

**UNIVERSIDADE FEDERAL DE PELOTAS**  
**Centro de Desenvolvimento Tecnológico**  
**Programa de Pós-Graduação em Computação**



Dissertação

**Exploring Independent Gates in FinFET-Based Transistor Network Generation**

**Vinicius Neves Possani**

Pelotas, 2015

**Vinicius Neves Possani**

**Exploring Independent Gates in FinFET-Based Transistor Network Generation**

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Leomar Soares da Rosa Junior

Coorientador: Prof. Dr. Felipe de Souza Marques

Pelotas, 2015

Universidade Federal de Pelotas / Sistema de Bibliotecas  
Catalogação na Publicação

P856e Possani, Vinicius Neves

Exploring independent gates in finfet-based transistor network generation / Vinicius Neves Possani ; Leomar Soares da Rosa Junior, orientador ; Felipe de Souza Marques, coorientador. — Pelotas, 2015.

80 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2015.

1. Vlsi design. 2. Tecnologia finfet. 3. Síntese lógica. 4. Ferramentas de CAD. 5. Redes de transistores. I. Rosa Junior, Leomar Soares da, orient. II. Marques, Felipe de Souza, coorient. III. Título.

CDD : 005

Vinicius Neves Possani

Exploring Independent Gates in FinFET-Based Transistor Network Generation

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 29 de janeiro de 2015

Banca examinadora:

.....  
Prof. Dr. Leomar Soares da Rosa Junior (Orientador)

Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul

.....  
Prof. Dr. Felipe de Souza Marques (Co-orientador)

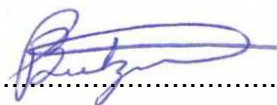
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

.....  
Prof. Dr. Bruno Zatt

Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul

.....  
Prof. Dr. Júlio Carlos Balzano de Mattos

Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul



.....  
Prof. Dr. Paulo Francisco Butzen

Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul

## **ACKNOWLEDGEMENTS**

Firstly, I would like to thank my family, especially my mother Mariane Possani and my father Jonas Possani for the love, affection, education and all the support along of my life. I also thank my sister Taise Possani and my brother in law Regis Ceretta for the incentive and especially for give us the Bruno Possani Ceretta, my nephew. Bruno, thank you for all the happiness that you bring for our family and for all the play, shouts and laughter around me, while I wrote this work.

An especial acknowledgement for my girlfriend Bárbara Cavalheiro for all the love, understanding and incentive while I was away for more two years during the master's degree. I also would like to thank her father, mother, and brother Antonio Cavalheiro, Lenir Cavalheiro, and Artur Cavalheiro, respectively.

This work is result of a long time of research under the orientation of the professors Leomar S. da Rosa Junior and Felipe S. Marques, which are my advisor and co-advisor respectively. Thank you for all the conversations, advices and patience during these years of work. Thank you also for the friendship, barbecues and traveling for conferences.

Since the undergraduate in Computer Science until today, I completed six and a half years studying at UFPEL. Thus, I would like to thank all the people who do this university works, especially my professors and colleagues. I am very happy for the opportunity of work as substitute professor at UFPEL in the last year, this was a very nice experience for me. Indeed, I learn a lot during the lessons together with the students.

***“But my hand was made Strong***

***By the hand of the Almighty***

***We forward in this generation***

***Triumphantly”***

***Redemption Song – Bob Marley***

## RESUMO

POSSANI, Vinicius Neves. **Explorando Gates Independentes na Geração de Redes de Transistores Baseada em FinFET**. 2015. 80f. Dissertação (Mestrado em Ciência da Computação) – Pós-Graduação em Ciência da Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, Pelotas, 2015.

Inicialmente, este trabalho apresenta uma análise, apontando o impacto da tecnologia FinFET na geração de redes de transistores durante a etapa de síntese lógica. Essa análise apresenta diversos estudos de casos para demonstrar que uma mudança de paradigma vem sendo introduzida pelos dispositivos *double-gate*, como os transistores *independent-gate* (IG) FinFET. Além disso, o presente trabalho mostra que essa mudança de paradigma deixa uma lacuna a ser explorada, tendo em vista que os métodos de geração de redes de transistores disponíveis na literatura não são capazes de explorar o potencial que os dispositivos *double-gate* oferecem. Então, neste trabalho são propostos dois métodos alternativos para geração de redes de transistors baseadas em dispositivos IG FinFET. Um dos métodos é baseado em grafos e visa encontrar padrões de arranjos promissores para explorar o potencial dos dispositivos *double-gate*. O segundo método proposto visa realizar defatorações em expressões Booleanas a fim de maximizar o uso dos *gates* independentes de cada transistor IG FinFET. Os experimentos realizados demonstram que os métodos propostos são capazes de gerar redes de transistors IG FinFET otimizadas, com um baixo custo em tempo de execução. Além disso, os resultados obtidos demonstram que de fato os métodos convencionais de geração de redes de transistors não são a melhor alternativa para gerar redes baseadas em dispositivos *double-gate*. Com isso, os resultados reforçam a existência de um novo paradigma introduzido pela tecnologia IG FinFET. Enfim, a análise apresentada neste trabalho dá suporte para o desenvolvimento de novas técnicas de geração de redes de transistors IG FinFET.

**Palavras-chave:** VLSI design; tecnologia FinFET; síntese lógica; EDA; ferramentas de CAD; redes de transistores; portas lógicas; fatoração; teoria de grafos.

## ABSTRACT

POSSANI, Vinicius Neves. **Exploring Independent Gates in FinFET-Based Transistor Network Generation**. 2015. 80f. Dissertation (Master Degree in Computer Science) – Pós-Graduação em Ciência da Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, Pelotas, 2015.

Firstly, this work presents an analysis pointing the impacts of the FinFET technology in the transistor network generation during the logic synthesis step. This analysis presents some case studies demonstrating that a new paradigm has been introduced by the double-gate devices, like the independent-gate (IG) FinFETs. Moreover, this work demonstrates that this new paradigm introduces a lack to be explored. Since the conventional methods for transistor network generation are not able to explore the potential provided by double-gate devices. Thus, this work proposes two alternative methods for IG FinFET-based transistor network generation. The first one is a graph-based method, which aims to find promising patterns to explore the potential provided by the double-gate devices. The second one aims to defactoring Boolean expression in order to maximize the use of the independent gates of each IG FinFET. The experiments have demonstrated that the proposed methods are able to generate optimized IG FinFET transistor networks, with a low cost in run time. Moreover, the obtained results demonstrate that, in fact, the conventional methods of transistor network generation are not the best alternative to design networks based in double-gate devices. This way, the results reinforce the existence of a new paradigm introduced by the IG FinFET technology. Finally, the analysis presented in this work provides support to design new methods to build transistor networks based in IG FinFETs.

**Key-words:** VLSI design; FinFET technology; logic synthesis; EDA; CAD tools; transistor network; logic gates; factorization; graph theory.



## LIST OF FIGURES

Figure 1 -	In (a), SG FinFET transistor and in (b) IG FinFET transistor. ....	15
Figure 2 -	Three different implementations of the Single-Gate (SG) FinFET transistor. Images source: (a) (SYNOPSYS, 2012), (b) (MARKOV, 2014), and (c) Intel (GSS, 2012).....	21
Figure 3 -	Usual Single-Gate (SG) FinFET configuration in (a) and Independent-Gate (IG) FinFET configuration in (b) and (c). Images source: (SYNOPSYS, 2012). ....	22
Figure 4 -	Different FinFET representations: in SG FinFET, IG FinFET parallel ( <i>low-V<sub>th</sub></i> ) and IG FinFET series ( <i>high-V<sub>th</sub></i> ). ....	24
Figure 5 -	Conventional implementation for 2-input NAND (a) and optimized solution obtained by merging parallel transistors in an IG FinFET <i>regular-V<sub>T</sub></i> ( <i>parallel</i> ) (b), proposed in (CHIANG, KIM, <i>et al.</i> , 2005). ....	25
Figure 6 -	Conventional implementation for 2-input NAND (a) and optimized solution obtained by merging series and parallel transistors in IG FinFETs, proposed in (CHIANG, KIM, <i>et al.</i> , 2006). ....	26
Figure 7 -	New implementation for the 6-input NAND gate using only six IG FinFET transistors, proposed in (CHIANG, KIM, <i>et al.</i> , 2006). ....	27
Figure 8 -	Representation of different kinds of FinFETs proposed in (ROSTAMI e MOHANRAM, 2011). ....	29
Figure 9 -	Defactorizations proposed in (ROSTAMI e MOHANRAM, 2011). ....	30
Figure 10 -	Generation of sub-functions until the 5th bucket, proposed in (MARTINS, DA ROSA JUNIOR, <i>et al.</i> , 2010). The number inside braces indicates the bucket composition step order. ....	33
Figure 11 -	Example of network composition proposed in (KAGARIS e HANIOTAKIS, 2007). ....	34

Figure 12 - Kernel structures: (a) NSP and (b) SP proposed in (POSSANI, CALLEGARO, <i>et al.</i> , 2013).....	35
Figure 13 - SP and NSP kernels in (a) and (c), respectively, and correspondent networks in (b) and (d). Final solution (e) obtained by compose networks (b) and (d).....	37
Figure 14 - In (a), transistor network obtained from Equation (2) and in (b), the optimized network obtained from Equation (3). ....	40
Figure 15 - In (a), network obtained from Equation (4) and in (b), optimized network obtained from Equation (5). ....	41
Figure 16 - Two possible solutions to implement the logic function $f$ using SG FinFETs: in (a) SP network and in (b) NSP network.....	42
Figure 17 - Two possible solutions to implement the logic function $f$ using IG FinFETs: in (a) SP network and in (b) NSP network.....	43
Figure 18 - Two possible solutions to implement the logic function $f$ using SG FinFETs: in (a) SP network and in (b) NSP network.....	43
Figure 19 - Two possible solutions to implement the logic function $f$ using IG FinFETs: in (a) SP network and in (b) NSP. ....	44
Figure 20 - In (a) SG FinFET implementation and in (b) IG FinFET implementation, both obtained from the factored form of Equation (12). In (c) IG FinFET implementation obtained by replicating the literal $a$ in the factored form of Equation (12). ....	46
Figure 21 - SP kernel template (a), auxiliary template graph (b) and resulting SG FinFET network (c). ....	49
Figure 22 - SP kernel (a) derived from Equation (14), auxiliary graph template (b), SG FinFET network (c) and IG FinFET network obtained after applying the edge reordering routine. ....	50
Figure 23 - Table obtained from Equation (15) that represents cubes relationship to determine wanted groups that share at least two literals.....	51
Figure 24 - Network delivered by the proposed method obtained from Equation (15).	
51	
Figure 25 - Network delivered by conventional methods obtained from Equation (15). ....	52

Figure 26 - Partial arrangements found from Equation (16) during the two first steps in (a), (b), and (c). Final solution achieved after to remove redundant transistors among the parallel branches in (d).....	53
Figure 27 - First defactorization proposed in (ROSTAMI e MOHANRAM, 2011): conventional arrangement in (a) and defactored arrangement saving one transistor in (b). ....	54
Figure 28 - Conventional arrangement in (a), network obtained by defactoring according (ROSTAMI e MOHANRAM, 2011) in (b), and the proposed merging in (c). ....	55
Figure 29 - Second defactorization proposed in (ROSTAMI e MOHANRAM, 2011): conventional arrangement in (a) and defactored arrangement saving one transistor in (b). ....	56
Figure 30 - Conventional arrangement in (a), network obtained by defactoring according (ROSTAMI e MOHANRAM, 2011) in (b), and the proposed merging in (c). ....	56
Figure 31 - Two adopted patterns to perform series defactorizations.....	57
Figure 32 - Promising arrangement to be defactorized, by replicating the transistor controlled by the literal $a$ .....	58
Figure 33 - Logic tree derived from Equation (16) in (a), tree obtained by merging nodes in (b), and resultant tree after performing the defactorization of literal $a$ in (c).....	60
Figure 34 - Optimized transistor network obtained by defactoring the factored form described in Equation (16).....	60
Figure 35 - Transistors increase or decrease obtained by the proposed method when compared to (MARTINS, DA ROSA JUNIOR, <i>et al.</i> , 2010), for the set of 5-input NPN-class functions. ....	63
Figure 36 - Transistors increase or decrease obtained by the proposed method when compared to (POSSANI, CALLEGARO, <i>et al.</i> , 2013), for the set of 5-input NPN-class functions. ....	64
Figure 37 - Transistors increase or decrease obtained by the proposed method when compared to (GOLUMBIC, MINTZ e ROTICS, 2008), for the set of 6-input Read-Once functions.....	65

## LIST OF TABLES

Table 1 -	Total number of transistors for the set of 5-inputs NPN-class Boolean functions.....	62
Table 2 -	Total number of transistors for the set of 6-inputs read-once functions..	65
Table 3 -	Results for decomposition of circuits into K-cuts, with k=4, k=6 and K=8. 67	
Table 4 -	Improvement obtained by the proposed defactorization method over the results provided by method (MARTINS, DA ROSA JUNIOR, <i>et al.</i> , 2010), considering the set of Boolean functions (1).....	69
Table 5 -	Improvement obtained by the proposed defactorization method over the results provided by method (MARTINS, DA ROSA JUNIOR, <i>et al.</i> , 2010), considering the set of Boolean functions (2).....	69
Table 6 -	Improvement obtained by the proposed defactorization method over the results provided by method (CALLEGARO, MARTINS, <i>et al.</i> , 2013) considering the set of Boolean functions (3).....	69
Table 7 -	Improvement obtained by the proposed defactorization method over the results provided by method (GOLUMBIC, MINTZ e ROTICS, 2008), considering the set of Boolean functions (4).....	70
Table 8 -	Comparative analysis between the proposed methods. ....	71

## LIST OF ABBREVIATIONS AND ACRONYMS

BDD	<i>Binary Decision Diagram</i>
CAD	<i>Computer Aided Design</i>
CMOS	<i>Complementary Metal-Oxide-Semiconductor</i>
DARPA	<i>Defense Advanced Research Projects Agency</i>
DG	<i>Double-Gate</i>
EDA	<i>Electronic Design Automation</i>
FC	<i>Functional Composition</i>
FinFET	<i>Field-Effect Transistor</i>
IC	<i>Integrated Circuit</i>
IG FinFET	<i>Independent-Gate FinFET</i>
ISOP	<i>Irredundant-Sum-Of-Products</i>
ITRS	<i>International Technology Roadmap for Semiconductors</i>
NSP	<i>Non-Series-Parallel</i>
NSP Kernel	<i>Non-Series-Parallel Kernel</i>
RO	<i>Read-Once</i>
RPO	<i>Read-Polarity-Once</i>
SG FinFET	<i>Single-Gate FinFET</i>
SP	<i>Series-Parallel</i>
SP Kernel	<i>Series-Parallel Kernel</i>
SOP	<i>Sum-Of-Products</i>
UTB-SOI	<i>Ultra Thin Body Silicon Over Insulator</i>
VLSI	<i>Very-Large-Scale Integration</i>

## SUMMARY

<b>1</b>	<b>INTRODUCTION .....</b>	<b>15</b>
1.1	Motivation.....	16
1.2	Objectives.....	18
1.3	Organization of this Work .....	19
<b>2</b>	<b>BACKGROUND .....</b>	<b>20</b>
2.1	The FinFET Transistor.....	20
2.2	Innovations and Applications of the FinFET Transistors.....	24
2.2.1	CHIANG et al. (CHIANG, KIM, <i>et al.</i> , 2005) e (CHIANG, KIM, <i>et al.</i> , 2006)...	24
2.2.2	Datta et al. (DATTA, GOEL, <i>et al.</i> , 2007).....	27
2.2.3	Wang (WANG, 2010).....	28
2.2.4	Rostami et al. (ROSTAMI e MOHANRAM, 2011) .....	29
2.3	The State of the Art on Transistor Network Generation .....	31
2.3.1	Mintz et al. (MINTZ e GOLUMBIC, 2005) .....	32
2.3.2	Martins et al. (MARTINS, DA ROSA JUNIOR, et al., 2010).....	33
2.3.3	Kagaris et al. (KAGARIS e HANIOTAKIS, 2007) .....	34
2.3.4	Possani et al. (POSSANI, CALLEGARO, <i>et al.</i> , 2013).....	35
<b>3</b>	<b>INVESTIGATING AND IDENTIFYING THE NEW CHALLENGES ESTABLISHED BY INDEPENDENT-GATE FINFET TRANSISTORS.....</b>	<b>39</b>
<b>4</b>	<b>PROPOSED METHODS FOR INDEPENDENT-GATE FINFET TRANSISTOR NETWORK DESIGN .....</b>	<b>47</b>
4.1	Generating Transistor Networks by Graph-Based Optimizations .....	47
4.1.1	Series-Parallel Kernel Finder .....	48
4.1.2	IG FinFET Dedicated Factorization.....	50

4.1.3	Network Composition.....	52
<b>4.2</b>	<b>Generating Transistor Networks by Defactoring Boolean Expressions ....</b>	<b>54</b>
4.2.1	First Observation on Defactorization.....	54
4.2.2	Second Observation on Defactorization.....	55
4.2.3	Third Observation on Defactorization.....	57
4.2.4	Integrating the Three Observations in a Single Method .....	58
<b>5</b>	<b>EXPERIMENTAL RESULTS .....</b>	<b>61</b>
5.1.1	Results for the Graph-based Approach.....	62
5.1.2	Results for the Defactorization Approach.....	68
5.1.3	Comparing the Proposed Approaches .....	71
<b>6</b>	<b>CONCLUSIONS AND FUTURE WORKS .....</b>	<b>73</b>
	<b>REFERENCES.....</b>	<b>75</b>
	<b>APPENDIX A - Publications .....</b>	<b>79</b>

## 1 INTRODUCTION

Although the continuous CMOS transistor scaling has been providing performance improvements of digital circuits, its technology process faces great challenges due to the devices shrinking and the fundamental materials limits (FRANK, DENNARD, *et al.*, 2001). Since 2001, the International Technology Roadmap for Semiconductors (ITRS) has pointed the FinFET (HUANG, LEE, *et al.*, 1999) and the UTB-SOI (CHOI, JEON, *et al.*, 2000) as promising technologies for transistor scaling beyond CMOS limits. In the last years, different solutions of double-gate (DG) devices were proposed (CHANG, CHOI, *et al.*, 2003) (NOWAK, ALLER, *et al.*, 2004) (ROY, MAHMOODI, *et al.*, 2005).

In this work we have a special interest for DG FinFET devices, which can be designed in two different ways according to the gate configuration. One possibility is to binding the two gates yielding a single-gate (SG) FinFET, as illustrated in Fig. 1(a). This configuration is also known as shorted-gate FinFET. The other one is to build an independent-gate (IG) FinFET, in such a way that each gate can be controlled by a different input signal, as illustrated in Fig. 1(b). These different configurations have allowed a rich project space and introduced new challenges to be explored both in logic and physical synthesis.



Figure 1 - In (a), SG FinFET transistor and in (b) IG FinFET transistor.



Based on the IG FinFET structure shown in Fig. 1(b), some researchers have observed the possibility of exploring the separated gates to merge two transistors associated in parallel, *i. e.*,  $(a + b)$ , in a single IG FinFET (CHIANG, KIM, *et al.*, 2005) (DATTA, GOEL, *et al.*, 2007) (MUTTREJA, AGARWAL e JHA, 2007). In this case, the threshold voltage must be low enough to turn on the transistor ( $I_{on}$ ) when at least one gate is active. This kind of transistor is called IG FinFET *low-V<sub>th</sub> (parallel)*.

Furthermore, some authors introduced the possibility of merging two transistors associated in series, *i. e.*,  $(a . b)$ , in an IG FinFET. It is also done based on the threshold voltage, which must be high enough to turn on the transistor ( $I_{on}$ ) if and only if both gates are active (CHIANG, KIM, *et al.*, 2006) (WANG, 2010) (ROSTAMI e MOHANRAM, 2011). Otherwise, the current is low enough to keep the transistor in the off state ( $I_{off}$ ). This kind of transistor is called IG FinFET *high-V<sub>th</sub> (series)*.

It becomes possible to design transistor networks by using only conventional single-gate SG FinFETs or combining both SG FinFETs and IG FinFETs (*series* and *parallel*). These three variations of FinFETs have enhanced the project space during the synthesis of transistor networks. This way, the merging of series and parallel transistors in IG FinFETs becomes a powerful strategy to decrease the transistor count in logic gates (CHIANG, KIM, *et al.*, 2005) (DATTA, GOEL, *et al.*, 2007) (MUTTREJA, AGARWAL e JHA, 2007) (CHIANG, KIM, *et al.*, 2006) (WANG, 2010) (ROSTAMI e MOHANRAM, 2011).

