2011.

Energy-efficient Cache Coherence Protocol for NoC-based MPSoCs

[1]Tales M. Chaves, [1]Fernando G. Moraes
tales.chaves@acad.pucrs.br, fernando.moraes@pucrs.br

[1] PUCRS – FACIN – Av. Ipiranga 6681 – Porto Alegre – 90619-900 – Brazil

Abstract

System-on-Chip designs benefit from the advances in the semiconductor industry, integrating several processors, specialized hardware units and memory modules on a single chip. As the number of functionalities offered by embedded devices increases, the amount of memory used by these devices also increases. This fact justifies the development of memory architectures that present scalability, low energy consumption and low latency. Most designs, which adopt a Network-on-Chip (NoC) as the interconnection mean, do not explore the physical services of the NoC in the cache coherence protocol. Multicast, for instance, can be used to optimize these protocols, leading to both traffic and energy consumption reduction. The goal of this work is to optimize a directory-based cache coherence protocol exploiting specific NoC services, such as multicast and priorities. To demonstrate our proposal an MPSoC described at the RTL level is used, enabling accurate performance and energy evaluation. Results show a reduction of 17% in the number of clock cycles and an average reduction of 39% in energy consumption for memory transactions.

1. Introduction

Multiprocessor System-on-Chip (MPSoCs) integrates multiple processing elements (PEs) on a single chip to exploit task level parallelism. In addition to PEs, MPSoCs also integrate memory elements and specialized hardware units. Nowadays, most designs employ a NoC to interconnect these elements due to the scalability, high bandwidth and parallel communication offered by NoCs [1].

The increasing number of functionalities and applications of current embedded systems raise the demand for processing, high-speed communication and memory. According to Wolf et al. [2], one of the most critical components that determine the success of an MPSoC architecture is its memory architecture. This is justified by the fact that applications might spend several cycles waiting for the conclusion of read/write memory operations. The use of cache memories is a simple and efficient way to increase software performance [3]. According to [4] caches are and will be one of the best solutions to achieve low latency accesses to data and instructions. The introduction of cache memories in MPSoCs might reduce the average access latency, since most memory accesses are local, and the amount of memory transactions issued by processors to the communication infrastructure is reduced.

The cache coherence protocol may have a huge impact in energy consumption and latency. Most MPSoCs which adopt a Network-on-Chip (NoC) as the interconnection mean do not explore the physical services of the NoC to implement the cache coherence protocol. Multicast, for instance, can be used to optimize these protocols, leading to both traffic and energy consumption reduction. Jarger et al. [5] propose a novel protocol, named *Virtual Tree Coherence* (VTC), which is based on a virtual ordered interconnection tree and explores the multicast service provided by the NoC. Each tree keeps a history of nodes sharing a common region of memory. Bolotin et al. [6] attributes different priorities to packets transmitted by the cache coherence protocol. Operations such as read and exclusivity request (short data packets) are transmitted using high priority. Long packets, such as packets containing data to be written in the memory or a block just read from the memory are transmitted using low priority. Barroso et al. [7] propose an invalidation-based directory protocol which exploits priority to differentiate packets. A low priority lane is used by request sent to a home node, while the high priority lane is used by forwarded requests and all replies. Also, to decrease the traffic in the network, a technique called cuise-missile-invalidates is used for sending a unique invalidation message to several nodes.

In this work, we propose the optimization of the MSI directory-based cache coherence protocol, which explores two physical channels and the dual-path multicast algorithm [9]. Also, multicast is used to reduce traffic generated by the cache coherence protocol, and priority services increase performance. Multicast messages can be used by cache coherence protocols when invalidation messages must be sent to all processors that are caching a given block. Without multicast, an invalidation message would have to be sent individually to all processors in the system, increasing the number of transactions in the network, as well as energy consumption and congestion. Priorities may be used to differentiate memory transactions, decreasing the response time to higher priority messages, as block invalidation.

Despite longer simulation times, this work adopts RTL modeling, since it enables accurate performance evaluation. The rest of this paper is organized as follows. Section II presents the architecture of the MPSoC

adopted by this work. Section III details the proposed optimizations in the cache coherence protocol. Section IV presents the experiments and results. Finally, Section VI concludes this paper.

2. MPSoC Architecture

This work adopts as reference a homogeneous NoC-based MPSoC [8], described in synthesizable VHDL. Each PE contains a MIPS-like processor (Plasma), a local memory (RAM), a DMA controller, a Network Interface and, optionally, a L1 cache memory and a shared L2 cache memory. A general view of a 2x2 instance of the MPSoC architecture is illustrated in Figure 3. Tasks communicate through message passing or through a shared L2 cache memory (when it exists).

PEs can be categorized in two types: slave and master. Slave-PEs are responsible for executing application tasks, while the Master-PE is responsible for mapping the tasks into the Slaves-PEs, task management and system debug. Each Slave-PE runs a tiny operating system, named *microkernel,* responsible for managing task execution and task communication. The network interface and DMA modules are responsible for sending and receiving packets. The 2-D mesh NoC used has the following features to support QoS: packet and circuit switching, priorities, duplicated physical channels (two 16-bit bidirectional links), and multicast.

A L1 cache can be used only when there is at least one shared L2 cache module in the MPSoC. These memories are used only for storing data. The local memory (RAM) stores the microkernel, application tasks and private data of each PE. The cache L1 memory stores private copies of blocks from the L2 cache, and it is managed by the cache controller. The L2 cache is shared among all PEs and is composed of: (*i*) a memory controller; (*ii*) a directory memory; (*iii*) a network interface (NI) and; (*iv*) a memory bank. Note that more than one L2 cache bank can be used in the system, resulting in a distributed shared memory (DSM) organization. In the scope of this work, only one L2 cache is used. All necessary instructions to execute a given task are stored in the local memory.

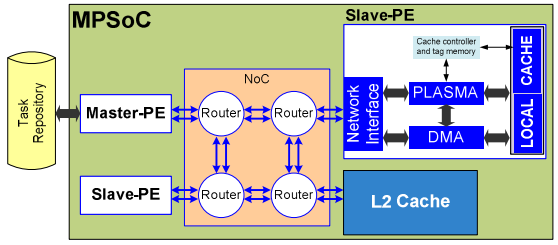


Figure 3 - MPSoC with a two level memory hierarchy – shared memory and caches (only Slave-PEs may contain caches).

Table 3 - Number of clock cycles and energy consumption of invalidate messages depending on the number of caches sharing a block.

	Platform	*3 caches*	*5 caches*	*8 caches*
Energy (nJ)	NO-OPT	1635	2584	3798
	OPT	685	2073	2916
	OPT gain vs NO-OPT	58.07%	19.76%	23.20%
Clock Cycles	NO-OPT	141	154	147
	OPT	129	127	129
	OPT Gain vs NO-OPT	8.51%	17.53%	12.24%

3. Cache Coherence Protocol

Cache coherence protocols are commonly divided in two classes: directory-based and snoopy-based protocols. The MSI cache coherence protocol is a directory-based protocol, where the directory keeps the current state of each block shared between PEs. Each shared block may be in one of the following states: *shared*, *exclusive* or *invalid*. In NoC–based systems, where the NoC offers physical services such as multicast, priority, and two physical channels, the MSI protocol can be optimized. To decrease the amount of traffic generated by the cache coherence protocol, this work identified four situations where the protocol can be optimized. Most of these optimizations are allowed by the physical services available in the NoC.

3.1. The Transition State

We propose the addition of a new state in the MSI protocol, named transition state (T). A block enters this state when the L2 cache controller receives a read request on a given block in exclusive state. The block stays in this state until it is updated in the L2 memory. Any additional read request arriving at the L2 cache controller when the block is in the T state is mirrored to the processor which is writing the block back to the L2 cache.

3.2. Invalidating cache lines

Before writing to a given block, a PE must acquire exclusivity on it. In order to do so, a request must be sent to the L2 cache controller. An exclusivity request on a shared block triggers the dispatch of invalidation messages to all PEs currently sharing the block. Most implementations of MSI protocol dispatch a unicast message for each PE sharing the block, which might represent a significant increase on the traffic in the NoC. In this case, multicast messages can be explored. Instead of sending several unicast messages, in the HeMPS MPSoC, in the worst scenario, at most two multicast messages are sent, independent of the number of PEs sharing the block. This is allowed by the Hamiltonian multicast algorithm implemented by the NoC.

3.3. Read request optimization

The use of multicast messages might optimize read operations on modified blocks. In the standard MSI protocol, read operations on modified blocks require the issue of a write-back request from the L2 cache controller to the PE holding the modified copy of the block. The cache controller only responds the read request after receiving and updating the block locally. Multicast messages can be exploited in this scenario, both to reduce the number of packets transmitted and increase performance of read operations.

Upon receiving a write-back request, the processor holding the modified copy of the block might send a multicast message to the L2 cache memory and also, to the PE which is trying to read the block. In some situations, it is not possible to send only one multicast message due to the limitations of the Hamiltonian routing algorithm and the labeling of the PEs/SM.

3.4. Write request optimization

To write on a shared block, a PE must read it beforehand (copy the block to its local cache). If the block to written is in modified state, a write-back operation must be performed by the PE holding the modified copy. Instead of sending the write-back response to the L2 cache controller, the PE might send its local copy of the block directly to the processor which wants to write on it. In this case, the L2 cache can be bypassed because the block will be modified right after. This optimization does not rely on any specific characteristic of the NoC.

3.5. Priority exploitation

The HeMPs platform allows two levels of priority for messages at the NoC level: high and low priority. High priority messages are routed first than low priority ones. Memory requests, such as read, read-with-exclusivty and exclusivity requests, are sent using high priority. These requests are short. Write-back responses are transmitted using low priority. This choice is based on the fact that if requests are transmitted using high priority, they will be attended earlier, decreasing latency of memory operations.

4. Experiments and Results

To evaluate the gains of the optimized MSI protocol, two different implementations of an MPSoC platform were simulated in RTL-level using the ModelSim simulator. The platform used as a case study is configured as follows: 5x5 NoC mesh topology, containing 24 PEs (1 master and 23 slaves) and 1 SM. The performed experiments analyze each proposed optimization separately against the non-optimized protocol. By doing this, it is possible to verify the gains individually. In all experiments, the results show the number of clock cycles, and the energy spent in communication between the PEs and the SM. This is justified by the fact that we focus mainly on the services offered by the NoC to optimize cache-coherence actions, therefore gains are obtained by decreasing communication cycles and energy consumption. The packets containing memory operations are triggered by application tasks. To evaluate the consumed energy per memory transaction, the present work adopts the volume-based energy model proposed by Hu et al. [9].

4.1. Invalidating cache lines

To evaluate the benefits of using multicast to propagate these messages, the number of caches sharing a copy of the same block of the SM varies. We evaluated three different scenarios, where a shared block is cached in: 3, 5 and 8 caches, respectively. The number of cycles and energy consumption are summarized in **Table 3**. Although with a smaller number of targets to invalidate, the first scenario (3 caches sharing a block) presents the best gain relative to the non-optimized implementation. This is due to the task mapping on the platform which allowed the sending of only one multicast message, which significantly reduces the amount of data transmitted on the NoC. For the other scenarios (5 and 8 caches sharing a block), the use of multicast messages saves energy and improves performance by at most 17.53%.

4.2. Other optimizations

The read request optimization showed for all experimented scenarios that the energy consumed is reduced up to12% in comparison with the NO-OPT implementation.

Figure 4 shows the energy spent during the operation varying the distance of the PE which requests the read and the L2 cache. For this optimization, particularly, the NO-OPT implementation takes, in average, 30 clock cycles less than the OPT to complete the read request, due to the higher complexity to treat multicast packets at each router, and the non-minimal path taken by these packets.

Figure 5 shows the results for the write request optimization. Results show that there is an average reduction of 17% in the number of cycles required to finish the write operation. Also, Figure 6 shows that there is a reduction of up to 86.8% on the energy spent during this operation by the OPT implementation over the NO-OPT. This reason of this significant reduction is that long messages, containing data blocks, are transmitted only once, from PE to PE.

The transition state optimization results show that the gains against the standard MSI protocol are really sensitive to the task and SM mapping. In scenarios where the PE that issues the second read request is closer to

the PE previously holding the modified copy of the block, there are gains both in performance of the protocol (decrease in clock cycles) and also a save on the energy spent during the operation.

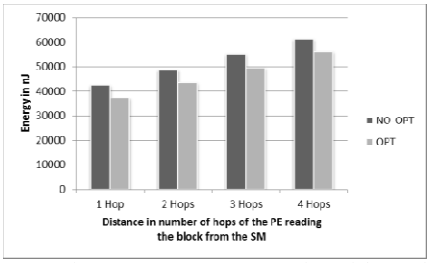


Figure 4 – Energy consumption of the read operation on a modified block as the number of hops increases.

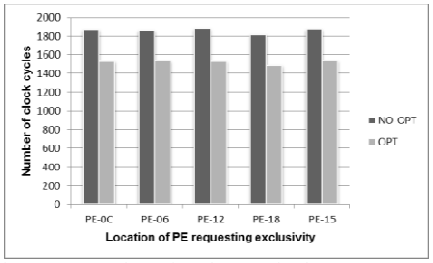


Figure 5 - Number of cycles required to execute a read operation on a modified block varying the location of the modified block.

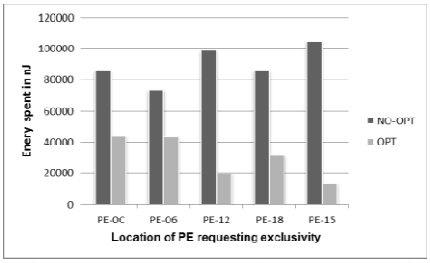


Figure 6 - Energy consumed to execute a read operation on a modified block varying the location of the modified block.

5. Conclusions

This work attempts to explore the benefits NoCs can bring to cache-coherence protocols, evaluating a complete system at the RTL level (PEs and the NoC), including the software (*microkernel* and applications) running on top of it. By using the proposed protocol optimizations, results show that it is possible to reduce the energy consumed by the operations up to 86.8% (average reduction: 39%) and to achieve an improvement of 17.53% in the execution time (clock cycles). All optimizations, except the Transitions state, always reported energy reduction. The Transition state optimization is sensible to the task mapping. This fact points to several future works, as couple the proposed techniques to mapping heuristics that consider the memory position in the MPSoC, and data migration policies to optimize the memory performance.

Future work also includes: (*i*) use of parallel benchmarks to characterize performance; (*ii*) extend the number of memory-IPs in the memory hierarchy, resulting in a DSM architecture; (*iii*) study and implement a way to distribute (or to hierarchize) the directory used by the cache coherence protocol; (*iv*) evaluation and implementation of a memory consistency model at the software level.

6. Referências

[1] Millberg, M.; Nilsson, E.; Thid, R.; Kumar, S.; Jantsch, A. The Nostrum backbone-a communication protocol stack for Networks on Chip. *17th International Conference on VLSI Design*, 2004. Page(s): 693 – 696.

[2] Wolf, W.; Jerraya, A. A.; Martin, G.; Multiprocessor System-on-Chip (MPSoC) Technology. *Computer-Aided Design of Integrated Circuits and Systems*. vol.27, no.10, pp.1701-1713, Oct. 2008.

[3] Petrot, F.; Greiner, A.; and Gomez, P. On Cache Coherency and Memory Consistency Issues in NoC Based Shared Memory Multiprocessor SoC Architectures. In: *EUROMICRO*, pp. 53-60, 2006.

[4] Leverich, J.; Arakida, H.; Solomatnikov, A.; Firoozshahian, A.; Horowitz, M.; Kozyrakis, C. Comparing memory systems for chip multiprocessors. *SIGARCH Comput. Archit. News* 35, 2 (June 2007), 358-368. 2007.

[5] Jerger, E. N. D.; Peh, L.; Lipasti, M. H. Virtual tree coherence: Leveraging regions and in-network multicast trees for scalable cache coherence. In *Proceedings of the 2008 41st IEEE/ACM international Symposium on Microarchitecture*, November 08 - 12, 2008, pp. 35-46.

[6] Bolotin, E.; Guz, Z.; Cidon, I.; et al. The Power of Priority: NoC Based Distributed Cache Coherency. *In International Symposium on Networks-on-Chip (NOCS'07)*, pp.117-126. 2007.

[7] Barroso, L. A.; et al. Piranha: a scalable architecture based on single-chip multiprocessing. In *Proceedings of the 27th annual international symposium on Computer architecture (ISCA '00)*, pp 282-29. 2000.

[8] Carara, E., Oliveira, R., Calazans, N., Moraes, F. "HeMPS - a Framework for NoC-based MPSoC Generation". In: ISCAS, pp.1345-1348, 2009.

[9] Carara, E.; Moraes, F.; "Deadlock-Free Multicast Routing Algorithm for Wormhole-Switched Networks-on-Chip". In: ISVLSI, pp. 341-346, 2008.

[10] Hu, J.; at al. Energy-aware mapping for tile-based NoC architectures under performance constraints. In: *ASP-DAC'03*, 2003, pp. 233-239.

Digital Logic Cancellation Block for a Cascade Feed-Forward Sigma-Delta Analog-to-Digital Converter

Paulo César C. de Aguirre, Felipe C. Lucchese, Lucas Teixeira, Crístian Müller and César Augusto Prior
{paulocomassetto, felipelucchese, lucasteixeira, cristian.muller}@mail.ufsm.br,
cesar.prior@ieee.org

Grupo de Microeletrônica – Gmicro
Universidade Federal de Santa Maria - UFSM

Abstract

This paper presents the characterization and synthesis results for a hardware implementation of a reconfigurable digital logic cancellation circuit auxiliary for multi-mode ΣΔ modulator that is capable to perform the analog-to-digital conversion for GSM, CDMA and WLAN standards. The ΣΔ modulator reconfigures its mash topology and building blocks in order to adapt the performance to the diverse standard specifications. The necessary integration and cancellation at three modulators output for baseband signal processing was designed in high level Matlab/Simulink and coded with VHDL for synthesis intended to prototyping in CMOS process.

1. Introduction

The evolution of band-base analog-digital conversion in telecommunications systems and signal processing lead with multi-mode capability of standards, comes from the 2G systems with high quality services for 3G systems, which include global system for mobile communications (GSM) and a wide-band code division multiple access system (WCDMA) to wireless systems that incorporate both Wireless Local Area Networks (WLAN) and cellular capability. Typical triple-mode base-band architecture and the required bandwidth to deal with GSM, CDMA and WLAN signals are shown in Fig.1 and Tab.1, respectively. In this context a multi-mode cascaded ΣΔ architecture have been reported in [1] whose wide programmability range of input frequency and dynamic range descends from modulator order programmability. Another reconfigurable ΣΔ modulator for a triple standard receiver has been introduced in [2] where a feedback path from the last to the third stages is done in order to further suppress the quantization noise power. Yet another multi-standard sigma-delta ADC has been explored in [3].

To allow the multi-mode cascade ΣΔ architecture a reconfigurable digital logic cancellation circuit is needed. This circuit is responsible for the modulators output signal processing in order to adapt the cascade architecture performance to diverse standard specifications. The development, hardware implementation and synthesis results of this reconfigurable architecture using two different technologies are presented.

The paper is organized as follows. Section 1 is the introduction. Section 2 focuses on selecting the appropriate architecture for the multi-standard specifications. Section 3 describes a logic cancellation function. Section 4 provides the implementation and simulation results. Finally, Section 5 concludes the paper.

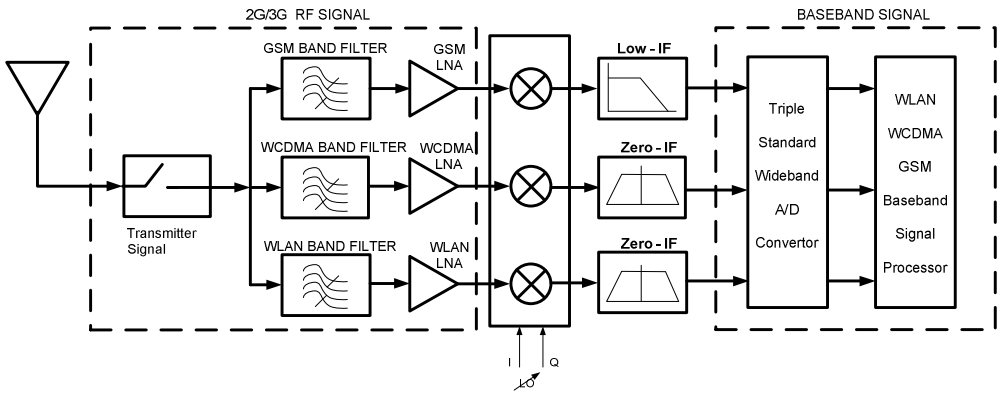


Fig. 1 – Typical Triple-mode base-band Architecture.

Tab.1 - Requirements for baseband

Standard	Frequency (MHz)	Chanel Banwidth	Dynamic Range
GSM	890-915(Tx) - 935-960(Rx)	200kHz	80dB
WCDMA	1850-1910(Tx) - 1920-1980(Rx)	5MHz	60dB
WLAN	2401-2473	20MHz	50dB

2. The overall circuit topology

A proposed cascaded second order single bit ΣΔ modulator 2-2-2 structure is shown in Fig. 2. In order to adapt the system modulator performance to GSM, CDMA and WLAN standard specifications the cascade topology and the logic cancellation block can be reconfigured to provide a second, fourth and sixth order system modulator. A Matlab/Simulink whole system simulation was performed and the simulation results shown that these bandwidth requirements, presented in Tab. 1, can be achieved at a frequency of 160 MHz and with the coefficients showed in Tab. 2.

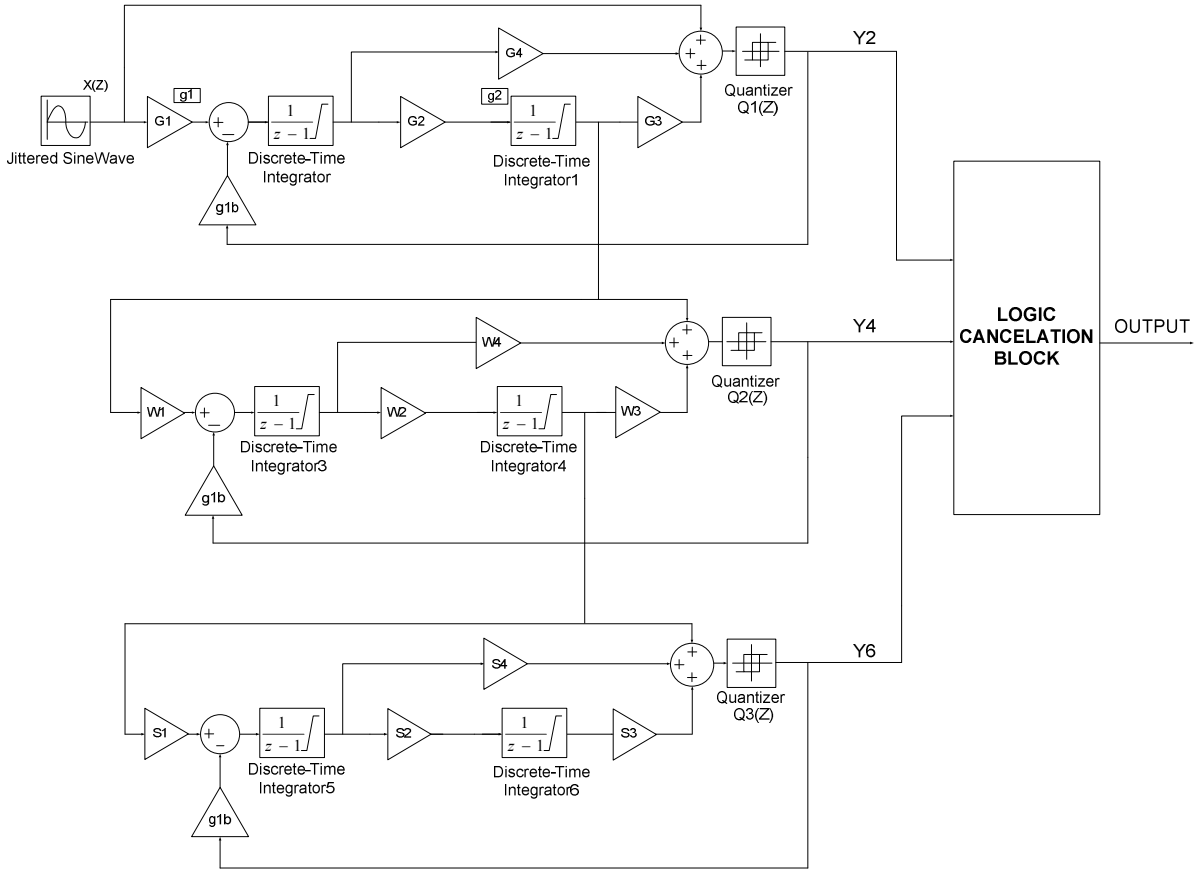


Fig. 2 – Mash Triple-mode SD modulator Architecture.

The system is composed for three second-order ΣΔ analog-to-digital converters based in a low distortion topology that was presented in [5]. The GSM output mode is directly obtained at the first stage output and the Eq. (1) presents the system transfer function:

$$Y_2(z) = Y_{GSM}(z) = X(z) + (1 - z^{-1})^2 Q_1(z) \tag{1}$$

The CDMA mode, Eq. (4), is obtained with a fourth order system modulator. The system second stage output is given by:

$$Y_4(z) = I_2(z) + (1 - z^{-1})^2 Q_2(z) \tag{2}$$

Where,

$$I_2(z) = -g_1 g_2 z^{-2} Q_1(z) \tag{3}$$

Tab. 2 – Coefficients of ΣΔ mash 2-2-2 architecture

Coefficients	$g_1/g_{1b}/w_1/s_1/g_2/w_2/s_2$	$g_3/g_4/w_3/w_4/s_3/s_4$
GSM	0.5	4
WCDMA	0.5	4
WLAN	0.5	4

Then,

$$Y_{CDMA}(z) = z^{-2}X(z) + g_5(1 - z^{-1})^4 Q_2(z) \tag{4}$$

To cancel this output and obtain the transfer function for a fourth order modulator based in 2-2 cascade structure, Eq. (5):

$$Y_{CDMA}(z) = X(z) + (1 - z^{-1})^4 Q_2(z) \tag{5}$$

The logic cancellation block must perform the transfer functions Eq. (6) and Eq. (7):

$$H_1(z) = z^{-2} \tag{6}$$

$$H_2(z) = g5(1 - z^{-1})^2 \tag{7}$$

Similarly, for the WLAN mode, a sixth order system modulator is needed. The third stage output (Eq. (8)), the transfer function for a sixth order modulator (Eq. (8)) and the operations to be performed by the logic cancellation block (Eq. (10) and Eq. (11)) are, respectively:

$$Y_6 = z^{-4}X(z) + g_6(1 - z^{-1})^6 Q_3(z) \tag{8}$$

$$Y_{WLAN} = X(z) + (1 - z^{-1})^6 Q_3(z) \tag{9}$$

$$H_3(z) = z^{-2} \tag{10}$$

$$H_4(z) = g_6(1 - z^{-1})^2 \tag{11}$$

The digital logic cancellation architecture block diagram is shown in Fig. 3 and its functionalities are explained in Section 3.

3. The Logic Cancellation Circuit

The logic cancellation circuit operates at a 160 MHz frequency, 6.25 ns period, and it is composed for three kinds of functions: Gain, Delay and Differentiator. This block, illustrated previously in Fig. 2, is simplified in Fig. 3. The Y2, Y4, and Y6 are single bit inputs and the Y(OUT) is a ten bit array output.

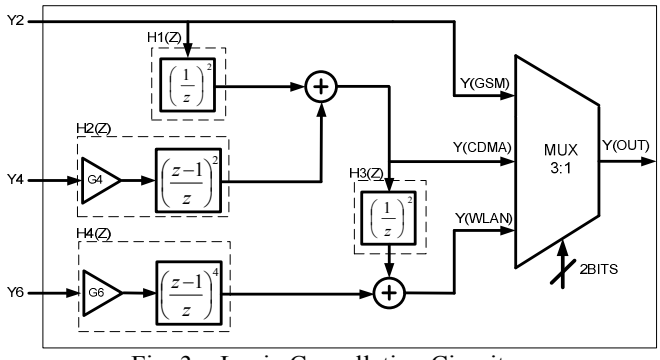


Fig. 3 – Logic Cancellation Circuit.

