

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JEFERSON JOSÉ BAQUETA

Evaluation of using MIGFET Devices in Digital Integrated Circuit Design

Dissertation presented in partial fulfillment of the
requirements for the degree of Master of Computer
Science

Prof. Dr. Renato Perez Ribas
Advisor.

Porto Alegre, March 2017

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Baqueta, Jeferson José

Evaluation of using MIGFET Devices in Digital Integrated Circuit Design / Jeferson José Baqueta. – 2017.

113 f.:il.

Avisor: Renato Perez Ribas;

Dissertation (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2017.

1. Digital Circuits. 2. Emerging Technologies 3. MIGFET 4. Adders 5. Cell libraries 6. Nanotechnology I. Ribas, Renato Perez. III. Evaluation of using MIGFET Devices in Digital Integrated Circuit Design.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Rui Vicente Oppermann

Vice-Reitor: Prof. Jane Fraga Tutikian

Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves

Diretor do Instituto de Informática: Prof. Carla Maria Dal Sasso Freitas

Coordenador do PPGC: João Luiz Dihl Comba

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

ACKNOWLEDGEMENTS

Gostaria de agradecer primeiramente a minha mãe, Roseni, e ao meu pai Anito, por todo o apoio que recebi durante esses dois anos de mestrado. Cheguei até aqui devido ao incentivo que recebi de vocês.

Agradeço a minha namorada Aline que esteve ao meu lado durante todo o mestrado, agradeço pelo companheirismo, compreensão e principalmente por estar sempre disposta a me ouvir e me incentivar nas horas difíceis.

Quero agradecer também a todos os colegas do Logics pela paciência e por toda a ajuda que me ofereceram durante esses dois últimos anos. Irei lembrar de cada um e levarei comigo todos os conhecimentos recebidos de vocês.

Gostaria de agradecer ao meu orientador o Professor Renato Perez Ribas pela paciência, compreensão e amizade, o que com certeza fez toda a diferença para o desenvolvimento desse trabalho.

E finalmente, agradeço a todos que diretamente ou indiretamente contribuíram para mais essa conquista.

Avaliação do Uso de Dispositivos MIGFET no Projeto de Circuitos Integrados Digitais

RESUMO

A diminuição das dimensões do transistor MOS tem sido a principal estratégia adotada para alcançar otimizações de desempenho na fabricação de circuitos integrados. Contudo, reduzir as dimensões dos transistores tem se tornado uma tarefa cada vez mais difícil de ser alcançada. Nesse contexto, vários esforços estão sendo feitos para encontrar dispositivos alternativos que permitam futuros avanços em relação à capacidade computacional. Entre as mais promissoras tecnologias emergentes estão os transistores de efeito de campo com múltiplos e independentes *gates* (MIGFETs). MIGFETs são dispositivos controlados por mais que um terminal de controle permitindo que funções Booleanas com mais de uma variável sejam implementadas por um único dispositivo. Redes de chaves construídas com dispositivos MIGFET tendem a ser mais compactas do que as redes de chaves tradicionais. No entanto existe um compromisso em relação a redução no número de chaves, devido à maior capacidade lógica, e um maior tamanho e pior desempenho do dispositivo. Neste trabalho, pretendemos explorar tal balanceamento no sentido de avaliar os impactos do uso de MIGFETs na construção de circuitos integrados digitais. Dessa forma, alguns critérios de avaliação são apresentados no sentido de analisar área e atraso de circuitos construídos a partir de dispositivos MIGFET, onde cada transistor é representado por um modelo RC. Em particular, tal avaliação de área e desempenho é aplicada no projeto de circuitos somadores binários específicos (metodologia *full-custom*). Além do mais, bibliotecas de células construídas a partir de dispositivos MIGFET são utilizadas na síntese automática de circuitos de referência através da metodologia *standard-cell*. Através dos experimentos, é possível ter-se uma ideia, mesmo que inicial e pessimista, do quanto o layout de um dado MIGFET pode ser maior do que um *single-gate* FinFET e ainda apresentar redução na área do circuito devido à compactação lógica.

Palavras-chave: Circuitos digitais, tecnologias emergentes, MIGFET, somadores, bibliotecas de células, nanotecnologia.

Evaluation of using MIGFET Devices in Digital Integrated Circuit Design

ABSTRACT

The scaling of MOS transistor has been the main manufacturing strategy for improving integrated circuit (IC) performance. However, as the device dimensions shrink, the scaling becomes harder to be achieved. In this context, much effort has been done in order to develop alternative devices that may allow further progress in computation capability. Among the promising emerging technologies is the multiple independent-gate field effect transistors (MIGFETs). MIGFETs are switch-based devices, which allow more logic capability in a single device. In general, switch networks built through MIGFET devices tend to be more compact than the traditional switch networks. However, there is a tradeoff between the number of logic switches merged and the area and performance of a given MIGFET. Thus, we aim to explore such a tradeoff in order to evaluate the MIGFET impacts in the building digital integrated circuits. To achieve this goal, in this work, we present an area and performance evaluation based on digital circuit built using MIGFET devices, where each MIGFET is represented through RC modelling. In particular, such an evaluation is applied on full-custom design of binary adder circuits and on standard-cell design flow targeting in a set of benchmark circuits. Through the experiments, it is possible have an insight, even superficial and pessimist, about how big can be the layout of a given MIGFET than the single-gate FinFET and still show a reduction in the final circuit area due to the logic compaction.

Keywords: Digital circuits, emerging technologies, MIGFET, adders, cell libraries, nanotechnology.