## 1.1 Motivation

Considering the new possibilities provided by the IG FinFET devices, several works available in the literature are concerned to use such possibilities to design basic gates and other few different Boolean functions. Recently, some works have presented novel solutions to implement elementary gates as AND, NAND, OR, and NOR, by merging series and parallel transistors in IG FinFETs (CHIANG, KIM, *et al.*, 2005) (DATTA, GOEL, *et al.*, 2007) (MUTTREJA, AGARWAL e JHA, 2007) (CHIANG, KIM, *et al.*, 2006) (ROSTAMI e MOHANRAM, 2011). Some of these authors have also synthesized benchmark circuits using such novel gates as cell library. Their experiments have demonstrated that decreasing the transistor count through IG FinFETs is an efficient way to reduce area and power in digital design (MUTTREJA, AGARWAL e JHA, 2007) (ROSTAMI e MOHANRAM, 2011).

In the literature there are many different methods to decrease the transistor count and to help the development of optimized logic gates (SENTOVICH, 1992) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO, MARTINS, *et al.*, 2013) (KAGARIS e HANIOTAKIS, 2007) (POSSANI, CALLEGARO, *et al.*, 2013). Such methods are able to deliver satisfactory solutions for single-gate devices, *i. e.*, *conventional CMOS or SG FinFET*.

In general, these methods aim to minimize the number of literals in a Boolean expression by applying Boolean and algebraic factorization (SENTOVICH, 1992) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010), graph-based factorizations (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO, MARTINS, *et al.*, 2013), and also more flexible graph-based optimizations (KAGARIS e HANIOTAKIS, 2007) (POSSANI, CALLEGARO, *et al.*, 2013). Traditionally, minimize the number of literals leads to a reduced transistor network. It occurs because there is a direct relation between literals and transistors. In other words, each literal in a factored expression can be directly mapped to a transistor in a network. This relation also happens when the literals represented by edges in a graph-based solution are mapped to transistors in a network.

However, a new paradigm has been introduced by merging series and parallel transistors in IG FinFETs, as mentioned before. It impacts in how to generate efficient transistor arrangements during the logic synthesis. Thus, when the goal is to decrease the transistor count by using IG FinFETs, just to minimize the number of literals in a given Boolean expression may not drive to the best solution. We have observed two main reasons for that:

- (1) Just consider the literal count as metric during the optimization process is not enough, we also need to adopt structural characteristics as criterion;
- (2) Some redundant literals, in a factored expression or in a graph-based solution, can contribute to find the best merging of series and parallel transistors in IG FinFETs.

In order to achieve efficient transistor arrangements, these two factors must be considered during the transistor network generation. Therefore, as both the conventional factorization methods and the graph-based techniques aim to decrease the maximum number of literals, these methods (SENTOVICH, 1992) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO,

MARTINS, *et al.*, 2013) (KAGARIS e HANIOTAKIS, 2007) (POSSANI, CALLEGARO, *et al.*, 2013) may not be the best alternative to generate IG FinFET transistor networks.

Considering that the merging of series and parallel transistors in IG FinFET devices has become an alternative way to reduce area and power dissipation in VLSI design (ROSTAMI e MOHANRAM, 2011), it is important to determine an automated method to find efficient IG FinFET transistor networks. We have observed that the co-related works (CHIANG, KIM, *et al.*, 2005) (DATTA, GOEL, *et al.*, 2007) (MUTTREJA, AGARWAL e JHA, 2007) (CHIANG, KIM, *et al.*, 2006) (WANG, 2010) (ROSTAMI e MOHANRAM, 2011) are concerned to demonstrate that the new possibilities provided by the IG FinFET devices are feasible for a real design.

In general, such works have investigated materials, electrical effects and geometric characteristics in order to solve problems for the physical synthesis step. Indeed, no one of them presents an automated method to find efficient transistor networks for IG FinFETs, during the logic synthesis step. Thus, the main motivation of this work consists in exploring the lack introduced by the IG FinFET technology in the logic synthesis level.

## 1.2 Objectives

The first objective of this work is to present a detailed analysis discussing that there is a new paradigm being introduced by the double-gate FinFET devices, during the transistor networks generation. This analysis will be performed through a set of case studies, demonstrating that the conventional methods of transistor networks generation may not be useful for an IG FinFET-based design. Thus, after characterizing the problem, the second objective of this work is to propose two alternative methods able to explore the potential of the IG FinFETs during the synthesis of transistor networks. One method tries to explore the double-gate devices by finding promising patterns that allow series and parallel merging of transistors in IG FinFETs. The other one is a defactorization method, which receives a factored expression as input and replicates some literals of the expression, in order to enable series and parallel merging in IG FinFETs. We also aim to reinforce the existence of a new paradigm by comparing the results of the proposed methods with the results provided by conventional ones.

As this is one of the first works that investigate the impact of double-gate FinFETs in the logic synthesis level, our overall objective is to provide support for further investigations related to double-gate devices in such level. In this sense, the main contributions of this work are:

- ✓ A set of case studies discussing the new paradigm to implement Boolean functions using IG FinFETs.
- ✓ A graph-based method for automatic generation of optimized IG FinFET transistor networks.
- ✓ An alternative method for defactoring Boolean expressions and improve the factored forms, generated by conventional methods, for an IG FinFET-based design.

### **1.3 Organization of this Work**

The remaining of this work is organized as follows. The Chapter 2 presents a brief review of all material that provided support to design this work. Thus, the Chapter 2 presents the origin and the characteristics of the FinFET transistor and also presents the main related works regarding FinFET technology and transistor network generation. The Chapter 3 presents the case studies that pointing for a new paradigm involving Boolean algebra, switch theory and the double-gate devices. In the sequence, the Chapter 4 presents the proposed graph-based method and the proposed defactorization method for IG FinFET transistor network design. The experiments performed in this work and the obtained results are presented in the Chapter 5. Finally, the Chapter 6 presents our conclusions, and also points some future works.

## 2 BACKGROUND

This chapter presents a brief review of related works, which were used as basis to develop this work. The section 2.1 presents some basic characteristics of FinFET transistors in order to provide a better understanding of this work. In the sequence, some papers that propose new approaches based on IG FinFETs are briefly discussed in the section 2.2. Finally, the most recent works related to transistor networks generation are discussed in the section 2.3. These three sections give us support to understand what are the new possibilities provided by the FinFET devices and how the most recent transistor network generation methods work.

### 2.1 The FinFET Transistor

The FinFET transistor arises in the end of 90's years, when the *Defense Advanced Research Projects Agency* (DARPA) announces a demand for sub 25nm technologies. In such days, the microelectronics industry was designing chips with technology of 250nm. Thus, Chenming Hu, professor and researcher from University of California, Berkeley, proposes a promising transistor structure feasible for sub 25nm design. In 1997 Chenming Hu and their research group receive support from DARPA to design and demonstrate the FinFET transistor (HUANG, LEE, *et al.*, 1999) (SPECTRUM, 2011).

Nowadays, it is possible to say that there is a race among the foundries to find the most accurate structure to implement FinFET transistors. Fig. 2 presents three different implementations of the FinFET transistor, among others. Although there are different ways to implement a FinFET, the main characteristics of this new device is the 3D construction of the *gate*, *source*, and *drain*, as we can see in all versions presented in Fig 2.

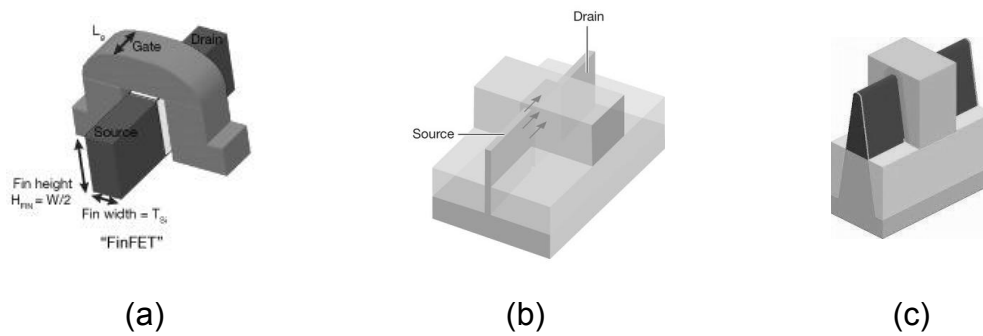


Figure 2 - Three different implementations of the Single-Gate (SG) FinFET transistor. Images source: (a) (SYNOPTIS, 2012), (b) (MARKOV, 2014), and (c) Intel (GSS, 2012).

As we can see in Fig. 2, the channel of the FinFET transistor is rounded by the gate in such a way that there is a contact of three faces of the channel with three sides of the gate. According to the experts, this approach provides greater control of the transistor channel, bringing the following characteristics:

- ✓ The gate has more control over the current flow in the transistor channel;
- ✓ The voltage in silicon substrate cannot impact in the current when the transistor is in the  $I_{off}$  state. In other words, the FinFET has lower static power dissipation;
- ✓ Due to the large inversion area of the transistor, the current flow is high when the transistor is in the  $I_{on}$  state;
- ✓ The FinFET structure does not impact in the density of transistors into the chip;
- ✓ The number of fins can vary according to the demand for performance.

The first two characteristics presented in the list above are related to low leakage. The last three characteristics are interesting because they allow greater performance with low energy consumption. Another issue that contributes for the integrated circuits (IC) design based on FinFET devices is the great similarity of such technology with the conventional planar CMOS fabrication process (HUANG, LEE, *et al.*, 1999).

As briefly presented in the introduction of this work, a FinFET transistor can be designed of different ways according the gate configuration. The Fig. 2 presents three different implementations for the usual configuration of the gate for a FinFET transistor. Notice that, in the usual configuration there is a single gate around the channel of the transistor. However, by removing the top of the gate, it is possible to produce two independent gates, as shown in Fig. 3(b) and Fig. 3(c) (LIU, MATSUKAWA, *et al.*, 2007). These different configurations can be explored according to the design constraints and the objectives of the project (WANG, 2010) (ROSTAMI e MOHANRAM, 2011) (MISHRA, MUTTREJA e JHA, 2011) (CAKICI, MAHMOODI, *et al.*, 2005).

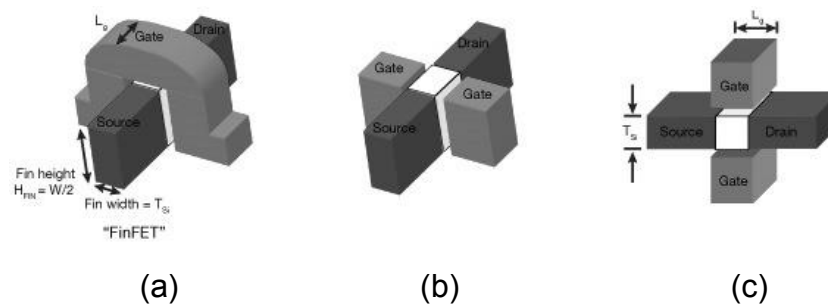


Figure 3 - Usual Single-Gate (SG) FinFET configuration in (a) and Independent-Gate (IG) FinFET configuration in (b) and (c). Images source: (SYNOPTIS, 2012).

Among the applications to explore the separated gates of an IG FinFET, the most common are:

- ✓ Decrease the transistor count by merging series and parallel transistors in IG FinFETs (CHIANG, KIM, *et al.*, 2006) (DATTA, GOEL, *et al.*, 2007) (MUTTREJA, AGARWAL e JHA, 2007) (ROSTAMI e MOHANRAM, 2011);
- ✓ Decrease the logic gate capacitances and power dissipation by disable one of the independent gates in each transistor (DATTA, GOEL, *et al.*, 2007) (MUTTREJA, AGARWAL e JHA, 2007) (ROSTAMI e MOHANRAM, 2011);

- ✓ Design optimized memory cells by exploring the independent gates (LIU, TAWFIK e KURSUN, 2008) (TAWFIK e KURSUN, 2008) (AMAT, ALMUDEVER, *et al.*, 2013).

Although the merging of pairs of transistors in IG FinFETs come with a slight deterioration in gate delay, it is shown that reducing the number of stacked devices by series merging is a good strategy to mitigate the loss in performance. Another point is that the merging of series and parallel transistors tends to increase the complexity and the wire length to perform intra-cell routing. However, this extra cost for routing the independent gates may be covered by the area reduction obtained by merging pairs of transistors in IG FinFETs.

Indeed, the FinFET technology brings a renewal for the microelectronics industry, allowing that the Moore's law remains in the next decades (MOORE, 1965). In the last two years the main companies of the microelectronics field have started to design ICs based on FinFETs. However, as mentioned before, there are some new possibilities to explore the separated gates of each IG FinFET, resulting in more challenges during the design.

The advances of the FinFET technology also bring challenges to the *Electronic Design Automation* (EDA) and *Computer-Aided Design* (CAD) tools. At the same time that the foundries have gradually adapted the fabrication flow to produce FinFET-base chips, the EDA industry also needs to adapt and design new feature to ensure that the tools are able to represent and model the complexities imposed by the FinFET technology (SYNOPSYS, 2012). In general, the main problems related to the FinFET design are in the physical synthesis step and in the fabrication process.

Although the impact of the FinFETs in the logic synthesis is small, such impact exists and demands for an efficient solution. As the IC design flow is composed of several steps, and the quality of the early steps can impact the following ones, it is important to perform optimizations in all phases of the project to attend the design constraints. Thus, in this work we will expose and propose alternative solutions for problems tied to the logic synthesis, more specifically during the transistor network generation. The problems and challenges tied to the physical synthesis are beyond the scope of this works.



In the remaining of this work the SG FinFET, the IG FinFET (*parallel*) and the IG FinFET (*series*) are represented according the notations illustrated in Fig. 4, respectively.

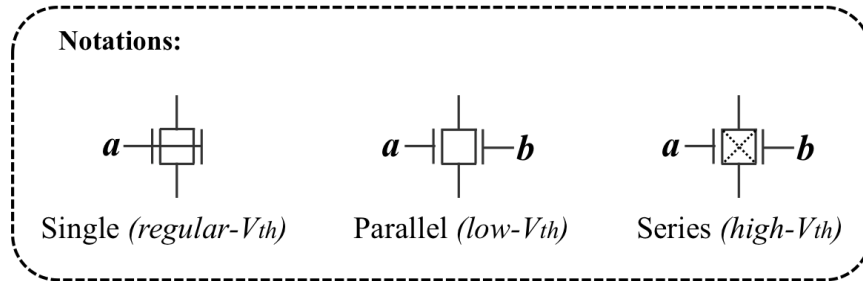


Figure 4 - Different FinFET representations: in SG FinFET, IG FinFET parallel (*low-V<sub>th</sub>*) and IG FinFET series (*high-V<sub>th</sub>*).

## 2.2 Innovations and Applications of the FinFET Transistors

This section presents some works that propose new approaches to implement logic gates exploring the IG FinFET transistors. In general, such works have demonstrated, through electrical simulations, the feasibility of using separated gates for different goals during the transistor network generation. These authors have presented results showing that the IG FinFETs allow the implementation of optimized logic gates, pointing reductions in different aspects as area, power dissipation, input capacitance and delay.

### 2.2.1 CHIANG et al. (CHIANG, KIM, *et al.*, 2005) e (CHIANG, KIM, *et al.*, 2006)

In 2005, Chiang et al. presented an IG FinFET-based implementation for the 2-input NAND and the 2-input NOR logic gates (CHIANG, KIM, *et al.*, 2005). For instance, as we can see in Fig. 5(a), a conventional implementation of the 2-input NAND has a parallel association of transistors in the *pull-up* plane. Thus, as an IG FinFET has two independent gates, the authors observed the possibility of merging two different input signals associated in parallel in a single IG FinFET, resulting in the networks illustrated in Fig. 5(b). The authors called this kind of transistor as IG FinFET *regular-V<sub>T</sub>* (*parallel*), since the parallel merging does not require changes in the threshold of the transistor. Thus, it means that the transistor will propagate the signal when at least one of the gates is active. Similarly to NAND, the optimized NOR gate is obtained by merging the parallel transistor of the *pull-down* plane.

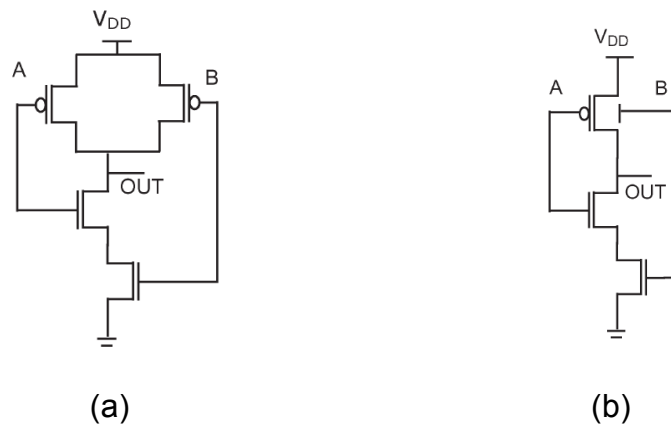


Figure 5 - Conventional implementation for 2-input NAND (a) and optimized solution obtained by merging parallel transistors in an IG FinFET *regular- $V_T$  (parallel)* (b), proposed in (CHIANG, KIM, *et al.*, 2005).

When comparing the optimized NAND gate illustrated in Fig. 5(b) with the conventional implementation it is possible to achieve 40% and 33% of reduction in static and dynamic power, respectively. Moreover, the authors report an improvement of 10% in performance for the optimized NAND gate. Another interesting issue is that the merging of transistors in IG FinFETs brings reductions in the input capacitances, resulting in a reduction of the total capacitance of the logic gate. For the proposed NOR gate, the authors report a reduction of 20% and 17% in the static and dynamic power, respectively. However, the performance is 2% worst than the conventional NOR gate. Although there is a small increasing in the gate delay, the proposed NOR implementation still being competitive by presenting significant reductions in area, power, and capacitance.

Afterwards, in 2006, Chiang *et al.* presented a new approach that allows the merging of two series transistors in a single IG FinFETs (CHIANG, KIM, *et al.*, 2006). The series merging is feasible by increasing the threshold of the transistors, resulting in a new kind of device called IG FinFET *high- $V_T$  (series)*. Thus, this kind of transistor will propagate the signal if and only if both gates are active. The possibility of merging series and parallel transistors in IG FinFETs contributes significantly to decrease the number of devices needed to implement a logic gate. In this sense, it is possible to improve the gains previously presented in (CHIANG, KIM, *et al.*, 2005).

The authors propose a new implementation for the 2-input NAND gate, which can be designed using only two transistors, one N-FinFET *high- $V_T$  (series)* and one

P-FinFET *regular- $V_T$  (parallel)*, as illustrated in Fig. 6(b). This new implementation has 2 times less area and capacitance than the conventional implementation, with an improvement of 19% in performance. The 2-input NOR also can be implemented using only two IG FinFETS.

Another interesting point demonstrated in (CHIANG, KIM, *et al.*, 2006) is that the proposed approaches for series and parallel merging of transistors allow the implementation of logic gates with larger number of inputs than the conventional CMOS implementation. Nowadays, the number of transistor associated in series is limited in order to achieve a practical performance. Thus, a conventional 6-input NAND implementation is not practical for the CMOS technology with a satisfactory performance. It is due to the six transistors associated in series in the *pull-down* plane of the 6-input NAND gate. However, by merging series and parallel transistors in IG FinFETs it is possible to reduce the transistor stack and achieve a practical implementation for the 6-input NAND gate, as illustrated in Fig. 7 (CHIANG, KIM, *et al.*, 2006).

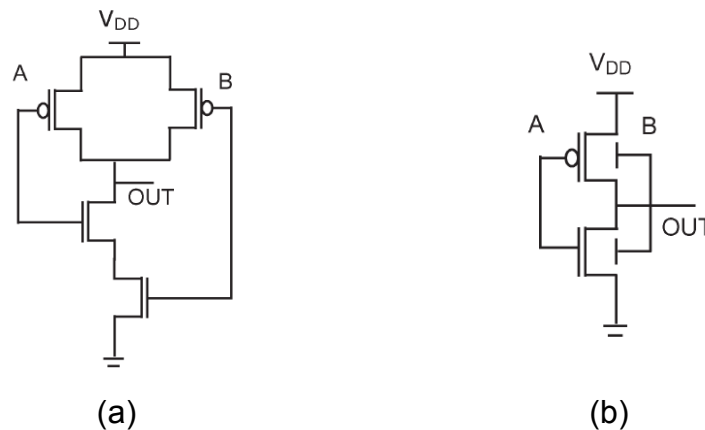


Figure 6 - Conventional implementation for 2-input NAND (a) and optimized solution obtained by merging series and parallel transistors in IG FinFETs, proposed in (CHIANG, KIM, *et al.*, 2006).