The H1(z) function implements a two clock cycle delay in the Y2 signal. The H2(z) function executes in a single clock cycle one gain and two differential operations. In the first operation a gain of 4 is applied at the Y2

input and this signal is the input of the first of two differentiators where the previous signal is subtracted from the actual signal in each differentiator operation. Adding the outputs of H1(z) and H2(z) the CDMA bandwidth output is achieved.

The additional blocks H3(z) and H4(z) are necessary for the WLAN bandwidth output. The H3(z) block implements a two clock cycles delay in the CDMA signal and the H4(z) implements a gain of 16 at the input Y3 and executes four differentiator operations, all in the same clock cycle. Hence, to reach the WLAN bandwidth the H3(z) and H4(z) functions must be added.

To choose the modulator order output needed by the user a simple multiplexer is presented at the digital cancellation block output.

4. Implementation and Simulation Results

The designed digital architecture was described in VDHL and synthesized for two different technologies: Altera Stratix II EP2S60F672C3N FPGA and X-FAB XC06 5V CMOS Process, using Quartus II and Synopsys Design Compiler tools, respectively.

Tab. 3 shows the synthesis results for the entire developed module considering maximum frequency operation and hardware consumption (number of DLRs and ALUTs for FPGA and logic cell number for standard cells).

Tab. 3 – Synthesis Results

Stratix II EP2S60F672C3N			**X-FAB XC06 (0.6µm)**		
Frequency	#ALUTs	#DLRs	Frequency	#Logic Cells	# NAND2 Equivalent Gates
575.04 MHz	35	36	195.7 MHz	254	604

In order to compare the architecture in high level (Matlab/Simulink) and hardware design (VHDL design) the stimulus injected by the testbench in the DUT were the same applied in the high level block. The same verification environment was used to validate the synthesized circuits. These stimulus were obtained from the three outputs of a mash sigma-delta AD converter structure simulated in Simulink, see Fig. 2.

5. Conclusions

This paper showed the design, validation and synthesis results of a digital logic cancellation architecture for a cascade ΣΔ AD converter. To achieve the CDMA and WLAN standard modes operation the quantization noise in the operation frequency range must be reduced. It is provided by the increment of the system modulator order.

The frequency requirement of the digital architecture, 160 MHz, was achieved for X-FAB XC06 standard cells technology.

It was observed that the circuit power consumption can be reduced both in the analog and digital circuits disabling modules when not needed.

The future works in the project will be the creation of a digital reconfigurable filter that will cover the three standard operation modes of the proposed system modulator and the power consumption reduced disabling analog and digital blocks when they are not being used.

6. References

[1] B. R. Jose, J. Mathew, P.Mythili and D. K. Pradhan, “A Multi-Mode Sigma-Delta ADC for GSM/WCDMA/WLAN Applications”, J Sign Process Syst (2011) 62:117–130, DOI 10.1007/s11265-008-0326-z.

[2] Andrea Xotta, Andrea Gerosa and Andrea Neviani, “A Multi-Mode ΣΔ Analog-to-Digital Converter for GSM, UMTS and WLAN,” IEEE International Symposium on Circuits and Systems (ISCAS 2005), May 2005, vol.3, pp. 2551-2554.

[3] Ling Zhang, Vinay Nadig and Mohammed Ismail, “A High Order Multi-Bit ΣΔ Modulator for Multi-Standard Wireless Receiver”, IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2004.) June , vol.3, pp. 379-382.

[4] B. Jalali-Farahani, and M. Ismail, “A Low Power Multi-Standard Sigma-delta ADC for WCDMA/GSM/Bluetooth Applications,” *IEEE Northeast Workshop on Circuits and Systems (NEWCAS 2004)*, June 2004, pp.241-243.

[5] J. Silva, U. Moon, J. Steensgaard, and G. C. Temes, “Wideband low-distortion delta-sigma ADC topology,” *Electronics Letters* , June 2001, vol. 37, no. 12, pp. 737-738.

Efficient Processing Element Unit for MPSoC NoC-based

[1,2]Paulo Santos, [1]Jonathan Martinelli, [1]Cezar Reinbrecht,
[1]Débora Matos, [1]Altamiro Susin
paulo-junior@uergs.edu.br, {jrmartinelli, cezar.reinbrecht, debora.matos}@inf.ufrgs.br,
altamiro.susin@ufrgs.br.

[1] Universidade Federal do Rio Grande do Sul
[2] Universidade Estadual do Rio Grande do Sul

Abstract

The evolution of the VLSI technologies allowed the development of more dense integrated circuits and complex designs resulting in Multi Processors Systems on Chip (MPSoCs). An MPSoC may consist of several of Intellectual Properties (IP) such as CPU or DSP cores, high-bandwidth I/O, memory subsystem, etc. This great number of resources became this architecture very popular in embedded systems. On the other hand, it offers several challenges to integrate these elements. The solution widely accepted and established for the interconnection is the Network on Chip (NoC). However, NoC imply in design the interface for each IP, where in many cases results in wrappers. In this context, we propose a generic processing element capable to supply all requirements of the several applications and allow better time to market, reducing the design effort. The main goal of this paper is to present the concept and the proposed design of a generic interface for processing element in order to compose an MPSoC NoC-based.

1. Introduction

The advance of deep-submicron technology and the increasing in the microelectronic systems complexity allowed the development of SoC designs. This idea allows building in a single integrated circuit a complete computational system, with a higher number of functionalities and complexity level [1, 2]. Furthermore, the trend for the processing parallel brought the concept of Multi-Processor Systems on Chip. This new paradigm may comprehend a great heterogeneity of devices, like RISC, VLIW, DSP and ASIP processors or even dedicated IPs. On the other hand, the intercommunication method becomes a new challenge, due to the increasing of nodes in the communication. In the last years, the common solution adopted for this issue is the network on chip.

In a near future, the systems will be composed by hundreds of IPs and the performance required for these systems will be elevated. As the NoCs allow connecting several and heterogeneous IPs [3, 4], they have been considered a scalable solution [1, 2] and for this reason, many related proposals have been presented in the literature. However, in order to integrate any IP with a NoC it is necessary to design wrappers and considering hundreds of heterogeneous IPs, the design costs could be prohibitive. Therefore, the processing elements have to be generic and customizable enough to "plug" in any design and "play" any application supporting its requirements, with minimum rework. Our idea is that this generality can be achieved with an architecture that provides a generic NI (Network Interface) and IP management unit where all communication issues are hidden from application.

In the state of the art, it can be observed several proposals of network interfaces to allow the integration between IP and Network on Chip. Nevertheless, these ideas use the concept of wrapper or resources sharing, increasing area or decreasing performance. Our proposal differs from the others because our goal is to avoid wrappers for each new IP architecture, using an element common in any IP, the memory.

This paper is organized as follows. In section 2 we present some related works. The proposed interface architecture is described in section 3. The tests and results are showed in section 4 and the conclusions at section 5.

2. Related Works

Singh [5] proposed an architecture of a network interface in which most of the interfacing functionality is incorporated in the generic part of the interface. In this manner, different cores can be attached in the nodes of a NoC with minimum redesign. This proposal guarantees a plug and play feature with a generic logic allied with specific wrappers. The generic logic is responsible for making transparent the network protocol and packet managing, and the specific wrapper is responsible to adapt the generic protocol to a specific IP interface.
However, this design effort is concentrated in the wrapper development, which is a weakness point of the proposal since, as the wrapper is embedded to NI, it is always used to connect a module even when it has a simple interface.

Another approach of network interface was presented in [6], where the authors proposed techniques to share hardware resources to save area. Its network interface uses the Open Core Protocol (OCP) integrated on the IPs. The trade-off is a timing cost once an application with hard communication rates requirements could imply in loss of performance and possible functional problems. It can occur because more than one IP can be connected in a single NI and thus, it requires the use of others control circuits like an arbiter to define the IP that can send a data word to the interface.

Following the idea of sharing resources, in [7] the authors proposed a network interface with shared-memory abstraction including a transaction based protocol to connect the IPs. The proposed NI configuration requires a very complex logic, which implies in development time. Besides, IPs with simplified protocols needs a great effort to connect in this architecture.

3. Generic Processing Element Architecture

Generic Processing Element (GPE) is a conceptual architecture which offers mechanisms to allow a suitable integration between IPs, even when they are heterogeneous. With the aim to compose a GPE, four basic modules are needed: an IP, a local memory, a network interface and an integrator manager.

In order to provide an appropriate communication with different IPs, it was developed an integrator manager based in memory mapping interface. This module is called as MMI (Memory Mapped Interface) and can be used for different processing elements, since the majority of them use external local memories.

3.1. Proposed Architecture Design

Fig.1 shows a blocks diagram with the structure of the proposed processing element (PE) with the MMI used as interface between NI and IP.

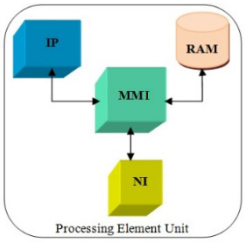


Fig.1: Proposed processing element with the NI connected by a general purpose processor by the MMI

As shown in fig.1, the MMI manages the communication inside the PE Unit (IP, RAM and NI), selecting and activating the module responsible by transferring data according to the need of the system. MMI manages the access to RAM by the data bus and by the address bus (in this case, are considered separated buses) as well as the control flags used for the reading and writing in the RAM.

Considering that the RAM will be accessed directly by the MMI and the IP, and that the NI will communicate only with the MMI, the memory mapped interface will be able to read and write in the RAM. In this case, the access to RAM will be made by the MMI only for reading and writing, according to the direction of the communication. To avoid conflicts, the MMI uses a hold enable in the IP.

3.1.1. Memory-mapped

With the purpose to save resources, the implementation and the transfer control between IP and MMI were made using a memory mapping where some addresses were reserved. This memory mapping allows the communication managing to be made by software, as shown in fig.2.

0xEFFF FFFF	VPB PERIPHERALS	0xE006 001E	ADDR_SOURCE_NI_IP
		0xE006 001A	ADDR_TAG_NI_IP
		0xE006 0016	ADDR_DATA_NI_IP
		0xE006 0012	ADDR_FLAG_WRITE
		0xE006 0008	ADDR_TARGET_IP_NI
		0xE006 0004	ADDR_DATA_IP_NI
0xE000 0000		0xE006 0000	ADDR_FLAG_READ

Fig.2: Reserved Addresses on IP for the control used by the MMI.

The reserved addresses correspond to the addresses available in VPB (VLSI Peripheral Bus) area in the IP core. As there are several addresses for sending and receiving of data, the defined protocol uses addresses for data, for target/source information and for the control flags.

The developed MMI presents two internal modules, MMI_IP_NI and MMI_NI_IP: one responsible to management the data for IP to NI and the other, in the contrary direction: for NI to IP, respectively. The top module is responsible for managing the multiplexers and demultiplexers changing the addresses and the RAM signal bus.

3.1.2. IP to NI module

The monitoring of the IP address bus is made to define the direction of the communication. When the IP writes at address ADDR_FLAG_READ, the MMI_IP_NI module is informed that a new data is available. In this case, MMI_IP_NI module reads the data and target from addresses ADDR_DATA_IP_NI and ADDR_TARGET_IP_NI. The module checks if there is some available NI FIFO slot and this information is indicate by FIFO_FULL control. When some slot in the FIFO is available, the MMI_IP_NI writes the data in the FIFO and update the ADDR_FLAG_READ of the RAM, informing to IP that it can write another data. While the MMI_IP_NI module communicates with the NI or waits for the available NI FIFOs, the RAM is available for the use of the IP, and while ADDR_FLAG_READ was not checked updated by the MMI, the IP will not write a new data. In fig.3a the flowchart of the MMI_IP_NI is showed.

3.1.3. NI to IP module

When the monitoring recognizes the ADDR_FLAG_WRITE address in the address bus, the module MMI_NI_IP identifies that the IP is waiting for a new data from NI. However, there is data available on the NI to be written on the IP. In this case, the MMI_NI_IP reads the data, the source and the tag from NI and it writes this information in the ADDR_DATA_NI_IP, ADDR_SOURCE_NI_IP and ADDR_TAG_NI_IP addresses, respectively. Once the MMI_NI_IP writes the data, the MMI_NI_IP writes in the ADDR_FLAG_WRITE address indicating to the IP that new data is available. Otherwise, the RAM remains available to the IP while the MMI_NI_IP reads data from NI.

When the MMI modules access the RAM at any time, the IP is put in *hold* mode. When the RAM is returned to IP, it resumes the execution of instructions without RAM data loss. The use of the *hold* mode is an important step for the MMI, since the RAM can only be written or read by a module at a time. A solution for that problem would be to use of a Dual Port RAM. Although the use of Dual Port RAM would avoid the use of hold mode, it would increase the area, which is not the focus in this work. The flowchart of MMI_NI_IP as described above is showed in fig.3b.

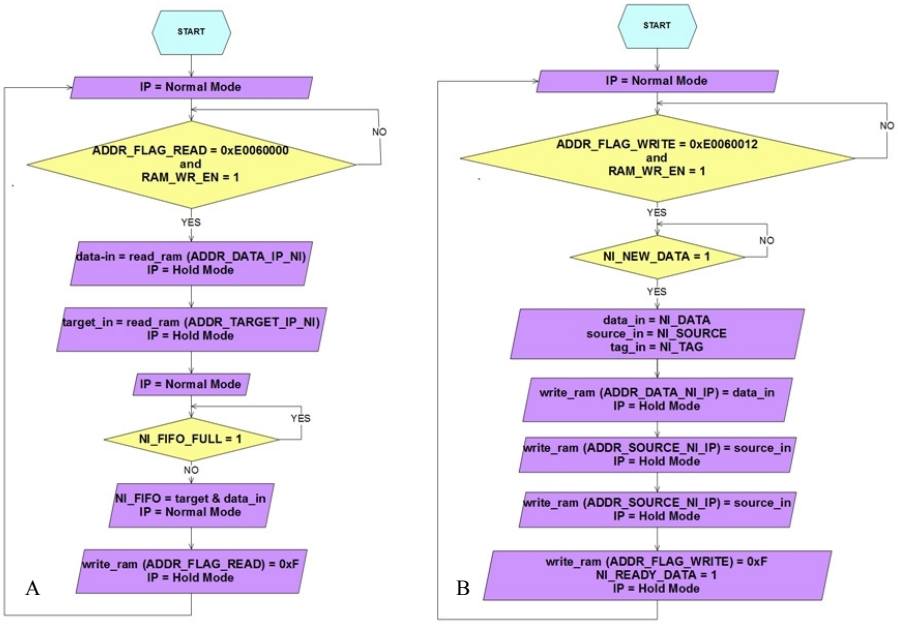


Fig.3: a) Flowchart of the MMI_IP_NI. b) Flowchart of the MMI_NI_IP

4. Experimental Results

4.1. Setup Description

The experiment consists of an MPSoC running an application, where it contains four PE interconnected through a NoC, as shown in fig.4.

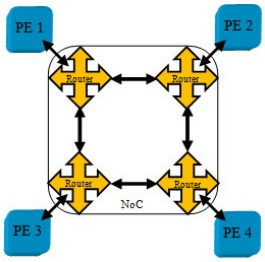


Fig.4: Tested system - MPSoC with NoC used on tested system

The PE consists of an IP, a RAM memory module, an MMI and an NI as shown in fig. 1. The IP used is an ARM7TDMI core with ROM memory. The RAM is modeled with a register bank. The NoC used is the SOCIN NoC [8]. The NI used is a generic network interface [9] that translates IP data to NoC protocol and vice-versa. This experiment was implemented in VHDL and simulated using the Modelsim simulator. In addition, this design was synthesized for standard cells in a 65nm technology using Cadence Synthesis tools.

4.2. Experiment Description

The application consists in a multiplication between two square matrices 3x3, whose elements are integers. The first matrix has each column distributed among the PE1, PE2 and PE3. The second matrix was fully initiated in PE4. The algorithm used was simpler algorithm known, multiplication and summing between the row elements of the first matrix and the column elements of the second matrix.

The PE4 starts forwarding the elements of its column to PE1, PE2 and PE3, which performs the multiplications and sums. After these calculations, the PEs returns the results to PE4. All data are 32 bits and the links inside of the PE are 32 bits too. To maintain a 32bits data and instructions, the compiler and the IP ARM has the Thumb mode disabled.

For comparison, a matrix multiplication program was written using the same algorithm and same matrices and run on a single ARM core. In the tab.1, a comparison with the single core and quad-core with NoC is shown.

Tab.1 – Comparison between single core and multi core.

DESIGN	Module	Latency (ns)	Area (um²)	Power @100MHz (mW)
Single Core	ARM	11495	40731	1.553
Multi-Core NoC	ARM	468445	40731	1.553
	MMI		933	0.05
	NI		934	0.06
	GPE		42598	1.564
	NOC		15733	1.433
	MPSoC		186125	7.689

5. Conclusions

In this work we have presented a proposal of a PE unit. We have run a matrix multiplication using four nodes connected in a mesh NoC, validating the proposal. However, a single 3x3 matrix multiplication is not the proper application to demonstrate the potentiality of the parallelism of this system. Hence, the costs of communication latency through a NoC were higher than the execution in a single core, shown in Tab 1. The GPE implemented has 4.5% increase in area and only 0,7% increase in power consumption at 100MHz compared with the IP, which means a low cost for a mechanism that enables a construction of a generic MPSoC.

6. References

[1] Benini L. and De Micheli G. “Powering Networks on Chips”. International Symposium on Systems Synthesis, Montreal, pp. 33-38, 2001.

[2] Kumar, S. et al. “A Network-on-Chip Architecture and Design Methodology”. Symposium on Very Large Integration Scale, pp. 117–124, 2002.

[3] Benini L. and De Micheli G. “Networks on Chips: a New SoC Paradigm”, IEEE Computer, pp. 70-78, 2002.

[4] Guerrier P and Greiner, A. “A Generic Architecture for On-Chip Packet-Switched Interconnections”. Design Automation and Test in Europe - DATE, pp. 250–256, 2000.

[5] Singh S. et al. “Generic Network Interfaces for Plug and Play NoC Based Architecture”. ARC, pp. 287-298, 2006.

[6] Ferrante, A., et al. “Network Interface Sharing Techniques for Area Optimized NoC Architecture”. EUROMICRO Conference on Digital System Design Architectures, pp. 10-17, 2008.

[7] Radulescu, A. et al. “An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration”**.** Design, Automation, and Test in Europe – DATE, pp. 20878 – 29883, 2004.

[8] Zeferino, C. “Redes-em-Chip: Arquiteturas e Modelos para Avaliação e Área e Desempenho”. 2003. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

Matos, D. “Interfaces Parametrizáveis para Aplicações Interconectadas por uma Rede-em-Chip”. 2010. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

DIGITAL DESIGN AND EMBEDDED SYSTEMS

Design and Verification of a Layer-2 Ethernet MAC Classification Engine for a Gigabit Ethernet Switch

Jorge Tonfat, Ricardo Reis
jorgetonfat@ieee.org, reis@inf.ufrgs.br

Grupo de Microeletrônica (GME) – PGMICRO – UFRGS, Porto Alegre, RS, Brasil

Abstract

This work presents the design and verification of the main block of a Gigabit Ethernet switch for an ASIC based on the NetFPGA platform [1]. The main function of the Layer-2 classification engine is to forward Ethernet frames to their corresponding output ports. To accomplish this task the block stores the source MAC address from frames in a SRAM memory and associates it to one of the input ports. This classification engine uses a hashing scheme that has been proven to be effective in terms of performance and implementation costs. It can lookup constantly 62.5 million frames per second, which is enough to work at wire-speed rate in a 42-port Gigabit switch. The main challenge was to achieve wire-speed rate during the learning process using external SRAM memory. This means that the bandwidth will not be reduced when new flows appear. This block was synthesized with an 180nm process and verified using System Verilog. A constrained random stimulus approach is used in a layered testbench environment with self-checking capability.

1. Introduction

Ethernet is the most popular layer-2(L2) protocol (data link layer according to the OSI model). It is widely used in Local Area Networks or LANs and also in Metropolitan Area Networks or MANs. Its popularity is mainly due to the low cost and high performance characteristics and also its fast standardizations: 10 Mbps in 1983, 100 Mbps in 1995, 1 Gbps in 1998, and 10 Gbps in 2002.

At the beginning, LANs were designed using one shared communication channel. During late 80s and early 90s, two main factors changed the way LANs were designed [2]: the LAN topology that change to a structured wiring system using central hubs and the improvement of computing systems and applications, which exceed the capacity of shared LANs, limiting the overall performance.

These factors together with the advances in microelectronics technology, allow the development of LAN switches that use the wiring structure already installed to create a micro segmented network. These changes have the following advantages: first, the possibility to eliminate collisions, if the full-duplex operation mode is used and the fact that each device has dedicated bandwidth and independent data rate.

Gigabit Ethernet switches are deployed in different kind of scenarios. Depending of it, some characteristics are different such as the lookup table size. This work focuses on scenarios where the L2 table size is large (more than 100k entries) like in an enterprise core. In the present work, we propose a layer-2 classification engine for a Gigabit Ethernet Switch. This paper is organized as follows: Section 2 presents a description of the NetFPGA Platform. Section 3 presents related previous works. Section 4 describes the design and implementation of our L2 classification engine. Section 5 presents the verification methodology. Section 6 shows the results for an ASIC implementation, and in Section 7 the conclusions and future work are presented.

2. The NetFPGA Platform

The NetFPGA platform [1] was developed by a research group at Stanford to enable fast prototyping of networking hardware. It basically contains an FPGA, four 1GigE port and buffer memory. The core clock of the board runs at 125 MHz. NetFPGA offers a basic hardware modular structure implemented in the FPGA as shown in Figure 1. Frames inside the data pipeline have their own header format as shown in Figure 2. The NetFPGA header contains information about the frame being processed such as frame size, source port and the destination port that is calculated by the classification engine. New modules can also add more headers. This pipelined structure allows us the implementation of specific functions on modules and integrate them quickly.

3. L2 Lookup Architectures

The L2 switching task needs three fields from the MAC layer header: the MAC address, the port related to that address and the VLAN (Virtual LAN) id (if present) for flow identification. This information needs to be stored in some kind of data structure. Previous works has shown different approaches. Solutions using binary and ternary CAMs (Content-addressable memory) were used but the cost per bit and power consumption as shown in [3] make them unviable for switches deployed at the enterprise core or metro edge where the lookup table's size is around the hundreds of thousands of entries. Another proposal could be the use of a software-only switch. Tests done in [4] confirm that the switching task needs to be implemented in hardware. One popular solution is the use of hashing functions to store MAC addresses in a SRAM. It uses simple hardware

compared with others, but has some disadvantages like hash collisions and a decreased table capacity. To deal with hash collisions, the table is organized in buckets that contain multiple entries. Since buckets are smaller than the table, the lookup is accelerated. Carefully chosen hash functions with relatively uniform distribution of output values (memory indexes) for an arbitrary set of input values (MAC addresses) will reduce the hash collisions and improve the table capacity as well [5]. This work uses a hashing scheme using the MAC address and the VLAN id to generate indexes to the SRAM.

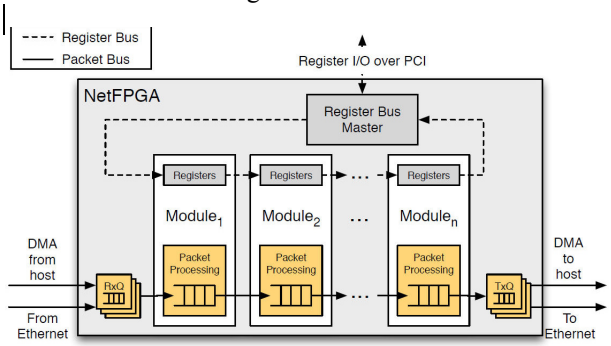


Fig. 1 – The NetFPGA Framework. [1].

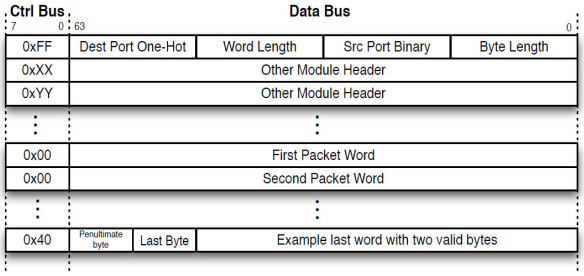


Fig. 2 – The NetFPGA packet format. [1]

4. L2 Classification Engine Architecture

The L2 classification engine is the block that implements the main function in a Gigabit Ethernet switch, the frame forwarding and learning, and is part of the data path as shown in Figure 3. It uses the MAC destination address (MACDA) and the MAC source address (MACSA) of frames to forward them to their proper destination port. In order to achieve this task, it will be needed to create a table entry with the MACSA and the input port in a process called learning. When the MACDA is not found on the table (miss), the frame will be sent to all ports except the source port (flooding). According to the IEEE 802.1D standard [6], it is necessary to age out all the entries that are not accessed for a programmable amount of time; this is not a priority task and should not interrupt the main learning/forwarding process. VLAN tags should also be considered during the frame learning/forwarding to be compliant with the IEEE 802.1Q [7] standard.

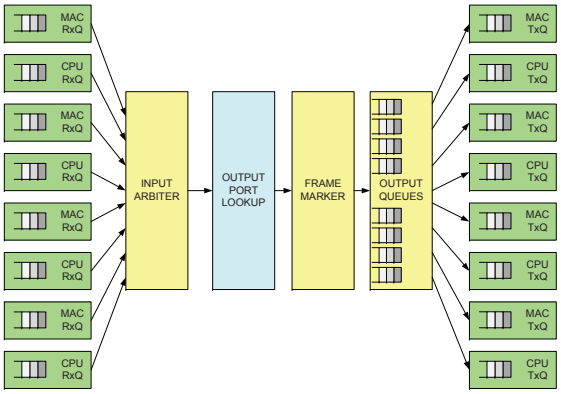


Fig. 3 – Gigabit Ethernet switch diagram block based on the NetFPGA platform.

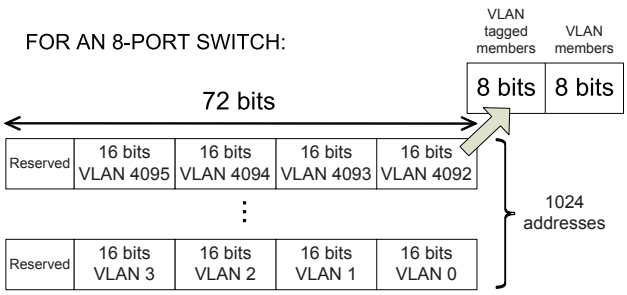


Fig. 4 – VLAN information memory format.

Every frame needs at least two read accesses from the lookup table, one for the MACDA (forwarding) and another for the MACSA (learning). If the MACSA is not found or the source port associated is different, then a third access (write) should be necessary to update the input port number or create a new entry in the table. In our design, four read and two write accesses are needed because a two-entry bucket mechanism is used, so each bucket is stored in two consecutives memory addresses. One extra read access is needed when a VLAN frame is processed. In the same SRAM where the lookup table is stored, VLAN information is stored as shown in Figure 4. The 4096 VLANs information are mapped directly into the SRAM, four ids for each memory entry. In that Figure is shown the organization for an 8-Gigabit port switch as an example. The 16 bits assigned for each VLAN id are divided in two groups: the tagged members and only members. It is important to know if a port is a tagged member or not. Tagged members should pass the VLAN tag in the frame header. This information is dispatched to the output ports module inserted into the NetFPGA header showed in Figure 2.