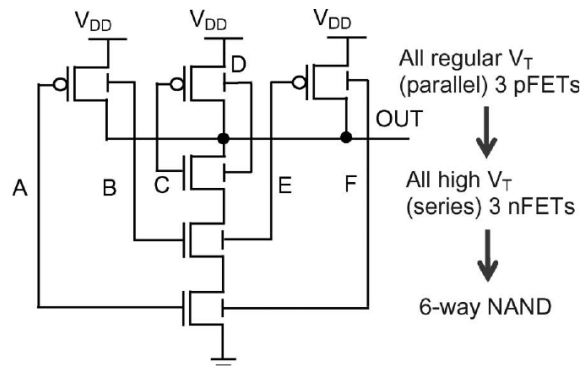


Figure 7 - New implementation for the 6-input NAND gate using only six IG FinFET transistors, proposed in (CHIANG, KIM, *et al.*, 2006).

### 2.2.2 Datta et al. (DATTA, GOEL, *et al.*, 2007)

The authors present different low power implementations for basic logic gates like 2-input NAND, 2-input NOR and INVERTER, where the independent gates of each IG FinFET are explored with different purposes (DATTA, GOEL, *et al.*, 2007). Some of these implementations use one gate as input and the other one is disabled (connected to the ground or supply rail), to reduce the switching capacitances. Other implementations use the independent gates for merging parallel transistors and decrease the transistor count, as early proposed in (CHIANG, KIM, *et al.*, 2005). However, the authors do not mention the possibility of merging series transistors in IG FinFETs. Such work also proposes a semi-analytical model to estimate power and performance of the FinFET-based circuits (DATTA, GOEL, *et al.*, 2007). A set of circuits from ISCAS85 benchmark was synthesized using two cell libraries composed of few elements, which were proposed by the authors.

**First Library:** The first library contains just three logic cells: 2-input NAND, 2-input NOR, and INVERTER. These cells were designed using single-gate SG FinFETs and each cell has different versions of size.

**Second Library:** The second library contains all logic cells from the first library and more five low power cells designed with IG FinFETs, which are presented in the paper.

The cells of these libraries were characterized and the circuits were synthesized through commercial EDA tools. This experiment demonstrated a significant reduction in power and area of the ISCAS85 circuits. The results obtained by using the second library, based on IG FinFETs, present an average reduction of

18% and 8.5% in power dissipation and design area, respectively, when compared to the first library results.

### 2.2.3 Wang (WANG, 2010)

Wang presents a methodology to explore the separated gates of the IG FinFETs (WANG, 2010). Basically, the paper presents electrical simulations demonstrating the possibility of merging series and parallel transistors in IG FinFETs, using different thresholds for series and parallel merging. The author reports that the increase in the threshold of an IG FinFET, to allow the series merging, degrades the transistor performance. Moreover, the transistor cannot work in a voltage lower than 1.0 V to enable the series merging. However, the author demonstrates that these problems do not prohibit the use of IG FinFET (*series*).

The paper presents an algorithm to explore the possibility of generating optimized transistor arrangements by merging series and parallel transistor in IG FinFETs. The algorithm starts from the set of minterms from a given logic function and generates all combinations of the minterms, producing a new set called by the author as *Power Set* (PS). In the sequence, the groups of PS that can be implemented in SG FinFETs or in IG FinFETs are separated in another set called SPS. The next step is to perform intersection and union operations among the elements from SPS. Finally, the results produced by such operations are mapped to series and parallel association of transistors in a network. This algorithm is repeated to generate a network for the complementary plan of the logic gate.

As results, the author presents new implementations for a majority gate, a 2-input MUX, a 3-input XOR, and for a comparator. These implementations were designed in static CMOS and also in *Pass-Transistor Logic* (PTL). In general, the solutions based in IG FinFETs presented reduction in power dissipation and area while the solutions based in SG FinFETs and PTL presented an increase of 52% and 98% in power and area, respectively. However, the solutions designed with SG FinFETs have greater performance.

Although the paper presents an algorithm for IG FinFET transistor networks generation, the author does not make clear whether such algorithm was implemented and tested for a large set of functions. The paper presents results for a very small set of five logic functions. Moreover, the algorithm presented in the paper is not enough

to understand how the networks are generated. The author does not present a complete example demonstrating how the networks are built.

#### 2.2.4 Rostami et al. (ROSTAMI e MOHANRAM, 2011)

Rostami et al. present a set of electrical and geometric configurations to determine the appropriated threshold needed to design IG FinFETs *high- $V_{th}$*  and IG FinFETs *low- $V_{th}$*  (ROSTAMI e MOHANRAM, 2011). A *high- $V_{th}$*  transistor implements a series arrangement of two input signal, as previous presented in (WANG, 2010) (CHIANG, KIM, *et al.*, 2006). On the other hand, the *low- $V_{th}$*  transistor implements a parallel arrangement of two input signals, equivalent to a *regular- $V_{th}$*  transistor previously presented in (CHIANG, KIM, *et al.*, 2006). Fig. 8 shows the notations proposed in (ROSTAMI e MOHANRAM, 2011) to represent different kinds of FinFETs.

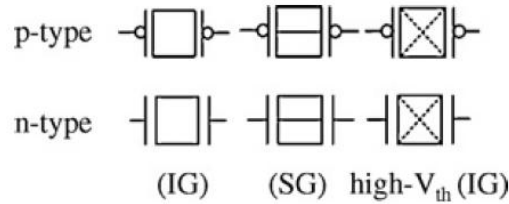


Figure 8 - Representation of different kinds of FinFETs proposed in (ROSTAMI e MOHANRAM, 2011).

The authors propose two techniques for defactoring Boolean expressions in order to leverage the merging of series and parallel transistor in IG FinFETs. For instance, a conventional network for the expression  $(a + (b \cdot c))$  can be implemented with three SG FinFETs, as illustrates in Fig. 9(a). However, by defactoring such expression we obtain  $((a + b) \cdot (a + c))$ , which needs only two IG FinFETs *low- $V_{th}$*  to implement the target function, as illustrated in Fig. 9(b). Similarly, the Fig. 9(c) presents a network composed of three SG FinFETs to implement the expression  $(a \cdot (b + c))$ . The defactorization produces  $((a \cdot b) + (a \cdot c))$ , allowing an optimized implementation using only two IG FinFET *high- $V_{th}$* , as shown in Fig. 9(d).

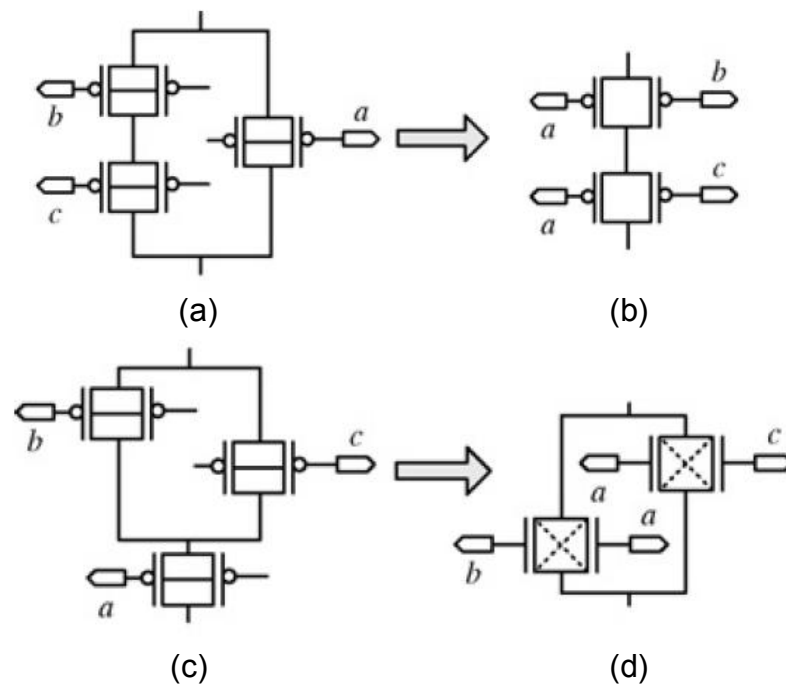


Figure 9 - Defactorizations proposed in (ROSTAMI e MOHANRAM, 2011).

Notice that, such defactorization technique tends to decrease the transistor count needed to implement a given logic function. In general, it contributes to design optimized logic gates and minimize area, capacitance and power dissipation. In this sense, the authors propose three libraries composed of low power logic cells implemented with SG FinFETs and IG FinFETs *high- $V_{th}$*  and *low- $V_{th}$* . These libraries were designed as follow:

**Basic library:** Contains some elementary gates as INV, 2-input AND, 2-input OR, 3-input NAND, 3-input NOR, and so on. All of the gates were implemented using only SG FinFETs.

**Library proposed in previous works:** This library has 41 gates, composed of the basic library presented above and adding other gates designed by merging transistors and disabling back gates, previously proposed in the literature (MUTTREJA, AGARWAL e JHA, 2007) (DATTA, GOEL, *et al.*, 2007).

**Complete library:** This library with 135 cells uses *high- $V_{th}$*  devices along with regular *low- $V_{th}$*  devices, and contains all the gates that are designed by merging series or parallel merging, along with the gates designed by

defactoring the Boolean equations. This library is a super-set of the two previous libraries.

Each gate is represented in the libraries by four different strengths, *i.e.*, 1X, 2X, 3X, and 4X. The strength of FinFET gates can be increased by adding parallel fins in each of its transistors. The ISCAS and OpenSPARC benchmark circuits were synthesized using the proposed libraries through the *Synopsys Design Compiler*. The experiment demonstrated that the complete library brings a significant reduction in area and power for the circuits. On average, the complete library reduces total power and number of fins by 36% and 37%, respectively, over the basic library based on conventional SG FinFETs in 32 nm technology. On the other hand, the previous work library achieves 20% and 21% reduction in total power and number of fins, respectively, over the basic library based on SG FinFETs in 32 nm technology (ROSTAMI e MOHANRAM, 2011).

Although the authors present a defactorization technique to improve the merging of series and parallel transistors, the paper does not present an automated method for IG FinFET transistor network generation. Thus, it is not clear how the 135 logic gates from the complete library were generated, insinuating that the library was manually generated.

### 2.3 The State of the Art on Transistor Network Generation

The switch network generation arises in the 1930s, when Claude Elwood Shannon studied techniques to build logic networks based on Boolean algebra. Thus, Shannon designed the Master's thesis entitled "*A Symbolic Analysis of Relay and Switching Circuits*" (SHANNON, 1938), which becomes the reference in the digital design of fewer years later.

In the last decades, different methods to generate optimized transistor networks were published in the literature. Part of them is based on factorization techniques and another part is based on graph optimizations. The main characteristics of the factorization method is that the final solution always is a series-parallel (SP) transistor network. It is due to the AND and OR operations present in the factored forms. On the other hand, the graph-based methods are able to achieve non-series-parallel (NSP) transistor networks, due to flexibility provided by the graph structure. In general, the SP arrangements contribute to design a more regular layout



compared to the NSP counterparts. The microelectronic industry has explored this characteristic through the *standard cell* libraries. However, the NSP arrangements can implement a given logic function using fewer transistors than the SP arrangements. These different characteristics can be explored according to the design constraints.

The following subsections present a review of the most recent works, published in the literature, for automatic transistor network generation. The main characteristics of such methods are briefly presented here, in order to identify why they are not the best alternative for IG FinFET design. Before to continue, it is important to mention that all these methods were designed thinking in single-gate devices *i. e.*, conventional planar CMOS or single-gate SG FinFETs. Thus, nowadays, these methods may be the best alternative for single-gate devices as SG FinFETs.

### 2.3.1 Mintz et al. (MINTZ e GOLUMBIC, 2005)

Mints et al. propose a factorization method based on graph partitioning (MINTZ e GOLUMBIC, 2005). The algorithm is recursive and operates over a logic function and their complement. For each iteration of recursion the function  $F$  is represented through the sum of two sub-functions  $F = F1 + F2$  or through the product of two sub-functions  $F = F3 * F4$ . Thus, the algorithm selects one of them to continue the recursion. The sub-functions used to perform a sum, or a product, are determined by computing two graphs  $G_f$  and  $G'_f$  from the function  $F$ . These graphs are used to calculate the intersections among the cubes from  $F$ . In the sequence, a graph partitioning is performed over  $G_f$  and  $G'_f$  and the best partitioning determines what sub-functions will be selected to continue the recursion.

In general, in the lowest levels of the recursion tree, *read-once* (RO) functions are found. Thus, a special routine is invoked to manipulate these parts of the function. A function is called *read-once* if it can be represented in a factored form where there are not repeated literals. In other words, each variable of the function appears once in the exact factored form (MINTZ e GOLUMBIC, 2005). The method proposed in (MINTZ e GOLUMBIC, 2005) and after improved in (GOLUMBIC, MINTZ e ROTICS, 2008) can achieve the exact solution (minimum number of literals) for *read-once* functions. In addition, this method has a competitive run time when compared with other techniques for *read-once* factorization.

### 2.3.2 Martins et al. (MARTINS, DA ROSA JUNIOR, et al., 2010)

Martins et al. propose a factorization technique based in a new paradigm called *Functional Composition* (FC), which consists in combining small (fewer literals) sub-functions until reaching the target function. The gradual composition allows controlling other criteria, beyond the literal count, of the partial functions and also of the final solution (MARTINS, DA ROSA JUNIOR, et al., 2010).

The proposed algorithm represents logic functions as a pair of  $\{functionality, implementation\}$ . The functionality is either a *Binary Decision Diagram* (BDD) node or a truth table. The implementation is either the root of an operator tree or a string representing a factored form. Together with the function implementation other characteristics can be stored like, number of literals, logic depth, series and parallel properties and so on.

The main idea of the algorithm is to produce pairs  $\{functionality, implementation\}$  of logic functions that can be represented by a single literal. After, the algorithm checks if the target function was already implemented with a single literal. If the target function was reached, then the algorithm ends. Otherwise, a loop is started where the sub-functions generated in the previous steps are combined by AND and OR operations to produce new sub-functions with an increase in the number of literals, as illustrated in Fig 10. This process is executed until reaching a factored form for the target function.

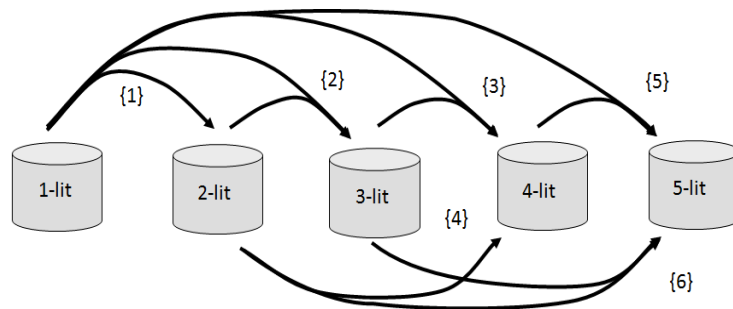


Figure 10 - Generation of sub-functions until the 5th bucket, proposed in (MARTINS, DA ROSA JUNIOR, et al., 2010). The number inside braces indicates the bucket composition step order.

This method is able to produce interesting results when compared to other factorization techniques available in the literature. Moreover, through the Functional Composition paradigm it is possible to adopt different criteria during the optimization process. This way, the method can deliver different solutions for a given logic function, each one with a particular impact in the quality of the final solution. However the limitation of this method is the high cost in run time, when the input function grows.

### 2.3.3 Kagaris et al. (KAGARIS e HANIOTAKIS, 2007)

Kagaris et al. propose a method that starts from a logic function  $f$  described in a sum-of-products (SOP) form and compose the transistor network gradually (KAGARIS e HANIOTAKIS, 2007). Each cube form  $f$  is inserted into the network according to the smallest cost to arrange their literals between two terminal nodes of the network. The authors called a set of literals (switches) in the network as *spine*, which can be divided in subsets to create new *spines*. Thus, a new cube can be inserted in the network by sharing their literals with other cubes (*spines*) already inserted in the network. If the new cube does not have common literals with the *spines* in the network, then a new independent *spine* is created between the terminal nodes of the network. For instance, let us consider the network composition illustrated in Fig. 11.

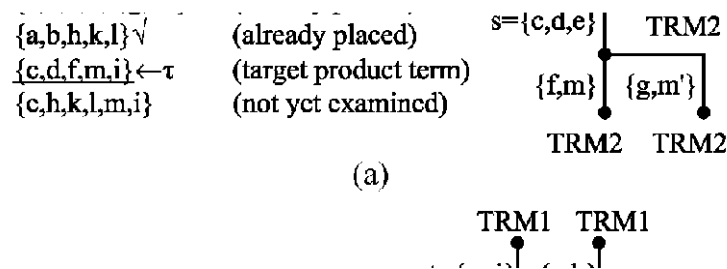


Figure 11 - Example of network composition proposed in (KAGARIS e HANIOTAKIS, 2007).

It is important to mention that the final solution depends on the order that the cubes are placed in the network. The authors propose different heuristics to start the insertion of cubes: examining the cubes in descending (or ascending) order of their literal counts, or examining next the cube that has maximum (or minimum) intersection with the already placed cubes (*spines*), or examining next the cube that

has maximum (or minimum) intersection with some terminal-to-terminal path in the currently constructed network, or simply examining the cubes in some random ordering (KAGARIS e HANIOTAKIS, 2007). As presented in the paper, if there are  $k$  candidate orders to start the optimization process, thus the method can try all of them and choose the best solution.

This method is able to generate NSP arrangements when a new *spine* shares literals with other two different *spines* already placed in the network. To ensure that the network is properly generated, a routine is executed to find and fix false paths in the network. A false path is a sequence of literals (switches) between the terminal nodes that is not covered by any cube from the input function. In some cases, a literal is inserted in the false path in order to transform it in a valid path. The paper presents a significant reduction in transistor count when compared with previous methods (CARUSO, 1991), (ZHU e ABD-EL-BARR, 1993), and (POLI, RIBAS e REIS, 2003).

#### 2.3.4 Possani et al. (POSSANI, CALLEGARO, *et al.*, 2013)

Recently, a new graph-based method for transistor network generation was proposed in (POSSANI, CALLEGARO, *et al.*, 2013). The method starts from an irredundant-sum-of-products (ISOP), and tries to combine the cubes of the function to build NSP and SP kernels. The NSP and SP kernels are graph structures used to determine the relationship among the cubes, as illustrated in Fig. 12(a) and in Fig. 12(b), respectively.

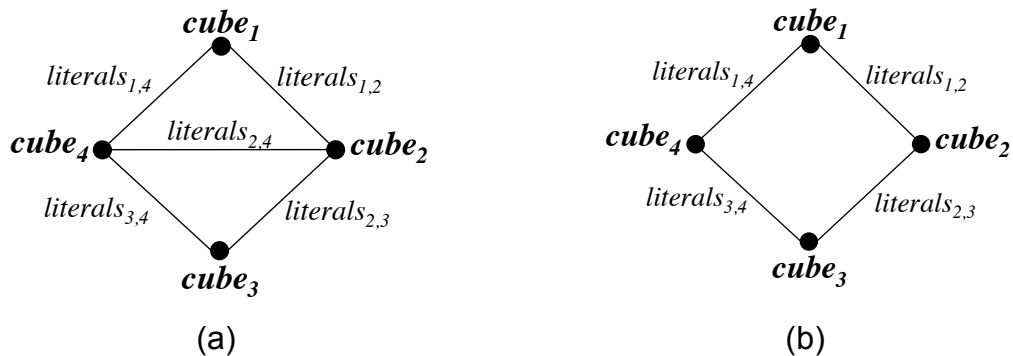


Figure 12 - Kernel structures: (a) NSP and (b) SP proposed in (POSSANI, CALLEGARO, *et al.*, 2013).