The lookup table is stored in a 72-bit wide SRAM. Each table entry is composed of a MAC address (48-bit), the input port(8-bit), the VLAN id(12-bit), and three status bits. These status bits are: the valid bit, the static bit and the age bit. When a new MAC address is learned, the valid and the age bit are set. If this MAC address is found later, the age bit is refreshed. In the aging process, the age bit of each valid entry will be

Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9	Cycle 10	Cycle 11	Cycle 12	Cycle 13	Cycle 14	Cycle 15
F0	F0	F0 RD REQ	F0 RD REQ	F0 RD REQ	F0 RD REQ	F0 RD REQ	F0	F0 WR REQ	F0 WR REQ	F0	F0				
F1 WR REQ	F1 WR REQ	F1	F1					F1	F1	F1 RD REQ	F1 RD REQ	F1 RD REQ	F1 RD REQ	F1 RD REQ	F1

cleared. If the age bit is not set, then the valid bit is cleared and the entry is aged out. The aging process will not modify the entries with the static bit set. This bit denotes an entry programmed by external access.

Fig. 5 – 16 cycles SRAM arbiter FSM, showing only the memory requests.

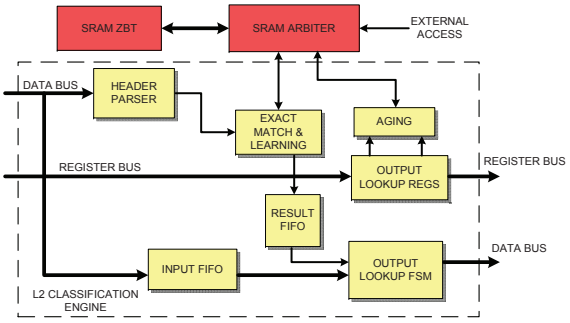


Fig. 6 – L2 classification engine block diagram.

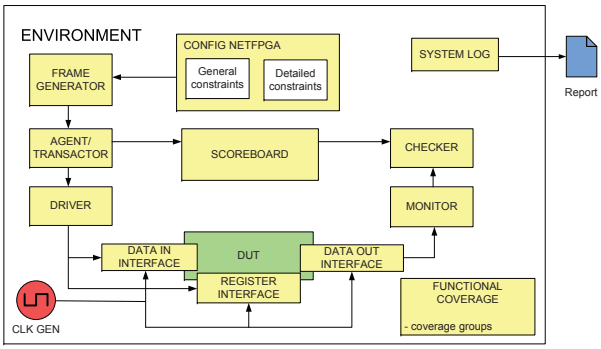


Fig. 7 – Classification Engine Testbench architecture.

The block diagram is presented in Figure 6. The header parser module will extract the MACSA, MACDA and the VLAN id from the frame and send this information to the Exact Match and Learning block. The frame will wait for the lookup using the input FIFO as a buffer. Considering the worst case (MACSA not found), the module have a constant latency of 12 clock cycles. This classification engine should be able to support 42 GigE ports at wire-speed or to process 62.5 million frames per second in the worst case. The worst case scenario appears when the switch deals with minimum frame length (64 bytes), minimum interframe gap and negligible propagation delays as explained in [2]. Considering this, a frame should be processed at least in 16 ns (672 ns for a single GigE port) or 8 clock cycles for a 500 MHz clock. To achieve this goal, this block needs to be tightly-coupled with the external SRAM memory controller. Considering that for each frame is needed 12 cycles (including read and write requests, header data fetch, hash mac addresses), the FSM from both modules (SRAM arbiter and Exact Match & Learning) have 16 cycles and are able to process 2 frames. Figure 5 shows how two frames are processed interleaved, using 14 of the 16 cycles to memory accesses. The two left cycles are used to external access and for the aging module. Since the SRAM is accessed by three different sources (the forwarding/learning module, the aging module and the external access through the register bus) an arbiter is needed. It can accept one request (read or write) per clock cycle and have a latency of five cycles due to the pipeline structure and the ZBT (zerobus turnaround) SRAM.

5. Verification Methodology

For the verification stage, we use System Verilog and Modelsim to create the testbench environment. The testbench architecture is better explained in Figure 7.

A testcase have been developed with particular constraints that will limit the random stimulus generation. With these constraints the generator will create a programmable amount of random frames that will be inserted in the DUT (Design under Test). The agent or transactor will take these frames (described in a high level variable) and will transform them into signals (bytes) and will send them through the interface (driver). The scoreboard will predict the expected result from each block and this result will be used by the checker to compare them with the received data from the DUT. During this process tens of bugs were found in the design and corrected. It is always preferable that the testbench and the design be designed by different persons; this will add some redundancy to the interpretation process of the specification.

6. Implementation Results

Table 1 shows a comparison between different classification engines. The work from [8] doesn't mention the operation frequency. [9] has better bandwidth results but is important to note that the bandwidth they show is an average that depends on the number of collisions and the size of the table. There is shown the results for a 64K table. If compared to the results obtained with a 32K table, the bandwidth is reduced by 25% while the results presented in this work are constant relative to the table size. This module was synthesized with different table sizes: 4K, 32K, 64K and 128K all of them obtain an operation frequency of 500 MHz and a bandwidth of 42 Gbps.

Tab.1 - Comparison with related works.

Solution	Op. Freq. (MHz)	Bandwidth (Gbps)	Technology Process
Our	500	42	TSMC 180nm
[10]	125	22	180nm
[8]	n/a	10	180nm
[9]	400	103.5	UMC 130nm

7. Conclusions and Future Work

A classification engine for a Gigabit Ethernet switch was presented. The verification stage is very important to find bugs that will only appear with random stimulus. The random constrained approach is more time-efficient to reach the coverage goal than other simpler methods such as direct-test.

The architecture presented in this work achieves the necessary throughput for a 42-port GigE. The next step is to work in order to be able to process layer-3 protocols such as IP. To accomplish this, some sort of search algorithm should be needed such as LPM (Longest Prefix Match).

8. References

[1] Jad Naous, Glen Gibb, Sara Bolouki, and Nick McKeown,"NetFPGA: reusable router architecture for experimental research," *in PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, New York, NY, USA, 2008, pp.

[2] Rich Seifert and James Edwards, *The All-New Switch Book: The Complete Guide to LAN Switching Technology*, Wiley, Hoboken, NJ, 2008.

[3] A.J. McAuley and P. Francis, "Fast routing table lookup using cams," in *INFOCOM '93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future. IEEE*, 1993, pp. 1382 –1391 vol.3.

[4] J Luo, J. Pettit, M. Casado, J. Lockwood, and N. McKeown, "Prototyping fast, simple, secure switches for ethane," in *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, 22-24 2007, pp. 73 –82.

[5] C. Huntley, G. Antonova, and P. Guinand, "Effect of hash collisions on the performance of lan switching devices and networks," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, 14-16 2006, pp. 280 –284.

[6] IEEE Std 802.1D-2004, "IEEE standard for local and metropolitan area networks media access control (MAC) bridges," p. 269, 2004.

[7] IEEE Std 802.1Q-2005, "IEEE standard for local and metropolitan area networks virtual bridged local area networks," p. 285, 2006.

[8] S.M. Mishra, A. Guruprasad, Chun Feng Hu, P.K. Pandey, and Ming Hung, "Wire-speed traffic management in ethernet switches," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, 25-28 2003, vol. 2, pp. II–105 – II–108 vol.2.

[9] V. Papaefstathiou and I. Papaefstathiou, "A hardwareengine for layer-2 classification in low-storage, ultra high bandwidth environments," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, 6-10 2006, vol. 2, pp. 1 –6.

[10] M.V. Lau, S. Shieh, Pei-Feng Wang, B. Smith, D. Lee, J. Chao, B. Shung, and Cheng-Chung Shih, "Gigabit ethernet switches using a shared buffer architecture," *Communications Magazine, IEEE*, vol. 41, no. 12, pp. 76 – 84, dec. 2003.

Functional Verification of logic modules for a Gigabit Ethernet Switch

Jorge Tonfat, Gustavo Neuberger, Ricardo Reis
jorgetonfat@ieee.org, {neuberg,reis}@inf.ufrgs.br

Grupo de Microeletrônica (GME) – PGMICRO – UFRGS, Porto Alegre, RS, Brasil

Abstract

This work presents the functional verification of logic modules for a Gigabit Ethernet (GigE) Switch for an ASIC based on the NetFPGA platform. A coverage-driven constrained random stimulus approach is used. It is implemented in a layered-testbench environment with self-checking capability. This environment implements the methodology presented by the Verification Methodology Manual (VMM) using SystemVerilog. The main advantage of this methodology is its reusability. This characteristic enables the development of a common testbench environment for our modules with minimum changes for each particular module. The four logic modules presented in this work implement functions of a Gigabit Ethernet switch. The common characteristic of these circuits is the close dependency between the time and its functionality. These modules need time information to deal with problems such as rate limiting, quality of service (QoS) or aging lookup tables in classification engines. As described in the literature, the transaction-level models used to predict the circuit behavior are time-independent when the implementation details are not relevant. But when time information influences the circuit functionality, the model needs to replicate the circuit latency to be functionally equivalent. We propose a simple solution to the synchronization process between the model and the design under verification (DUV). This solution preserves the main advantage of transaction-level models (faster simulation time than the RTL model) and generates the result data with the same circuit latency. These features made possible to run a considerable amount of testcases that helps to find and correct bugs in the circuit with a high confidence measured by the functional and code coverage results.

1. Introduction

The improvement in the semiconductor manufacturing processes and design methodologies led to the possibility of bigger and complex digital designs. This scenario along with tighter time-to-market budgets reduces the possibility of success within the first attempt. To mitigate this situation, industry drives to the development of new design methodologies such as the reuse of Intellectual Property (IP) cores design. In this context, the design verification acquires relevant attention from the academic and industrial environments. High confidence design verification means high quality designs and less circuit functional bugs transferred to the end user.

Design verification is the process to certify if the design functionality is according to the design specification. It is the inverse process of design as shown in Figure 1. Two methods are commonly used: functional and formal verification.

Functional verification or simulation-based verification is the most used technique for industrial applications. This technique is easy to deal with, but almost always the most resource and "bottleneck" phase of the design flow. Formal verification methods were also proposed but only applied to low complexity circuits. In this scenario, few functional verification experiences of communications systems were reported [3]. Most of them were applied to processors designs [4] [5]. This work presents the functional verification of four modules of the datapath of a Gigabit Ethernet switch.

This paper is organized as follows: Section II presents a description of the modules verified in this work. Section III presents the verification environment implemented. Section IV presents the technique used to synchronize the DUV and the reference model. Section V shows the results from the verification of these modules, and in Section VI the conclusions and future work are presented.

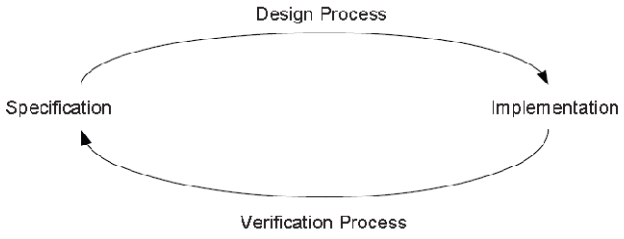


Fig. 1 – The verification process vs. the design process. [6]

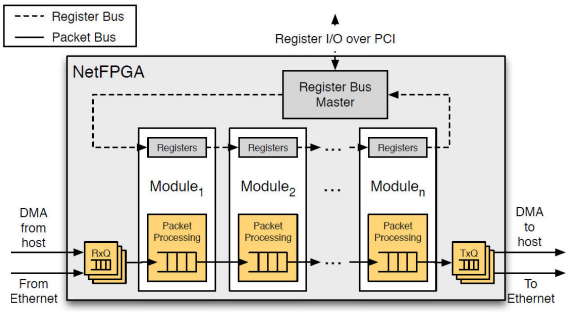


Fig. 2 – The NetFPGA Platform modular structure. [1]

2. Designs Under Verification

The NetFPGA platform [1] was developed by a research group at Stanford to enable fast prototyping of networking hardware. NetFPGA offers a basic hardware modular structure implemented in the FPGA as shown in Figure 2. Due to the modular structure, these modules have the same input/output interface and the same register interface because they are connected in a ring topology.

The designs chosen to be verified in this work are part of the user datapath of a Gigabit Ethernet switch. These modules implement the functionality of the switch such as the frame forwarding, frame classifying or output queuing. These modules are shown in Figure 3. The four modules are functionally verified.

The first of them is the module Input Arbiter. This module decides, based on a scheduler algorithm, which Rx queue to service next. We propose an improvement from original NetFPGA design changing it from Round-Robin to Deficit Round Robin. DRR, as described in [7], it provides a credit based quantum per queue that avoids an unfair behavior between queues. As the old scheduler, this one has a state machine cycling between each queue. This time, each FIFO has a credit counter, indicating the number of words to be copied.

Instead of serving one packet of each queue, the scheduler server packets as the queue credits allow. The next one is the module Output Port Lookup. The main function of this module is to forward Ethernet frames to their corresponding output ports. To accomplish this task the block stores the source MAC address from frames in a SRAM memory and associates it to one of the input ports. This module uses a hashing scheme to search the address in the lookup table. A background process erases unused addresses in the lookup table. This process is done at a programmable time. The frames can be unicast, multicast and broadcast. Due to its implementation, the time needed to process one frame is not constant. This feature increases the complexity of the reference model to predict the expected output.

The third module is the Frame Color Marker. This module is part of the QoS mechanism. It should mark each frame with a color that represents the frame priority. It is based on the following request for comments: RFC2697, RFC4115 and RFC2698. The last module is the Output Queues. This module looks which output queue to store the frame in, until the Tx queue is available to accept the frame for transmission.

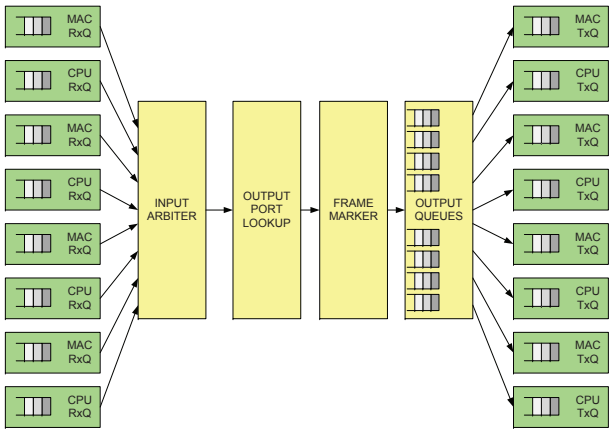


Fig. 3 – The User Datapath diagram block.

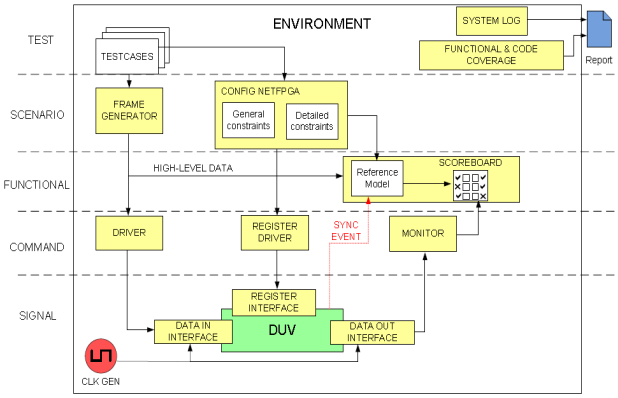


Fig. 4 – The testbench architecture.

3. The Verification Environment

The testbench environment is shown in Figure 4. This environment is organized in a hierarchical layered structure. This structure helps to maintain and reuse it with different Designs Under Verification or DUVs. It was implemented following the recommendations shown in [2]. The following subsections explain the functionality of each layer in the testbench environment.

3.1. The Signal and Command Layer

The signal layer implements the interface between the testbench and the DUV. Since all modules have the same interface, these interfaces are shared among all the verification environments. Minimal modifications are needed to the modules "Input Arbiter" and "Output Queues" because they multiplex and demultiplex respectively the frames from the Ethernet ports. In the case of the Input Arbiter module, the input interfaces are expanded to number of input ports of the GigE Switch. And in the case of the Output Queues module, the output interfaces are also expanded.

The command layer implements the driver and monitor functions. The driver receives high-level data and breaks into signals that are sent to the DUV through the interfaces. Another driver (register driver) receives the register configuration of the module and creates the register transactions to configure the DUV. The monitor implements the complementary function. It receives the signals generated by the DUV and composes a high-level data that will be used to compare against the results from the reference model.

3.2. The Functional Layer

The functional layer implements the scoreboard. In the literature are different definitions for the scoreboard. In this work the scoreboard includes the reference model, the data structure that holds the results from the DUV, and the output comparison function. The reference model receives high-level data and computes the expected output of the circuit. This output is stored in a data structure of results. When the monitor receives a result generated by the DUV, this will be compared against the results stored from the reference model. If the comparison succeeds, the circuit produced the expected result; otherwise an error report is generated.

3.3. The Scenario Layer

The scenario layer creates the configuration of each of the modules in the verification environment. This configuration sets the parameters of the DUV such as aging time in the output lookup module or the data rate in the Frame Color Marker module. This configuration is also sent to the reference model to compute the expected output. Based on the parameters created for this testcase, the generator will created constrained random data (Ethernet frames) that will stimulate the DUV. Depending on the module that is verified, the generator will create different kind of frames. For example, in the environment created for the Input Arbiter module, the frames generated have different input ports. In the case of the Output Queues module, the output port field is more interesting so frames with different output ports are generated.

3.4. The Test Layer

The test layer includes the testcases that will be applied to the DUV. Most of them will be generated randomly and some directed cases were added such as bad register configurations. These directed cases are needed to verify the module response under uncommon situations that are possible. This layer will control the execution of the current simulation and will manage the reports of each of function in the testbench environment. This layer sets the error tolerance, and classifies the unexpected behavior with different levels of events. The events are classified into six different categories, starting with the information or verbose events, going through debug, normal, warning and error events, finishing with fatal events. A maximum number of errors is defined for each testcase, this feature will let the simulation to continue running in the presence of errors and will help to find more bugs in the DUV. One fatal event will necessary stop the simulation because this kind of events makes unviable the execution of the current testcase.

4. The Circuit Reference Model

In this section, the mechanism used to synchronize the reference model with the DUV is presented. When is mentioned the use of transaction-level models, the tendency is to relate them with the high-level hardware modeling language SystemC as shown in [8]. In this work all transaction-level models were written in SystemVerilog. As explained in [9], there is no technical reason for a model written at the same level of abstraction to run faster if it is described in other language.

In a first approach to model the circuit behavior, a simple transaction-level model was created for each of the modules. This approach showed that if the latency of the circuit is not taken into account, some expected results are different from the actual RTL model results. These differences appear because these circuits have a close relationship between the time and functionality, so the classic reference model cannot be used.

In a second approach the circuit latency was tried to be modeled without changing the transaction-level characteristic of the reference model. But the complexity and the variety of operation modes in these modules lead to a complex reference modeling that reduce the reliability of the reference model.

The final approach to model the circuit functionality was to synchronize the computation of the expected output with some key points (states) in the RTL model. This synchronization was done with one of the features of the SystemVerilog language. SystemVerilog “Event” objects are used to trigger the reference model computation when certain conditions are met in the RTL model. The observation of the DUV is done without modifying the original code of the RTL model. The reference model continues being a simple transaction level model and the expected result is computed at the same execution time of the RTL model.

5. Verification Results

In table 1 are shown the results from the functional verification of the four modules. The ModelSim 6.6b simulation tool is used on a server with an Intel Xeon processor @ 2.5 GHz with 32GB of RAM. Some common characteristics can be extracted such as the high percentage of functional coverage on each module. These results are mostly due to the specification of constrained random stimulus directed to meet the functional coverage objective. It is important to mention that the functional coverage defined for each module is configured according to their respective specifications. The results from the code coverage show that the testcases could be improved to obtain a higher confidence of the functional verification process.

The most important information is found in the simulation time columns. In the first simulation time column the reference model is replaced with the same RTL model of the circuit. The second simulation time

column shows the time simulation for the proposed reference model with time synchronization with the RTL model. It is shown a clear run time improvement with an average of 23,37% for this experience. These results demonstrated that due to transaction-level of the model, the run time is smaller.

Tab.1 - Detailed results from the functional verification of the four modules.

Module	# Stimuli per testcase	# TestCases	Simulation Time 1 (hours)	Simulation Time 2 (hours)	Functional Coverage (%)	Code Coverage (%)
Input Arbiter	1600	1000	10.28*	8.8	94.44	91.70
Output Port Lookup	10000	1000	93.83*	76.66	99.60	90.18
Frame Color Marker	10000	1000	54.8	35.18	99.21	85.71
Output Queues	10000	1000	64.68*	48.52	91.74	86.57

* = These values were approximated taking as reference the average run time of 1 testcase repeated 10 times.

6. Conclusions and Future Works

The functional verification of the modules of the datapath of a Gigabit Ethernet switch was presented. These experiences shown that using classic timeless reference models are not suitable for this kind of circuits where the time influences in the circuit response. A simple method to synchronize the reference model with the DUV was proposed. This method maintains the benefits of transaction-level models as well as correctly predicts the circuit output.

Future improvements to the functional verification process should be the use of a mechanism to eliminate testcases that do not increase the functional coverage in order to reduce more the simulation run time. Some solutions are shown in [3] and [10].

7. References

[1] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: reusable router architecture for experimental research," in *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*. New York, NY, USA: ACM, 2008, pp. 1–7.

[2] C. Spear, *SystemVerilog for Verification, Second Edition: A Guide to Learning the Testbench Language Features.* Springer Publishing Company, Incorporated, 2008.

[3] M. Strum, W. J. Chau, and E. Romero, "Comparing two testbench methods for hierarchical functional verification of a bluetooth baseband adaptor," *in Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on,* 2005, pp. 327 –332.

[4] Y. Wu, L. Yu, W. Zhuang, and J. Wang, "A coverage-driven constraint random-based functional verification method of pipeline unit," in *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, 2009, pp. 1049 –1054.

[5] Z. Gu, Z. Yu, B. Shen, and Q. Zhang, "Functional verification methodology of a 32-bit risc microprocessor," in *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on*, 2002.

[6] S. Vasudevan, *Effective Functional Verification.* Springer, 2006.

[7] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *Networking, IEEE/ACM Transactions on,* vol. 4, no. 3, pp. 375 –385, Jun. 1996.

[8] M.-K. You, Y.-J. Oh, and G.-Y. Song, "Implementation of a hardware functional verification system using systemc infrastructure," in *TENCON 2009 - 2009 IEEE Region 10 Conference,* 2009, pp. 1 –5.

[9] J. Bergeron, *Writing Testbenches using SystemVerilog*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[10] M. Bose, J. Shin, E. Rudnick, T. Dukes, and M. Abadir, "A genetic approach to automatic bias generation for biased random instruction generation," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 2001.

A Direct Memory Access Controller (DMAC) IP-Core using the AMBA AXI protocol

[1]Ilan Correa, [2]José Luís Güntzel, [1]Aldebaro Klautau and [1]João Crisóstomo Costa
{ilan, aldebaro, jweyl}@ufpa.br, guntzel@inf.ufsc.br

[1]Sensors and Embedded Systems Laboratory (LASSE)
Federal University of Pará, Belém, Brazil
[2]Embedded Computing Lab. (LCE)
Federal University of Santa Catarina, Florianópolis, Brazil

Abstract

This work presents the design of a hardware controller that allows a peripheral of an embedded computer system to directly access the memory, known as direct memory access controller (DMAC). This DMAC is inspired in the Dalton Project DMAC, also being compatible with the widely used Intel I8237. In order to make it suitable for current embedded applications, its communication pattern follows the AMBA AXI protocol, which allows more flexibility such as fewer timing restrictions and data transfer in the system with minimum CPU load.

1. Introduction

The tremendous advances in integrated circuit fabrication technology have allowed a myriad of new products that are present in everyday. Most of these products fall into the embedded systems class. Particularly, personal mobile devices (PMDs), such as smatphones, tablets etc, became very popular, currently responding for a significant part of the consumer electronics market. New electronic products, mainly PMDs, present very stringent design requirements in terms of energy consumption, reliability, cost and performance (speed).

Currently, the major challenge in embedded systems design concerns energy efficiency, which corresponds to achieve the performance required by the application with as low power consumption as possible. Unfortunately, those are conflicting targets. Thereby, new solutions in terms of hardware, embedded software or both are greatly demanded, as those presented in [1], [2].

To meet performance requirements of embedded systems, a widely used design technique is the adoption of blocks that are designed to efficiently perform a particular task or a critical part of a larger algorithm. Those blocks are generally referred to as Intellectual Property (IP) blocks.

This paper presents the design of an IP block that acts as a direct memory access controller (DMAC). It is based on the widely used Intel I8237 DMAC [3], as well as on Dalton Project's DMAC [4]. But unlike those DMACs, ours follows the AMBA AXI [5] protocol, which provides ways to achieve those requirement aforementioned and is becoming widely used in industry.

As basic design requirement, it was assumed that this DMAC should allow data transfer between a peripheral and the memory consuming the least amount of CPU cycles possible. The DMAC design is the result of a partnership in the context of NAMITEC Project [6], involving two research groups, one from the Federal University of Santa Catarina and another from the Federal University of Pará.

2. AMBA AXI overview

The Advanced Microcontroller Bus Architecture (AMBA) was developed by ARM Ltd in 1996 and has several versions. In 2003 a new AMBA with many features was released, looking for achieving better performance. In this new version there are five communication channels which are defined as: Read address channel, Write address channel, Read data channel, Write data channel and Write response channel. The channels Read/Write address channel carry information needed to begin a transfer, Read/Write data channel carry data after handshake in any address channel and Write response channel is used with Write data channel to indicate how a write operation ended or whether there was an error during handshake in a transfer or address channel.

The AMBA AXI architecture was conceived to allow only two devices to be directly connected. In a given transfer, one device is the "Master", while the other is the "Slave". Since in the DMAC there might exist various devices, an "interconnect block" must be used, as the one available from ARM [7]. The topology of the interconnect block is such that for each Master there is a Slave, and for each Slave there is a Master (shown in Fig. 1). Hence, the interconnect block acts as a data multiplexer, allowing a Master to connect with any other Slave by using the information provided through Read or Write address channel [7].

A transfer is always started by a Master. Initially, it releases control information in Read or Write address channel, such as word length, number of words and addresses. After control information has been transferred

(and Slaves have answered), data transfer starts through Write or Read data channel. Finally, in the case of write operation the response is sent by Write response channel, from Slave to Master.

In order to allow an energy-efficient operation, in this work the simplest AXI protocol was adopted, which only manages the handshaking and data transfer as specified in version two. However, may be interesting to add some of the features of newer versions, (mainly from version four), whenever they contribute to save power.

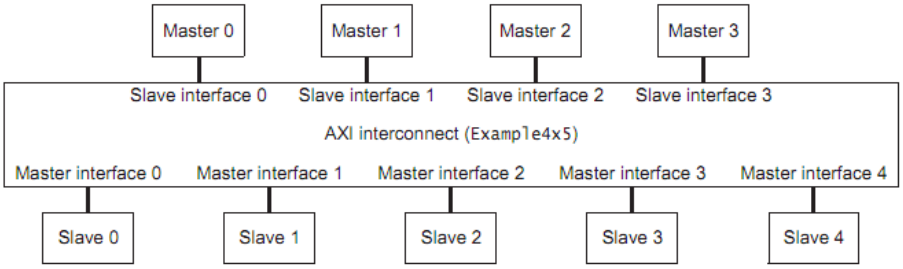


Fig. 1 – Interconnect block (taken from [7]).

3. Inside the controller

The direct memory access controller (DMAC) acts as a bridge, where it receives an amount of data, stores the data in its internal buffer and forwards the data to the destination. This operation is repeated until all data has been transferred. The DMAC also handles the priority between all devices requesting memory access.

For the operation to be performed the DMAC should have a set of components that assists it in the job (e.g. Registers, buffers, multiplexers, etc), in addition to the interfaces of the protocol, two Masters and a Slave. Fig. 2 shows the internal organization of the designed DMAC.

These components keep and forward information concerning transfer direction, number of words to be transferred, size of each word, priority of a transfer, which device will transfer, temporary information about current transfer and a memory to store the data to be forwarded.

Fig. 2 shows the set of components that performs all data manipulation inside the DMAC. This set consists of simple components, what makes this controller less power hungry, causing less impact in the overall system consumption. Registers, multiplexers, comparators, a buffer and adders (not shown) compose its datapath. The datapath is coordinated by the control block. The DMAC components are described in the following sub-sections.

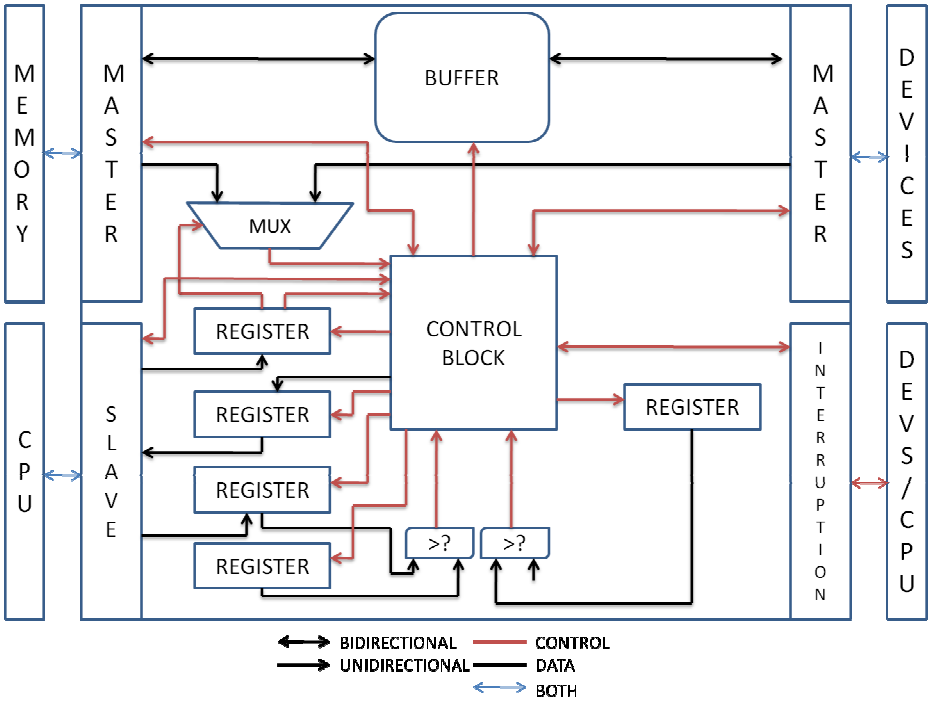


Fig. 2 – DMAC internal structure and its interfaces to the system.

3.1. Registers

The registers store information coming from CPU, information that CPU should read to know which device will transfer, temporary information generated performing the transfer and priority control.

Information coming from CPU corresponds to the direction of transfer, number of words to be transferred and addresses.

3.2. Multiplexers

A DMA transfer can occur from memory to a device, from device to memory or from memory to memory. In the first type of transfer the device is referred to as Receiver, whereas in the second type of transfer, it is referred to as the Provider.

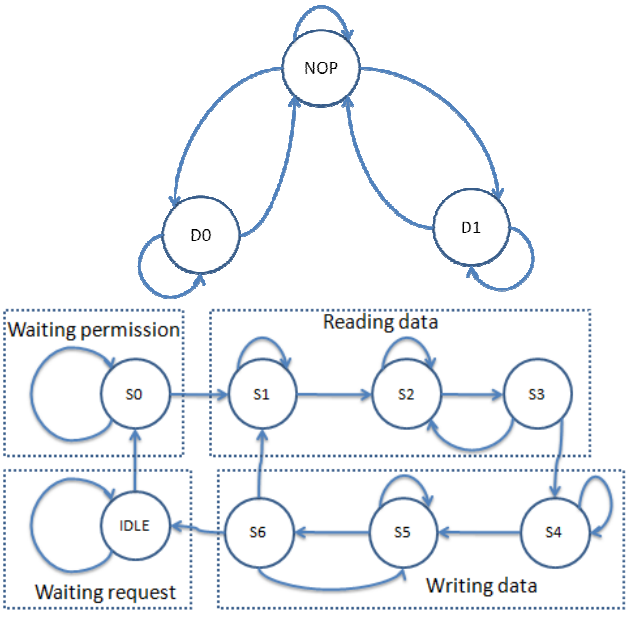
In each transfer the DMAC reads a block of words from the Provider and forwards to the Receiver. To properly accomplish this, extern signal should be selected among those signals coming from memory or from devices. This is the role of multiplexers: they generate internal signal from external signals, and thereby the control block deals only with internal signals, which make its design easy. For instance, in the case of a transfer operation from memory to a device, the controller generates control information concerning Read address channel connected to memory and waits the response coming from it. After memory response, it starts the transfer putting words on Read data channel, thus informing the controller that data is on this channel by selecting its signals through the multiplexers. After internal buffer is full the controller generates signals on Write address channel connected to device and starts forwarding the buffer contents. Such operation is similar to the first transfer previously described.

3.3. Buffer

This component provides a way to store a block of words and then forwards when it is full. It avoids excessive handshakes.

3.4. Control block

This component provides the control signals to the datapath, being highly dependent on how datapath components are organized. The control block consists of two state machines: the "priority machine" and the "transfer machine". The former manages priority between devices and controls the beginning of the operation, whereas the latter manages internal signals used to control the datapath and external signals, used in interruptions and in the AXI protocol. Fig. 3 depicts these state machines.



(a) (b)

Fig. 3 – State machines of DMAC control block: priority machine (a) and transfer machine (b).

The priority machine (Fig. 3 (a)) controls priority of accessing memory for two or more devices (it can be easily adapted for more devices). While there is no request on interruption interface, it stays in the NOP state. Whenever there is a request the machine goes either to state D0 or D1, depending on the device requesting access. In D0 and D1 a signal is generated and the transfer machine (Fig. 3 (b)) starts its operation.

The transfer machine generates control signals to the datapath and externals signals. It performs four tasks: waits request (IDLE), waits for permission from the CPU (S0), read data from the Provider and write data to the Receiver. While waiting for request, the transfer machine is basically waiting for the signal coming from the priority machine. This signal changes its state from IDLE to S0. In S0 it writes data from the CPU in its internal register (information about transfer) and waits for the signal from CPU that indicates that the bus is free. In S1, S2 and S3 the controller reads data from the Provider and writes these data to the Receiver in S4, S5 and S6.

These states are triggered by signals of AXI (they are properly selected on the multiplexers). They generate output signal of AXI too. For instance, in a read operation from memory, S1, S2 and S3 generate signals on Read address channel and Read data channel. In the second part of the transfer, the controller forwards the data using Write address channel, Write data channel and Write Response channel.

4. DMA operation

This section presents a simplified description of the DMAC operation when two devices disp0 and disp1 request a transfer at the same time, supposing that device disp0 has higher priority and therefore, will transfer first. In this examples memory is the Provider and a device is the Receiver.

Initially, the DMAC waits for request and the two state machines are in NOP and IDLE states, respectively. Then, devices disp0 and disp1 request a transfer at the same time. The priority machine goes to state D0, writes on proper register which device should transfer and generates a signal that makes the operation machine goes from IDLE to S0.

In S0 the DMAC interrupts the CPU and the signals coming from Slave interface connected to CPU are selected. The CPU reads some internal registers and with these data writes on other register information about the transfer. After that, it releases a signal indicating that the bus is free and the operation machine goes to S1.

In S1 the Read address channel of Master connected to Provider is used. When Provider is complete it sends a signal and the transfer machine progresses to S2. In S2 Read data channel is used to transfer data, with each received data goes to S3, stores data, and verifies if buffer is full. If the buffer is not full, the machine proceeds to S2. Else, it goes to S4.

S4 and S5 are similar to S2 and S3, respectively, except that the controller writes using Write address channel and Write data channel. S6 verifies if all data has been transferred. If not, the machine goes to S2. Else, it goes to IDLE states.. State S6 also verifies the Write response channel, looking for errors.

5. Synthesis Results

In order to validate the DMAC architecture, its datapath was described in VHDL and synthesized for Stratix II and Stratix III Altera FPGAs by using Quartus II software. We estimate that the impact of the control block on the DMAC performance and FPGA resources is not as relevant as that of the datapath.

The obtained results, in terms of resources and critical delay, are showed in Table 1. The small difference of delay for these FPGA families may indicate that the DMAC performance is bounded by the internal FPGA wire connections and by capacitive load represented by the chip pads. Therefore, for a more accurate evaluation, an ASIC synthesis is demanded.

Table 1. Synthesis results of DMAC datapath for Altera FPGAs

family/device	# of ALUTs	# of flip-flops	logest delay (ns)
Stratix II - EP2S15F484C4	269	573	9.296
stratix III - EP3SE50F484C4	268	573	9.172

6. Conclusion

This paper presented the design of an AMBA AXI based direct memory access controller (DMAC). The internal structure of DMAC was showed, highlighting the simplicity of design which reduces power consumption, a critical subject in embedded systems. The DMAC operation also allows for CPU time savings, due to the small number of cycles for preparing each transfer. A device can be easily adapted to the designed DMAC because AMBA AXI does not have hard timing constraints. The DMAC was designed in such a way that the number of devices can be easily expanded.

The compatibility with commercial IPs and systems is ensured by the use of AMBA AXI protocol. Although such protocol requires a significant number of signals, it is not a relevant problem since the DMAC is to be used in systems-on-a-chip (SoCs).

The DMAC is currently being described in VHDL. For the final version of this paper, we intend to add synthesis results targeting Altera FPGA devices.

7. References

[1] Mutlu, O.; Hyesoon Kim; Patt, Y.N.; "Address-Value Delta (AVD) Prediction: A Hardware Technique for Efficiently Parallelizing Dependent Cache Misses," Computers, IEEE Transactions on, vol.55, no.12, pp.1491-1508, Dec. 2006.

[2] Jones, R.B.; Allan, V.H.; , "Software pipelining: a comparison and improvement," Microprogramming and Microarchitecture. Micro 23. Proceedings of the 23rd Annual Workshop and Symposium. Workshop on, pp.46-56, 27-29 Nov 1990.

[3] 8237A High Performance Programmable DMA Controller Datasheet, Intel Corporation, 1993.

[4] “The UCR Dalton Project - IP-Based Embedded System Design”, 2010; http://www.cs.ucr.edu/~dalton/.

[5] AMBA AXI Protocol Specification ver 2.0, ARM Ltd., 2003.

[6] “Instituto Nacional de Ciência e Tecnologia de Sistemas Micro e Nanoeletrônicos”, 2010; http://namitec.cti.gov.br/.

[7] PrimeCell® AXI Configurable Interconnect, ARM Ltd., 2004

GenCode: A tool for generation of Java code from UML class models

Abilio G. Parada, Eliane Siegert, Lisane B. de Brisolara
{agparada,esiegert,lisane}@inf.ufpel.edu.br

Universidade Federal de Pelotas

Abstract

The development of embedded software differs from the development of traditional software, mainly due to hard constrains related to the embedded platform and the tiny time-to-market. Allied to the high complexity found on emergent embedded systems, these difficulties motivate for the use of Model-driven Engineering (MDE) approaches to provide abstraction and automation for embedded software design process. To enable the effective use of the MDE paradigm, tools are required to capture models, transform models, and to generate code. This paper presents the development of a tool to support the use of MDE in the embedded software development, which reads a UML model, capturing its elements and generating code from them. The developed tool prototype is able to generate skeleton of Java code from UML class diagrams.

1. Introduction

Embedded software differs from traditional software, since engineers should take into consideration the hardware platform in order to develop efficient algorithms to run on it, considering memory size, performance and power constrains. Besides, engineers should worry about the speed of product delivery (time-to-market), while producing good quality software at small costs and, consequently, competitive products.

At the same time, the complexity of embedded applications grows up following advances in processing power of embedded devices. Usually models are used to deal with high complexity [1] because model-based approaches offer several benefits. Firstly, models have fewer details so are easiest to build, thus facilitating the understanding and detection of a problem. The Unified Modeling Language (UML) [2] has become the standard modeling language for object-oriented software development and has also been used in the embedded domain [1][3].

To automate the process of embedded software development and deal with the complexity of emergent embedded software, a new paradigm promises automation and abstraction. This paradigm is known as Model-Driven Engineering – MDE [4], because it considers the model as a primitive artifact of the software project. Several authors advocate its use in embedded systems design [5][6]. With MDE models are not only used in the software documentation, the software models are used for all software engineering phases (analysis, design, implementation, and testing). Following this paradigm, models evolved and are transformed until to be able to obtain an implementation from it, speeding the process of software development and offers abstraction for engineers.

To allow the effective use of the MDE paradigm, tools are required to capture models, transform models, and generate code. This paper presents the development of a tool to support the embedded software development process, which reads a UML model, capturing its elements and generating code. The tool prototype is able to generate skeleton of Java code from UML class diagrams.

This paper is organized as following. Concepts of the MDE, including used standards and tools are presented in Section 2. Section 3 presents the proposed tool, presenting its structure, describing the model capturing and code generation approaches, and illustrating its functionality. Finally, Section 4, presents conclusions and future works.

2. Background

The Model-Driven Engineering – MDE considers the models as part of the software production process, differing from the other software engineering paradigms that use the models only for documentation. In MDE approaches, models are considered as important as the code, because they can be transformed into other models or codes. This way, automation is supported, speeding up processes, and facilitating errors detection.

Several MDE approaches are based on UML models, which are stored using the XMI [7] model interchange standard, which is based on XML (*eXtensible Markup Language*). Thus, the first functionality required for a MDE supporting tool is the model capturing. After capturing the model, it must be transformed into code in the target language. This process typically uses a template engine to transform model into code, given the format specified by the template. Templates are a flexible approach to convert models to text. The most popular template engine is named Velocity [7]. Other methods to generate source code include the use of rules, writing programs that generate programs (code generators), as used in this work.

3. The code generator

This section discusses the development of a prototype of a code generation tool. This prototype is able to generate Java code from UML class diagrams. The proposed tool is divided in two main steps: XML file capturing and Java code generation. The next subsections describe these steps, discuss the tool developing and demonstrate its functionality.

3.1. Capturing information from XML file

The first step consists in capturing information from UML diagrams, represented using the XMI standard. This step is divided in two substeps. The first substep is the reading of "xmi:id" that consists in an identification of the diagram components (class, attribute, operation, association and properties of UML tool). This information is captured and stored into a trie (retrieval tree) to facilitate and accelerate the search. This is required because in the file has references forwards, for example, the class X has a attribute that is a reference to the class Y, and the information of the class Y is next the information of the class X. The last substep is responsible for storing and organizing information captured using an object-oriented approach like a UML structure, which contains the information of the model, and sub divide on components of UML like, for example, classes, packages, associations, attributes, methods, and parameters.

3.2. Code generation from UML class model

After capturing the model, it must be transformed in Java code. This process uses the structure generated at first step. The method used to generate code is a simple conversion from information of the model captured in first step to text. This is the usual approach and it is used by [8].

Using object-oriented approach, each element of structure is responsible to its code generation that is performed by the method named GenCode. For example, the Class element, which represents a UML class, is responsible to generate code for an instance of the relative class found at the model

The code generated by the tool includes specification of package, specification of super class (if there is a specialization/generation relationship between classes), declaration of attributes (defined explicitly in the model) considering its type, and method signatures generation. Beside of the methods declared into the model, constructor methods, and get and set methods to access private and protected attributes are also generated. The constructor method generation includes the superclass constructor invocation, if necessary, and initialization of the class attributes. When the attributes are of primitive type, these are defined and comments are generated to remember the programmer to initialize them.

Besides the attributes explicitly defined in the class model, other attributes are generated to represent return parameters of methods or relationship between classes (association, aggregation or composition). When one attribute represents a relationship and its multiplicity is "*", this attribute is declared as an Array of objects of the associated class, otherwise the attribute is declared as a reference for one object of the related class.

Methods in a UML model have parameters that can be set as in, out, inout or return. The method signature generation includes definition of method visibility, and type for all parameters including return. The method return type is defined according to the type of output parameter. A method can have more than one output parameter, if this is defined with another type, a comment is generated to warn the designer. The declarations of out parameters are into the scope of the method and declaration of the return is in the class scope.

When there is an inheritance hierarchy including an abstract class, abstract methods defined in the abstract class are generated as concrete methods into the code of its immediate concrete subclass. Beside the code itself, the tool generates also clarifying comments and warning comments into a local of code.

3.3. Tool developing

The tool prototype was implemented using the Java language. Fig. 1 illustrates the UML class diagram of the tool prototype built using Papyrus [9], an open source tool for UML2 modeling. The tool input is a XML file and for the output, a directory with the name of model is created where the generated Java code is enclosed.

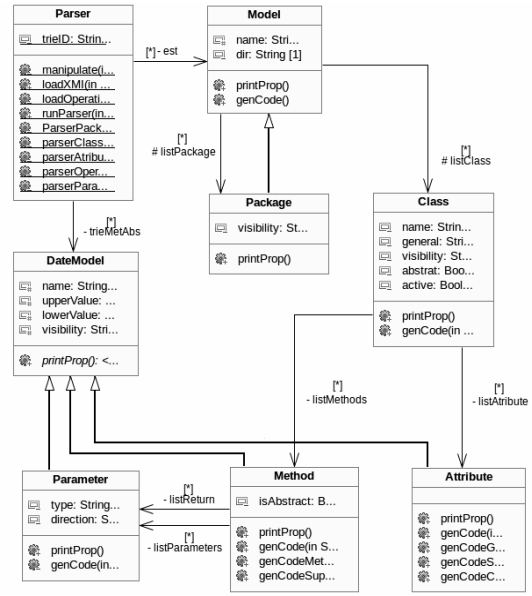


Figure 7: Class diagram – Structural view of tool prototype

3.4. Tool validation

Fig. 2 illustrates a fragment of the code generated by the tool using as input the UML class diagram depicted in Fig. 1. Although code was generated for all classes, the Fig. 2 depicted code generated for the class "Parser". In the generated code, firstly there are the class definition (line 3) and the list of attributes (line 5 to 13), in which is possible to observe attributes generated as an Array because of the multiplicity of the classes relationship. In line 17 is declared a variable to store the value returned by the method named "manipulate" (line 65). Method "get" and "set" are found from line 30 until line 60 and class methods from line 62 until 91.

```
1  import java.util.ArrayList;
2
3  public class Parser{
4
5      /*
6       *Attributes
7       */
8      private ArrayList<Model> est;
9
10     private ArrayList<String> trieID;
11
12     private ArrayList<Operation> trieOpe;
13
14     /*
15      *Attribute of Return Method manipulate
16      */
17     private String str;
18
19     /*
20      *Constructor
21      */
22     public Parser(){
23         this.est = new ArrayList<Model>();
24         this.trieID = new ArrayList<String>();
25         this.trieOpe = new ArrayList<Operation>();
26         this.trieMetAbs = new ArrayList<DataModel>();
27     }
28
29
30     /*
31      *Get
32      */
33     public Model getEst(){
34         return this.est;
35     }
36
37     public String getTrieID(){
38         return this.trieID;
39     }
40
41     public Operation getTrieOpe(){
42         return this.trieOpe;
43     }
44
45
46     /*
47      *Set
48      */
49     public void setEst( Model est ){
50         this.est = est;
51     }
52
53     public void setTrieID( String trieID ){
54         this.trieID = trieID;
55     }
56
57     public void setTrieOpe( Operation trieOpe ){
58         this.trieOpe = trieOpe;
59     }
60
61
62     /*
63      *Methods
64      */
65     private String manipulate(String str, String Key){
66         return str;
67     }
68
69     public void loadXMI(String inputFileName){
70     }
71
72     private void loadOperation(String bf, String key){
73     }
74
75     public void runParser(String input FileName){
76     }
77
78     private void parserPackage(String bf, String Key){
79     }
80
81     private void parserClass(String bf, String Key){
82     }
83
84     private void parserAttribute(String bf, String Key){
85     }
86
87     private void parserOperation(String bf, String key){
88     }
89
90     private void parserParameter(String bf, String key){
91     }
92
93 }
```

Figure 2: Code generated from the model loaded in the tool – class "Parser"

```
1
2  public class Attribute extends DataModel{
3
4      /*
5       *Attributes
6       */
7      private String name;
8
9      /*
10      *Constructor
11      */
12     public Attribute(){
13         super();
14         this.name = new String();
15     }
16
17
18     /*
19      *Get
20      */
21     public String getName(){
22         return this.name;
23     }
24
25
26     /*
27      *Set
28      */
29     public void setName( String name ){
30         this.name = name;
31     }
32
33
34     /*
35      *Method
36      */
37     public void printProp(){
38     }
39
40     public void genCode(void out){
41     }
42
43     public void genCodeGet(void out){
44     }
45
46     public void genCodeSet(void out){
47     }
48
49     public void genCodeConst(void out){
50     }
51
52     /*
53      *Abstract Method of Super
54      */
55     public void printProp(){
56     }
57
58 }
```

Figure 3: Code generated from the model loaded in the tool – class "Attribute

Fig. 3 illustrates the generated code from the class "Attribute" of UML class diagram from Fig. 1. In this code is possible to note the declaration of this class as subclass of the superclass named "DataModel", and the

call of the superclass constructor method (line 13) inside of the constructor method of class "Attribute" (from line 12 until 15), and the declaration of a method early defined as abstract in the superclass (line 53 until 56).

4. Conclusions and Future Work

This paper presented a new tool to generate Java code from UML class diagram. As the generator uses only class diagrams, only static information are used in the code generation, which limits the code generation supported by this tool. As future work, we plan to extend the tool in order to support also other diagrams, enabling a more complete code generation, including the sequence of method calls.

Since UML extensions (profiles) are available for the embedded domain, the capturing of UML models should be also extended to allow the capturing of stereotypes used for this specific profiles, allowing consider this information also during the model analysis or code generation.

5. References

[1] SELIC, B. (2003). Models, software models, and UML. UML for real: Design of embedded realtime systems (pp. 1-16). Boston: Kluwer Academic Publishers.

[2] OMG. Unified Modeling Language (UML). Available at: http://www.omg.com/.

[3] BRISOLARA, Lisane; KREUTZ, Márcio Eduardo; CARRO, Luigi . UML as front-end language for embedded systems design. In: Luis Gomes; Joao M. Fernandes. (Org.). Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation. Hershey: IGI Global, 2009, Chapter 1, p. -. ISBN: 978-1-60566-750-8.

[4] SELIC, B. UML 2: A model-driven development tool. Model-Driven Software Development. IBM Systems Journal, Riverton, v. 45, n. 3, p. 607-620, 2006.

[5] TERRIER, F.; GERARD, S. MDE Benefits for Distributed, Real Time and Embedded Systems. In: From Model-Driven Design to Resource Management for Distributed Embedded Systems. Springer Boston v. 225/2006. p. 15-24. Jan, 2007.

[6] ESPINOZA, H.; CANCILA, D.; SELIC, B.; GÉRARD, S. Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems. In: Proc. of the 5th European Conference on Model Driven Architecture - Foundations and Applications, 2009.

[7] OMG. XMI 2.1.1: XML Model Interchange. (OMG document formal/2007-12-01). Available at: <http://www.omg.org>.

[8] Usman, M. and Nadeem, A. (2009). "Automatic generation of Java code from UML diagrams using UJECTOR". International Journal of Software Engineering and its applications (IJSEIA), Daegu, vol. 3, No. 2 (April), pp. 21-37.

[9] Draft Tutorial for Profile usage in Papyrus, Papyrus. http://www.papyrusuml.org

Review of Localization Schemes Using Artificial Neural Networks in Wireless Sensor Networks

[1]Stephan Hermes Chagas, [1]Leonardo Londero de Oliveira, [1]João Baptista S. Martins
{stephan.chagas,leonardo}@mail.ufsm.br, batista@inf.ufsm.br

[1]GMICRO – Universidade Federal de Santa Maria

Abstract

Wireless sensor networks (WSNs) have been applied in several areas, such as military surveillance, environmental monitoring, robotics, domotics, animal tracking, and many others. Localization awareness is an important issue because the collected/transmitted data by network nodes could become meaningless without correct positioning information. Artificial neural networks (ANNs) are not commonly applied in localization tasks, even more when we are talking about WSNs. However, this paper corroborates that ANNs could be used for nodes localization estimation, and with reasonable accuracy. Among the families of ANNs available, the Multi Layer Perceptron (MLP) is the most suitable to application in WSNs, because it has the best computational and memory resource requirements. Since the network nodes are embedded devices with several constraints of energy consumption and computational power, low memory usage and low floating point operations are desirable. This article shows some advantages of using ANNs in localization issues and does also some suggestions about future works and further research directions.

1. Introduction

Due to several enhancements on sensing technology, embedded systems and wireless communication technologies, the wireless sensor networks experienced a reasonable interest growth by the scientific community.

The sensor nodes are now capable of operating with a 3 Volt DC coin-sized battery that could work for many years, depending on the sample rate. These tiny sensors are portable, unobtrusive and can be easily integrated into small devices. There are many applications where these sensor nodes could be used, such as residential, commercial, industrial, medical and military.

One of the most significant problems to be solved is the localization of network nodes without using GPS (Global Positioning System) devices. This fact can be easily justified because the sensor nodes, in most cases, are low-cost small form factor equipment and have several constraints of computational power and energy consumption.

In order to solve this problem, several localization schemes have been proposed in recent years. However, there are few related works that uses neural networks as the operating principle of the localization algorithms. Neural networks have been tested in localizations tasks [1]-[4] with good simulation results. In further sections, localization schemes and artificial neural networks will be more detailed, as well as the interaction between them.

The reminder of this paper is organized as follows. In the next section, the localization task and its relevance will be better described. In section 3, will be introduced the neural networks principles. Section 4 presents some families of artificial neural networks and comments about their performance in localization task, as well some advantages in using ANNs for localization algorithms. This article is concluded in section 5.

2. Localization Schemes

The goal of a localization scheme (or algorithm) is to estimate the coordinates of network nodes in a coordinate system. Several sources of information to do a location estimation of the network nodes could be used. As sources, the schemes can only use connectivity information (content of the message exchanged among the nodes), only signals measurements obtained by some specific hardware modules (acoustic signal strength, RF signal strength, and others), or both connectivity and signal measurements data.

The kind of source of the used data to do the localization task generally determines the classification of the algorithm as range-free or range-based system. The range-free algorithms use only nodes connectivity information. The range-based type operates using signal measurements. Hybrid systems also could be implemented. The nodes collected/transmitted data could probably become meaningless if no location information were supplied within.

2.1. Example of a Range-based localization scheme

The RSSI (received signal strength indication) technique uses the radio signal strength to estimate the distance between two nodes that could communicate to each other. Since the signal strength diminishes with the square of the distance between transmitter and receiver, the node that receives data from other should be able to calculate its distance from the signal source.

However, RSSI ranging measurements could contain noise, because radio propagation tends to be highly uniform due to reflection, diffraction, multipath effects, scattering and possible interferences. Hence, the RSSI measurements are highly dependent on the characteristics of the propagation channel, as well described in [5]. If the nodes environment changes to another place, same collected data with RSSI measurements must probably means different distance estimation.

Based on the estimated distance among nodes with unknown positions and some reference nodes (that have location awareness), the self-localization task could be accomplished by the unknown position nodes using, for example, lateration.

2.2. Example of a Range-free localization scheme

The Centroid algorithm proposed by Bulusu et al [6] is the one of the most well-known localization schemes that uses range-free approach. This technique requires low computational power, due to reduced amount of calculation needed, as well as low power requirements, owing to the receiver-based characteristic, since the communication needed consists only of receiving packets from the reference nodes (nodes that have location awareness).

These nodes, that know their own localization, transmit packets in broadcast mode that contains their coordinates and identification on the network, in a regular period. These packets could be named as *beacons*. The unknown nodes select a group of more appropriated reference nodes based on a calculation defined as *Connectivity Metric,* and then compute their estimated locations as the intersection of connectivity regions covered by such anchors, defined as the centroid of these seeds. As shown in fig. 1, as higher the number of reference nodes, higher the number of intersection of connectivity regions. This fact gives more accuracy for the localization process.