The main idea behind the kernels is to avoid greedy choices during the

beginning of the optimization process. In general, the greedy choices are related to local minima, prohibiting that more efficient solutions can be found. Thus, the kernels allow to find optimal sub-structures, which are merged to implement the final network. The method is divided in five well defined steps. The first three steps aim to build the kernels. The last two steps try to compose the found kernels in order to deliver an optimized switch network. Such steps are run in the following sequence: (A) NSP Kernel Finder, (B) SP Kernel Finder, (C) Fake Cube Insertion, (D) Kernel Composition, (E) Edges Compression.

Basically, the step (A) selects four cubes at a time and tries to build a NSP kernel for each combination. The found kernels are stored in a list and the remaining cubes are carried to the step (B). This second step selects four cubes at a time in order to built SP kernels. Similarly, the SP kernels are stored in a list and the remaining cubes are carried to to the step (C). The third step selects three cubes at a time and tries to build a NSP kernel by inserting a fourth cube that is a false cube. A false cube does not contribute with any logic to the network, but is structurally necessary to achieve the optimized arrangement. This way, the cubes of the input ISOP are gradually taken into account according to the kernels that are found. Hence, the number of possible combinations to select the cubes and to build the kernels tends to decrease from the step (A) to the step (C). Finally, the found kernels are associated in parallel and the last two steps (D) and (E) try to share equivalent switches<sup>1</sup> among the kernels, in order to perform local optimizations and remove redundancies.

According to the characteristics of the input ISOP, different situation may occur during the optimization process. A possible situation is that, one of these three first steps may not find any kernel. Thus, the following step receives all cubes still available at that moment. Other situation is that, the routines (A), (B), and (C) may not find any kernel. Thus, the switch network is generated during the steps (D) and (E), by applying an edge sharing technique associated to an edge compression technique. In some cases, multiple kernels can be found during the steps (A), (B), and (C). This way, the method is able to generate more complex networks by merging these kernels during the steps (D) and (E).

---

<sup>1</sup> Switches controlled by the same literal are considered equivalentes.

For instance, let us consider the Equation (1) to exemplify how the method works. In this case, the method is able to find a SP kernel and a NSP kernel as illustrated in Fig. 13(a) and Fig 13(c), respectively. Thus, each kernel is translated to their respective network by reordering and mapping the literals of the edges to switches in the network, as illustrated in Fig. 13(b) and Fig. 13(d). In the sequence, the kernel composition step removes redundant switches between these partial networks, resulting in the optimized transistor network illustrated in Fig. 13(e).

$$f = !a.!b.!c.!d + !a.b.d + !a.b.c + a.!b.d + a.!b.c + a.b.!c.!d + b.c.d \quad (1)$$

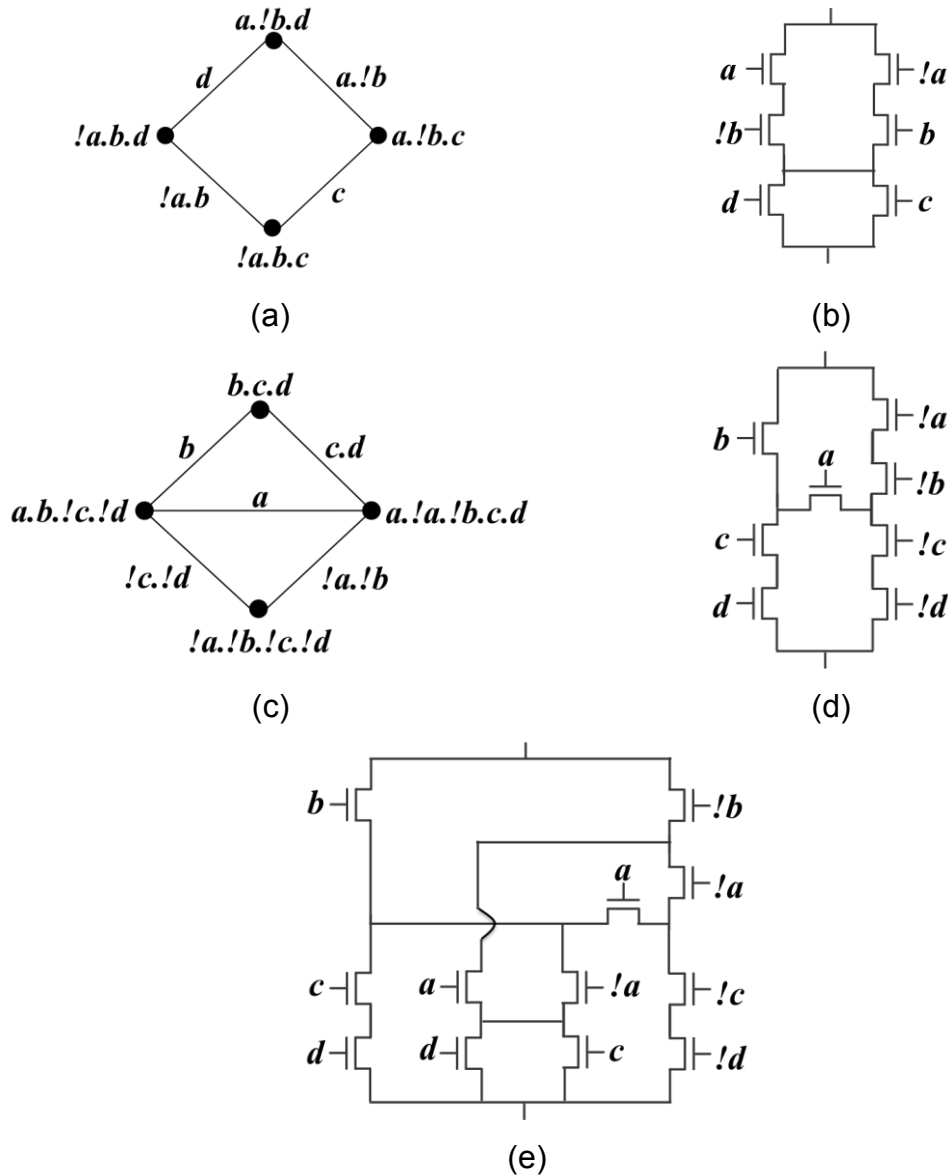


Figure 13 - SP and NSP kernels in (a) and (c), respectively, and correspondent networks in (b) and (d). Final solution (e) obtained by compose networks (b) and (d).

The paper presents some experiments over different set of logic functions. The obtained results demonstrate improvements in the transistor count reduction when compared to the results obtained by the most recent methods (KAGARIS e HANIOTAKIS, 2007) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010). The main contribution of this method is the possibility to find promising arrangements in the beginning of the optimization process without performing greedy choices.

### 3 INVESTIGATING AND IDENTIFYING THE NEW CHALLENGES ESTABLISHED BY INDEPENDENT-GATE FINFET TRANSISTORS

This chapter presents some case studies that demonstrate how the merging of series and parallel transistors in IG FinFETs impacts in the transistor network generation. A brief discussion regarding this impact was previously presented in (POSSANI, REIS, *et al.*, 2014). We consider this analysis the main contribution of this work. Although, in the literature, there are many works proposing novel solutions based in IG FinFET transistors, no one of them presents an analysis demonstrating the existence of a new paradigms as we are presenting here. This analysis provides support to design future works related to IG FinFET devices.

In general, a transistor network can be generated by different ways and using different logic styles. In this work we are concerned to generate transistor networks to implement logic gates based in FinFET devices. As a logic gate can be implemented using two complementary networks, one for the *pull-up* and the other one for *pull-down* plane, the proposed methods aim to generate the networks for each plane separately. Thus, in order to simplify the case studies and the examples along of this work, we will consider the transistor network generation for a single plane of the logic gates.

**Case 1:** The first interesting point is that two different factored forms, which represent the same logic function  $f$  and are composed of the same number of literals, can result in transistor networks with a different number of devices. Thus, in order to demonstrate that, let us consider the equivalent factored forms represented by Equation (2) and Equation (3), both composed of 8 literals.

$$f = (a + (d \cdot (b + c))) \cdot (!a + (!b \cdot (!c + !d))) \quad (2)$$

$$f = ((!a \cdot d) \cdot (c + b)) + ((a \cdot !b) \cdot (!c + !d)) \quad (3)$$



When considering single-gate devices, *i. e.*, conventional CMOS or SG FinFETs, these factored forms can be used to build two different transistor networks to implement the function  $f$ , both composed of 8 devices. However, when considering the possibility of merging series and parallel transistor in IG FinFETs, it is possible to save 2 transistors by using the Equation (3) instead of using the Equation (2). The IG FinFET transistor network obtained from Equation (2) is illustrated in Fig. 14(a) and the more optimized arrangement, obtained from Equation (3), is illustrated in Fig. 14(b). As can be seen, it is difficult to determine what is the best factored form to be used if just the literal count is adopted as metric. ■

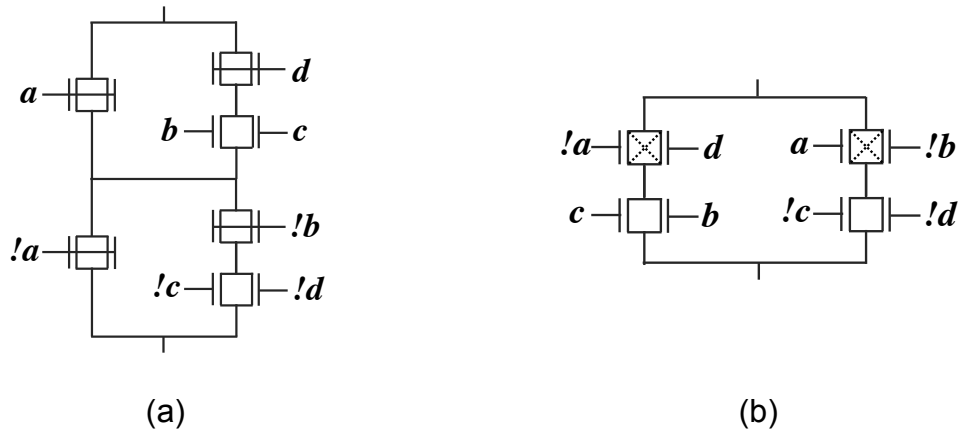


Figure 14 - In (a), transistor network obtained from Equation (2) and in (b), the optimized network obtained from Equation (3).

**Case 2:** This second case demonstrates that just minimizing the number of literals in a Boolean expression may not lead to the minimum number of IG FinFETs. Let us consider the logic function  $f$ , which can be factored of two different ways. One resulting in 14 literals and the other one resulting in 15 literals, as represented in Equation (4) and in Equation (5), respectively.

$$f = (!a + ((!c + !d) \cdot (b + (c + d)))) \cdot ((a \cdot !b) + ((!c + d) \cdot (c + (!d \cdot (a + !b))))) \quad (4)$$

$$f = ((!b \cdot a) \cdot ((!d \cdot c) + (d \cdot !c))) + ((!d \cdot !c) \cdot ((!b \cdot !a) + (b \cdot a))) + (!a \cdot (d \cdot c)) \quad (5)$$

These two factored forms can be used to build transistor networks composed of single-gate devices. In this case, the networks obtained from Equation (4) and

Equation (5) are composed of 14 and 15 transistors, respectively. As expected, the network obtained from Equation (4) will save one transistor compared to the network obtained from Equation (5). On the other hand, when considering devices like IG FinFETs to implement the function  $f$ , we have the opposite. Notice that, through the Equation (4) it is possible to build a network composed of 9 transistors, as shown in Fig. 15(a). However, through the Equation (5), it is possible to build an efficient arrangement using only 8 transistors, as shown in Fig. 15(b). In this case, more literals in the factored form results in fewer transistors in the network. Indeed, the best factored form for single-gate devices may not be the best candidate equation for independent-gate devices and vice versa. ■

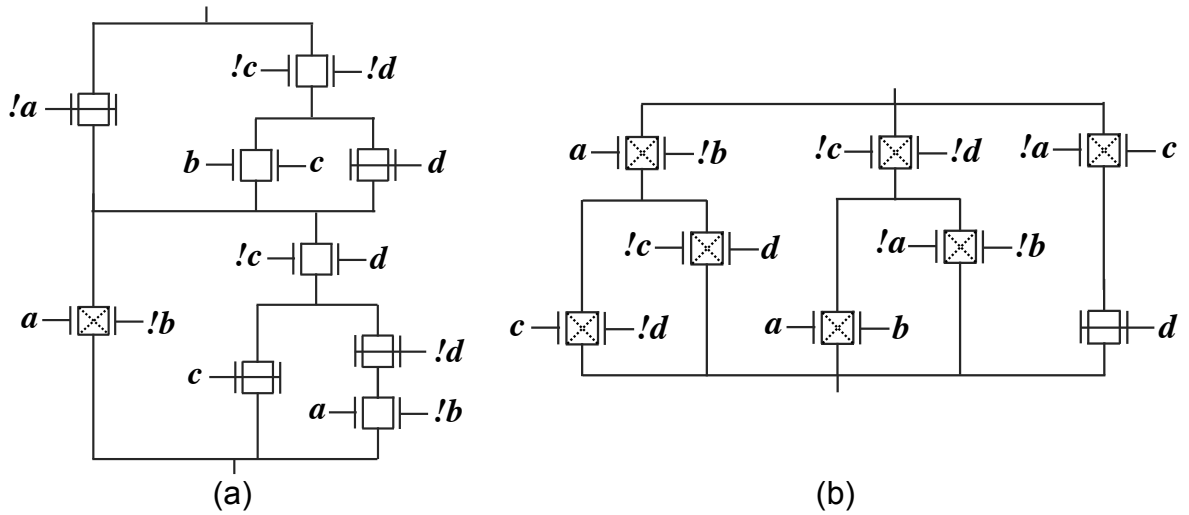


Figure 15 - In (a), network obtained from Equation (4) and in (b), optimized network obtained from Equation (5).

**Case 3:** When the goal is to minimize the transistor count, the non-series-parallel (NSP) arrangements are a feasible way to do that (POSSANI, CALLEGARO, *et al.*, 2013). In general, the graph-based techniques are able to find NSP arrangements while the factorization methods are limited to series-parallel (SP) arrangements. Thus, when considering single-gate devices, the NSP arrangements can result in a network with the same number of transistors or with fewer transistors than the SP counterpart. However, when considering the merging of series and parallel transistors in IG FinFETs, in some cases, the SP arrangements can result in a network with fewer transistors than the NSP solution.

To demonstrate an instance of this problem, let us consider the Equation (6). This equation can be factored resulting in an expression composed of 7 literals, as presented in Equation (7).

$$f = a.e.b + !a.d + b.c.d \quad (6)$$

$$f = b . ((a . e) + (c . d)) + (!a . d) \quad (7)$$

Such factored form can be used to build a SP transistor network with 7 single-gate devices, as shown in Fig. 16(a). Another alternative is to apply a graph-based technique over the Equation (6), resulting in a NSP transistor network comprising 6 single-gate devices, as shown in Fig. 16(b). Notice that, the NSP arrangement contributes to decrease the transistor count. Thus, in a first analysis we can claim that the NSP solution is better than the SP one, considering SG FinFETs.

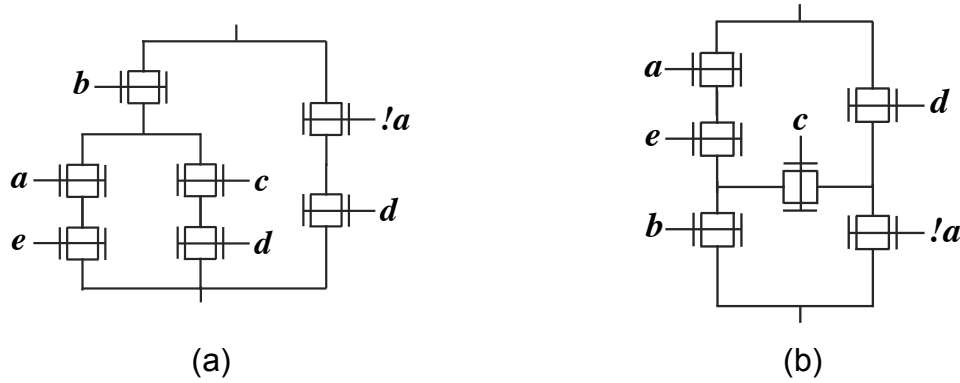


Figure 16 - Two possible solutions to implement the logic function  $f$  using SG FinFETs: in (a) SP network and in (b) NSP network.

However, when considering IG FinFETs, the series transistors of the SP network illustrated in Fig. 16(a) can be merged resulting in the network illustrated in Fig. 17(a). Similarly, by merging transistors in the NSP network illustrated in Fig. 16(b), it is possible to achieve the network illustrated in Fig. 17(b). As can be seen, the NSP network results in 5 transistors while the SP network needs only 4 transistors to implement the function  $f$ . Thus, we have observed that, in some cases, the NSP configuration precludes the merging of transistors. ■

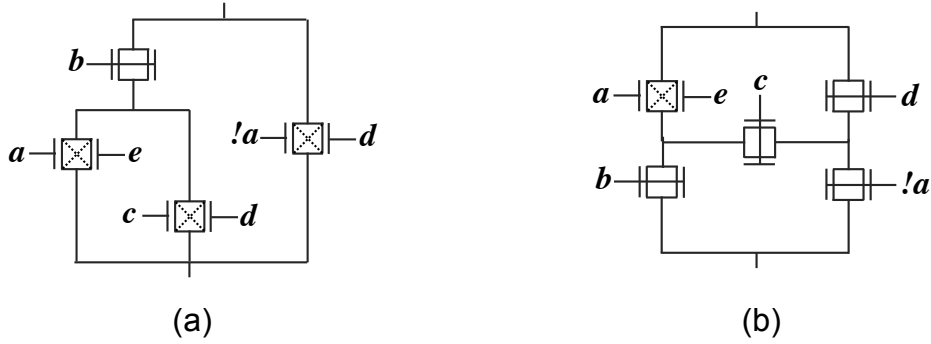


Figure 17 - Two possible solutions to implement the logic function  $f$  using IG FinFETs: in (a) SP network and in (b) NSP network.

**Case 4:** This case demonstrates that in some cases, the NSP arrangements still being the best solution to decrease the transistor count. For instance, let us consider the Equation (8). By applying a factorization technique over this equation we obtain a factored form composed of 10 literals, as presented in Equation (9). Such factored form can be directly mapped to a SP network composed of 10 single-gate devices, as shown in Fig. 18(a). However, when applying a graph-based technique over the Equation (8), it is possible to achieve an efficient NSP arrangement with 7 transistors, as shown in Fig. 18(b).

$$f = a.b + a.c + a.d.g + c.e + b.c.d.g \quad (8)$$

$$f = c . (e + (b . e . g)) + a . ((d . g) + b + c) \quad (9)$$

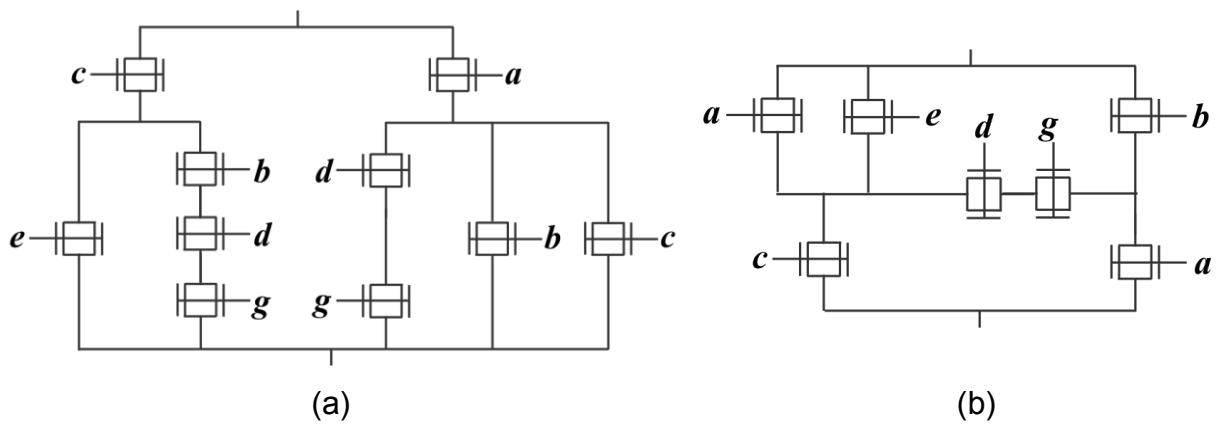


Figure 18 - Two possible solutions to implement the logic function  $f$  using SG FinFETs: in (a) SP network and in (b) NSP network.

In this case, the gains obtained by using the NSP arrangement remain when considering IG FinFETs. It is illustrated in Fig. 19(a) and Fig. 19(b), which present the SP and NSP solutions, respectively. Notice that the NSP arrangement is the best solution, keeping a reduction of 2 transistors when compared to the SP implementation. ■

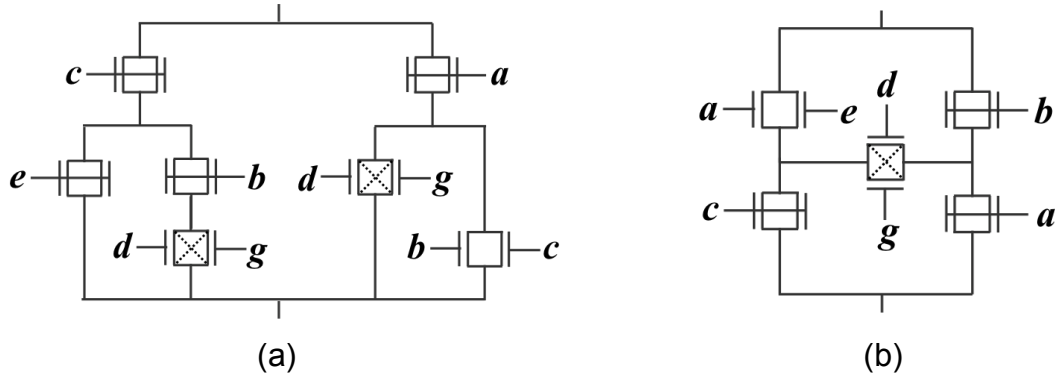


Figure 19 - Two possible solutions to implement the logic function  $f$  using IG FinFETs: in (a) SP network and in (b) NSP.

**Case 5:** The previous cases have demonstrated some challenges introduced by the IG FinFET transistors, when generating minimal transistor networks for general Boolean functions. However, this case aims to demonstrate how the IG FinFETs affect the implementation of transistor networks derived from special classes of Boolean functions like read-once (RO) and read-polarity-once (RPO) classes. A Boolean function is considered RO if it can be represented in a minimum factored form where each variable appears only once in the expression (GOLUMBIC, MINTZ e ROTICS, 2008). The function represented by Equation (10) is RO.

$$f = (a + b) \cdot ((c \cdot d) + e) \quad (10)$$

Similarly, a Boolean function is considered RPO if it can be represented in a minimum factored form where each polarity (positive or negative) of a variable appears at most once in the expression (CALLEGARO, MARTINS, *et al.*, 2013). The function represented by Equation (11) is RPO.

$$f = (!a \cdot d + c) \cdot (a + b) \quad (11)$$

Functions from RO and RPO classes are attractive, especially because the optimality can be guaranteed for these classes. In other words, RO and RPO functions can be always factored with the minimum number of literals (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO, MARTINS, *et al.*, 2013). This way, it is possible to generate efficient transistor networks composed of the minimum number of single-gate devices. However, when considering double-gate devices, as IG FinFETs, these classes cannot guarantee the optimality in terms of transistor count during the network generation. It occurs due to the same fact presented in the *Case 2*, where a factored form with the minimum number of literals does not deliver the minimum network. Thus, we have observed that, although the definition of RO class says that each variable must appear once, in some cases, it is opportune to repeat some literals in the factored expression in order to achieve the best merging of series and parallel transistors in IG FinFETs. This observation is also valid for Boolean function from RPO class.

For instance, let us consider the RO function represented by Equation (12). This factored form can be used to build the network illustrated in Fig. 20(a), where each literal present in the expression was directly mapped to a single-gate transistor in the network. Another possibility is to use this factored form to build a network by merging series and parallel transistors in IG FinFETs, resulting in the network illustrated in Fig. 20(b). However, the network presented in Fig. 20(b) is not the best solution in number of transistors. Observe that, by replicating the literal *a* in the factored form of Equation (12), it is possible to achieve the network presented in Fig. 20(c), which saves one transistor compared to the network illustrated in Fig. 20(b). ■

$$f = a \cdot ( (b \cdot (d + e)) + (c \cdot h \cdot g) ) \quad (12)$$

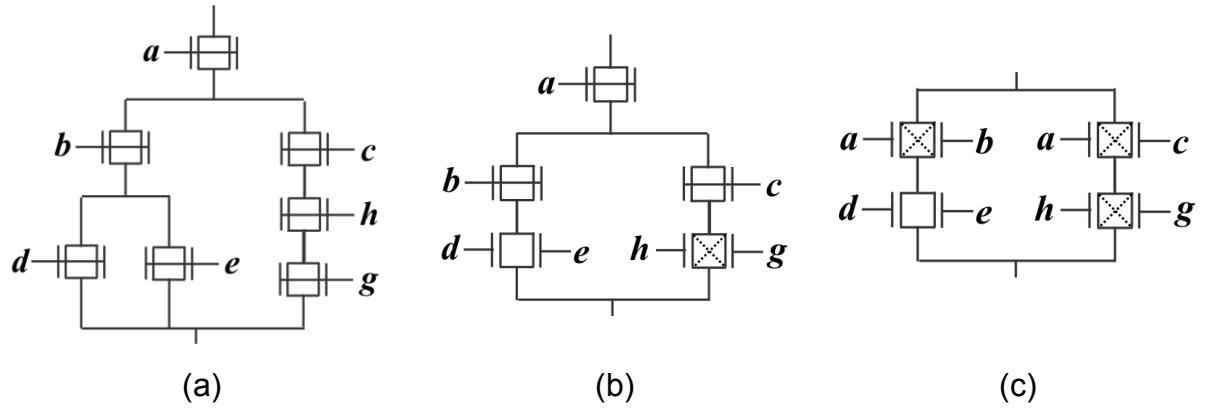


Figure 20 - In (a) SG FinFET implementation and in (b) IG FinFET implementation, both obtained from the factored form of Equation (12). In (c) IG FinFET implementation obtained by replicating the literal *a* in the factored form of Equation (12).

The analysis presented in this chapter pointed the main challenges that we have observed during the IG FinFET transistor networks generation. Thus, this study provides support to understanding that, in fact, a new paradigm was introduced by the IG FinFET technology. Once we have discovered what are the main problems to be solved during the automatic synthesis of transistor networks, in the next chapter, two alternative methods are proposed based in such analysis.

## 4 PROPOSED METHODS FOR INDEPENDENT-GATE FINFET TRANSISTOR NETWORK DESIGN

This chapter presents the two proposed methods for IG FinFET transistor network generation. The first one is a flexible graph-based method, which starts from a Boolean expression and tries to find promising patterns to be merged in IG FinFETs (*series*) and (*parallel*). The second one is a defactorization method, which manipulates a factored form produced by a conventional factorization method. The main idea behind the defactorization is to improve the possibilities for merging series and parallel transistors in IG FinFETs.

### 4.1 Generating Transistor Networks by Graph-Based Optimizations

In this section we propose an alternative method that tries to find the best merging of series and parallel transistors during the beginning of the optimization process. This way, it is possible to efficiently explore the separated gates of the IG FinFETs during the network generation. The proposed method operates over an irredundant-sum-of-products (ISOP) in three steps (A) *Series-Parallel Kernel Finder*, (B) *IG FinFET Dedicated Factorization*, and (C) *Network Composition*. Before to proceed, let us consider a brief overview of the proposed method:

- (A) The first step tries to find efficient arrangements through a graph-based structure called SP kernel, which was previously proposed in (POSSANI, CALLEGARO, *et al.*, 2013). The arrangements obtained through the SP kernels are useful for merging series and parallel transistors in IG FinFETs.
- (B) The second step presents a strategy to perform simple factoring operations, considering some constraints to maximize the merging of series and parallel transistors.



- (C) Finally, the third step is responsible for building the final network by composing the SP kernels with the factored arrangements obtained during the previous steps (A) and (B).

#### 4.1.1 Series-Parallel Kernel Finder

The SP kernels presented here allow efficient arrangements with a large sharing between the paths of the network. Besides that, such arrangements are promising to be merged in IG FinFETs. These are two attractive characteristics that contribute to decrease the transistor count. The main idea behind the SP kernels is to combine the cubes of the input ISOP, taking four of them at a time, in order to build an undirected graph where the cubes are represented through the vertices and the edges represent common literals between pairs of cubes (vertices). A SP kernel is determined according to the definitions and restrictions presented below.

For each combination, the algorithm selects four cubes composing a function  $h$  that is represented by an ISOP form. In the sequence, an undirected graph  $G = (V, E)$  is built, where the vertices in  $V = \{v_1, v_2, v_3, v_4\}$  represent different cubes in  $h$ , so  $|V| = 4$ . To ensure that the obtained graph is a valid SP kernel, two rules must be checked:

**Rule 1** – Let  $E_{v_i}$  be the set of edges that are connected to  $v_i$  and  $|E_{v_i}|$  the number of edges in this set. For each vertex  $v_i \in V$ , the set of literals of  $v_i$  (i.e.,  $literals(v_i)$ ) must be shared through the edges  $e_j \in E_{v_i}$ . This rule is satisfied if and only if the following equation results in the value 1:

$$\prod_{i=1}^{|V|} \left( \left( \bigcup_{j=1}^{|E_{v_i}|} label(e_j) \right) \equiv literals(v_i) \right) \quad (13)$$

**Rule 2** – The graph  $G$  obtained must be isomorphic to the graph shown in Fig. 21(a), referred herein as *SP kernel*.

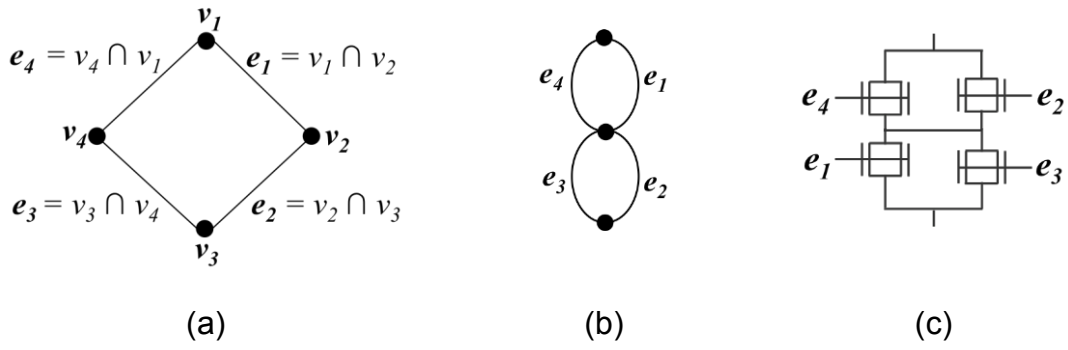


Figure 21 - SP kernel template (a), auxiliary template graph (b) and resulting SG FinFET network (c).

The SP kernel finder step must apply some transformations over the graph in order to map each kernel found to a switch network. Therefore, the first step consists on mapping the kernel edges, illustrated in Fig. 21(a), to an auxiliary template graph depicted in Fig. 21(b). After, an edge reordering routine is applied over the auxiliary template graph, moving the edge  $e_1$  to the place of  $e_3$ , thus, the edge  $e_3$  is moved to the place of  $e_2$ , and finally, the edge  $e_2$  is moved to the place of  $e_1$ . This edge reordering results in the transistor network shown in Fig. 21(c). Through these operations, it is possible to ensure that each path of the transistor network represents a cube from the function  $h$ .

**Example 1:** In order to demonstrate how a transistor network is obtained from a SP kernel, let us consider the kernel illustrated in Fig. 22(a), which was obtained from the Equation (14).

$$f = a.c + a.d + b.c + b.d \quad (14)$$

The edge labels are mapped from the kernel illustrated in Fig. 22(a), to the auxiliary graph shown in Fig. 22(b). Consequently, by applying the edge reordering over the auxiliary graph, the kernel found is mapped to the transistor network depicted in Fig. 22(c). Notice that the auxiliary template graph and the edge reordering routine are necessary to implement the cubes from Equation (14) through the sharing between the paths of the network illustrated in Fig. 22(c). As can be seen in Fig. 22(c), the parallel transistors  $(a + b)$  and  $(c + d)$  can be merged in IG FinFETs (*parallel*), resulting in a network with only two transistors, as illustrated in

Fig. 22(d). It is worth to mention that the SP kernel allows the generation of networks with more transistors, enabling also the merging of series transistors. ■

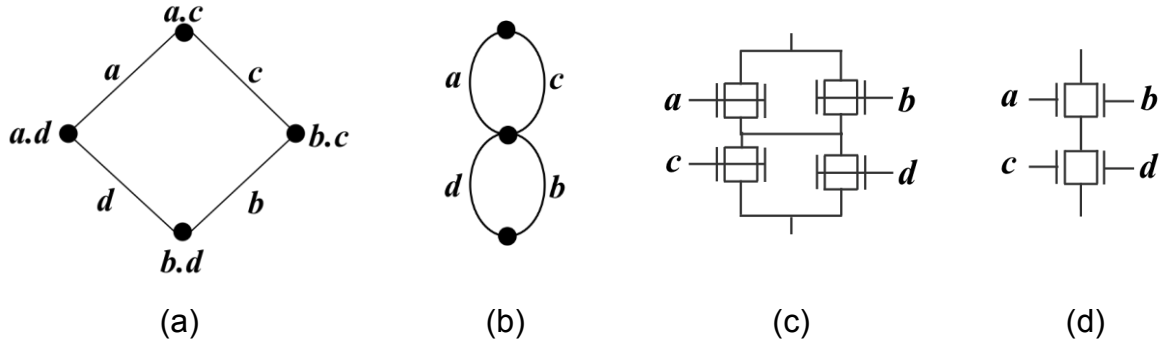


Figure 22 - SP kernel (a) derived from Equation (14), auxiliary graph template (b), SG FinFET network (c) and IG FinFET network obtained after applying the edge reordering routine.

#### 4.1.2 IG FinFET Dedicated Factorization

This step is responsible for factoring the cubes from the input ISOP in such a way that there are at least two common literals to a group of cubes. This constraint was adopted in order to prioritize factorizations that allow the merging of two literals in an IG FinFET and so, minimize the use of SG FinFETs in the network. Thus, the algorithm uses a table to determine the relationship between the cubes. This is demonstrated in the following example.

**Example 2:** For instance, consider the Boolean function described by Equation (15).

$$f = c.d + !a.b.!c.!d + a.b.d + a.b.c \quad (15)$$

Firstly, the algorithm builds a table using the cubes and the variables from the input ISOP as presented in Fig. 23. The occurrence of a variable in the direct or complementary polarity is represented in the table by 1 or 0, respectively. Then, the next step consists in performing a search over the table in order to find the **wanted** groups, like this circled by a dashed line in Fig. 23. The idea behind this group is to indicate that there are at least two common literals, *i. e.*,  $(a . b)$ , in *cube3* and *cube4*.

Thus, these cubes will be factored as follow:  $(a \cdot b) \cdot (c + d)$ . This local factorization can be implemented using only two IG FinFETs, one (*series*) and one (*parallel*), as can be seen in the leftmost branch of the network illustrated in Fig. 24. In the sequence, the remaining cubes, *cube1* and *cube2*, which not belong to any group, are implemented as independent branches using IG FinFETs (*series*), as can be seen in the rightmost branches of the network shown in Fig. 24.

		<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
<i>cube1</i>	<i>c.d</i>			1	1
<i>cube2</i>	<i>!a.b.!c.!d</i>	0	1	0	0
<i>cube3</i>	<i>a.b.d</i>	1	1		1
<i>cube4</i>	<i>a.b.c</i>	1	1	1	

Figure 23 - Table obtained from Equation (15) that represents cubes relationship to determine wanted groups that share at least two literals.

Notice that, by adopting factorizations with at least two literals, it was possible to avoid the sharing of the literal ***b***, which appears three times, as circled by a dotted line in Fig. 23. Traditional factorization methods tend to perform the ***unwanted*** sharing, resulting in the network illustrated in Fig. 25. As can be seen in Fig. 24, the network obtained by the proposed method saves one transistor if compared to the conventional solution presented in Fig. 25. ■

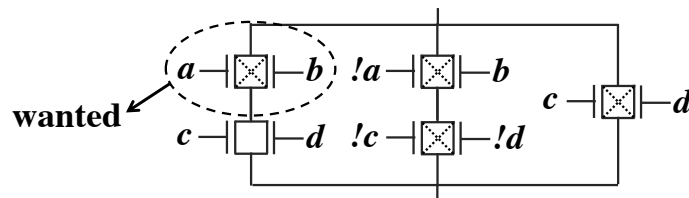


Figure 24 - Network delivered by the proposed method obtained from Equation (15).

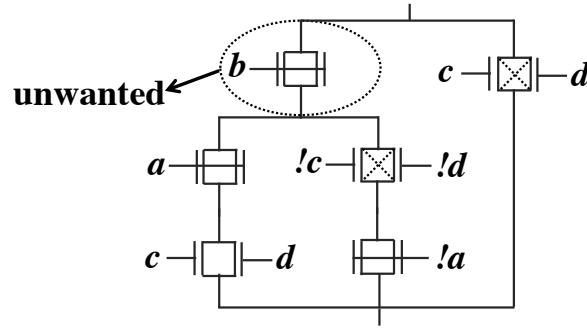


Figure 25 - Network delivered by conventional methods obtained from Equation (15).

#### 4.1.3 Network Composition

As mentioned before, this step aims to compose the SP kernels and the factored arrangements obtained during the previous steps in order to build the final network. Thus, such network can be generated by different ways according to the input ISOP: (i) a network can be implemented using one or more SP kernels associated in parallel; (ii) a network can be implemented using one or more SP kernels, associated in parallel to other factored arrangements obtained in the step (B); (iii) no SP kernels are found during the step (A), so, the network is completely generated in the step (B).

After finding the sub-arrangements, the network is generated by associating these partial solutions as parallel branches. Thus, the edge sharing technique proposed in (POSSANI, SOUZA, *et al.*, 2012) is applied in order to perform local optimizations and remove some redundancies, which may eventually exist between the SP kernels and the factored arrangements. This process is very similar to the kernel composition step described in (POSSANI, CALLEGARO, *et al.*, 2013). An interesting characteristic of the proposed method is that, by applying such edge sharing technique in this last step, the method is able to find series-parallel SP and also non-series-parallel NSP arrangements. As presented in the Example 4, in some cases, the NSP arrangements can contribute to decrease the transistor count. In general, the networks generated by the proposed method tend to be composed only by series-parallel arrangements. It is due to the fact that the steps (A) and (B) produce series-parallel arrangements, becoming very similar to a factorization method.

**Example 3:** To demonstrate how a network is composed, let us consider that the partial networks presented in Fig. 26 were found during the two first steps of the method by processing the Equation (16). In this case, the arrangement illustrated in Fig. 26(a) was obtained through a SP kernel along of the step (A). Furthermore, the arrangements illustrated in Fig. 26(b) and Fig. 26(c) were found during the step (B). Thus, these three partial solutions will be associated as parallel branches of a network and the last step of the optimization process is executed in order to remove redundant transistors among the branches. In this case, the redundant IG FinFET *parallel* ( $c + d$ ) was shared, resulting in the optimized networks illustrated in Fig. 26(d). ■

$$f = a.c + a.d + b.c + b.d + e.g.c + e.g.d + !a.!b.!e.!g \quad (16)$$

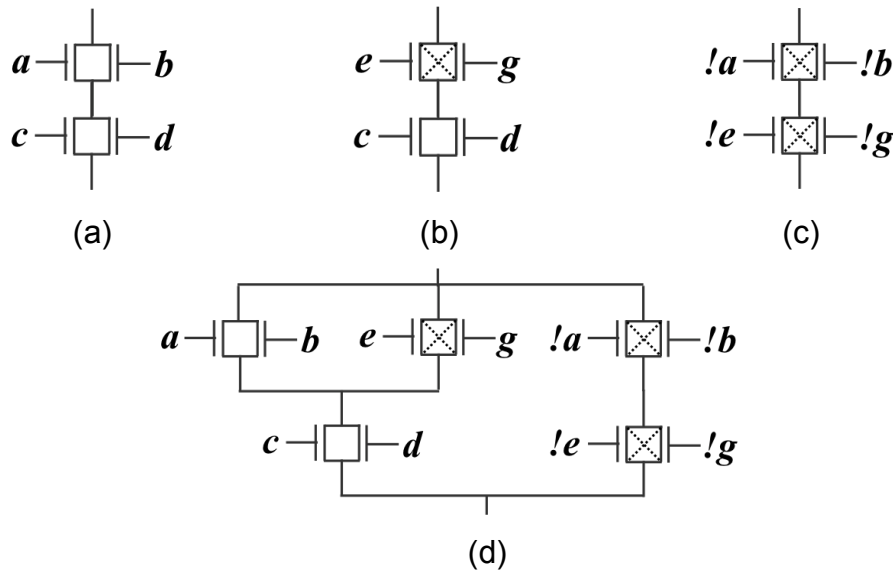


Figure 26 - Partial arrangements found from Equation (16) during the two first steps in (a), (b), and (c). Final solution achieved after to remove redundant transistors among the parallel branches in (d).