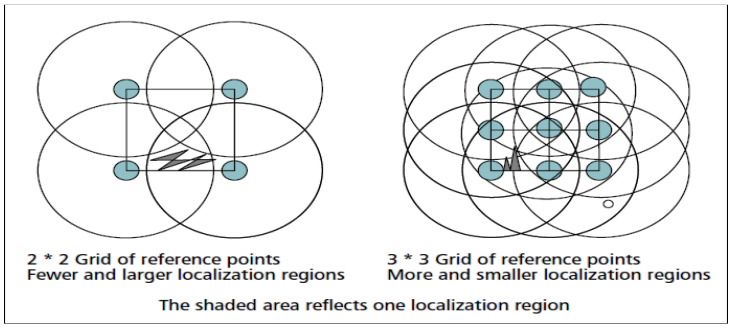


Fig. 1 – Intersection of connectivity regions of reference nodes [6].

3. Artificial Neural Networks

Artificial neural networks (ANNs) are computational and mathematical models based on a simplified vision of biological nervous system. In the network there are groups of neurons that perform a specific physiological function. Basically, an ANN is an adaptive system that receives an input, process the data and provides an output.

3.1. Artificial Neuron

A neuron operates by receiving signals from other neurons through connections, called *synapses*. The combination of these signals, in excess of a certain *threshold* or *activation* level, will result in the neuron *firing*, that is sending a signal on to other neurons connected to it. Some signals act as *excitations* and others as *inhibitions* to a neuron firing.

In nature, a neuron consists basically of three parts: the cell body, the axon and the dendrites.
The signal from other neurons is received by the dendrites and its combination, in excess of a certain threshold, will lead to a change on the neuron output response through the axon. An artificial neuron, by its turn, try to model this behavior in a abstract way. The main idea of the artificial neuron in computational intelligence applications is not to fully model the biological operation but to allow its most basic functionality. Each artificial neuron consists of multiple inputs, weights and a single output. The neuron combines the weighted inputs with reference to a threshold value and the activation function to determine its output.

3.2. Neural Network Structure

The way the artificial neurons connect to each other determines the neural network structure. There are several types of structures, although the most common is called *backpropagation network*.

On this type of structure, there's an input layer with the number of neurons equal to the number of inputs of the net, a variable number of hidden layers (the ones that are not directly connected to the inputs or the outputs) and an output layer with a number of neurons equal to the number of outputs of the net. Usually, this structure is fully connected, which means each neuron from one layer have a weighted connection to each neuron of the following layer.

3.3. Operation and Training

For operation of each neuron of the neural network follows formula (1), where w_{kj} is the weight of each input, x_j is the input signal of each variable and where m is the number of inputs. The bias input is x_0 and its value is arbitrarily equal to +1.

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right) \quad \textbf{(1)}$$

The function φ is the activation function, which is responsible for combining for giving the output from the weighted inputs. There are usually three types of activation function: threshold, piecewise-linear and sigmoid.

After all the structure is done, there's still need to train the network before using it. This process is performed by giving the correct answers for a set of inputs and setting the weights of the ANN until its results match the given answers. This type of learning process is called supervised learning and it is usually performed by the Error Back Propagation algorithm.

4. Artificial Neural Networks in Localization

Different types of artificial neural networks can be obtained by varying the activation functions of neurons and structures of weighted interconnections between them. Examples of these classes are the *Multi-Layer Perceptrons* (MLP), *Radial Basis Function* (RBF), and the *Recurrent Neural Networks* (RNN).

The way through which the MLP network is trained is by testing the performance of the network to all sets of inputs and trying to minimize the measured error between the desired output and the network output. The weights are modified based on this error until it reaches a minimum tolerance.

The weights are obtained by the solution of Green's functions in RBF networks and can be solved, unlike the case of MLP networks. The major drawback is that it spans the input/output data space of the application affecting memory and computational requirements. It is possible however to use fewer nodes to approximate the performance of the network and consequently reducing the requirements. The networks of this type are called Reduced RBF (RRBF) networks.

The structure of the MLP network is very similar to the RNN networks. The only difference is that MLP interconnections are straight forward from the inputs to the outputs while in RNN networks it is possible to have feedbacks to previous layers.

As shown by Shareef et al [1], the use of MLP is more suitable for localization in wireless sensor networks due to its low computational and memory costs. Its accuracy is slightly lower than the RBF (that shown best accuracy, but higher computation resources requirement), but the best trade-off between accuracy and resource requirements is clearly seen.

One advantage of using neural networks to perform localization tasks is that it does not require prior knowledge about the noise distribution of the network environment. The noisy distance measures can be used directly as inputs to train the artificial neural network. ANNs are able to characterize the noise and compensates it for more precision in estimating the location of the node. Other techniques such as Kalman filter, for example, depend on knowledge about the environment distribution of noise to estimate the node's location [7]. When the parameters of noise are not changed, the localization using MLP could be a good option.

Simulation results show that neural networks can achieve high level of accuracy in estimating the location, and requires fewer anchor nodes. This technique is not affected by the NLOS (Non-Line-of-Sight) environment, neither by the irregularity of the power transmitted by the anchors [3].

Locating people in underground environments is a task where artificial neural networks are employed with success. Due to the nature of the environment in these situations, the signals suffer from various multipath effects caused by reflection, refraction, diffraction and collision with humid rough surfaces. In those kinds of circumstances, where the signals are blocked due to NLOS, the traditional location techniques such ones based on RSSI, AoA (angle of arrival) and TDOA (time difference of arrival) lead to high rates of error in location estimate. Dayekh et al [8] proposes a solution based on extracting channel impulse response (CIR) fingerprints with reference to one or more receivers and then use artificial neural networks as a matching algorithm to localize.

Other localization approach that can be applicable to WLAN (Wireless Local Area Network) is shown in [9], where Fingerprint technique is integrated with Trilateration. This algorithm has 44% better accuracy compared to basic Fingerprint technique and 73% better accuracy when compared to basic Trilateration approaches. Moreover, the algorithm that integrates Trilateration and Fingerprints techniques is independent of the environmental parameters such as attenuation loss and path loss exponent, and knowledge of transmitted power is not necessary.

5. Conclusions

In this paper, the basic theory about node localization in wireless sensor networks and the basic principles of the artificial neural networks were discussed, as well some possibilities of the interaction between them (due to the small number of related work found by the authors). The ANNs could be applied in localization algorithms with reasonable accuracy and without the need of prior knowledge about de environment noise, unlike the techniques that apply Kalman filters. Among the artificial neural networks families mentioned, the *Multi-Layer Perceptron* was the most suitable for applications on embedded systems due to its good accuracy and low computational resource requirements, showing better trade-off between these items than other families. Another important characteristic of the using of ANNs is the immunity against errors of localization estimative due to non-line-of-sight environments and the irregular power transmitted by the anchors. As future work, an implementation of a wireless sensor network running localization scheme based on MLP family of ANNs would be made on commercial models of motes, such as the Crossbows' MicaZ.

6. References

[1] Ali Shareef, Yifeng Zhu, and Mohamad Musavi, "Localization Using Neural Networks in Wireless Sensor Networks," Mobilware *Proc. 1st international conferenceon MOBILe Wireless middleware, Operating systems, and Applications,* 2007.

[2] Mohammad Shaifur Rahman, Youngil Park, and Ki-Doo Kim, "Localization of Wireless sensor Network Using Artificial Neural Network," *Proc. 9th international conference on communications and information technologies,*2009.

[3] Jin-Peng TIAN, and Hui-Chang SHI, "Study of Localization Scheme base on Neural Network for Wireless Sensor Networks," *IET conference on Wireless, mobileand Sensor Networks,* Dec. 2007, Pp. 64-67.

[4] Gianni Giorgetti, Sandeep K. S. Gupta, and Gianfranco Manes, "Wireless Localization Using Self-Organizing Maps," *IPSN*, 2007, p. 25-27.

[5] Salvador Jauregui, and Mario Siller, "A Big Picture on Localization Algorithms Considering Sensor Logic Location,". *Proc. IEEE International Conference on Systems, Man and Cybernetics,* 2009.

[6] Nirupama Bulusu, John Heidemann, and Debora Strin, "GPS-less Low-Cost Outdoor Localization for Very Small Devices ," IEEE *Personal Communications*, 2000, p. 28-34.

[7] Ali Shareef, and Yifeng Zhu, "Comparisons of Three Kalman Flter Tracking Models in Wireless Sensor Network," *Proc. 26th EUROMICRO Conf.* (EUROMICRO'00), 2000.

[8] Shehadi Dayekh, Sofiène Affes, Nahi Candil, and Chahé Nerguizian "Cooperative Localization in Mines Using Fingerprinting and Neural Networks," *IEEE Wireless Communications and Networking Conference, p. 1, 2010.*

[9] N.S. Kodippili, and Dileeka Dias "Integration of Fingerprinting and Trilateration Techniques for Improved Indoor Localization," *Seventh International Conference on Wireless and Optical Communications Networks, p. 1, 2010.*

[10] Anthony Taok, Nahi Kandil, and Sofiène Affes, "Neural Networks for Fingerprinting-Based Indoor Localization Using Ultra-Wideband," *Journal of Communications, v. 4 n. 4, p. 267-275, 2009.*

Power Analysis of a Floating Point Unit for a Reconfigurable Architecture

Bruno Heckttheuer, Eduardo Nicola, Mateus Grellert, Júlio C. B. Mattos
{bbhecktheuer,enicola.ifm, mgdsilva, julius}@inf.ufpel.edu.br

Grupo de Arquiteturas e Circuitos Integrados – GACI
Centro de Desenvolvimento Tecnológico - CDTec
Universidade Federal de Pelotas / Pelotas – Brazil

Abstract

With the complexity of embedded systems growing day by day, there are several electronic devices with different applications in a single device. To cope with this heterogeneous behavior of these applications it is necessary embedded architectures providing high performance. Reconfigurable architectures present a solution based on high performance maintaining low power consumption. This paper presents the power analysis of a combinational Floating Point Unit (FPU) for a reconfigurable architecture. The FPU are supports four basic arithmetic (addition, subtraction, multiplication and division). The power results were generated and analyzed for the three target architectures (sum/subtraction, multiplication and division).

1. Introduction

Currently, the majority of electronic devices available on the market have embedded computational units. With the growing market for embedded systems, the complexity of these systems is increasing due to different functionalities in one device, such as mobile handsets. The applications run on these devices have heterogeneous behaviors, i.e., it is necessary different hardware resources for each application to perform high performance. The use of general purpose processors can not guarantee an efficient implementation and produces high consumption int terms of power and energy.

Reconfigurable architectures [1] have the ability to adapt (to reconfigure) according to the type of application. Moreover, these architectures have been presented as a good solution to increase performance and provide low power consumption.

Multimedia and communication algorithms applied in the field of embedded systems make heavy use of floating point arithmetic. Due to the complexity and cost of implementations of floating point arithmetic in hardware, algorithms often make use of emulation in software or conversion (manually or automatically) of floating point operations in fixed point.

The target architecture used in this work consists of a Reconfigurable Functional Unit (reconfigurable array), a unit capable of performing the reconfiguration and a general purpose processor [2]. The basic idea of this approach is to find pieces of code that can be performed more efficiently in the array during program execution. This unit is implemented in a combinational way and it is composed by functional units that perform operations on integer data, multipliers and units of memory access.

This reconfigurable array did not use floating point operations in hardware, which produced the solutions are not adequate in terms of performance in floating point arithmetic. This work uses the Floating Point Unit (FPU) developed in [3] to insert Reconfigurable Unit at the target architecture. This FPU implements operations of addition, subtraction, multiplication and division using the hardware description language VHDL.

This work aims to develop a methodology to analyze the power dissipation of four PFU modules. The paper is organized as follows: section 2 presents the modules used in the paper. Section 3 shows the power analysis methodology. Section 4 presents the results of power dissipation and finally, section 5 presents the conclusions and future work.

2. Target Architectures

Three architectures are analyzed: sum/subtraction, multiplication and division to floating point numbers in single precision, ie, using 32-bit representation in IEEE 754 [4]. This standard defines how to represent floating point numbers are stored in memory, rounding algorithms, exception handling, etc. The modules were developed through a fully combinational logic, using the standard algorithms, taking for granted that the entries are in the format specified by IEEE 754. The modules were implemented using VHDL and synthesized using Xilinx tool 10.1.

- *Sum/Subtraction Module*

The implementation of the algorithm of sum / subtraction in sequential logic is relatively simple, but its hardware implementation in combinational way can be considered a challenge. Figure 1 illustrates the structure of module sum / subtraction used in the workplace. This module contains a sum/subtractor and normalization steps. The stages of normalization have a complex hardware.

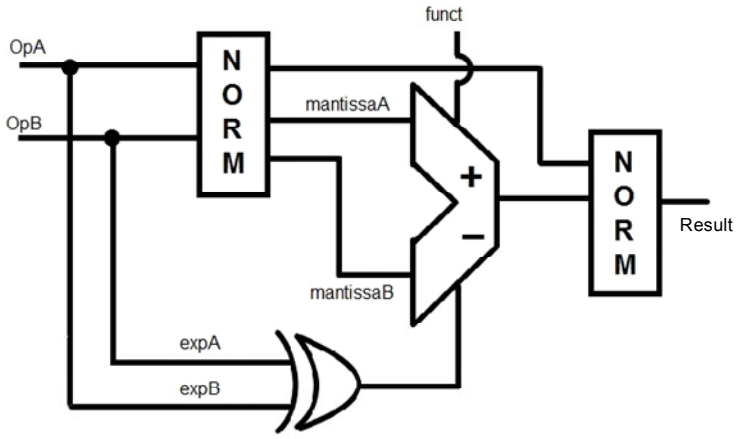


Fig. 4 - Structure Module Sum / Subtraction

- *Multiplication Module*

The multiplication algorithm performs basically the sum of the exponents of the operands and the multiplication of mantissas, check overflow / underflow and normalize the result (if necessary). Steps have been implemented in a way similar to the steps of module sum / subtraction. Figure 2 shows the structure of module multiplication.

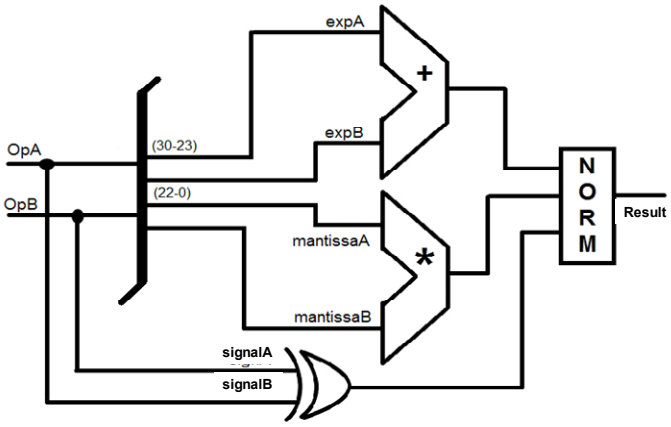


Fig. 2 - Structure Module of the Multiplication

- *Division Module*

The division module has been simplified using the architecture of the multiplication module. In other words, divide X by Y is equivalent to multiplying X by the inverse of Y. The values of 1 / Y is stored in a ROM memory. Memory size ranges from 1KB to 32MB providing different solutions in terms of area and accuracy. Larger memories provide greater accuracy, however, require a larger area. Despite occupying a large area, this procedure presents a low complexity. Figure 3 illustrates the division module.

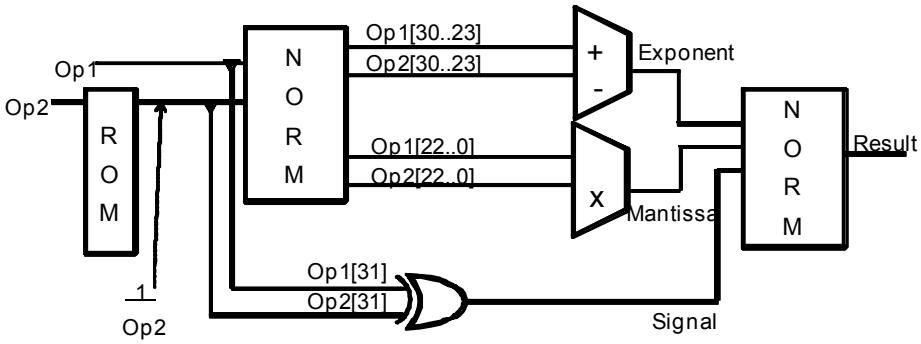


Fig. 3 - Structure Module of the Division

3. Power Analysis Methodology

The PowerCompiler [5] tool was used to generate power data for the three target architectures. The library used was the logic cells StandardCell 0.18 [6]. To use the tool PowerCompiler took some procedures. Because the modules were described in VHDL, it was necessary, through a script, to convert to the Verilog language each module, since the tool only supports Verilog language. After the generation of the Verilog files for each architecture, it is necessary to generate input data for each unit separately. A Verilog testbench was used also described to simulate the inputs of the circuits. Figure 4 illustrates the design flow of power analysis metodology.

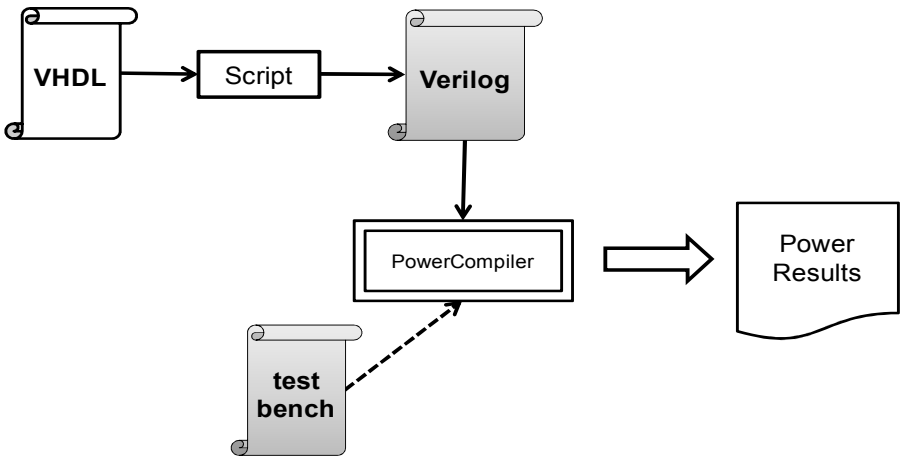


Fig. 4 – Power Analysis Diagram

4. Results

This section presents the results of power modules for sum / subtraction, multiplication and division. The results obtained for power as the technology StandardCell 0.18 TSMC [6]. The Table 1 presents the results in terms of dynamic power, static power (leakage) and total power (dynamic plus static) provided by PowerCompiler tool. The module sum / subtraction showed the worst results in terms of power. The results of the power division of multiplication modules are very close. However, the results of division module did not consider the power dissipated by ROM memory access, because the sofwtare used can not support analysis of memories.

Tab. 1. Power Results for FPU components

Modules	Dynamic Power (mW)	Static Power (nW)	Total Power (mW)
Sum/Subtract	55.6611	71.5245	55.7326
Multiplication	18.4919	168.1223	18.6600
Division	18.6585	170.1281	18. 8286

5. Conclusions and Future Works

This paper presented the analysis of power module sum/subtraction, multiplication and division to a Floating Point Unit to fully combinational for use in a reconfigurable architecture. Data were analyzed from the static and dynamic power architectures, and can see that the data were quite satisfactory. As future work we intend to compare the results of FPU power architectures developed entirely with combinational logic to sequential. In addition, we intend to integrate the modules of the FPU Reconfigurable Architecture.

6. References

[1] COMPTON, K. AND HAUCK, S. Reconfigurable computing: a survey of systems and software. ACM Comput. Surv. 34, 2 (Jun. 2002), 171-210.

[2] BECK, A. C., RUTZIG, M. B., GAYDADJIEV, G. e CARRO, L. Transparent reconfigurable acceleration for heterogeneous embedded applications. In: DESIGN, AUTOMATION AND TEST IN EUROPE, New York, 2008. In Proceedings of DATE 2008. New York: ACM, 2008. 1208-1213.

[3] SILVA, Mateus Grellert et al. Implementação de uma Unidade de Ponto Flutuante para uma Arquitetura Reconfigurável. In: XVI IBERCHIP Workshop, 2010, Foz do Iguaçu, Brazil. Proceedings of XVI IBERCHIP Workshop. Foz do Iguaçu : IBERCHIP, 2010.

[4] IEEE Standards Committee 754. IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985. New York: IEEE, 1985.

[5] SYNOPSYS. Synopsys Power Compiler; http://www.synopsys.com/products/logic/design_compiler.html

[6] ARTISAN COMPONENTS. TSMC 0.18 μm 1.8-Volt SAGE-XTM Standard Cell Library Databook. 2001.

ARITHMETIC AND DIGITAL SIGNAL PROCESSING

Impact of Process Variability considering Transistor Networks Delay

Jerson Paulo Guex, Cristina Meinhardt, Ricardo Reis
{jpguex, cmeinhardt, reis}@inf.ufrgs.br

PGMICRO, PPGC, Instituto de Informática
UFRGS

Abstract

This paper presents an analysis of process variability considering the use of extracted layouts with their respective capacitance and parasitic resistance. The effects on PullDown and PullUp networks of transistors are verified separately. The performed experiments aimed to analyze the variability when using serial and/or parallel transistors. Also it analyzes the influence of process variability in fall delay, rise delay. Preliminary results demonstrated that both networks could be less sensitive to process variability if the network is composed by more than 2 transistors in parallel. The use of larger transistors than the minimum possible size for the used technology node also reduces the effects of variability [10].

1. Introduction

The continuous shrinking of devices adds new challenges to integrated circuit design due to variability of transistor's technology parameters and dimensions. Variability can provide power consumption outside the design specifications, which can allow faster circuit degradation. In some cases, it can occur erroneous circuit functionality, that makes it unsuitable or with use restrictions. Prediction of such variability in the design phase can increase the success rate in the circuit development.

In nanometer scale, decreasing parasitic capacitances increase the transistor speed. On the other hand, there is an increase of the current mismatch, especially if device geometries vary in the manufacturing process [1]. The mismatch of parameters occurs in various steps of manufacturing process and the main parameters affected are transistor dimensions, mobility of electrons or holes and oxide thickness of the transistor gate [2].

According to [4], the variability may be classified by environment factor, commonly understood by variations in power supply and temperature; reliability factor, related to aging of the transistors and their effects like negative bias instability temperature (NBTI), Hot Carries and electromigration; and physical factors related to variations in parameters that induce a mismatch in transistor performance.

These parameters variations turn the circuits less predictable and demand a great effort to create suitable techniques to deal with this uncertain performance. Design for Manufacturability (DFM) is an effort to deal with this issue. DFM has a set of techniques that can be used to improve the manufacturability. According to [5], a regular layout is one of DFM techniques that can improve predictability in the physical implementation of a circuit. Thus, it can be obtained more accuracy in delay estimates.

Parameters like oxide thickness (Tox), width (W), length (L), threshold voltage (Vth), effective channel length (Leff), and change in resistance values of metals after CMP and mobility of transistor's charges (electrons or holes) are the most important parameters affected by process variability [7]. In Fig. 1, it is possible to see some trends on parameter variability.

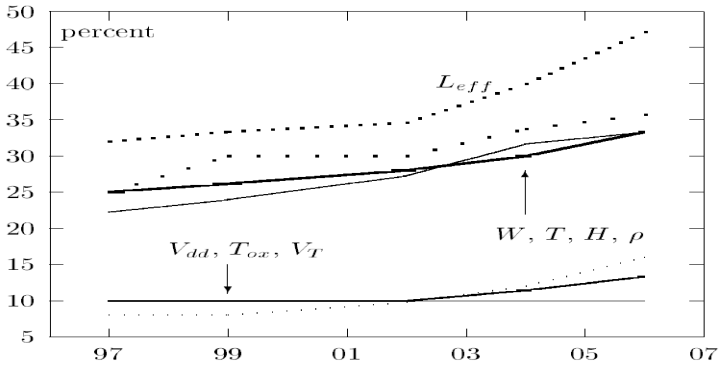


Fig.1-Estimates on Parameters Variation [7]

Hierarchical or spatial mechanism can classify the variability mechanisms in physical factor. These mechanisms can be systematic or random [3] or design dependent according to [4]. The systematic mechanism can be found in many dies or wafers like a rapid annealing. The random mechanisms are related to doping the transistor channel and correction of threshold voltage (Vth).

This paper is focused on physical factors that are imposed in each stage of manufacturing. In this experiment, variability effects over extracted layouts are analyzed considering the parasitic capacitances and resistances. All layouts explore regular patterns on poly layers.

The remainder of this paper is as follow. In section 2 we describe the experimental work and the applied methodology. Results are explained in section 3 and conclusions are presented in section 4.

2. Methodology of the Experiments

The main objective of this work is to analyze the behavior of circuits designed using a 65nm technology considering the effects of process variability in rise and fall signal transitions. It was taken into account the effects in PullUp and PullDown networks. This allows evaluating the influence on delay of stacked transistors, parallel transistors and mixed arranges. The layouts of the experiments consider the logical effort, method used to estimate delay in CMOS circuits [11], to determine the transistor sizing for each layout. The experiments were done as following:

1. Choosing of 18 circuits to test;
2. Layout Design of selected circuits in a 65nm technology;
3. Verification of layouts versus schematics and extraction;
4. Modification the technology model files to reflect the variability effects;
5. Running of Monte Carlo simulations using HSPICE;
6. Analysis of the results.

Each transistor network was considered separately. For PullDown network was designed 9 layout versions: three layouts to represent schematic diagrams with 2, 3 and 4 transistors in series; three layouts that represent schematic diagrams with 2, 3 and 4 transistors in parallel; and three layouts that represent the union of 2, 3 and 4 transistors in series/parallel. For PullUp network 9 layouts were also designed in the same conditions and specifications that PullDown.

Fig. 2 and 3 show the schematic diagram for some of the layouts. All layouts designed for this experiment were built aiming to be as regular as possible. The layouts were implemented using Cadence Virtuoso. After design rule check (DRC) of the layouts, they were compared with the schematic (LVS) to ensure the correctness of the layout. Following, the layout was passed through the extraction tool to obtain the parasitic capacitances and parasitic resistances.

In order to simulate the process variability, transistors parameters like width (W), length (L), oxide thickness (Tox), effective channel length (Leff) and threshold voltage (Vth) were varied by 10% of their nominal values. The correlations between these parameters is considered. All parameters are changed in the model and this model employs these basic parameters to determine all the others parameters values. The variability adopts a Gaussian distribution with three sigmas around the default values.

The Monte Carlo technique was adopted with a Gaussian distribution to deal with simulation. Even though this method is very expensive in terms of computing power and is not suitable for complex circuits with over one hundred transistors [6], in this work the method works efficiently because only small circuits are explored. These small circuits reflect blocks often found in complex circuits. Each of the extracted layouts was simulated using Hspice with the Monte Carlo method and doing 10.000 iterations per simulation.

The transistor model used for those experiments is a Low Power Typical model from the same foundry that provides the design kit of 65nm. A load capacitance of 10fF was placed in each extracted circuit output to avoid floating terminals to emulate a real situation of operation. Each input was fed with the same signal with a 1ns of period.

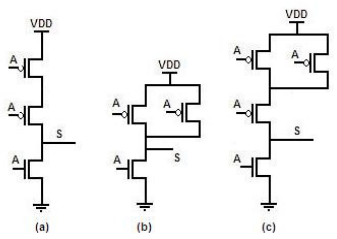


Fig. 2 -Schematic diagram of pulldown network with: (a) 2 transistors in series, (b) 2 in parallel and (c) 2 transistor in parallel and 1 in series

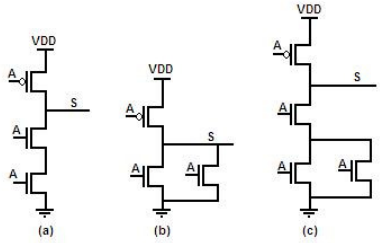


Fig. 3 - PullUP network with: (a) 2 transistors in series, (b) 2 transistors in parallel and (c) 2 transistors in parallel with 1 in series.