## 4.2 Generating Transistor Networks by Defactoring Boolean Expressions

As demonstrated in Section III, in fact the IG FinFET has established innovations and challenges during the transistor network generation. Consequently, the arrangements delivered by conventional techniques, based on factorization or graph optimizations, may not be useful enough to explore the potential of merging series and parallel transistors. In this sense, an alternative is to apply a defactorization technique over the factored forms produced by the conventional methods. It was firstly demonstrated in (ROSTAMI e MOHANRAM, 2011), where the authors presented two ways for defactoring Boolean expressions, as previous discussed in the subsection 2.3.4 of the Chapter 2.

### 4.2.1 First Observation on Defactorization

The first defactorization technique presented in (ROSTAMI e MOHANRAM, 2011) aims to map conventional arrangements, like the network illustrated in Fig. 27(a), to an optimized network by defactoring and merging parallel transistors (literals), as shown in Fig. 27(b). As can be seen, it leads to decrease the number of transistors. However, we have observed that it may not be the best solution in terms of transistor count when considering arrangements similar to this illustrated in Fig. 27(a), but composed of more than four transistors. Thus, when there is a similar configuration to this illustrated in Fig. 27(a), we propose the merging of series transistors instead parallel ones. This process will be demonstrated in following example.

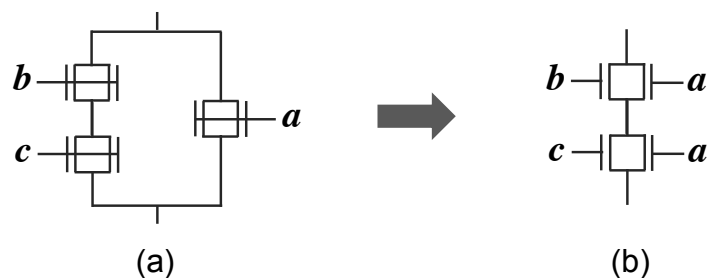


Figure 27 - First defactorization proposed in (ROSTAMI e MOHANRAM, 2011): conventional arrangement in (a) and defactored arrangement saving one transistor in (b).

**Example 4:** For instance, let consider the conventional arrangement illustrated in Fig. 28(a). In this case, by applying the parallel defactorization proposed in (ROSTAMI e MOHANRAM, 2011), it is possible to achieve the network illustrated in Fig. 28(b). However, by merging series transistor instead defactoring, it is possible to achieve the network illustrated in Fig 28(c). Notice that, the alternative proposed in (ROSTAMI e MOHANRAM, 2011) leads to a reduction of one transistor when compared to the conventional arrangement. On the other hand, our alternative leads to a reduction of two transistors when compared to conventional arrangement. ■

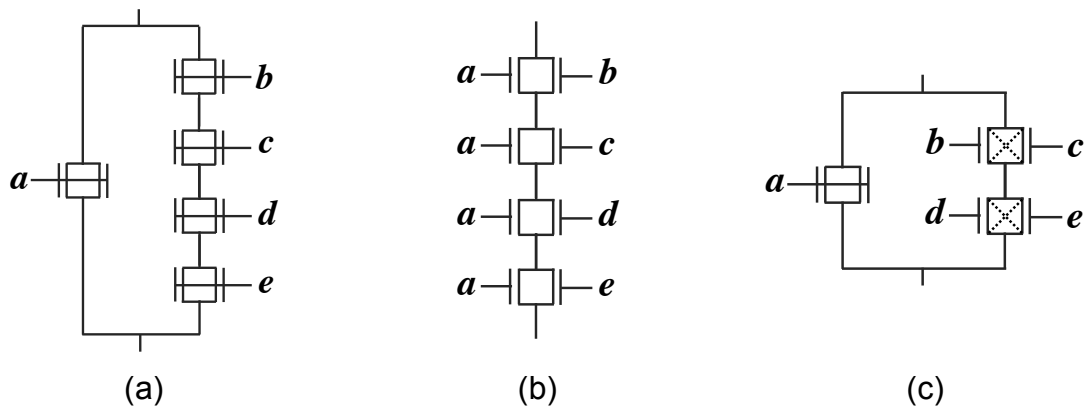


Figure 28 - Conventional arrangement in (a), network obtained by defactoring according (ROSTAMI e MOHANRAM, 2011) in (b), and the proposed merging in (c).

#### 4.2.2 Second Observation on Defactorization

The second defactorization technique presented in (ROSTAMI e MOHANRAM, 2011) aims to map a conventional arrangement as illustrated in Fig. 29(a) to a more efficient network, by defactoring and merging series transistors (literals), as shown in Fig. 29(b). Again, we have observed that it is not the best alternative for optimizing arrangements similar to this illustrated in Fig. 29(a), but composed of more than four transistors. This problem is presented in the next example.



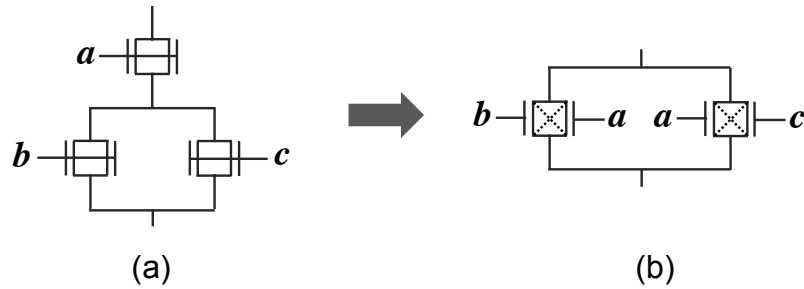


Figure 29 - Second defactorization proposed in (ROSTAMI e MOHANRAM, 2011): conventional arrangement in (a) and defactored arrangement saving one transistor in (b).

**Example 5:** Let us consider the conventional arrangement illustrated in Fig. 30(a). In this case, by defactoring and merging series transistors as presented in (ROSTAMI e MOHANRAM, 2011) it is possible to achieve the network illustrates in Fig. 30(b), which saves one transistor when compared to the conventional arrangement. However, we have observed that it is opportune to hold the factored transistor and merge the parallel ones, as illustrated in Fig. 30(c). Notice that, our alternative allows a reduction of two transistors when compared to the conventional arrangement. ■

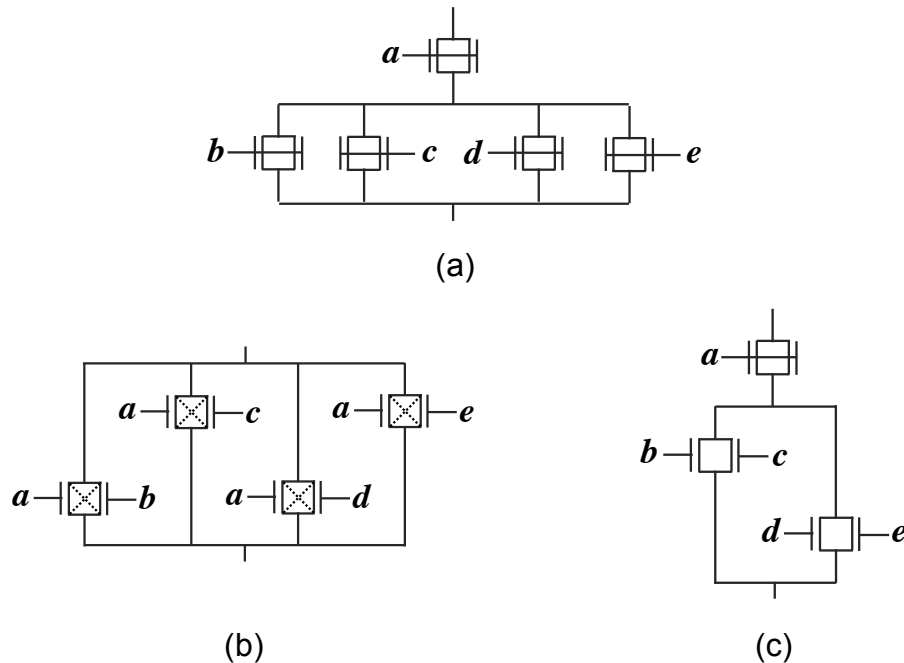


Figure 30 - Conventional arrangement in (a), network obtained by defactoring according (ROSTAMI e MOHANRAM, 2011) in (b), and the proposed merging in (c).

### 4.2.3 Third Observation on Defactorization

We have observed that, to achieve optimized results we cannot stay limited to these patterns presented in the previews examples of Fig. 28(c) and Fig. 30(c). In some special cases, series defactorizations may contribute to decrease the transistor count. Such cases are defined by the patterns presented in the Fig. 31(a) and Fig. 31(b).

The first pattern is based on a *factored* literal  $I$  associated to a set of *AND* nodes through an *OR* node, as illustrated in Fig. 31(a). The restriction adopted here is that each branch (*AND* nodes) must have exact one *single* literal, as demonstrated by  $s1$  and  $s2$  in the Fig. 31(a). This pattern allows replicating the *factored* literal  $I$  to be merged with the *single* ones. Besides that, we also have considered a second pattern, which is a simple variation of the first one. It is demonstrated in Fig. 31(b), where exact one *single* literal  $s3$  can appears connected directly to the *OR* node as shown in Fig. 31(b).

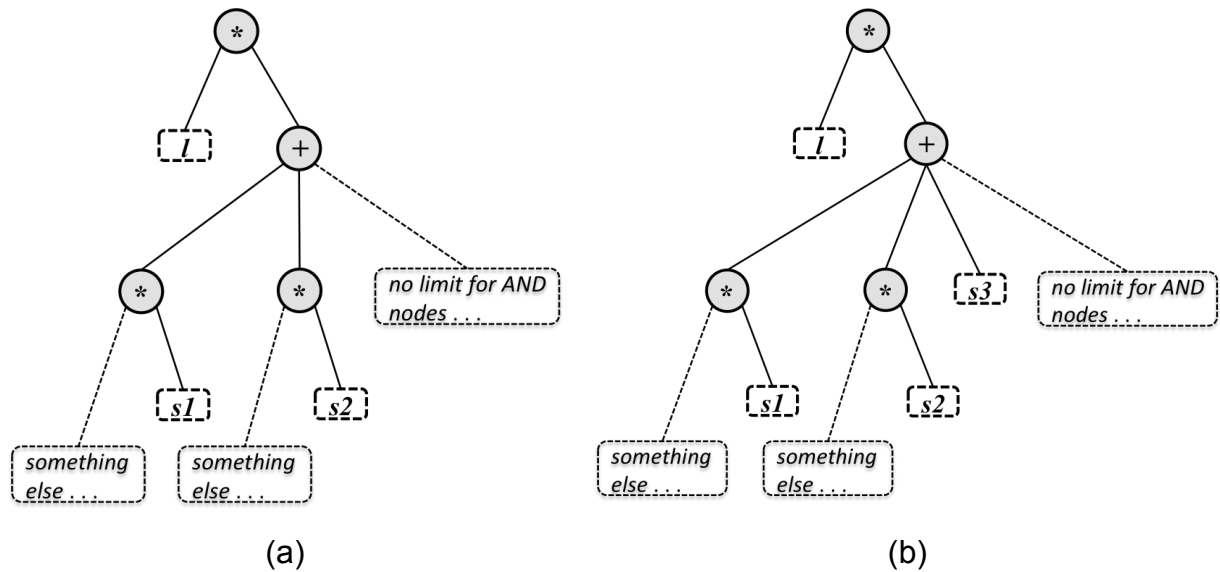


Figure 31 - Two adopted patterns to perform series defactorizations.

**Example 6:** For instance, consider the network illustrated in Fig. 32, which was obtained from Equation (17). Notice that the second pattern is found in this case, where we have a conventional arrangement composed of a *factored* transistor and exactly one *single* transistor associated in each branch of the network, as shown in Fig. 32. As can be seen, it is opportune replicating the *factored* transistor and merging it with *single* ones. ■

$$f = (a.(!b.(!c.!d))+e+(b.(d+c)))) \quad (17)$$

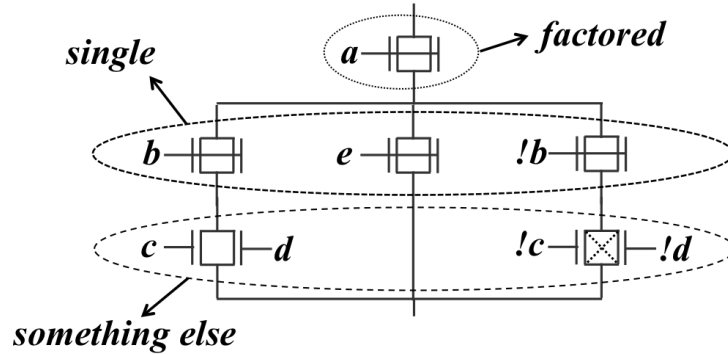


Figure 32 - Promising arrangement to be defactorized, by replicating the transistor controlled by the literal **a**.

#### 4.2.4 Integrating the Three Observations in a Single Method

When the purpose is to generate reduced transistor networks by applying defactorization techniques, we suggest our two alternatives for merging transistors, as illustrated in Fig. 28(c) and Fig. 30(c). In general, when there are more than four transistors involved in the arrangement to be defactorized, our alternatives will contribute to save more transistors than the alternatives proposed in (ROSTAMI e MOHANRAM, 2011). Wherein, when there are four or less transistors in the arrangement to be defactorized, our alternatives will result in the same transistor count obtained by the defactorizations proposed in (ROSTAMI e MOHANRAM, 2011). Moreover, we will improve the solution by applying series defactorizations only when its really contributes to minimize the number of FinFET in the network.

Considering the optimizations proposed in the previews three examples, we have designed a method which starts from a factored expression and, when possible, performs series and parallel merging, *i.e.*, as proposed in Fig. 28(c) and Fig. 30(c), as well as defactorizations, *i.e.*, as defined by the patterns presented in Fig. 31. The proposed defactorization approach is divided in four simple steps:

- (A) The first one is translating the input factored form to a logic tree;
- (B) The second one aims to merge nodes of the logic tree, which represent the same operator and are connected one to the other. It is done in order to find pairs of literals to be merged in IG FinFETs. By doing that, we produce the results demonstrated in Fig. 28(c) and Fig. 30(c);
- (C) The third one consists in traverse the logic tree looking for the patterns presented in Fig. 31. If one of such patterns is found, so the algorithm performs a defactorization by replicating the factored literal in the tree. It is important to mention that such patterns may be found more than once and in different branches of the same logic tree;
- (D) Finally, the transistor network is obtained by mapping single literals to SG FinFETs and by mapping pairs of literals to IG FinFETs. If a pair of literals is connected to an *AND* node it results in an IG FinFET *series*. Otherwise, if a pair of literals is connected to an *OR* node it results in an IG FinFET *parallel*.

**Examples 7:** Let us consider as input of the proposed method the factored form represented by Equation (17), which represents the network illustrated in Fig. 32. In the first step the algorithm will translate the expression to the logic tree illustrated in Fig. 33(a). Thus, by merging nodes with the same operator we obtain the logic tree illustrated in Fig. 33(b). Notice that in this step some pairs of literals are already grouped. Now, evaluating the logic tree illustrated in Fig. 33(b) it is possible to identify a factored literal *a* and exactly one single literal in each branch of the tree *e. g.*, literals *b*, *e* and *!b*. It matches with the second pattern, illustrated in Fig. 31(b). Thus, the factored literal *a* is replicated in each branch, resulting in the logic tree illustrated in Fig. 33(c). In order to merge pairs of literals in IG FinFETs, the literals not yet grouped are arranged in pairs as presented in Fig. 33(c). Finally, the optimized transistor network is obtained by mapping each leaf of the logic tree to a FinFET in the network, as illustrated in Fig. 34. ■

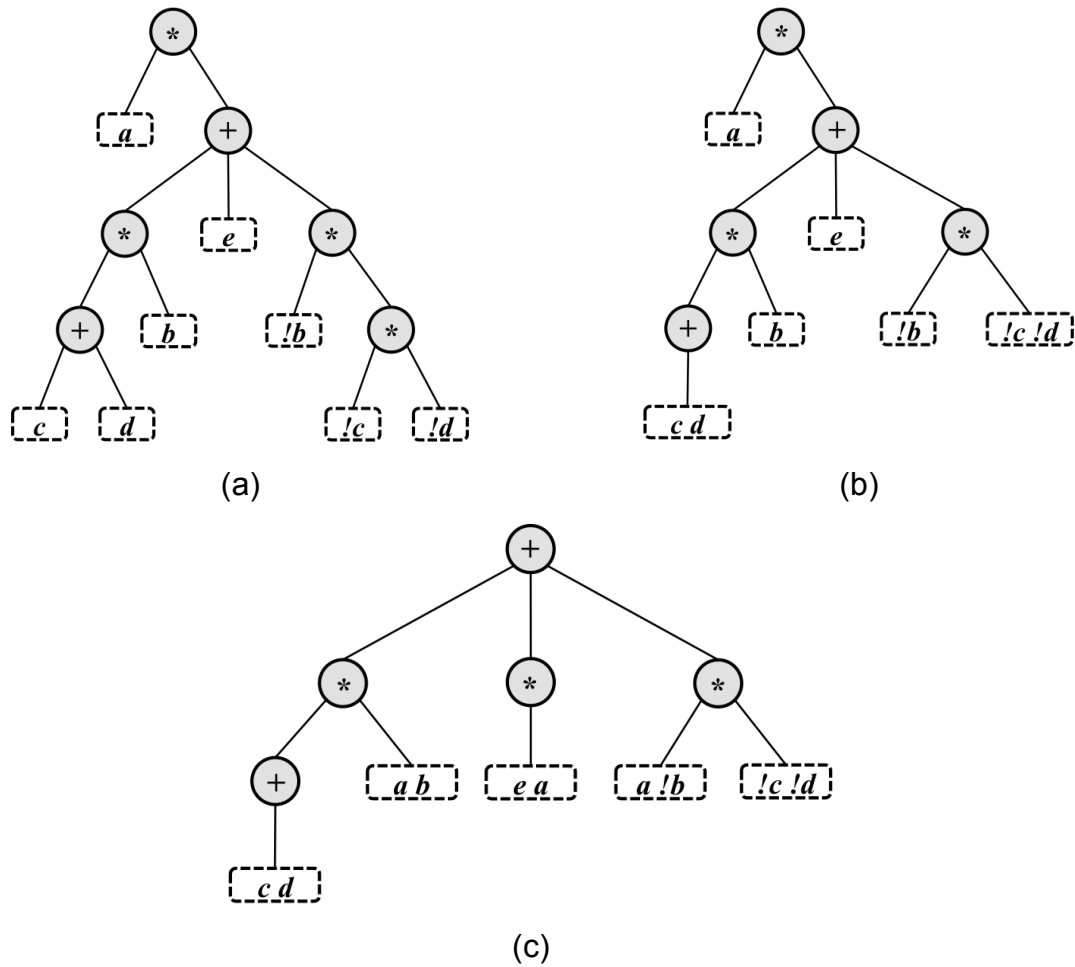


Figure 33 - Logic tree derived from Equation (16) in (a), tree obtained by merging nodes in (b), and resultant tree after performing the defactorization of literal  $a$  in (c).

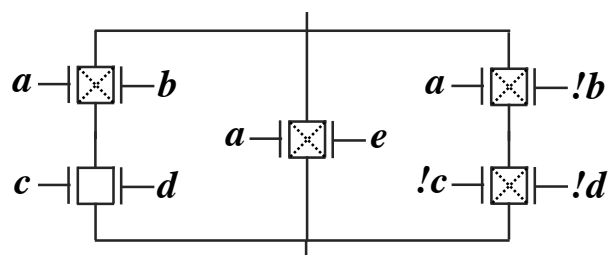


Figure 34 - Optimized transistor network obtained by defactoring the factored form described in Equation (16).

## 5 EXPERIMENTAL RESULTS

This section presents the results obtained by each method proposed in this work. Firstly, an evaluation for each method is presented separately. In the sequence, a brief discussion is presented in order to point out when it is opportune to use the proposed graph-based method or the proposed defactorization. To perform the experiments we have applied the proposed methods over different classes of Boolean functions and also over benchmark circuits. All experiments were executed in a platform with an Intel Core i5 processor at 2.8GHz with 4GB of RAM. The sets of functions used herein are detailed as follow:

- (1) 4-input P-class: 3,982 functions;
- (2) 5-input NPN-class: 616,126 functions;
- (3) 6-input Read-Polarity-Once class: 59,963 functions;
- (4) 6-input Read-Once class: 456,772 functions;
- (5) ACM/SIGDA benchmark circuits (IWLS, 2005).

For each set above we have compared the solutions generated by the proposed methods with the solutions obtained by conventional factorization (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO, MARTINS, *et al.*, 2013) and a graph-based method (POSSANI, CALLEGARO, *et al.*, 2013). We have compared the proposed methods with the conventional ones in order to demonstrate that these methods are not the best alternative for IG FinFETs. Moreover, we do not found in the literature automated methods able to explore the potential of the IG FinFET during the transistor network generation.

It is worth to remember that all conventional methods used in the following experiments were designed for factoring and generating networks based in single-gate devices. Thus, we have generated conventional transistor networks using such methods and, when possible, merged series and parallel transistors in

IG FinFETs. This way, it is possible to perform a fair comparison of the proposed methods with these other techniques.

### 5.1.1 Results for the Graph-based Approach

The first experiment was performed over the set (2), 5-inputs NPN-class (negation-permutation-negation). This set was applied to the proposed method and also to the most recent published factorization (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) and graph-based methods (POSSANI, CALLEGARO, *et al.*, 2013). Table 1 presents the total number of transistors obtained by each evaluated method, when considering the 5-inputs NPN-class Boolean functions. As can be seen, the proposed method was able to minimize more transistors than the other techniques.

Table 1 - Total number of transistors for the set of 5-inputs NPN-class Boolean functions.

	MARTINS et al., 2010	POSSANI et al., 2013	Graph-Based Method
<i>Total Number of Transistors</i>	7,350,852	8,037,628	6,843,272
<i>Run Time</i>	6.35 hours	441.30 seconds	371.21 seconds

For this experiment two histograms are presented in Fig. 35 and Fig. 36, showing gains, losses and draws obtained by the proposed method when compared to the (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) and the (POSSANI, CALLEGARO, *et al.*, 2013), respectively. In the graphics, negative values represent losses and positive values represent gains.

As can be seen in Fig. 35, for a small subset (15.90%) of functions, the proposed method presents an overhead of transistors (losses) if comparing to (MARTINS, DA ROSA JUNIOR, *et al.*, 2010). In general, these losses are concentrated between 1 and 3 transistors. On the other hand, the proposed method presents gains for 60.74% of functions. In the most of cases, the gains are concentrated between 1 and 4 transistors. For 23.36% of functions the proposed method and the method presented in (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) achieve the same transistor count.

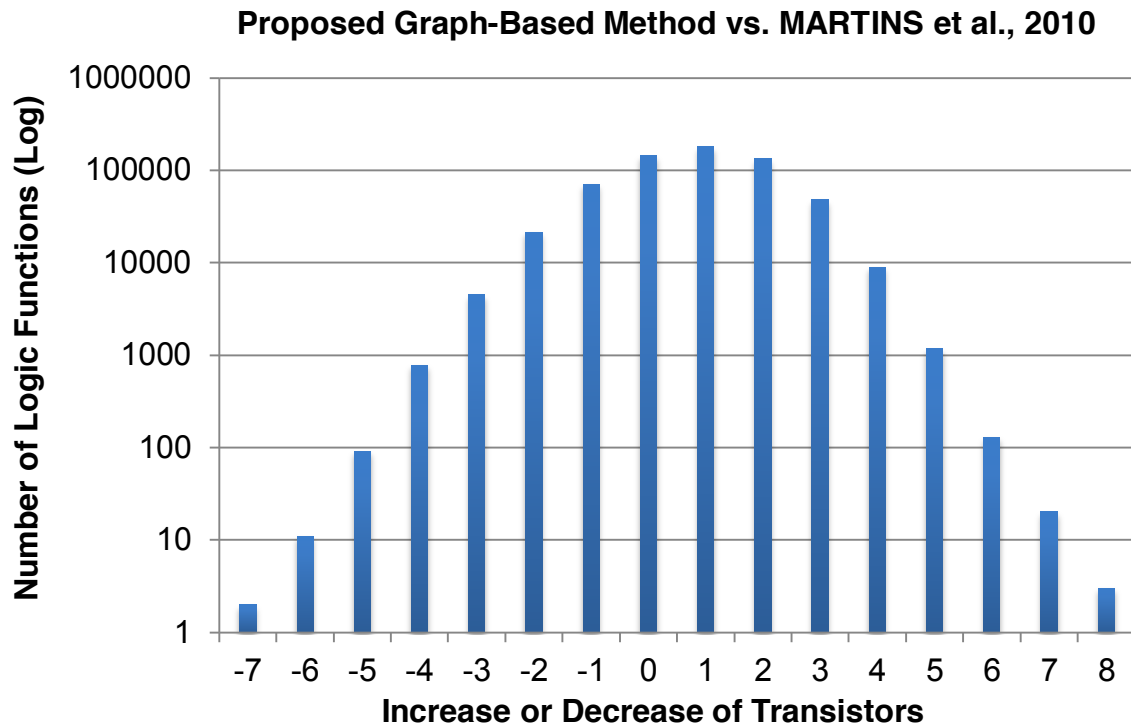


Figure 35 - Transistors increase or decrease obtained by the proposed method when compared to (MARTINS, DA ROSA JUNIOR, *et al.*, 2010), for the set of 5-input NPN-class functions.

As can be seen in Fig. 36, for a small subset (2.58%) of functions the proposed method presents losses, if comparing to (POSSANI, CALLEGARO, *et al.*, 2013). Basically, these losses represent an overhead of 1 and 2 transistors. On the other hand, for 87.12% of functions the proposed method was able to decrease transistors. In general, these gains are concentrate between 1 and 6 transistors. The draw between the two methods represents 10.30% of functions. The main reason for the bad results of the method proposed in (POSSANI, CALLEGARO, *et al.*, 2013) is that such method tends to find NSP networks, which may not be the best alternative for merging transistors in IG FinFETs.

The methods (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) and (POSSANI, CALLEGARO, *et al.*, 2013) are the most promising for generating optimized transistor networks for single-gate devices. Thus, these results reinforce the existence of a new paradigm and demonstrate how far the conventional methods can be of the optimized solution for double-gate devices.



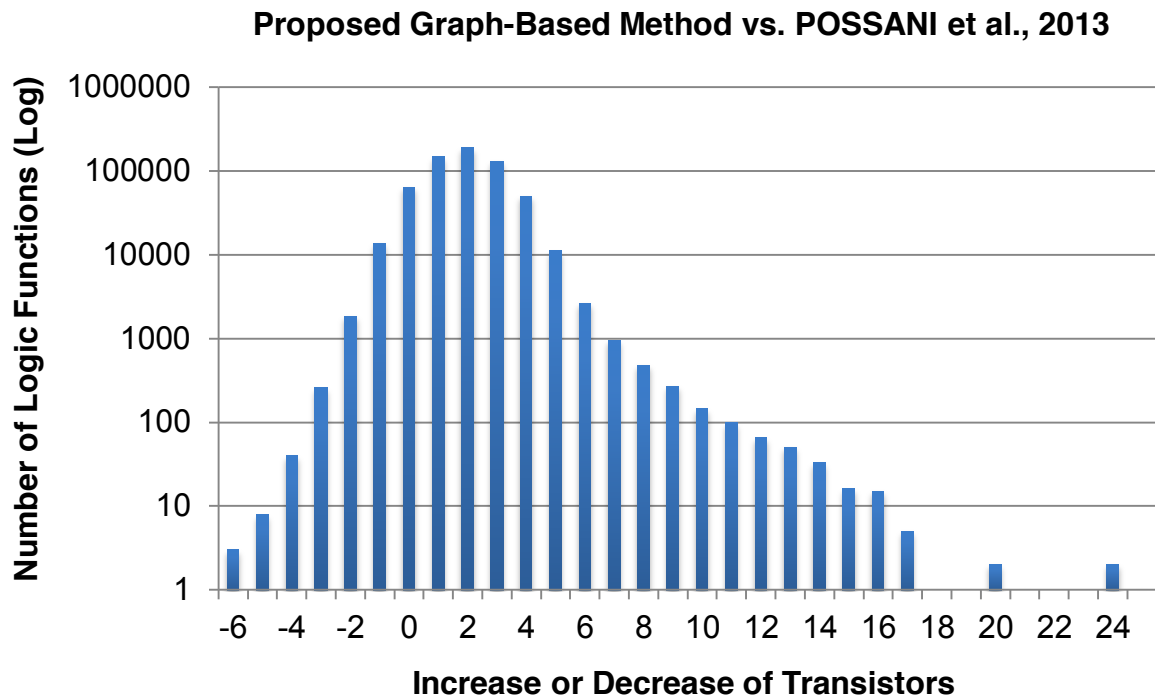


Figure 36 - Transistors increase or decrease obtained by the proposed method when compared to (POSSANI, CALLEGARO, *et al.*, 2013), for the set of 5-input NPN-class functions.

The second experiment was performed over the set (4), 6-input Read-Once class. This set was applied to the proposed method and to a method dedicated to read-once functions (GOLUMBIC, MINTZ e ROTICS, 2008). Table 2 presents the total number of transistors obtained by each method, when considering this set of functions. Again, the proposed method was able to minimize more transistors than the factorization dedicated to read-once functions. In other words, the method (GOLUMBIC, MINTZ e ROTICS, 2008) always delivers the exact (minimum) solution for this class of functions when considering single-gate devices. However, this method cannot ensure the optimality for devices like IG FinFETs.

Table 2 - Total number of transistors for the set of 6-inputs read-once functions.

	Dedicated to Read-Once GOLUMBIC et al., 2008	Graph-Based Method
<i>Total Number of Transistors</i>	1,760,260	1,714,131
<i>Run Time</i>	56.48 seconds	36.23 seconds

Fig. 37 presents a histogram comparing the proposed method with the factorization proposed in (GOLUMBIC, MINTZ e ROTICS, 2008). Basically, the proposed method presents losses for 5.40% and presents gains for 16.82% of the functions. In general, the losses and gains represent increase or decrease of 1 transistor, respectively. For 77.78% of the functions both methods achieve the same transistor count. This large occurrence of draws may indicate that in the most of cases the exact factored form of a read-once function is the optimum solution for IG FinFET.

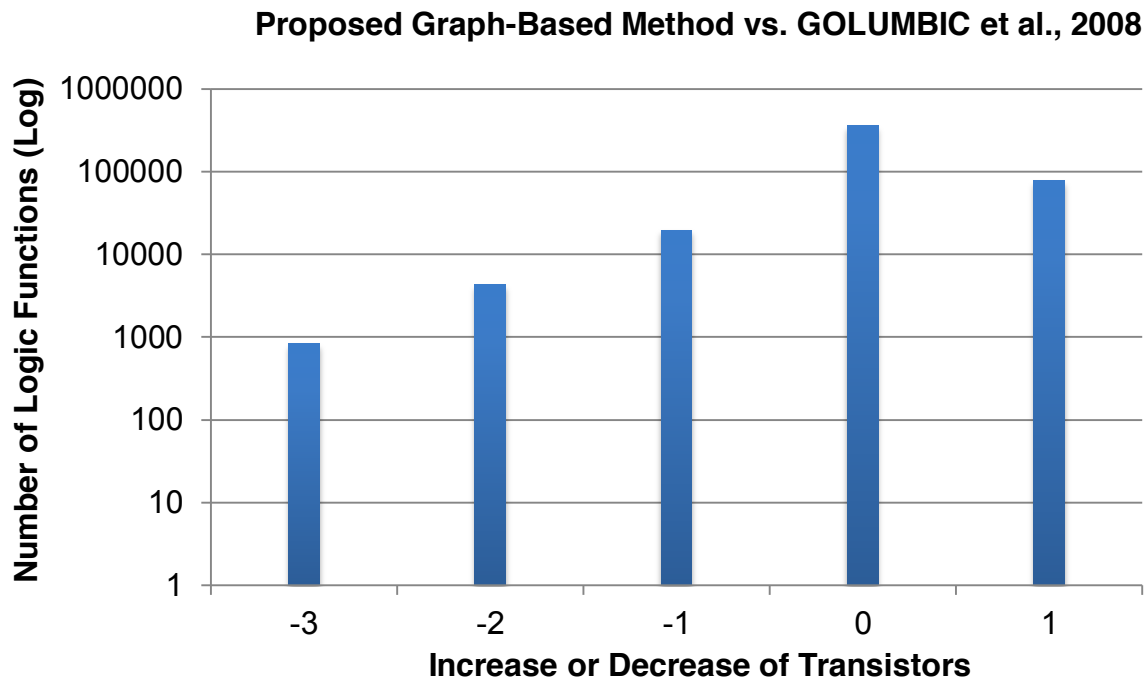


Figure 37 - Transistors increase or decrease obtained by the proposed method when compared to (GOLUMBIC, MINTZ e ROTICS, 2008), for the set of 6-input Read-Once functions.

Another experiment was performed over the set (5), ACM/SIGDA benchmark circuits (IWLS, 2005). Functions with 4, 6, and 8 inputs were extracted from the circuits by performing a cover using K-cuts with maximum size of 4, 6, and 8, respectively. We have used the command “*resyn2 ; if -K k*” (where  $k$  means the maximum number of inputs of each K-cut) from ABC tool (ABC, 2005) to perform the covering. In this experiment, logic gates were generated for all Boolean functions extracted from the circuits listed in the Table 3. The results obtained by each method, considering three different configuration of  $k$ , are presented in Table 3. Consider the following caption to indicate the method in the Table 3: [1] (MARTINS, DA ROSA JUNIOR, *et al.*, 2010); [2] (POSSANI, CALLEGARO, *et al.*, 2013); [3] Proposed Graph-based Method.

As can be seen, in general, the proposed approach was able to achieve smaller transistor network to implement the circuits. The transistor count reduction is not so expressive when considering circuits composed by 4-input functions, since other approaches (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (POSSANI, CALLEGARO, *et al.*, 2013) can also achieve networks closer to the minimum solution. When considering circuits composed of 6 and 8-input functions, the proposed method was able to decrease more transistors. Also, it is important to notice that for all configuration of  $k$ , the proposed method was able to synthesize the circuits in a short time when compared to the conventional methods. This experiment demonstrates the feasibility of using the proposed method in real design when considering circuits with random functions, even when scaling the optimization problem due to the use of functions with more than 4 inputs.

Indeed, the proposed graph-based method is an alternative way to generate IG FinFET transistor networks, being able to decrease the transistor count by efficiently exploring double-gate devices. It is worth to mention that the proposed method presented in this work only explores some characteristics that can lead to a reduced transistor network implementation. We have identified that, in some few cases, the proposed method delivers bad solutions with an overhead of transistors when compared to the traditional techniques. We believe it is related to the large number of possible combinations to merge two transistors in IG FinFETs *series* and *parallel*. Thus, it becomes difficult to avoid local minima solutions. These cases will be carefully investigated in the future in order to improve the proposed technique.

Table 3 - Results for decomposition of circuits into K-cuts, with k=4, k=6 and K=8.

	<b>K = 4</b>			<b>K = 6</b>			<b>K = 8</b>		
<b>Circuit</b>	<b>[1]</b>	<b>[2]</b>	<b>[3]</b>	<b>[1]</b>	<b>[2]</b>	<b>[3]</b>	<b>[1]</b>	<b>[2]</b>	<b>[3]</b>
<b><i>apex6</i></b>	2,200	2,200	2,192	2,351	2,346	2,271	2,477	2,486	2,398
<b><i>apex7</i></b>	736	738	735	748	746	735	760	760	759
<b><i>c8</i></b>	324	324	323	317	318	311	324	326	318
<b><i>cm152a</i></b>	48	48	48	58	58	58	56	56	54
<b><i>cm162a</i></b>	132	132	132	132	134	130	119	120	117
<b><i>cm163a</i></b>	130	130	128	110	110	108	126	132	129
<b><i>cordic</i></b>	152	159	150	159	144	141	173	146	149
<b><i>cmb</i></b>	128	128	128	120	120	120	116	116	116
<b><i>count</i></b>	410	400	400	380	380	373	434	442	433
<b><i>cu</i></b>	154	155	154	143	142	141	135	136	134
<b><i>dalu</i></b>	3,880	4,008	3,856	4,518	4,254	4,189	5,049	4,644	4,651
<b><i>decod</i></b>	154	154	154	192	192	192	192	192	192
<b><i>frg1</i></b>	348	348	348	295	292	288	312	306	294
<b><i>i2</i></b>	446	446	446	422	422	420	362	362	362
<b><i>i5</i></b>	832	832	832	998	998	963	1,068	1,068	1,017
<b><i>i7</i></b>	2,134	2,130	2,130	1,724	1,600	1,596	1,724	1,600	1,596
<b><i>pair</i></b>	4,184	4,198	4,166	4,390	4,386	4,311	4,720	4,690	4,658
<b><i>pcle</i></b>	302	234	232	216	296	213	243	344	244
<b><i>vda</i></b>	2,228	2,228	2,228	2,823	2,826	2,815	3,945	3,896	3,864
<b><i>x2</i></b>	132	132	132	149	148	147	115	114	114
<b><i>x3</i></b>	1,932	1,932	1,921	2,313	2,322	2,257	2,379	2,398	2,340
<b><i>Total # of Transistors</i></b>	<b>20,986</b>	<b>21,056</b>	<b>20,835</b>	<b>22,558</b>	<b>22,234</b>	<b>21,779</b>	<b>24,829</b>	<b>24,334</b>	<b>23,939</b>
<b><i>Run Time (Seconds)</i></b>	<b>0.44</b>	<b>2.26</b>	<b>1.79</b>	<b>109.92</b>	<b>5.56</b>	<b>2.87</b>	<b>630,939.82</b>	<b>48.40</b>	<b>12.60</b>

### 5.1.2 Results for the Defactorization Approach

In order to evaluate the proposed defactorization method we have considered the four first sets of Boolean functions described before, (1), (2), (3), and (4). We have generated factored forms for each function of these four sets using three different factorization methods available in the literature (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO, MARTINS, *et al.*, 2013). Such methods were chosen because they are considered the state-of-the-art in factorization, respecting the classes of functions that each one can manipulate. Thus, it is important to mention that the method (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) can be applied to Boolean function of general classes, while the method (CALLEGARO, MARTINS, *et al.*, 2013) can be applied to functions from the Read-Once and to the Read-Polarity-Once classes, and the method (GOLUMBIC, MINTZ e ROTICS, 2008) is restrict to manipulate function from the Read-Once class.

Considering this scenario, we have performed the following experiments: the method (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) was applied over the first two sets of functions (1) and (2); the method (CALLEGARO, MARTINS, *et al.*, 2013) was applied over the third set of function (3) and the method (GOLUMBIC, MINTZ e ROTICS, 2008) was applied over the fourth set of function (4). In the sequence, we have applied the proposed defactorization method over the factored forms obtained by each method, considering these four sets of functions. The main idea behind these experiments is to observe how the proposed defactorization method can improve the solutions produced by the conventional factorization methods in terms of FinFET transistor count.

The results obtained in these experiments are summarized in the next four tables. In the top of these tables, the first column shows the applied method, the second column shows the set of functions used as input, the third column shows the original transistor count obtained by each method, the fourth column shows the transistor count obtained after performing the proposed defactorization, and the last column shows the reduction from the original transistor count. In the bottom of these tables, the first column presents how many transistors were decreased, the second column presents the occurrence of each reduction and the last column presents this occurrence in percentage.

Table 4 - Improvement obtained by the proposed defactorization method over the results provided by method (MARTINS, DA ROSA JUNIOR, *et al.*, 2010), considering the set of Boolean functions (1).

Method	Set of Functions	FinFET Transistor Count		
		Original	Defactored	Reduction
MARTINS et al., 2010	(1)	21,788	21,764	24
Decrease of Transistors		Occurrence		Percent of Functions
0		3,958		99.40%
1		24		0.60%

Table 5 - Improvement obtained by the proposed defactorization method over the results provided by method (MARTINS, DA ROSA JUNIOR, *et al.*, 2010), considering the set of Boolean functions (2).