3. Experimental Results

The simulation experiments run over HSpice, performing Monte Carlo method with 10000 iterations using the BSIM4 model. For PullDown networks (series/parallel and mixed with 2, 3 and 4 transistors) were observed the mean values for fall delay, fall time of the analyzed network and their respective standard deviations. In the same way, the results for mean value and standard deviations of rise delay, rise time and power estimation of PullUP network were acquired.

The results for the PullDown network show that the normalized fall delay deviation tends to decrease in function of the number of stacked transistors and their respective width. The normalized fall delay deviation decay approximately by 9% comparing the scenarios with 2 and 4 stacked transistors.

Looking to the result of normalized fall delay deviations it is possible to conclude that parallel arrangements with 3 or 4 transistors could be a good strategy for the design of circuits more tolerant to process variability. The junction of series and parallel transistor shows that this combination could be more sensitive to the process variability. This experiment shows a linear trend in increasing of normalized fall delay deviation when both parallel and series arrangement are put together with 3 or 4 transistors. According to the obtained data, the implementation with 2 transistors in both parallel and stacked seems to be a good strategy to cope with process and to deal with fall delay constraints. Tab. I show the results of normalized fall delay deviation during the simulations. The results are normalized dividing the standard deviation by its mean value. This normalized process was made to provide a fare comparison between the experiments.

Tab.1 – Normalized Fall Delay Deviation

# Transistors	Series	Parallel	Both
2	0.03263	0.03140	0.02917
3	0.03016	0.02358	0.03244
4	0.02987	0.02016	0.03458

The PullUP network results show that the rise delay deviation tends to decay according to the number of stacked transistors. This effect is noticed because of the continuous increment of transistors width according to the number of transistors in series observing the relation of $W_p/W_n=2$ mentioned in Section II and the position of transistors according to the Figure 2. Parallel transistors arrangements have also the rise delay deviation decay in all the situations, which include 2, 3 and 4 transistors.

With the arrangements to perform this experiment for the PullUP network, it is possible to compare the decrease of the rise delay deviation in function of the number of transistors in series or parallel and with both topologies. The decay of rise delay deviation in paralle and series arrangements is a direct product of the area of each transistor, as showed in Tab 2. As greater is the transistor, less is the rise delay deviations Adopting a mixed arrangement, in the booth columm, the experiments show a tendency of stability in rise delay deviation with 2, 3 and 4 transistors. Taking in account all the aspects related above, the best solution to implement a PullUP network regarding the effects of process variability is by using a parallel structure that guarantee a minor effect on rise delay deviation. This consideration is only valid if the transistors are not using minimal dimensions for the technology used.

Tab.2 – Normalized Rise Delay Deviation

# Transistors	Series	Parallel	Both
2	0.03039	0.02546	0.02805
3	0.02642	0.01965	0.02761
4	0.02164	0.01563	0.02887

4. Conclusion

The experiments performed over the PullDown and PullUp networks show the dependence of variability in relation to the size of transistors. It can be verified in Figure 4, that the normalized fall and rise delays have the same behavior for variability decay when the number of transistors and their sizes in serial and parallel arrangements is increased. This indicates that the use of the minimum possible dimensions of the technology could be avoided to decrease the effects of process variability. Other conclusion highlights regarding the use of more parallel arrangements in PullDown and PullUp networks when the target is minimize the fall and rise delay deviation fluctuations under influence of process variability.

Fig. 4 – Comparison between rise and fall delay deviation.

5. Acknowledgment

The authors would like to thank the CAPES, Brazilian Funding Agency and also to the Walter E. C. Bartra for his help and collaboration.

6. References

[1] G. Tulunay,; G. Dundar,; A. Ataman,; , "A new approach to modeling statistical variations in MOS transistors,". ISCAS 2002. IEEE Int. Symp. on Circuits and Systems, vol.1, pp. I-757- I-760 vol.1, 2002.

[2] M. J. M. Pelgron, A. C. J. Duinmaijer, A. P. G. Welbers,Matching Properties of MOS Transistor, IEEE Journal of solid state circuits, October 1989.

[3] A J Strojwas, Conquering Process Variability :A Key Enabler for Profitable Manufacturing in Advanced Technology Nodes,. IEEE International Symposium on Semiconductor Manufacturing, ISSM 2006, San Jose, USA. pp xxiii `a xxxii

[4] S. R Nassif, Process variability at the 65nm node and beyond,. IEEE Custom Integrated Circuits Conference 2008. San Jose, USA, pp 1-8

[5] C. Menezes,; C. Meinhardt,; R. Reis,.; R. Tavares,; , "Design of Regular Layouts to Improve Predictability,", 6th Int. Caribbean Conference on Devices, Circuits and Systems, pp.67-72, April 2006.

[6] J. A. Power, B. Donnellan, A. Mathewson, W. A. Lane, Relating Statistical MOS-FET Model Parameter Variabilities to IC manufacturing Process Fluctuations Enabling Realistic Worst Case Design. IEEE Trans. on Semiconductor Manufacturing, August 1994.

[7] S.R. Nassif,; , "Design for variability in DSM technologies [deep submicron technologies]," ISQED.IEEE 2000 First Int. Symp. on Quality Electronic Design,, pp.451-454, 2000.

[8] D. da Silva, A. I. Reis, R. P. Ribas, "Gate delay variability estimation method for parametric yield improvement in nanometer CMOS technology", Microelectronics Reliability, vol. 50, no. 9-11, 21st European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis, 2010, pp 1223-1229.

[9] Singh A, Mani M, Orshansky M. "Statistical technology mapping for parametric yield" IEEE/ACM Int, Conf. on Computer-Aided Design (ICCAD), November 2005, pp 511-8 [12] R. Bartle, "Early MUD History," Nov. 1990; http://www.ludd.luth.se/aber/mud-history.html.

[10] Kheterpal, V., Rovner, V. et al. "Design Methodology for IC Manufacturability Based on Regular Logic-Bricks"Design Automation Conference (DAC), June 2005, Anaheim, California, USA.

[11] Sutherland, I. E., "Logical Effort: designing fast CMOS circuits", Morgam Kaufmann, São Francisco, USA, 1999.

Area and power Optimization of Radix-2 Decimation in Time (DIT) FFT Implementation Using MCM Approach Along the Stages

[1,2]Sidinei Ghissoni, [3]Eduardo Costa, [1]Ricardo Reis
{sghissoni,reis}@inf.ufrgs.br, ecosta@ucpel.tche.br

[1] Universidade Federal do Rio Grande do Sul - UFRGS
[2] Universidade Federal do Pampa - UNIPAMPA
[3] Universidade Católica de Pelotas - UCPEL

Abstract

This paper proposes the optimization of area and power of architecture radix-2 Decimation in Time (DIT) FFT using the Multiple Constant Multiplication (MCM) approach along the stages. The MCM problem has been largely applied to the reduction of the multipliers in digital filters. In MCMs, the operations over the constants are implemented by using addition/subtractions and shifts rather than the use of general multipliers. In FFT filters, the butterfly algorithm plays a central role in the complex multiplications by constants. Thus, the use of the MCM in the butterflies can reduce significantly the number of real and imaginary multiplications by constants. It can be obtained by sharing the twiddle factors of the butterflies as much as possible. In this work, we have implemented two and three stages of 16 bit-width butterfly radix-2 with decimation in time for 4 and 8-point FFT respectively, using both the MCM and gate level approaches. For each stage of the real and imaginary parts of the butterflies we are able to apply the sharing of partial coefficients using MCM. The results were obtained by synthesizing the circuits in the CADENCE Encounter RTL Compiler tool for the UMC130nm technology. The results show that reductions of 10% in area, and 7% in power are achievable when compared with the synthesis logic of the implemented behavioral unrestricted architecture.

1. Introduction

Fast Fourier Transform is an important tool which is largely used in Digital Signal Processing (DSP) applications, such as audio and video process, wireless communication. The various existing FFT algorithms use the partitioning of the set of input samples into sequences of half of the length of the original sequence. This is done recursively until there are only sequences of length 2, where the input samples cannot be more partitioned. This algorithm is named radix-2 FFT [1]. Partitioning the input sequence into more than two subsequences leads to higher radices of the FFT algorithm, what leads to the increase of the number of arithmetic operations. The complexity of the FFT architecture design is mainly given by the multiplication of the inputs by a large number of coefficients.

A large amount of researches has concentrated the implementation of efficient multipliers, in order to optimize area and power consumption of this arithmetic operator. However, in the case of the FFT circuit, where a large amount of complex multiplications are performed by the butterflies, even the use of these efficient multipliers has not enabled optimizations in fully-parallel FFT architectures.

The Multiple Constant Multiplication (MCM), i.e., the use of multiplication of a set of constants by a variable, approach has enabled significant impact on the design of Digital Signal Processing (DSP) area. In the MCM operation, each constant is implemented using only addition/subtraction and shift operations rather than using a general multiplier for each constant. A large amount of algorithms has been proposed to optimize the multiplier block in digital filters by using the MCM approach [2]. However, only a few of them has been applied to FFT in matrix form, as in this work. Thus, we propose to optimize a radix-2 DIT butterfly by using the MCM approach along the stages, in order to allow for further implementation of an efficient FFT architecture. We have used the algorithm of [2] for the generation of the tree of adder/subtraction and shift operations. After, the algorithm presented in [12] was used for the optimization of the circuits by considering the use of the efficient metric gate-level.

This paper is organized as follows: in Section 2, we present an overview of Multiple Constant Multiplications with FFTs algorithms. In section 3 it is presented the application of the MCM approach in a butterfly, as matrix. In Section 4 some result comparisons between butterflies using MCM and gate level are presented. Finally in Section 5, we present the conclusions of this work and some ideas for future works.

2. Application of the MCM Problem in the FFT Architecture

The Fast Fourier Transform (FFT) algorithm is a simpler form to calculate the Discrete Fourier Transform (DFT) efficiently. In the last years, many algorithms have been proposed for the improvement of performance of the FFT butterflies. Equation 1 presents the radix-2 butterfly with decimation in time [1], where the N^2 multiplications obtained in the direct DFT are reduced to $\log_2 N$ multiplications. This aspect enables a real increase of computational performance in the solution of the Fourier transform. The FFT can reduce the

computational complexity of the DFT, because its butterfly can process the calculation of two samples at a time.

Several other algorithms were developed to further reduce the computational complexity of the butterfly, such as radix-4, split-radix, radix-2^2, radix-2/4/8, and higher radix versions. However, all of them are based on the butterfly of [1].

$$X(K) = \sum_{n=0}^{N-1} x(n) W_N^K, K = 0,1,N-1 \quad (1)$$

where $W_n = e^{-j\frac{2\pi}{N}}$ is the twiddle factor.

2.1 Related Work

In the last years several works have been proposed for the optimization of FFT architectures. As an example, in [3] it is proposed a new radix-2/4/8 algorithm, which can effectively minimize the number of complex multiplications in pipelined FFTs. In [4], an optimization of FFT architecture based on multirate signal processing and asynchronous circuit technology is proposed. In [5], solutions based on parallel architectures for high throughput and power efficient FFT cores are presented. In the work of [5], different combinations of hybrid low-power techniques are exploited, such as i) the use of multiplierless units, which replace the complex multipliers in the FFTs, ii) the use of low-power commutators, which is based on advanced interconnection, and iii) the use of parallel-pipelined architecture. The proposed methods also use MCM for the sharing of various multipliers that are located in the same stage of the hybrid architectures. Although this methodology is efficient, in some cases area has been increased.

In [6] it is proposed the optimization of the twiddle factors using trigonometric identity for few points of FFT architecture. The presented methodology proposes the replacement of the adders of the circuits by the use of multiplexers. Based on the same idea, the work of [10] proposes a Low-Complexity Reconfigurable Complex Constant Multiplication for FFTs for the reduction of area for a larger number of points (32 points). This new methodology proposed by [6] and [10] was compared against the works [7-9], where reductions in terms of the number of adders could be achievable. Although the authors of [6] and [10] do not present power and delay results in their work, they comment that probably these metrics may lead to large circuits, due to the limitation of the proposed architecture, where only FFT computation in serial form is performed. In [11] it is presented methods for designing multiplierless implementations of fixed-point rotators and FFTs, in which multiplications are replaced by additions, subtractions, and shifts. These methods minimize the adder-cost (the number of additions and subtractions), while achieving a specified level of accuracy, but it takes much more processing time, with increasing the width of the bits.

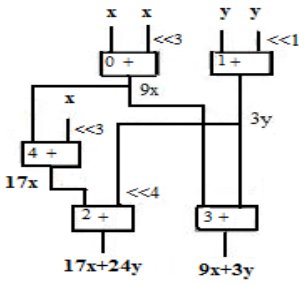
Recently Aksoy [12] proposed the implementation of the MCM parts of FIR filters by determining the cost of each operation in terms of HAs, FAs and logic gates in a given technology library under two inputs. Thus, the cost function of an operation at the gate-level depends on: i) the type of operation (addition or subtraction); ii) the shifted input in a subtraction (minuend or subtrahend); iii) the number of shifts at the inputs; iv) the position of the operation in the architecture (influences the number of bits); and v) the range and type of numbers considered (unsigned or signed). Since the shifts are free in terms of hardware, the filter coefficients and partial terms are considered as odd numbers.

Although there are several techniques to reduce the complexity of the FFT architectures, only a few of them uses the MCM approach for the sharing of the real and imaginary twiddle factors in the butterflies. In our work, we have presented the use of Matrix-MCM algorithm based on the proposed algorithm of [2] associated to the gate level metric of [12] in order to reduce the complexity of the radix2 butterfly of the FFT with decimation in time.

3. Implementation of the Radix-2 DIT FFT by Using MCM Along the Stages

In this method each output of the DIT FFT was represented as a vector composed by the association of their respective twiddle factors multiplied by each respective input. Firstly, each matrix is divided into real and imaginary parts in order to achieve maximum optimization. After that, the matrices results are optimized by the algorithm based on [2], where the results are now composed by adder/subtrators and shifts. In the resultant terms of each butterfly the gate level metric is applied. As an example, figure 1, suppose we have the following matrix representation:and their linear transform representation:

$$\begin{pmatrix} 17 & 24 \\ 9 & 3 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$



z_10=x+x<<3

z_11=y+y<<1

z_12=z_10+x<<3

z_1=z_10+z_11<<4

z_2=z12+z_11

Fig. 1 –Linear transforms representation using subexpresion sharing:

In fig. 2, it is presented the flow for the fully-parallel 8-point radix-2 FFT architecture with decimation in time. The FFT is divided into three stages and each one of them is composed by for butterflies. The butterflies allow the calculation of complex terms, where one complex addition, one complex subtraction and one complex multiplication are involved in the butterfly block. The blank rectangles shown in fig. 2 represent circuit registers. The multipliers present in the butterflies were implemented by using adders and subtractors with gate level metric and MCM approach.

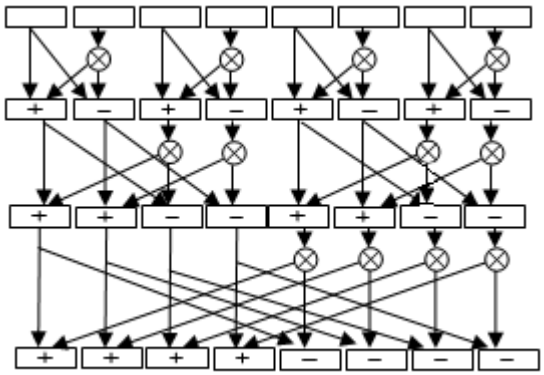


Fig. 1 – 8-point radix-2 DIT FFT architecture.

In an 8-point Fully-parallel FFT implementation, 12 real and 12 imaginary terms are performed in the butterflies (3 stages with 4 butterflies). Thus, the architecture presents large hardware requirements in terms of arithmetic operators. In order to reduce the number of operators, we have applied the MCM approach to the twiddle factors of the butterflies. Particularly, we have used the algorithms of [2], which enable the optimization of the number of operations under a delay constraint. Firstly, the algorithm removes the repeated twiddle factors at each stage. After the initial checking, it is possible to observe that, in fact, in the 8-point radix-2 DIT algorithm, only three different coefficients are generated (by considering that the negatives coefficients are easily obtained by the positive ones using the 2´s complement operation). With the coefficients the matrix is generated and then, finally the gate level metric [12] is applied.

4. Results

This section shows the obtained results of 4 and 8-points, 16-bit radix-2 DIT FFT implemented in hardware description language - VHDL. Table 1 presents area and delay results obtained from the implementation of fully-combinational and optimized multipliers (using adder/subtractors), that are guided from the MCM algorithm of [2] and gate level metric[12] approaches. The logic synthesis was performed with the CADENCE Encounter RTL Compiler tool for the UMC 130nm technology. In the behavioral implementations the FFTs are described in VHDL language and the Compiler tool optimizes the arithmetic operators automatically. All the simulations are limited by the path delay of the circuits.

Tab.1 – Architecture Radix-2 DIT FFT results

Circuits	Cells	Area (mm2)	Power(w)	Path delay(ps)
FFT 4P,16b- CMM (2 stages)	7132	18760	20,12	8814
FFT 4P,16b- CMM (2 stages and gate level)	6981	18519	19,85	8792
Behavioral	7448	19746	20,01	8753
FFT 8P,16b- CMM (3 stages)	12515	35198	35,0	12689
FFT 8P,16b- CMM (3 stages and gate level)	11734	34211	33,3	12494
Behavioral	12741	36127	35.2	12439

In table 1, except for the 4-point FFT, there is a reduction of up to 5% in power consumption for the architectures implemented with MCM and using Ripple Carry adders (RCA) compared with the architectures described in behavioral form and synthesized by the commercial tool automatically. Another point to be emphasized it that with the increase in the number of points, area has been reduced in the FFTs with the use of CMM (Constant Matrix Multiplication) approach, mainly when the gate level approach is used. Area reduction occurs because the use of MCM enables a greater sharing of subexpressions. Moreover, the replacement of multipliers in the butterflies at the stages of the FFTs for adders/subtractors results in the reduction of power consumption of the circuit, when compared with the behavioral description, where the arithmetic operators form the tool are used. The 4-point FFT implemented with Ripple Carry adder does not reduce power consumption directly, because the reduced possibility of sharing the coefficients between the stages. However, the use of MCM and gate level together, allows for the reduction of power consumption, even for the 4-point FFT, as can be seen in Table 1.

Observing the reduction of area, when the adders are implemented with gate, and when the MCM method is applied between the stages of the FFTs, it is clear that the larger number of points allows for a higher reduction of area and power consumption under the same delay restrictions imposed for the synthesis of behavioral descriptions obtained by the commercial tool Encounter RTL.

5. Conclusions and Future Works

In this study we have applied the MCM approach in the stages of of 4 and 8-point radix-2 DIT FFT. The obtained results show that the use of the MCM and gate level approaches simultaneously, can reduce area and power of the fully-parallel FFTs, when compared with the implementations done in behavioral description. It occurred due to the fact that the multipliers of the butterflies are replaced by adders/subtractors and shifts, when using the MCM approach. We have used the algorithm proposed in [2] and [12] for the implementations with MCM and gate level. Since the MCM algorithm reduces the number of operations under a delay constraint, we were able to reduce area and power of the FFTs simultaneously.

With obtained results, we are convinced that the large impact on area and power reductions provided by the MCM and gate-level can allow a low power FFT implementation. Thus, as future work, we intend to verify the low power aspect that can be obtained by the use of the MCM and gate-level approaches in FFTs with larger number of points.

6. References

[1] J. W.;Cooley, J. W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation”, [S.l.], v.19, n.90, p.297–301, 1965.

[2] Aksoy, L., Costa, E., Flores, P. and Monteiro J., 2008. Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(6), 1013-1026.

[3] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, “A New VLSI-Oriented FFT Algorithm and Implementation,” Proc. of Eleventh Annual IEEE Int’l ASIC Conference, 1998, pp. 337-341.

[4] K. Stevens and B. Suter, “A Mathematical Approach to a Low Power FFT Architecture,” IEEE Int’l Symp. on Circuits and Systems, vol. 2, 1998, pp. 21-24.

[5] W. Han, T. Arslan, A. T. Erdogan and M. Hasan, “High-performance low-power FFT cores,” ETRI Journal, vol. 30, no. 3, pp. 451–460, June 2008.

[6] J.-E. Oh and M.-S. Lim, “New radix-2 to the 4th power pipeline FFT processor,”IEICE Trans. Electron, vol. E88-C, no. 8, pp. 1740–1764, Aug. 2005.

[7] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, “Simplified design of constant coefficient multipliers,” Circuits, Systems and Signal Processing, vol. 25, no. 2, pp. 225–251, Apr. 2006.

[8] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, “Simplified design of constant coefficient multipliers,” Circuits, Systems and Signal Processing, vol. 25, no. 2, pp. 225–251, Apr. 2006.

[9] Y. Voronenko and M. P¨uschel, “Multiplierless multiple constant multiplication,” ACM Trans. Algorithms, vol. 3, no. 2, May 2007.

[10] F. Qureshi, Gustafsson, F. O.” Low-complexity reconfigurable complex constant multiplication for FFTs ” Circuits and Systems. ISCAS 2009. IEEE International Symposium on, pp. 1137-1140 May 2009.

[11] M D Macleod, " Multiplierless Implementation of Rotators and FFTs ", EURASIP Journal on Applied Signal Processing, 2005:17 (2005) pp 2903-2910, Sept 2005.

[12] L. Aksoy , E. Costa, , Flores, P. and J. Monteiro “Optimization of Area in Digital FIR Filters using Gate-Level Metrics”DAC 2007.

Development of the Overlap and Add Block for SoC-SBTVD Audio MPEG4–AAC Decoder and Hardware Interface with the wm8731 CoDec

[1]Renê A. Benvenuti, [2]Adriano Renner, [3]Altamiro A. Susin
[1]augusto.benvenuti@ufrgs.br, [2]adriano.renner@ufrgs.br, [3]susin@eletro.ufrgs.br

Universidade Federal do Rio Grande do Sul

Abstract

SoC-SBTVD is a complex project that involves a few Brazilian universities focused on development of the Brazilian digital television Set Top Box (STB). In the LaPSI laboratory at UFRGS we are developing the hardware of the STB in the System-on-Chip (SoC) approach. The architecture of the SoC will contain a CPU and memory, the audio and video decoders and peripherals. The SoC is described in VHDL and synthesized to FPGA as a validation process to be afterwards mapped to silicon. This paper presents a way to test the audio MPEG4–AAC decoder enabling to play a decoded audio stream. The focus is the creation of the overlap and add part of the decoder and also the creation of an interface between the decoder and the codec wm8731, which will permit to appreciate the recovered sound in the audience room.

Key words: Overlap and Add, codec wm8731, audio output.

1. Introduction

When we talk about audio measurements many times we are interested both in a quantitative and in a qualitative point of view. Referring to audio coding and decoding, the quantitative measure may be the difference of the original stream and the recovered one. Here we are mainly interested in a qualitative measure, which means that we want to reproduce the audio signal and to have a feeling about how good it "sounds".

Looking for one way to qualitative assessment of the decoded stream, was proposed an interface which could connect the output of the MPEG-4 AAC audio decoder to the wm8731 codec chip included in the development kit used for design, and allow to play the decoded data.

Overlap and add is a computational low cost method to calculate the discrete convolution between an extensive signal and a finite impulse response filter. The overlap and add block was developed to be part of the audio decoder and to be connected to the audio output interface.

2. Development Environment

It was used in this project the Altera's DE2 development kit, which includes the Ciclone II FPGA and the Wolfson wm8731 audio codec.

The codec uses the following standard protocols to communicate itself:

- I2C interface: Used for sending command data and to set the desirable functionalities. Will be denominated as control interface;
- I2S interface: Used to exchange data and clocks such as bit rate clock and the left right selection clock. Will be denominated as data interface.

For simulation and synthesis of the code were used Quartus II and ModelSim development tools.

For validation of the blocks is used a C++ source code developed at Universidade de Brasília (UnB).

3. Audio Output Interface

The goal of this work is, therefore, to create an interface as shown in figure 1:

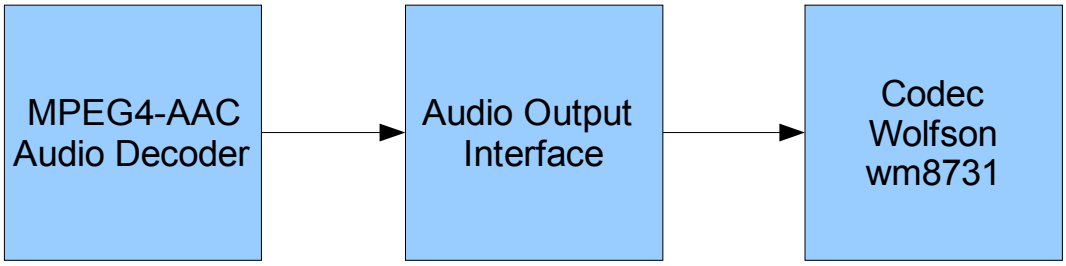


Fig. 5 – Proposed audio output interface

To create such interface was proposed one blocks architecture, allowing individual blocks synthesis and simulation.

Figure 2 shows the established design for the audio output interface of the decoder.

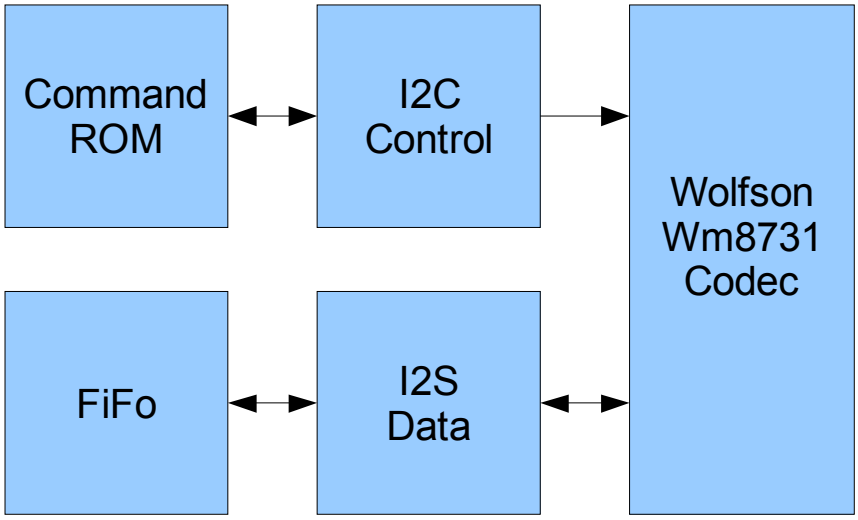


Fig. 2 – Audio Output Interface main blocks

The functions of the blocks in figure 2 are:

- Command ROM: The command ROM block is used to store the commands, to initialize the wm8731 codec and also to manage the sending order.
- I2C Control: The I2C control block has the function to create the standard I2C message, generate the needed clocks and serialize the data within the command ROM.
- I2S Data: The I2S data block is used to manage the data transmission to the wm8731 codec, depending on the sample rate established before by commands. It uses a left/right select signal to choose the left/right channel, when a stereo audio stream is played.
- FIFO: It is a queue that is needed to match the decoder and wm8731 rates. It was developed to let the written data available to read on the very first read clock rising edge, to avoid breaks on playing the audio stream.

Synthesizing the structure described in figure 2 the following resources were used (table 1):

Tab. 1 Ciclone II FPGA used resources

Resource	Used	Total Available
Logic Elements	240	33,216
RAM Block [bits]	32,768	483,840

4. Overlap and Add Block

The Overlap and Add Block (OAB) composes the final step of the audio stream and FIR filter convolution, being responsible, basically, by the sum and storage of the data from IMDCT [1].

At this moment, the OAB is synthesizable and at validation stage. To validate the hardware implementation we get the input and output streams from the C++ code, developed by Universidade de Brasilia (UnB) team, of the OAB structure, and then, using test benches, we generate the same input vector in our hardware OAB design to compare the output streams. Therefore we can see and adjust possible fails on the design.

Figure 3 illustrates the test and validation structure.