Method	Set of Functions	FinFET Transistor Count		
		Original	Defactored	Reduction
MARTINS et al., 2010	(2)	7,350,853	7,335,567	15,286
Decrease of Transistors		Occurrence		Percent of Functions
0		600,955		97.54%
1		15,056		2.44%
2		115		0.02%

Table 6 - Improvement obtained by the proposed defactorization method over the results provided by method (CALLEGARO, MARTINS, *et al.*, 2013) considering the set of Boolean functions (3).

Method	Set of Functions	FinFET Transistor Count		
		Original	Defactored	Reduction
CALLEGARO et al., 2013	(3)	420,226	414,704	5,522
Decrease of Transistors		Occurrence		Percent of Functions
0		54,564		91.00%
1		5,276		8.80%
2		123		0.20%

Table 7 - Improvement obtained by the proposed defactorization method over the results provided by method (GOLUMBIC, MINTZ e ROTICS, 2008), considering the set of Boolean functions (4).

Method	Set of Functions	FinFET Transistor Count		
		Original	Defactored	Reduction
GOLUMBIC et al., 2008	(4)	1,760,260	1,683,460	76,800

Decrease of Transistors	Occurrence	Percent of Functions
0	379,972	83.19%
1	76,800	16.81%

In fact, the results presented in these tables show that conventional factorization methods may not be the optimal solution in some cases. Through the proposed defactorization method it is possible to improve the solutions provided by conventional methods with a very small cost in execution time. For these experiments, the proposed defactorization method was able to defactore all functions of the four sets in a total execution time of 23.6 seconds. In general the obtained reduction is limited to one or two transistors as presented in the tables above. However, it is opportune to use the proposed defactorization, considering their low cost and that it never brings losses for the original factored form. It is worth to mention that the defactorization method depends on the input factored form. Thus, by trying different input expressions it is possible to achieve more optimized networks.

It is important to mention that the original factored forms provided by the methods (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (GOLUMBIC, MINTZ e ROTICS, 2008) (CALLEGARO, MARTINS, *et al.*, 2013) for the sets (1), (3), and (4), respectively, are composed of the minimum number of literals. Just the set (2) was factorized in a heuristic mode using the method (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) due to the high time complexity of the exact mode when the number of variables from the input function increases. Thus, the condition of the sets (1), (3), and (4) contributes to prove that just minimizing the number of literals is not enough to achieve efficient IG FinFET transistor networks.

### 5.1.3 Comparing the Proposed Approaches

As presented before, both the proposed graph-based and the defactorization methods contribute to achieve reduced FinFET transistor networks. However, in some cases it may be more convenient to apply one method instead the other. It depends of the characteristics of the Boolean functions to be implemented. For instance, for the classes of functions that the exact factored form is easy obtained for dedicated methods, it is opportune to generate the exact factored form through such factorization methods and, in the sequence, apply the proposed defactorization. As demonstrated in Table 4, Table 5, Table 6, and Table 7, in general this strategy allows converging for efficient FinFET arrangements.

However, when the exact factorization is unknown or cannot be easily found, it is opportune to apply the proposed graph-based approach instead applying the defactorization. This was demonstrated for the set of 5-input NPN-class, where the results obtained by the graph-base method in Table 1 are better than the results provided by defactoring the solution provided by (MARTINS, DA ROSA JUNIOR, *et al.*, 2010), as presented in Table 5. These two situations mentioned above are summarized in the top rows of the Table 8.

Table 8 - Comparative analysis between the proposed methods.

Situation	Graph-based Method	Defactorization Method
Exact factorization is known		✓
Exact factorization is unknown	✓	
Standard Cells		✓
Full Custom	✓	

Another point to be considered is what kind of arrangement is desired to the circuit design. As mentioned before, the proposed graph-based method is able to achieve NSP arrangements. On the other hand, the solutions obtained by the proposed defactorization method are limited to SP arrangements. It is due to the *AND* and *OR* operators present in the factored forms used as input. These different



kinds of arrangements are directly related to the design constraints adopted during the project.

For instance, SP arrangements are attractive for a *standard cells* design due to the layout regularity. The NSP arrangements are more attractive for a *full custom* design, considering that the methods for automatic layout generation may not work well with NSP topologies. Nevertheless, the NSP arrangements can be used to generate logic gates for a *standard cells* design. It requires an adaptation in the tools for automatic layout generation or requires interactive interventions of the designers to achieve an optimized layout diagram. This comparative analysis between the proposed methods, considering the design constraints, is summarized in the bottom rows of the Table 8. Finally, this work presents two alternative methods to be applied with different purposes, which are able to produce efficient results quickly.

## 6 CONCLUSIONS AND FUTURE WORKS

This work presented an analysis demonstrating that the independent-gate FinFET technology brings advances and new possibilities while introducing a new paradigm in the transistor network design. Through this analysis it was possible to identify that the conventional methods for transistor network generation are not the best alternative to explore the independent gates of the IG FinFETs.

Considering the potential and the gap introduced by the FinFET technology, this work proposes two methods to explore such potential and cover the gap. The first method was based in graph optimizations aiming to find promising arrangements to be merged in IG FinFETs. This method is flexible to generate series-parallel and non-series-parallel transistor arrangements, contributing to achieve reduced transistor networks.

The second method aims to defactore Boolean expressions in order to increase the merging of series and parallel transistors in IG FinFETs. This method is able to improve the solutions provided by conventional factorization methods for an IG FinFET-based design. An interesting point is that the proposed defactorization is fast and lossless. Thus, the final solution will be better or equals to the input factored form, but never will be worst.

The results obtained through the proposed methods also reinforce the existence of a new paradigm, considering that conventional methods of transistor network generation present an overhead in transistor count for all experiments performed in this work. The experiments demonstrate that the proposed methods are practical alternatives to generate optimized IG FinFET transistor networks. However, there is still a wide field to be explored in this context. We consider that the analysis presented in the Chapter 3 is a good point to start new investigations and propose smarter algorithms to explore the independent gates of each FinFET.

As future works, we intend to investigate alternatives to improve the proposed graph-based method, in order to increase the possibilities of merging series and parallel transistors in IG FinFETs. In general, the bad solutions achieved by the proposed graph-based method are related to combinational problems. It is not trivial determine what are the best combination to perform series and parallel merging of transistors. Thus, a possibility for future works is to investigate a smart heuristic to test different combinations to merge pairs of transistors in IG FinFETs. Moreover, we also aim to investigate what is the impact of decreasing the number of internal nodes in the network, instead decreasing only the transistor count. Decrease the internal nodes may be a good strategy to save area in a FinFET-based transistor network.

Finally, as presented in the literature, minimize the number of devices by merging series and parallel transistors in IG FinFETs contribute to save circuit area, power and, in some cases, improve the circuit performance. This work presents some contributions in the logic synthesis step that can drive to such improvements. However, it is important to mention that, although this work and many other works show and discuss such benefits, it is no clear how the microelectronics industry has accepted the possibility to design integrated circuits based on IG FinFETs. Since the FinFET is an emerging technology, there are still some challenges to be considered and solved in the design flow.

## REFERENCES

- ABC, B. L. S. A. V. G. ABC: A System for Sequential Synthesis and Verification, 2005. Disponivel em: <<http://www.eecs.berkeley.edu/~alanmi/abc/>>. Acesso em: Jan 2015.
- AMAT, E. et al. Impact of FinFET Technology Introduction in the 3T1D-DRAM Memory Cell. **IEEE Transactions on Device and Materials Reliability**, v. 13, n. 1, p. 287 - 292, 2013.
- BUTZEN, P. et al. Standby power consumption estimation by interacting leakage current mechanisms in nanoscaled CMOS digital circuits. **Microelectronics Journal**, v. 41, p. 247-255, Jan 2010.
- CAKICI, T. et al. **Independent gate skewed logic in double gate SOI technology**. Proceedings IEEE International SOI Conference. [S.l.]: [s.n.]. 2005. p. 83–84.
- CALLEGARO, V. et al. **Read-Polarity-Once Functions**. 26th Symp. on Integrated Circuits and Systems Design. Curitiba, PR: [s.n.]. 2013. p. 1-6.
- CARUSO, G. Near optimal factorization of Boolean functions. **IEEE Trans. Comput. Aided Design Integr. Circuits Syst.**, v. 10, p. 1072–1078, Aug. 1991. ISSN 8.
- CHANG, L. et al. Extremely scaled silicon nano-CMOS devices. **Proceedings IEEE**, v. 91, n. 11, p. 1860–1873, Nov 2003.
- CHIANG, M. et al. Novel High-Density Low-Power Logic Circuit Techniques Using DG Devices. **IEEE TRANSACTIONS ON ELECTRON DEVICES**, v. 52, n. 10, p. 2339-2342, Oct 2005.
- CHIANG, M. et al. High-Density Reduced-Stack Logic Circuit Techniques Using Independent-Gate Controlled Double-Gate Devices. **IEEE TRANSACTIONS ON ELECTRON DEVICES**, v. 53, n. 9, p. 2370- 2377, Sep 2006.
- CHOI, Y. et al. **30 nm ultra-thin-body SOI MOSFET with selectively deposited Ge raised S/D**. 58th DRC Device Research Conference. Denver, CO: [s.n.]. 2000. p. 23 - 24.
- COLINGE, J. **FinFETs and Other Multi-Gate Transistors**. [S.l.]: Springer US, 2008.

DA ROSA JUNIOR, L. et al. **Fast Disjoint Transistor Networks from BDDs**. 19th ACM Symposium on Integrated Circuits and Systems Design (SBCCI). Ouro Preto: [s.n.]. 2006. p. 137-142.

DA ROSA JUNIOR, L. et al. **A Comparative Study of CMOS Gates with Minimum Transistor Stacks**. 20th ACM Symposium on Integrated Circuits and Systems Design (SBCCI). Rio de Janeiro: [s.n.]. 2007. p. 93-98.

DATTA, A. et al. Modeling and Circuit Synthesis for Independently Controlled Double Gate FinFET Devices. **IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 26, n. 11, p. 1957-1966, Nov 2007.

FRANK, E. et al. Device scaling limits of Si MOSFETs and their application dependencies. **Proceedings of the IEEE**, v. 89, n. 3, p. 259-288, Mar 2001.

GOLUMBIC, M.; MINTZ, A.; ROTICS, U. **Factoring and recognition of read-once functions using cographs and normality**. Proc. of the 34th Annual Design Automation Conference. [S.l.]: [s.n.]. 2001. p. 109-14.

GOLUMBIC, M.; MINTZ, A.; ROTICS, U. An improvement on the complexity of factoring read-once Boolean functions. **Discrete Appl. Math**, v. 156, p. 1633-1636, 2008. ISSN 10.

GSS, G. S. S. Simulation analysis of the Intel 22nm FinFET. **Gold Standard Simulations**, 9 May 2012. Disponível em: <[http://www.goldstandardsimulations.com/news/blog\\_search/simulation-analysis-of-the-intel-22nm-finfet/](http://www.goldstandardsimulations.com/news/blog_search/simulation-analysis-of-the-intel-22nm-finfet/)>. Acesso em: 10 Jan 2015.

HUANG, X. et al. **Sub 50-nm FinFET: PMOS**. Technical Digest. International Electron Devices Meeting. Washington, DC: [s.n.]. Dec 1999. p. 67 - 70.

ITRS. International Technology Roadmap for Semiconductors. **International Technology Roadmap for Semiconductors**, 2001. Disponível em: <[www.itrs.net](http://www.itrs.net)>. Acesso em: Jan 2014.

IWLS, B. Benchmarks, IWLS. **Benchmarks, IWLS**, 2005. Disponível em: <<http://www.iwls.org>>. Acesso em: Jan 2015.

KAGARIS, D.; HANIOTAKIS, T. A Methodology for Transistor-Efficient Supergate Design. **IEEE Trans. On Very Large Scale Integration (VLSI) Systems**, p. 488-492, 2007.

LIU, Y. et al. Cointegration of High-Performance Tied-Gate Three-Terminal FinFETs and Variable Threshold-Voltage Independent-Gate Four-Terminal FinFETs With Asymmetric Gate-Oxide Thicknesses. **IEEE Electron Device Letters**, v. 28, n. 6, p. 517 - 519, June 2007.

LIU, Z.; TAWFIK, S. A.; KURSUN, V. **Statistical Data Stability and Leakage Evaluation of FinFET SRAM Cells with Dynamic Threshold Voltage Tuning under Process Parameter Fluctuations**. 9th International Symposium on Quality Electronic Design (ISQED). [S.l.]: [s.n.]. 2008. p. 305-310.

MARKOV, I. L. Limits on fundamental limits to computation. **Nature**, v. 512, p. 147–154, August 2014.

MARTINS, M. et al. **Boolean Factoring with Multi-Objective Goals**. IEEE Int. Conf. on Computer Design. Amsterdam: [s.n.]. 2010. p. 229-234.

MINTZ, A.; GOLUMBIC, M. Factoring Boolean functions using graph partitioning. **Discrete Applied Mathematics**, v. 149, p. 131-53, 2005.

MISHRA, P.; MUTTREJA, A.; JHA, N. FinFET Circuit Design. **Springer Science+Business Media, LLC**, p. 23-54, 2011.

MOORE, G. Cramming more components onto integrated circuits. **Electronics Magazine**, v. 38, n. 8, p. 114–117, Apr 1965.

MUTTREJA, A.; AGARWAL, N.; JHA, N. **CMOS Logic Design with Independent-gate FinFETs**. 25th International Conference on Computer Design. Lake Tahoe, CA: [s.n.]. 2007. p. 560 - 567.

NOWAK, E. J. et al. Turning silicon on its edge. **IEEE Circuits Devices Mag.**, v. 20, n. 1, p. 20–31, Jan./Feb. 2004.

POLI, R.; RIBAS, R.; REIS, A. **Unified Theory to Build Cell-Level Transistor Networks from BDDs**. Symposium on Integrated Circuits and System Design (SBCCI). São Paulo: [s.n.]. 2003. p. 199–204.

POSSANI, V. et al. Optimizing Transistor Networks Using a Graph-Based Technique. **Analog Integrated Circuits and Signal Processing**, v. 73, p. 841-850, May 2012.

POSSANI, V. et al. **Improving the methodology to build non-series-parallel transistor arrangements**. 26th Symposium on Integrated Circuits and Systems Design (SBCCI). Curitiba, PR: [s.n.]. 2013. p. 1-6.

POSSANI, V. N. et al. **Exploring Independent Gates in FinFET-Based Transistor Network Generation**. 27th Symposium on Integrated Circuits and Systems Design - SBCCI '14. Aracaju, SE: ACM. 2014. p. 1-6.

POSSANI, V. N. et al. **New challenges on Independent Gate FinFET Transistor Network Generation**. 23rd International Workshop on Logic and Synthesis, 2014. San Fransisco, CA: [s.n.]. 2014.

ROSTAMI, M.; MOHANRAM, K. Dual-Vth Independent-Gate FinFETs for Low Power Logic Circuits. **IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 30, n. 3, p. 337- 349, Mar 2011.

ROY, K. et al. **Double-gate SOI devices for low-power and high-performance applications**. Proc. IEEE Int. Conf. on CAD (ICCAD 2005). [S.I.]: [s.n.]. 2005. p. 217–224.

SENTOVICH, E. **SIS: A system for sequential circuit synthesis**. Technical Report No. UCB/ERL M92/41, EECS Department, University of California. Berkeley. 1992.

SHANNON, C. A symbolic Analysis of Relay and Switching Circuits. **Transactions American Institute of Electrical Engineers**, New York, v. 57, p. 38-80, May 1938.

SPECTRUM, I. IEEE SPECTRUM. **IEEE SPECTRUM**, May 2011. Disponivel em: <[spectrum.ieee.org/semiconductors/design/the-origins-of-intels-new-transistor-and-its-future](http://spectrum.ieee.org/semiconductors/design/the-origins-of-intels-new-transistor-and-its-future)>. Acesso em: Jan 2014.

SYNOPSYS. FinFET: The Promises and the Challenges. **Synopsys Insight Newsletter**, 2012. Disponivel em: <[www.synopsys.com/COMPANY/PUBLICATIONS/SYNOPSYSINSIGHT/Pages/Art2-finfet-challenges-ip-IssQ3-12.aspx](http://www.synopsys.com/COMPANY/PUBLICATIONS/SYNOPSYSINSIGHT/Pages/Art2-finfet-challenges-ip-IssQ3-12.aspx)>. Acesso em: Jan 2015.

TAUR, Y.; NING, T. **Fundamentals of Modern VLSI Devices**. New York: Cambridge University Press, 1998.

TAWFIK, S. A.; KURSUN, V. **Portfolio of FinFET memories**: Innovative techniques for an emerging technology. International SoC Design Conference (ISOC). [S.l.]: [s.n.]. 2008. p. I-101 - I-104.

WANG, M. Independent-Gate FinFET Circuit Design Methodology. **IAENG International Journal of Computer Science**, v. 37, n. 1, p. 50, Feb 2010.

ZHU, J.; ABD-EL-BARR, M. On the optimization of MOS circuits. **IEEE Trans. on Circuits and Systems: Fundamental Theory and Applications, Theory Appl.**, v. 40, p. 412–422, 1993. ISSN 6.

## APPENDIX A - PUBLICATIONS

### Published Papers:

POSSANI, VINICIUS N. ; REIS, ANDRE I. ; RIBAS, RENATO P. ; MARQUES, FELIPE S. ; DA ROSA JUNIOR, LEOMAR S. **New challenges on Independent Gate FinFET Transistor Network Generation**. In: 23rd International Workshop on Logic and Synthesis – IWLS, 2014, San Fransisco, CA.

POSSANI, VINICIUS N. ; REIS, ANDRÉ I. ; RIBAS, RENATO P. ; MARQUES, FELIPE S. ; DA ROSA JUNIOR, LEOMAR S. **Exploring Independent Gates in FinFET-Based Transistor Network Generation**. In: 27th Symposium on Integrated Circuits and Systems Design – SBCCI, 2014. Aracaju, p. 1-6.

POSSANI, VINICIUS N. ; MARQUES, FELIPE S. ; DA ROSA JUNIOR, LEOMAR S. ; CALLEGARO, VINICIUS ; REIS, ANDRE I. ; RIBAS, RENATO P. **Transistor-level optimization of CMOS complex gates**. In: 4th IEEE Latin American Symposium on Circuits and Systems – LASCAS, 2013, Cusco. p. 1.

POSSANI, VINICIUS NEVES ; CALLEGARO, VINICIUS ; REIS, ANDRÉ INÁCIO ; RIBAS, RENATO PEREZ ; MARQUES, FELIPE DE SOUZA ; DA ROSA JUNIOR, LEOMAR SOARES. **Efficient transistor-level design of CMOS gates**. In: 23rd ACM international conference on Great lakes symposium on VLSI – GLSVLSI, 2013. Paris. p. 191.



POSSANI, V. N. ; CALLEGARO, V. ; REIS, A. I. ; RIBAS, R. P. ; MARQUES, F. S. ; DA ROSA JUNIOR, L. S. **A New Algorithm to Implement Combinational Logic Cells with Reduced Number of Switches.** In: 28th South Symposium on Microelectronics – SIM, 2013, Porto Alegre.

POSSANI, VINICIUS N. ; CALLEGARO, VINICIUS ; REIS, ANDRE I. ; RIBAS, RENATO P. ; MARQUES, FELIPE S. ; DA ROSA JUNIOR, LEOMAR S. **A Structural Algorithm to Minimize Switch Count in CMOS Logic Gates.** In: 22th International Workshop on Logic and Synthesis – IWLS, 2013, Austin, TX.

POSSANI, VINICIUS N. ; CALLEGARO, VINICIUS ; REIS, ANDRE I. ; RIBAS, RENATO P. ; MARQUES, FELIPE S. ; DA ROSA, LEOMAR S. **Improving the methodology to build non-series-parallel transistor arrangements.** In: 26th Symposium on Integrated Circuits and Systems Design – SBCCI, 2013, Curitiba. p. 1-6.

#### **Accepted Paper:**

POSSANI, VINICIUS N. ; CALLEGARO, VINICIUS ; REIS, ANDRE I. ; RIBAS, RENATO P. ; MARQUES, FELIPE S. ; DA ROSA, LEOMAR S. **Graph-Based Transistor Network Generation Method for Supergate Design.** IEEE Transactions on Very Large Scale Integration Systems, 2015.

- ✓ First Submission in March 19, 2014
- ✓ Accepted with major reviews in November 02, 2014
- ✓ Resubmitted in December 18, 2014
- ✓ Acceptance notification in February 03, 2015