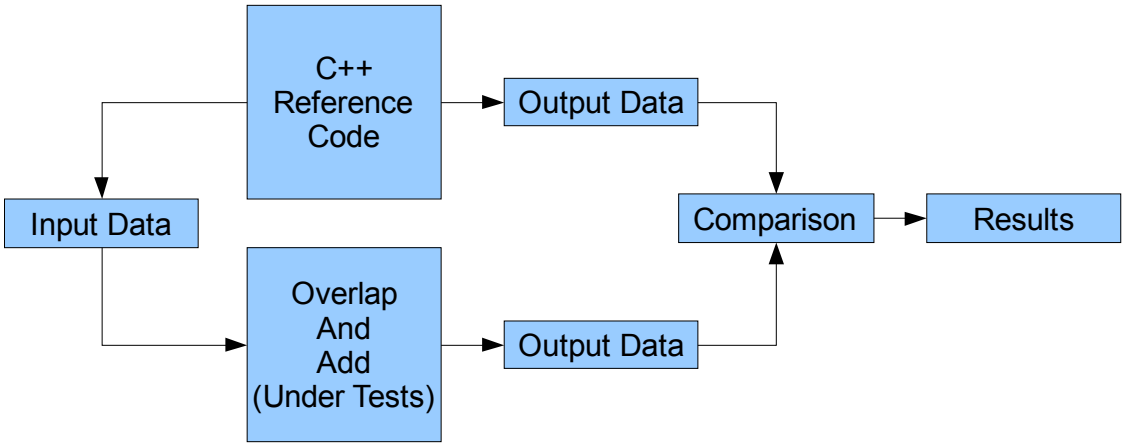


Fig. 3 – Overlap and Add Block validation structure

Although yet in validation stage, the OAB structure and the audio output interface might be incorporated to the rest of the decoder project, allowing an easy way to qualitative measurement of the audio decoder.

Figure 4 shows how it is organized the audio output interface with OAB layout.

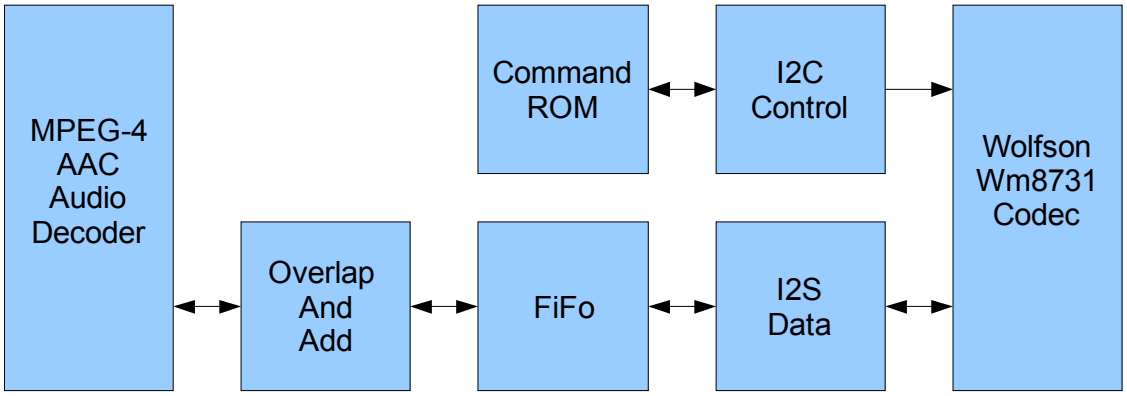


Fig. 4 – OAB and Audio Output interface layout

Until now, the entire structure of the OAB and the audio output interface, built to allow stereo sound, was synthesized using the resources described on table 2.

Tab. 2 – Ciclone II FPGA resources used to the structure

Resource	Used	Total Available
Logic Elements	358	33,216
RAM Block [bits]	65,536	483,840

Both structures, OAB and audio output interface, are totally developed in hardware description language, and may be used in the final project.

5. Conclusion and Future Works

Once integrated to decoder, the audio output interface might be helpful to qualitative measurement of the audio decoder, accomplishing the main goal of this work.

The audio output interface could be used with any codec that communicates using I2C and I2S protocols, in our case the Wolfson wm8731.

The synthesized audio output interface and the OAB use a small amount of the total FPGA resources, around 1 % of the total logic elements and around 13.5 % of the total internal RAM bits, and it is an interesting alternative to the use of a processor dedicated to this function.

Using an own developed HDL code avoids the need of purchase some kind of intellectual property (IP) to do such task.

Properly tested the overlap and add block may be incorporated into the audio decoder, comprising part of the SoC-SBTVD audio decoder.

The next steps to complete this work are the on board validation of the structure and the integration to the audio decoder, after that we will be able to "hear" the decoder.

6. References

[1] ISO/IEC 14496, "Informational Technology – Coding of Audio Visual Objects" part 3: audio, 3rd ed., Dec 2005.

Cell-Based VLSI Implementations of the Add One Carry Select Adder

[1]Jucemar Monteiro, [1]Pedro V. Campos, [1]José Luís Güntzel, [2]Luciano Agostini
{jucemar, pedropaulovc, guntzel}@inf.ufsc.br, agostini@inf.ufpel.edu.br

[1]Embedded Computing Lab. (LCE) - UFSC - Florianópolis, Brazil
[2]Group of Architectures and Integrated Circuits (GACI) - UFPel - Pelotas, Brazil

Abstract

This paper proposes an add-one carry-select adder architecture optimized for cell-based VLSI generation. This architecture is compared to a previously proposed A1CSA architecture, as well as, to Carry-Select Adder (CSA) and Carry-Ripple Adder (CRA) architectures. Synthesis results for 45nm technology showed that, for higher order adders (32 to 256 bits), the proposed A1CSAS architecture is, on average, 13% faster than the investigated "select adders". Power and area estimates showed that, for the same range, the A1CSAS is smaller and consumes less power than the others "select adders". Power-delay results reveals that, for a bit range between 16 and 256, the A1CSAS is the most energy-efficient architecture among all investigated adders.

1. Introduction

Addition is the most commonly used arithmetic operation within contemporary electronic systems. It has great importance not only in general purpose CPUs, but also in acceleration blocks, as part of arithmetic circuits. Besides used as a proper operation, it serves as the basis to many other arithmetic operations.

The choice of which adder architecture to use is of utmost importance, since the performance of adders may determine the whole system performance [1]. Area and power consumption are also relevant figures of merit to be considered, especially when the design targets VLSI realization. Recently, energy-efficiency has also become an important metric due to the growth of battery-powered portable device marked.

Most works addressing adder architectures are focused on critical delay reduction by optimizing the carry propagation chain [1-3]. Such optimization can be accomplished at the logic-level, at transistor-level or at both. Transistor-level optimizations may use pass transistors, dynamic logic, etc. On the other hand, conventional physical design flow relies on standard-cells for layout generation. Although typical standard-cells libraries contain up to hundreds of cells, they are all designed using static CMOS style. The inclusion of user-created cells with other styles, as pass transistor or dynamic logic, is generally not allowed due to limitations of the design flow itself. Therefore, when designing high performance addition-based arithmetic circuits using cell-based VLSI design, designers must rely on logic-level optimized fast adder architectures.

The Add-One Carry-Select Adder (A1CSA) [4-5] has originally been proposed as a low-cost version of the Carry-Select Adder (CSA) [6]. Later, a low power version of the A1CSA was also proposed [7]. However, in order to achieve area/transistor reduction, those A1CSA circuits, and a few others as well, explore pass transistor logic, which prevents their implementation in a standard-cells design flow. Recently, Mesquita et al. proposed a logic-level only A1CSA architecture, targeting FPGA-based design [8].

The main contribution of this paper relies on a new logic-level A1CSA architecture, hereinafter referred to as A1CSAS, specially tailored for standard-cells based design. The A1CSAS was compared to Carry-Ripple Adder (CRA), CSA and Mesquita's A1CSA (for the sake of simplicity, referred to as A1CSA). Synthesis results, obtained with Synopsys Design Compiler (DC) [9], reveal that, for bit-widths ranging from 32 to 256, the A1CSAS is the fastest adder architecture among the investigated ones. They also show that, for bit-widths between 16 and 256, the A1CSAS is the most energy-efficient adder architecture. Finally, the synthesis results pointed out that, for the considered range of bit-widths (8 to 256), the A1CSAS occupies less area and consumes less power than both the A1CSA and the CSA.

The remaining of this paper is organized as follows. Section 2 presents both A1CSA and A1CSAS architectures and gives a brief review on related works, concerning the A1CSA. Synthesis results are presented and discussed in section 3. Section 4 presents the most relevant conclusions of this work.

2. Add One Carry Select Adders (A1CSA) for Cell-Based Design

The Carry-Ripple Adder (CRA) is known as the most area efficient, and at the same time, the slowest adder architecture [2]. It has both time and area complexities O(n) [3]. In the CSA, addition is divided into m modules of k bits each. Each module has two k-bit wide adders (generally, CRAs), which perform two additions at the same time: one with CIN=0 (CIN means carry-in) and another with CIN=1. Such degree of parallelism makes the CSA one of the fastest adder architectures. On the other hand, the CSA area is about as twice as that of the CRA, for a given bit-width. The time and area complexities of the CSA are O(sqrt(n)) and O(2n), respectively.

The philosophy behind the A1CSA relies on optimizing the CSA by replacing the adder with CIN=1 by a less expensive logic, known as "add-one logic". Most works on the A1CSA architecture perform transistor-

level optimization on the "add-one logic" by using pass transistors [4,5,7]. However, the design at the transistor-level prevents the standard-cells based A1CSA realization. On the other hand, the A1CSA architecture proposed by Mesquita et al. was designed at the logic-level only [8], to allow FPGA-based synthesis. In this work we consider an adapted version of Mesquita's A1CSA (referred to as A1CSA, simply) which block diagram is showed in Fig. 1. In this diagram, the box labeled with "A1" represents the "add-one logic". An explicit multiplexer is responsible for choosing the correct result for each module, except for the least significant one.

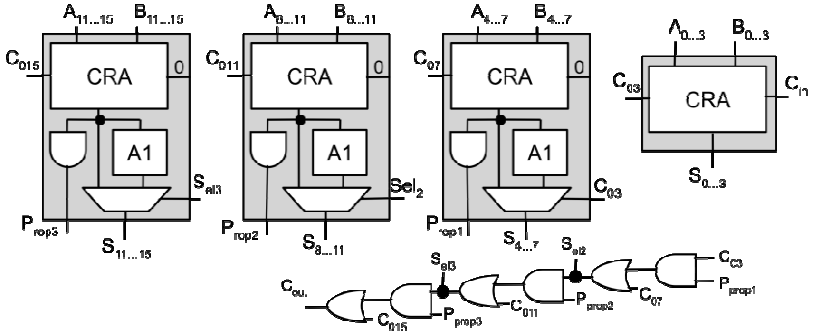


Fig. 1 – 16-bit A1CSA.

The add-one logic can be implemented based on the following properties of the binary addition [5] [8].

Property 1: *In an addition of two n-bit numbers with the least significant bit being "0" for both numbers, if the carry-in bit is changed from one value to another, the LSB of the addition is complemented and the other bits remain unchanged.*

Property 2: *If the addition of two n-bit numbers with a carry-in of "0" has m "1s" before the first occurrence of a "0" (starting from the LSB), then the least significant m+1 bits of the addition with a carry-in of "1" will have values complementary to the first m+1 bits of the addition with carry-in "0".*

Cin	0		Cin	1	
A	1 1 0 0	(12)	A	1 1 0 0	(12)
B	0 1 0 0	(4)	B	0 1 0 0	(4)
Output	1 0 0 0 **0**	(16)	Output	1 0 0 0 **1**	(17)

(a)

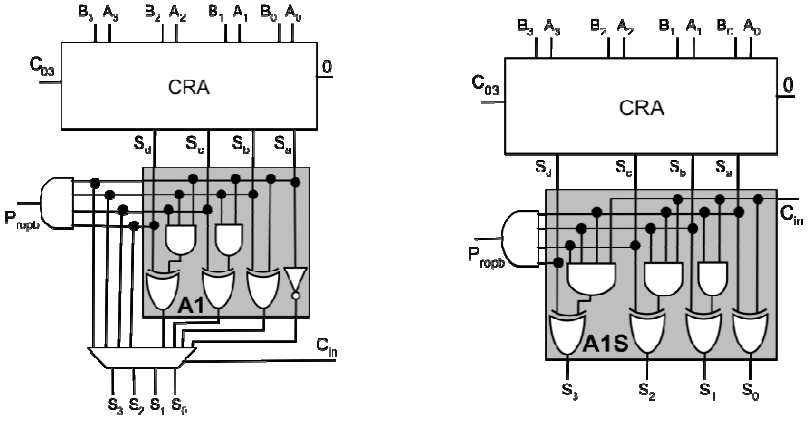
Cin	0		Cin	1	
A	0 0 0 1	(1)	A	0 0 0 1	(1)
B	0 1 1 0	(6)	B	0 1 1 0	(6)
Output	0 **1 1 1**	(7)	Output	**1 0 0 0**	(8)

(b)

Fig. 2 – Example of Property 1 (a) and Property 2 (b).

The properties showed the Fig. 2 were used to derive the circuit for the "add-one logic". Fig. 3(a) shows the designed circuit for the "A1" block in Fig. 1. (i.e., for the "add-one logic" of the A1CSA).

Analyzing the equations of the A1 block, one can notice that it is possible to incorporate the result selection into the A1 existing XOR gates, therefore eliminating the multiplexer. By doing so, the logic depth of an A1CSA module is reduced, resulting in delay reduction. It also contributes to reduce circuit area, and thus, to improve energy-efficiency. The modified "add-one logic", called A1S, is showed in Fig. 3(b). It was used to replace both the A1 block and the multiplexer in the A1CSA of Fig. 1, given rise to a new A1CSA, named A1CSAS.



(a) (b)

Fig. 3 – A1 block (a) and A1S block (b)

3. Results and Discussion

The analyzed adder architectures A1CSAS and A1CSA were evaluated and compared against CRA and CSA, which is fast "carry select" adder architectures that may be adopted in a cell-based design flow. The CRA was also included in the study to serving as reference.

In order to obtain reliable estimates for area, critical delay and power, 8, 16, 32, 64, 128 and 256-bit wide adders were described in Verilog and synthesized for TSMC 45nm standard-cells library using Synopsys Design Compiler (DC) [8] in Topographical mode. The DC mapped the CRAs by using the full adder (FA) cell available from the standard-cells library, thus providing the best possible CRA realizations within the automated design flow.

The fast adder architectures were designed by splitting the total bit width into 4-bit wide modules (i.e., m=4). Each output function of block A1 was described in a separated Verilog module, in order to prevent DC from sharing logic. The same strategy was used to describe block A1S. The CRA adder was also described in a separate Verilog file. The modules were joined in another Verilog module called basic block. Fig. 1 shows the 16-bit wide A1CSA composed by three basic blocks and one 4-bit wide CRA. A1CSAS were described in Verilog in a similar manner. Such Verilog organization allows to restrict logic optimization and mapping to the bounds of each sub-circuit, thus preventing DC from transforming an A1CSA into a CRA.

Tab. 1 shows critical delays (in ps) as estimated by Synopsys DC. As it would be expected, the CRA exhibits the worst delay for all bit widths. On the other hand, for 8 and 16 bit wide adders the CSA exhibits the shortest delays, whereas for 32 to 256 bit wide adders the A1CSAS exhibits the shortest delays. The delay comparison reveals that the A1CSAS is, on average, 3% faster than CSA and it is, on average, 5% faster than A1CSA, respectively. For adder with bit width larger than 64 bits, A1CSAS become faster than both CSA and A1CSA. For 64 bits, the A1CSAS is 15% faster than the CSA and 13 % faster than A1CSA. For 256 bits the A1CSAS is 18% and 17 % faster than the CSA and the A1CSA, respectively.

Tab.1 - Adders critical delay [ps]

	Bit-width					
Adder	**8**	**16**	**32**	**64**	**128**	**256**
A1CSAS	365.9	440.7	608.1	888.5	1550.3	2855.7
A1CSA	315.4	413.0	624.8	1015.1	1854.1	3429.1
CRA	406.4	740.1	1415.5	2681.1	5187.8	10316.4
CSA	293.8	386.9	618.8	1037.0	1860.7	3481.6

Tab. 2 shows area estimates provided by Synopsys DC, in μm^2. As one would expect, the CRA requires the smallest amount of area among the investigated architectures. Among the remaining architectures, referred to as "select adders", A1CSAS is the architecture requiring the smallest amount of area, followed by A1CSA and CSA. The main result of this table is that A1CSAS is, on average, 26% smaller than CSA and 13% smaller than A1CSA, respectively.

Tab.2 - Adders area [μm^2]

	Bit-width					
Adder	**8**	**16**	**32**	**64**	**128**	**256**
A1CSAS	45.9	102.3	215.2	441.0	892.6	1795.8
A1CSA	50.5	116.1	247.3	509.8	1034.8	2084.7
CRA	35.3	70.6	141.1	282.2	564.5	1129.0
CSA	57.0	135.7	293.0	607.7	1237.1	2495.9

Tab. 3 shows the power consumption of each adder, expressed in mW, as estimated by Synopsys DC. The CRA is the adder architecture requiring the least power, followed by A1CSAS and A1CSA. On average, A1CSAS demands 18% and 11% less power than CSA and A1CSA, respectively, for all bit widths.

Tab.3 - Adder power estimates [mW]

	Bit-width					
Adder	**8**	**16**	**32**	**64**	**128**	**256**
A1CSAS	27.0	57.6	118.3	238.4	482.8	976.8
A1CSA	29.3	63.6	133.4	270.6	550.4	1114.0
CRA	23.4	47.2	95.2	190.0	381.0	771.7
CSA	30.7	67.9	144.0	294.4	599.9	1212.1

To establish a comparison in terms of energy efficiency, the power-delay product (PDP) of each adder was calculated. The PDP of an adder can be interpreted as the amount of energy it requires to perform each addition. Tab. 4 shows PDP values, in fJ. A1CSAS presents the lowest PDP values for bit widths ranging from 16 to 256. The A1CSA presents the second lowest PDP values, considering bit widths from 16 to 256. For 8 bit wide adders, the CSA presents the lowest PDP, followed by A1CSA. The CRA has the highest PDP among all

adders from 16 to 256 bits. The comparison of PDP between the "select adders" shows that the PDP of A1CSAS is, on average, 19% lower and 15% lower than that of CSA and A1CSA, respectively.

Tab.4 - PDP of adders [fJ]

	Bit-width					
Adder	**8**	**16**	**32**	**64**	**128**	**256**
A1CSAS	9.87	25.37	71.95	210.59	748.50	2789.39
A1CSA	9.26	26.28	83.36	274.65	1020.48	3820.19
CRA	9.51	34.92	134.69	509.45	1976.72	7960.79
CSA	9.02	26.28	89.08	305.33	1116.17	4220.88

4. Conclusion

This paper presented two version of Add-One Carry-Select Adders suited for cell-based VLSI design flow. Those architectures were synthesized for TSMC 45nm standard-cells library by using Synopsys Design Compiler in Topographical mode. Carry-Ripple Adders (CRA) and conventional Carry-Select Adders (CSA) were also synthesized. Results showed that A1CSAS requires, on average, 13% and 26% less area than A1CSA and CSA, respectively. Such area reduction comes from the use of an add-one logic (replacing one of the two 4-bit CRA modules encountered in conventional CSA) and elimination of the multiplexer present in the original A1CSA.

For adders with bit width ranging from 32 to 256, critical delay estimates showed that A1CSAS is significantly faster than CSA and A1CSA. Considering the bit-widths between 16 and 256, the A1CSAS is the most energy-efficient adder architecture. Finally, for the whole range of considered bit widths, the A1CSAS occupies less area and consumes less power than both the A1CSA and the CSA. Only the CRA consumes less power and requires less area than the A1CSAS. Therefore, the A1CSAS is an excellent high performance and energy-efficient adder alternative for cell-based VLSI design.

5. References

[1] J. M. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits: a design perspective, 2nd ed., Upper Saddle River, N. J.: Prentice Hall, 2003, pp.559-586.

[2] M. D. Ercegovac and T. Lang, Digital Arithmetic, San Francisco: Elsevier Science, 2004.

[3] V. Oklobdzija, "High-speed VLSI arithmetic units: adders and multipliers", in Design of High-Performance Microprocessor Circuits, A. Chandrakasan, W. J. Bowhill and F. Fox, Eds. Piscataway, N. J.: IEEE Press, 2001, pp. 181-204.

[4] T. Y. Chang and M. J. Hsiao, "Carry-select adder using single ripple-carry adder," *Electronics Letters*, vol. 34, no. 22, Oct. 1998, pp. 2101-2103.

[5] Y. Kim and L. S. Kim, "64-bit carry-select adder with reduced area", *Electronics Letters*, vol. 37, no. 10, May 2001, pp. 614-615.

[6] O. J. Bedrij, "Carry-Select Adder", *IRE Transactions on Electronic Computers*, June 1962, pp. 340-346.

[7] Y. He, C.-H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," *Proc. IEEE Intl. Symposium on Circuits and Systems* (ISCAS2005), pp. 4082–4085.

[8] E. Mesquita, et al, "RIC fast adder and its SET-tolerant implementation in FPGAs," *Proc. Intl. Conference on Field Programmable Logic and Applications* (FPL 2007), pp. 638-641.

[9] Synopsys's Design Compiler User Guide, Version C-2009.06, June 2009.

Iterative Mode Hardware Implementation of CORDIC Algorithm

Raphael A. Camponogara Viera, Paulo César C. de Aguirre, Leonardo Londero de Oliveira and João Baptista Martins
{raphaelviera, paulocomassetto, leonardo}@mail.ufsm.br, batista@inf.ufsm.br

Grupo de Microeletrônica - GMICRO
Universidade Federal de Santa Maria - UFSM

Abstract

This paper presents different hardware implementations of CORDIC (Coordinate Rotation Digital Computer) algorithm in iterative mode. CORDIC is a shift-add algorithm for computing a wide range of functions including trigonometric and logarithmic, for instance. We have used a C model of the CORDIC to validate hardware design functionally. The algorithm was described in VHDL, synthesized and tested with a SPARTAN3E XCS500E FPGA. After that, they were synthesized using standard cells. We have achieved the maximum frequency of 420 MHz for the 16 iterations approach using the FPGA and 125 MHz in X-FAB XC06 standard cells synthesis.

1. Introduction

The Digital Signal Processing (DSP) has been dominated by low cost microprocessor-based-systems. While these systems offer much flexibility, they are not fast enough for the current DSP applications because yours software algorithms do not meet the demanding [1]. In order to solve this problem, hardware dedication systems can be used for high speed DSP applications.

Trigonometric functions is one of the mainly tasks performed in DSP applications. Among the existing hardware algorithms for trigonometric solutions the CORDIC (Coordinate Rotation Digital Computer) algorithm is one of the most used. Several works and hardware implementations of CORDIC algorithm are presented and the literature, like [2], [3] and [4]. These alternatives exist for implementing "shift-and-add" algorithms functions in FPGA-based systems like logarithm and exponential functions.

This work focuses on a hardware implementation of the Bit-Parallel iterative mode (rotational) CORDIC algorithm with 8, 16 and 32 iterations. To validate this architecture a CORDIC algorithm software implementation was developed. The hardware architectures were implemented in a Spartan3E XC3S500E FPGA and synthesized to X-FAB XC06 standard cells. A comparative study on the performance of these implementations is presented.

The paper is organized as follows. Next we introduce the basis of CORDIC algorithm. Section 3 describes the implementation of the serial mode CORDIC in hardware and software. Section 4 provides simulation and synthesis results. Finally, we conclude the paper in Section 5.

2. The CORDIC Algorithm

The CORDIC algorithm is an iterative method for computing elementary functions like sine, cosine, and arctangent. It was introduced by Volder [5] [6]. The method can also be easily extended to compute square roots, hyperbolic functions as well polar to rectangular and rectangular to polar conversions [7]. It works by reducing the calculation into a number of micro-rotations for which the arctangent value is precomputed and loaded from a table. This method reduces the computation to addition, subtraction, compares and shifts [6]. Those functions are easily performed by a Field Programmable Gate Array (FPGA).

2.1. Basis of CORDIC

The algorithm is derived from the general rotation transform that gives the equations for the vector (X_{n+1}, Y_{n+1}) in terms of vector (X_n , Y_n) and θ as:

$$\begin{aligned} X_{n+1} &= X_n.\cos(\theta) - Y_n.\sin(\theta) \\ Y_{n+1} &= X_n.\sin(\theta) + Y_n.\cos(\theta) \end{aligned} \qquad (1)$$

In order to simplify the calculations, the CORDIC algorithm now uses a number of rotations, each one can be easily computed to build up the θ angle. We decompose the θ angle as a sum of the angles, so that:

$$\theta = \sum_{n}^{\infty} \alpha_n \qquad (2)$$

and each α_n is chosen as: $$\alpha_n = \pm \tan^{-1}(2^{-n}) \Leftrightarrow \tan(\alpha_n) = 2^n \qquad (3)$$

Rearranging the equations for a single rotation in terms of $\tan(\alpha_n)$:

$$\begin{aligned} X_{n+1} &= X_n.\cos(\alpha_n) \quad Y_n.\sin(\alpha_n) = \cos(\alpha_n).[X_n - Y_n.\tan(\alpha_n)] \\ Y_{n+1} &= X_n.\sin(\alpha_n) + Y_n.\cos(\alpha_n) = \cos(\alpha_n).[X_n.\tan(\alpha_n) + Y_n] \end{aligned} \tag{4}$$

Using $\tan(\alpha_n) = \pm 2^{-n}$ and let $\sigma_n = \pm 1$, the equations for X_{n+1} and Y_{n+1} become:

$$\begin{aligned} X_{n+1} &= \cos(\alpha_n).(X_n - \alpha_n.2^{-n}.X_n) \\ Y_{n+1} &= \cos(\alpha_n).(Y_n + \alpha_n.2^{-n}.Y_n) \end{aligned} \tag{5}$$

If we ignore for the moment the $\cos(\alpha_n)$ terms, then these equations may be implemented with an adder and a shifter. The solution to the term $\cos(\alpha_n)$ is to ignore it! This in effect means that the values computed by the CORDIC algorithm are $1/\cos(\alpha_n)$ greater than they should be at each iteration. The total scaling factor as $K \to n$ is:

$$K = \prod_{k=0}^{n} \frac{1}{\cos(\alpha_n)} = \prod_{k=0}^{n} \sqrt{1 + \sigma_n^2.2^{-n}} \approx 1.6468 \tag{6}$$

The value of K depends on the number of iterations and the accuracy of the number in the look-up table. Note that this scaling factor is constant, independent of the angle. In many applications, the scaling factor can be ignored. The angle of rotation is accumulated by summing $\sigma_n \tan^{-1}(2^{-n})$. The values of $\sigma_n \tan^{-1}(2^{-n})$ are tabulated and then added to an accumulator. CORDIC is normally used in one of two modes: rotation mode and vectoring mode.

Rotation Mode: In this mode the initial vector (X_{in} , Y_{in}) is rotated by an angle θ. The resulting values of x and y provide the trigonometric functions, and follows the sequence below:

1. An initial vector (X_{in} , Y_{in}) is rotated by an angle θ;
2. The angle accumulator is initialized with the value of θ;
3. A sequence of rotations where each angle is of magnitude $\tan^{-1}(2^{-n})$ is applied;
4. At each step the sign of the angle σ_n is chosen to reduce the angle accumulator;
5. So σ_n is simply chosen according to the value of the angle accumulator (Z is the angle accumulator).

The equations for rotation mode are:

$$\begin{aligned} X_{n+1} &= X_n - Y_n.\sigma_n.2^{-n} \\ Y_{n+1} &= Y_n + X_n.\sigma_n.2^{-n} \\ Z_{n+1} &= Z_n - \sigma_n.\tan^{-1}(2^{-n}) \end{aligned} \tag{7}$$

where $\sigma_n = -1$ if $Z_n < 0$, $+1$ otherwise. At the end of the algorithm, the results are:

$$\begin{aligned} X &= K[X_{in}.\cos(Z_0) - Y_{in}.\sin(Z_0)] \\ Y &= K[Y_{in}.\cos(Z_0) + X_{in}.\sin(Z_0)] \\ Z &= 0 \end{aligned} \tag{8}$$

$$K = \prod_{k=0}^{n} \sqrt{1 + 2^{-2n}} \tag{9}$$

These equations provide both the length of the vector and add the tangent to any initial angle placed in angle accumulator. The sine and cosine functions are performed in the rotation mode. In order to compute the sine or cosine of an angle θ, the Z register will be initialized with θ, Y with 0, and the X with 1. The X register corresponds to the scaled cosine value, and the Y register to the sine value. We can determine the unscaled value by dividing the final values by K (1.6468). For example, after 16 iterations, the algorithm will be as (8) and the results of sine and cosine:

$$\sin(30^\circ) = \frac{Y_{16}}{K} = \frac{0.8237}{1.6468} = 0.5001 \approx 0.50 \qquad \cos(30^\circ) = \frac{X_{16}}{K} = \frac{1.4261}{1.6468} = 0.8660 \approx 0.87$$

The method to compute polar to cartesian coordinate transformation is the same to compute sine and cosine. The transformation is defined by:

$$x = r.\cos(\theta) \qquad y = r.\sin(\theta)$$

To perform this transformation is necessary to load again the θ into Z, load X with K and Y with 0.

3. C Modeling and Hardware Implementations

The iterative (in Bit-Parallel mode) CORDIC algorithm was implemented in C language and VHDL. We have used C modeling to validate the hardware architecture and also find errors that could lead to different results between both implementations [8].

3.1. Software Implementation

The software implementation was developed in C language and provides circular, rotational and hyperbolic functions. The same input vectors for the software algorithm were applied to the hardware input architectures in order to evaluate yours functional behaviors.

3.2. Hardware Implementations

The Bit-Parallel CORDIC hardware architecture was described in VDHL and uses a single hardware block to perform the *n* iterations. This block is composed for three adders/subtractors, three registers, two shifters, a ROM and some multiplexers as can be seen in Fig. 1. Then, there is a latency of *n* clock cycles between the input and output data. This implementation is faster than the Bit-serial one because the second has a latency of (*n x n*) between the input and outputs data [3].

Typically, about 10 to 20 iterations are enough to give a good result with a lower error rate. More iterations provide more accuracy but request longer adders and consequently reduce the maximum frequency operation and increasing the logic area cell.

In this work three different implementations were developed. The first one performs 8 iterations, the second, 16 iterations and the last, 32 iterations. An evaluation between these implementations is presented in Section 4.

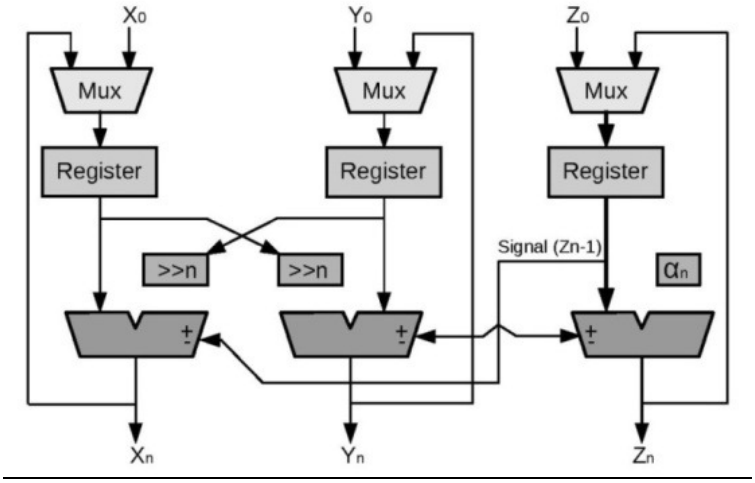


Fig. 1 – Block diagram architecture

4. Simulation and Synthesis Results

The designed modules were synthesized for two different technologies: Spartan3E XCS500E FPGA and X-FAB XC06 5V CMOS Process, using Xilinx ISE 12.1 and Synopsys Design Compiler tools, respectively.

The simulations were carried out using the ISE Simulator and the validation of these one was performed comparing the hardware and software outputs. These results are shown in Tab.1. There is a slightly difference between the hardware and software results because the constant factor (K) was ignored in VHDL implementation.

Tab.1 – Simulation Results

	Entries			Software		Hardware	
	Xin	Yin	Zin	Xout	Yout	Xout	Yout
8 bits	100	5	30	-28.62	162.17	-32.00	149.00
	128	2	60	-37.80	207.39	-35.00	196.00
	288	4	30	-84.24	466.76	-78.00	465.00
16 bits	100	5	30	-36.37	160.83	-33.00	150.00
	128	2	60	-39.41	207.10	-36.00	197.00
	288	4	30	-87.43	466.18	-82.00	460.00
32 bits	100	5	30	-36.37	160.81	-33.00	159.00
	128	2	60	-39.42	207.09	-36.00	198.00
	288	4	30	-87.89	466.09	-85.00	465.00

Tab.2 shows the FPGA synthesis results for the entire developed module considering maximum frequency and resource utilization (number of CLBs, LUTs and IOBs) for three different hardware implementations with 8, 16 and 32 iterations.

Tab.2 – Device Utilization Summary

XC3S500E		**Used**	**Utilization**	**Maximum Frequency**
8 bits	Number of CLB	24	Less than 1%	264 MHz
	Number of 4 input LUTs	9	Less than 1%	
	Number of bonded IOBs	25	10%	
16 bits	Number of CLB	76	Less than 1%	194 MHz
	Number of 4 input LUTs	31	Less than 1%	
	Number of bonded IOBs	49	21%	
32 bits	Number of CLB	100	Less than 1%	177 MHz
	Number of 4 input LUTs	43	Less than 1%	
	Number of bonded IOBs	97	41%	

Tab. 3 presents the number of logic cells and equivalent gates (based in a NAND2) for three different frequency operations for standard cells synthesis. This approach provides a tradeoff between maximum frequency operation and logic cells number.

Tab. 3 – X-FAB XC06 Standard Cells Synthesis Results

	Logic Cells	**NAND2 Equivalent Gates**	**Maximum Frequency**
8 bits	1095	3417	125 MHz
16 bits	10826	21200	125 MHz
32 bits	51725	83808	100 MHz

5. Conclusion

In this paper, we have presented a hardware implementation of the CORDIC algorithm in a Bit-Parallel iterative mode. This approach uses less hardware than a Bit-Serial or pipelined structure [3]. The initial study has compared FPGA and logic synthesis and standard cells results. The standard cells synthesis proves that this design can operate at higher frequencies (more than 100 MHz) even for a 600 nanometer technology. Bit-Parallel iterative mode takes *n* clock cycles, where *n* is the number of algorithm iterations. In this work, were implemented three different architectures, the first one with 8 iterations, the second one with16 iterations and the third one with 32 iterations. It was proved that more iterations you have, more precision is given and more clock cycles are needed to process an input data.

6. References

[1] Andraka, R., "A survey of CORDIC algorithms for FPGA based computers", Andraka Consulting Group, Inc, 1998.

[2] Angarita, F., Perez-Pascual, A., Sansaloni, T., Vails, J., "Efficient FPGA Implementation of CORDIC Algorithm for Circular and Linear Coordinates", International Conference on Field Programmable Logic and Applications. Aug 2005.

[3] Valls, J., Kuhlmann, M., Parhi, K.K., "Efficient mapping of CORDIC algorithms on FPGA", IEEE Workshop on Signal Processing Systems, 2000. SiPS 2000. Oct 2000.

[4] Vadlamani, S., Mahmoud, W., "Comparison of CORDIC algorithm implementations on FPGA families", IEEE Southeastern Symposium on System Theory. Nov 2002.

[5] Volder, J., "The CORDIC Trigonometric Computing Technique", IRE Trans. Electronic Computing, Vol EC-8, pp330-334. Sept 1959.

[6] Bhakthavatchalu, R., Sinith, M.S., Parvathi N., Jismi K.,"A Comparison of Pipelined Parallel and Iterative CORDIC Design on FPGA", 5th International Conference on Industrial and Information Systems. Aug 2010.

[7] Duprat, J. and Muller, J.M., "The CORDIC Algorithm: New Results for Fast VLSI Implementation", IEEE Transactions on Computers, Vol. 42, pp. 168-178. 1993.

[8] Andraka, R. J., "Building a High Performance Bit Serial Processor in an FPGA", On-Chip System Design Conference. 1996.

Test-Chip Structures for Local Random Variability Characterization in CMOS 65 nm

[1]Felipe Correa Werle, [2]Juan Pablo Martinez Brito, [1,2]Sergio Bampi
{ fcwerle, juan, bampi}@inf.ufrgs.br

[1]GME, Microelectronics Group – Informatics Institute
[2]PGMICRO, Graduate Program on Microelectronics
UFRGS, Federal University of Rio Grande do Sul

Abstract

This paper describes the design of a 65nm technology test chip, aimed at investigating and characterizing the truly local random variations. The first structure is a matrix-style transistor array with closely spaced MOS transistors. The second structure comprises three arrays of ring oscillators with different numbers of stages and other two arrays of ring oscillators composed solely by single-type transistors. The third structure is based on a procedure to measure an array of stacked-pairs of identical MOS transistors. The design is done in 65nm CMOS bulk technology and the final chip area is 1580 x 1580 µm.

1. INTRODUCTION

Process variations impose a very important challenge to future nano-scaling of VLSI technology below 32nm. For the past several years, variation in CMOS process has been a concern in the design, manufacture and accurate operation of integrated circuits. Nowadays, intradie variations (further discussion on the differences between interdie and intradie variation are in [1]) are considered as one of the limiting factors for further CMOS scaling [2] and certainly a drawback to the continuing Moore's law [3] scaling. Intradie fluctuations [4] originate mainly from the fluctuation of dopants in the channel region, which is determined by a stochastic energetic ion implantation process [5]. That is, this type of characteristics fluctuation cannot be eliminated in principle.

Mismatch differences among MOSFETs are of utmost concern, since the fluctuation of their characteristics becomes significant as their physical size decreases. Therefore, to model and characterize these variations, highly accurate measurement data on local variability are needed to feed and verify transistor mismatch models [6, 7] with realistic statistical data. Thus, one of the most serious challenges in process variations for sub-100-nm technologies is the effective and reliable way to obtain statistical data from FETs within a reasonable time. To obtain meaningful variation data, special care must be taken concerning the test structure and measurement setup. Due to its statistical random nature, the local random variation effect must be characterized by measuring a large number of individual devices closely placed.

A basic approach to obtain statistical data from a given process is to use arrays of identical transistors [8, 9]. Transistor arrays unavoidably occupy a large area and require sequentially long measurements, reducing the measurement throughput. Nevertheless, the attractiveness of transistor arrays as test structures has led to recent efforts [10] that aim to come up with new ways to create optimized structures for fast and semi-automatic measurement procedures. This structure relies on a multiplexed transistor array with high-density access to multiple devices by means of address decoding and access circuits comprised of CMOS transmission gates. An alternative to the first class of measurements is to convert an analog signal to a more robust measurable quantity. Doing so simplifies the requirements for the test equipment and environment.

Frequency measurements using on-chip circuitry can help with signal-to-noise and bandwidth problems and provide a minimally invasive probing strategy. Ring oscillator-based approaches [11] are effective for test time and are good general-purpose indicators of process variations on digital performance. However, they typically cannot help to predict unique device variation because they tend to indicate the mean parameter strength from its frequency value.

A more recent technique for variability measurement is to use a common gate series-connected MOSFET structure [12-14] suitable for process monitoring purposes. In this structure the mismatch behavior of a large number of MOSFETs pairs is promptly evaluated in a small area using simple voltage measurements. In this paper we propose a test chip designed with new structures for local random variability measurements. Design, simulation and variance simulations using Monte-Carlo were done for a 65 nm CMOS bulk process. The designed test chip has the following structures:

1) A MOSFET matrix-style array composed of multiplexed/biased closely spaced identical MOS transistors.
2) An array of a new type of ring oscillator composed solely by single-type transistor called: NMOS-only [or PMOS-Only] ring oscillator.
3) An array of MOSFET stacked-pairs in which their gates are internally connected.
4) An array of MOSFET stacked-pairs in which the gates are externally connected, providing the possibility to evaluate layout/distance mismatch.

2. A MOSFET MATRIX

The purpose of this structure is to evaluate the transistor mismatch in the traditional way (data analysis of current-voltage curves). Based on [9], Fig. 1 shows the entire diagram of the test structure. The MOSFET matrix includes: bias circuitry, level shifters and address decoding.

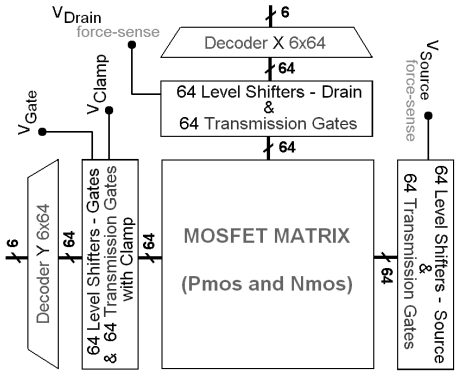


Fig. 1 - Entire MOSFET Matrix Test Structure

The transistors are arranged in individually addressable cells. The matrix contains a total of 1024 devices (512 NMOS and 512 PMOS) placed in 64 columns with 16 rows. Ten groups of ten different size transistors compose the MOSFET matrix. Table 1 summarizes the transistor sizes (60nm is the minimum channel length):

Tab. 1 - Transistor Sizes

#	L [nm]	W [nm]
1	60	6000
2	90	
3	120	
4	240	
5	600	
6	6000	
7	600	180
8		360
9	60	120
10		180

All transistor drains on each column are connected together. All gates and sources on each row are connected together. Figure 2 draws the connection to each Device Under Test (DUT) within the matrix.

Using a row/column address decoder, only one transistor is selected. The other terminals of the non-selected transistors are clamped to respective (N-Fet or P-Fet) clamp voltages into the accumulation mode. The transmission gates connection to each DUT, the force/sense lines, and the DUT selection procedure are shown in Fig. 3.

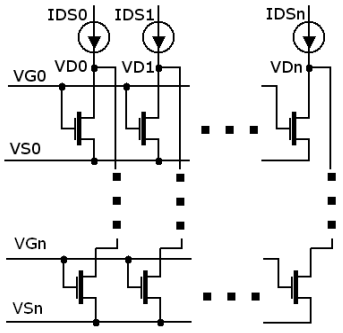


Fig. 2 - Transistor Connections.

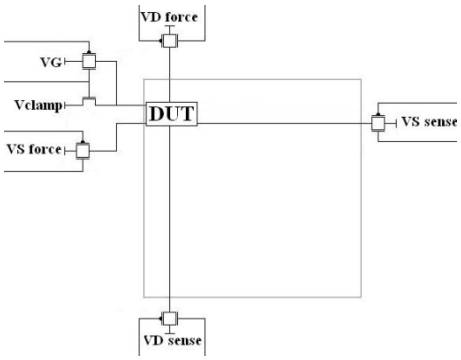


Fig. 3 - DUT Connection

Transmission gates are composed by thick oxide transistors (I/O transistors 2.5V) to drive current of the selected device to the I/O pin. The transmission gate lies in series between the drain and the source terminal of the DUT, causing a difference between the applied pin voltage ('external' voltage) and the 'real' bias voltage applied to the device. To minimize the effects of these IR drops, a ***Kelvin measurement*** technique is used. This technique increases the pin count because a Sense pin is needed. The entire layout of this structure including the PMOS and NMOS MOSFET Matrix, the bias circuitry, level shifters, address decoding and routing is about **330 x 280 µm²** as shown in figure 4:

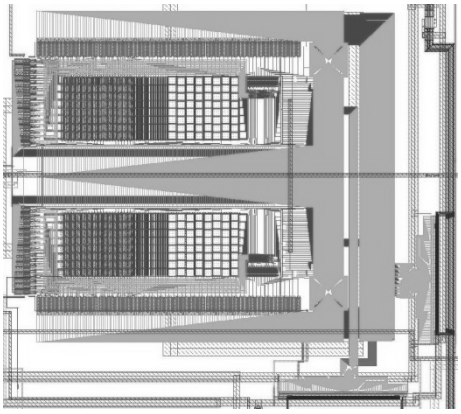


Fig. 4 - MOSFET Matrix Layout

3. RING OSCILATOR

The frequency of a traditional CMOS ring oscillator (RO) is typically sensitive to many process parameters, e.g., transistor gate length and width, threshold voltage, oxide thickness, etc. For equally designed ROs (that is, same number of stages and same W and L for the inverters), the respective difference on the measured output frequency is related to process variations [11]. In our test chip, we designed three different CMOS ring oscillators with 67, 101 and 151 stages that are planned to achieve (with full supply power. 1.2V) 148.4 MHz, 221.5 MHz and 332.9 MHz respectively. To control and avoid interferences in our measurement the last stage of each ring oscillator is controlled by a Tristate inverter, thus only one oscillator oscillates at time. By now one group of oscillator was measured in a temporary test platform. The definitive setup is being planned. The measurement was made in appropriated voltage range to show variations between the oscillators.

Tab. 2 –Voltage and frequency of oscillation.

VDD OSC 67	Mean Freq.	Variation
230	17 kHz	15%
280	6.1 kHz	14.6%
300	100 kHz	14.8%
400	1.2 MHz	13.3%

In our test chip, we designed a novel ring oscillator whose frequency is maximally sensitive to variations in a single type of transistors and minimally sensitive to other variation sources. For this, we introduced the use of NMOS-Only (or PMOS-Only) ring oscillators [15], with the inverter as an enhancement load/enhancement driver configuration. Figure 5 has an example of a three (3) stage NMOS-Only Ring Oscillator:

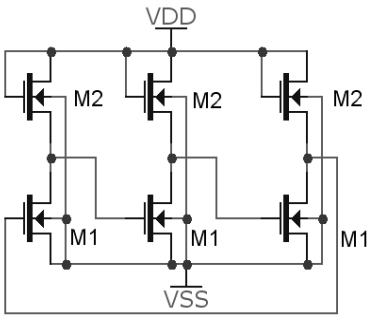


Fig. 5 - Three (3) stages NMOS-Only Ring Oscillator

Usually, the variation in transistor threshold voltage is the strongest contributor to the measured fluctuations in the frequency of ring oscillators. Since this type of ring oscillator contains only NMOS or PMOS transistors, the output frequency variation will be only dependent respectively on NMOS or PMOS threshold voltage variation. In 65 nm technology, this circuit consists of a pull-up device (M2) that must be sized considerably more resistive than the pull-down device (M1). Consequently, for this circuit the sizes shown in Table 3 for M1 and M2 were chosen:

Tab. 3 - transistor size

	M1	M2
$W/L\,[nm]$	W_1=20000	W_2=200
	L_1=100	L_2=10000

The oscillation frequency of this type of RO is typically low (~20MHz), due to the large load and large size transistors. In Fig. 6, an example of the output waveform from the output inverter nodes for an asymmetric supply, e. g. VDD=1V e VSS=-0.2V:

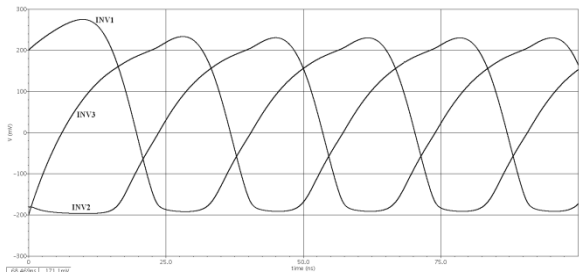


Fig. 6 - Output of a 3 stage RO – extracted simulation

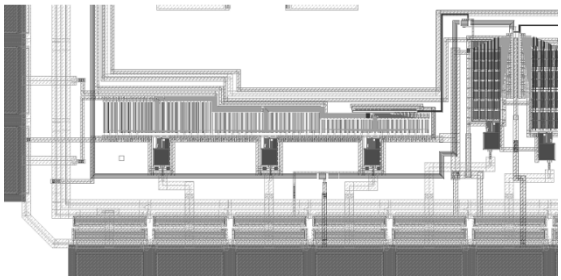


Fig. 7 - Ring Oscillators layout

The entire test structure includes 320 (5 x 64) Ring Oscillators of which: 64 NMOS-Only 3 stage ring oscillators, 64 PMOS-Only 3 stage ring oscillators, 64 CMOS 67-stage ring oscillators, 64 CMOS 101-stage ring oscillators, 64 CMOS 151-stage ring oscillators. Each group of oscillators has its individual supply pin. The layout of this structure including all types of ring oscillators and appropriate buffering measures **200 x 480 µm^2** as shown in Fig. 7.

4. TEST CHIP OVERVIEW

The design size of this test-chip (Fig. 8) is 1.58 mm x 1.58 mm (with pads, without scribe lines), or 1mm x 1mm core size, under 65 nm CMOS design rules. An array of 44 Pads for packaging is needed to access the internal test structures previously described.

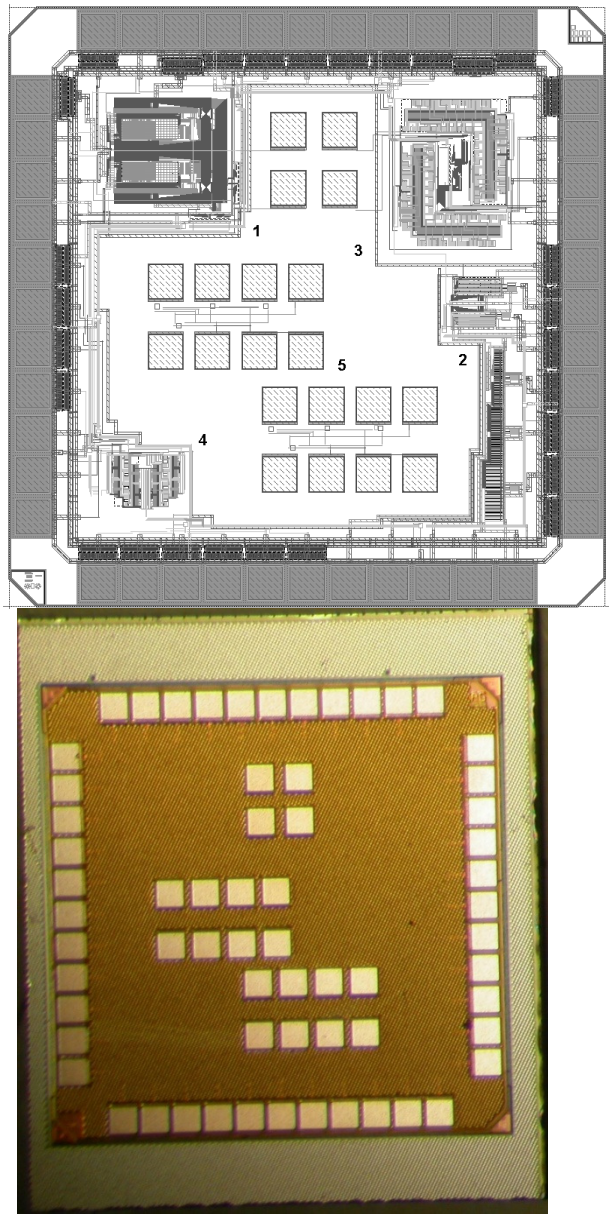


Fig. 8 - Complete layout of the test chip and photography of the chip

Tab. 4 - Shows the area of all blocks included in the test chip, including the pads (which were obtained from the foundry library):

Circuit	Area [µm x µm]
MOSFET Matrix	330 x 280
Ring Oscillator	200 x 480
Stacked-Pair Matrix version 1	390 x 320
Stacked-Pair Matrix version 2	205 x 155
µProbe Test Structures	2 x (930 x 280)

5. CONCLUSIONS

The test structures that we developed for the local random variations characterization are composed of transistor arrays of identical size MOSFETs, ring oscillators and stacked-pair transistors. The simulations were done to verify the sensitivity of each structure to random local variations. Electrical measurements are indispensable for the complete validation of the test structures proposed in this chip. The measurements in our test vehicle are multiplexed for thousands of test transistors and pairs, using only 44 pads. The first measurements show variations between the frequency of ring oscillator with the same number of stages and same power supply. Currently the test platform and setup is being developed to allow an automatic measurement of all structures that are implemented on the chip.

6. REFERENCES

[1] Gary S. May,Costas J. Spanos. Fundamentals of semiconductor manufacturing and process control. John Wiley & Sons, Inc 2006.

[2] Wilson R.; "The dirty little secret: Engineers at design forum vexed by rise in process variations at the die level," EE Times, p. 1, Mar. 25, 2002. Web: http://www.eetimes.com/issue/fp/OEG20020324S0002.

[3] Kuhn, K. et al. "Managing Process Variation in Intel's 45nm CMOS Technology". Intel Technology Journal, [S.l.], 2008.

[4] S. Springer et al., "Modeling of Variation in Submicrometer CMOS ULSI Technologies," IEEE Transactions on Electron Devices, vol. 53, Issue 9, pp. 2168–78, September 2006.

[5] Asenov, A.; Brown, A.R.; Davies, J.H.; Kaya, S.; Slavcheva, G., "Simulation of intrinsic parameter fluctuations in decananometer and nanometer-scale MOSFETs," Electron Devices, IEEE Transactions on , vol.50, no.9, pp. 1837-1852, Sept. 2003.

[6] Galup-Montoro, C.; Schneider, M.C.; Klimach, H.; Arnaud, A., "A compact model of MOSFET mismatch for circuit design," Solid-State Circuits, IEEE Journal of , vol.40, no.8, pp. 1649-1657, Aug. 2005.

[7] Chung-Hsun Lin; Dunga, M.V.; Darsen Lu; Niknejad, A.M.; Chenming Hu, "Statistical Compact Modeling of Variations in Nano MOSFETs," VLSI Technology, Systems and Applications, 2008. VLSI-TSA 2008. International Symposium on , vol., no., pp.165-166, 21-23 April 2008.

[8] Wang, V.; Shepard, K.L., "On-chip transistor characterization arrays for variability analysis," Electronics Letters , vol.43, no.15, pp.806-807, July 19 2007.

[9] Agarwal, K.; Liu, F.; McDowell, C.; Nassif, S.; Nowka, K.; Palmer, M.; Acharyya, D.; Plusquellic, J., "A Test Structure for Characterizing Local Device Mismatches," VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on , vol., no., pp.67-68, 0-0 0.

[10] Agarwal, K.; Hayes, J.; Nassif, S., "Fast Characterization of Threshold Voltage Fluctuation in MOS Devices," Semiconductor Manufacturing, IEEE Transactions on , vol.21, no.4, pp.526-533, Nov. 2008.

[11] Bhushan, M.; Ketchen, M.B.; Polonsky, S.; Gattiker, A., "Ring oscillator based technique for measuring variability statistics," Microelectronic Test Structures, 2006. ICMTS 2006. IEEE International Conference on , vol., no., pp. 87-92, 6-9 March 2006.

[12] Rahul Rao; Jenkins, K.A.; Jae-Joon Kim, "A Completely Digital On-Chip Circuit for Local-Random-Variability Measurement," Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International , vol., no., pp.412-623, 3-7 Feb. 2008.

[13] Wils, N.; Tuinhout, H.P.; Meijer, M., "Characterization of STI Edge Effects on CMOS Variability," Semiconductor Manufacturing, IEEE Transactions on , vol.22, no.1, pp.59-65, Feb. 2009.

[14] Test Circuit for Evaluating Characteristics Mismatch in Metal–Oxide–Semiconductor Field-Effect Transistor Pairs by Estimating Conductance Variation through Voltage Measurement Mamoru Terauchi and Kazuo Terada Jpn. J. Appl. Phys. 47 (2008) 4480.

Gonzalez, Christopher J. (Shelburne, VT, US), Ramadurai, Vinod (Burlington, VT, US), Rohrer, Norman J. (Underhill, VT, US) 2008 Circuit and method to measure threshold voltage distributions in SRAM devices United States International Business Machines Corpo

Author Index