LIST OF FIGURES

Figure 1: Transistor structures: (a) traditional MOSFET, (b) FinFET and (c) silicon nanowire FET. Source: (ZHANG, 2016).....	21
Figure 2: Boolean equivalences used to group functions in classes. Source: (HINSBERGER; KOLLA, 1998).....	26
Figure 3: Logic switch representations: (a) direct or NMOS, and (b) complementary or PMOS.	27
Figure 4: Switch networks: (a) series association $f = (x_0 \cdot x_1)$, (b) parallel association $f = (x_0 + x_1)$, and (c) arrangement corresponding to the function $f = !x_4 + ((x_1 \cdot x_2) \cdot (!x_0 + x_3))$	28
Figure 5: NSP logic switch associations: (a) Delta, and (b) Wye. Source: (SHANNON, 1938).....	28
Figure 6: Switch networks corresponding to $f = (!x_4 \cdot !x_3) + (!x_4 \cdot x_0 \cdot x_2) + (x_1 \cdot x_2) + (x_1 \cdot x_0 \cdot !x_3)$: (a) branch-based, (b) factored form, and (c) NSP.....	29
Figure 7: Logically complementary switch networks: (a) $f = (x_0 \cdot x_1) + (x_0 \cdot x_2) + (x_1 \cdot x_2)$ and (b) representation for $!f$	30
Figure 8: Topologically complementarity switch networks (a) representation for $f = x_0 + (x_1 \cdot x_2)$, and (b) representation for $!f$	30
Figure 9: Generic n -inputs CMOS logic gate.....	31
Figure 10: Examples of static CMOS logic gates: (a) inverter, (b) NAND2, (c) NOR2, and (d) or-and (OA).....	31
Figure 11: Example of RC network.....	32
Figure 12: SG-FinFET: (a) physical structure, (b) direct or N-type symbol, and (c) complementary or P-type symbol.....	33
Figure 13: MIGFET logic switch behavior.	34
Figure 14: IG-FinFET: (a) physical structure, (b) N-type IG-FinFET- HV_{th} symbol, (c) N-type IG-FinFET- LV_{th} symbol, (d) P-type IG-FinFET- LV_{th} symbol and (e) P-type IG-FinFET- HV_{th} symbol...	35
Figure 15: DG-SiNWFET and possible configurations: (a) physical structure, (b) N-type device, (c) P-type device and (d) XNOR operation.....	36
Figure 16: Signal degradation in SiNWFET: (a) degradation of signal in the N-type and P-type transistor, and (b) logic gate using transmission gate structure to restore the output signal.....	37
Figure 17: TIG-SiNWFET and possible configurations (a) physical structure (b) GAMBLE operation, (c) N-type device, (d) P-type device, (e) series nFET, (f) series pFET, and (h) XNOR configuration.	38
Figure 18: Signal degradation in TIG-SiNWFET: (a) degradation of signal in N-type and P-type transistors, and (b) logic gate using transmission gate structure to restore the output signal.....	39
Figure 19: TIG-FGMOSFET and possible configurations: (a) physical structure (b) MAJ3 operation, (c) N-type device, (d) P-type device. Source: (DAVILA-SALDIVAR, <i>et al.</i> , 2014).	40
Figure 20: Switch networks corresponding to equation (13): (a) SG devices, (b) IG-FinFET, (c) TIG-FGMOSFET, (d) DG-SiNWFET, and (e) TIG-SiNWFET.....	42
Figure 21: Defactorization technique: (a) original network and (b) defactored one.....	43
Figure 22 Series and parallel defactorizations: (a) switch network obtained from $f = x_0 + (x_1 \cdot x_2 \cdot x_3 \cdot x_4)$, and (b) switch network obtained from $f = x_0 \cdot (x_1 + x_2 + x_3 + x_4)$ Source: (POSSANI <i>et al.</i> , 2016).....	44
Figure 23: Defactorization method proposed in (POSSANI <i>et al.</i> , 2016): (a) series and parallel patterns, and (b) use of defactorization technique over the logic expression $f = x_0 \cdot (x_1 + (x_2 + x_3 + x_4)) + x_5 \cdot (x_6 + x_7 + x_8) + x_9$. Source: (POSSANI <i>et al.</i> , 2016).....	45
Figure 24: Logic gate implementation for $f = (x_0 \cdot x_1 \cdot !x_2) + (x_1 \cdot x_2 \cdot !x_3) + (x_0 \cdot x_2 \cdot !x_3)$ using TIG-FGMOSFET, where $f_M = (x_0 \cdot x_1) + (x_0 \cdot x_2) + (x_1 \cdot x_2)$. Source: (AMARÚ <i>et al.</i> , 2015).....	47
Figure 25: Direct relationship between BBDD node and transistor network composed by DG-SiNWFET. Source: (GAILLARDON <i>et al.</i> , 2014).	48
Figure 26: Transistor layout: (a) MOSFET, and (b) FinFET and SiNWFET.....	51
Figure 27: Dimension parameters for SG-FinFET and DG-FinFET and layout representation.....	52
Figure 28: Layouts for FinFET devices with N_{fin} equals to 3: (a) SG-FinFET and (b) IG-FinFET.....	53

Figure 29: Example of application of folding technique on two-finger FinFETs: (a) SG-FinFET, and (b) DG-FinFET.....	53
Figure 30: Physical dimensions for DG-SiNWFET and TIG-SiNWFET, and layout representation considering minimum W.....	55
Figure 31: Layouts of SiNWFETs with N_{SiNW} equals to 3: (a) DG-SiNWFET, and (b) TIG-SiNWFET.....	56
Figure 32: Example of application of folding technique on two-finger SiNWFET devices: (a) DG-SiNWFET, and (b) TIG-SiNWFET.....	56
Figure 33: Examples of DSD for DG-SiNWFET devices: (a) NAND2 gate layout, and (b) possible layouts for XOR2 gate. Source: (BOBBA <i>et al.</i> , 2015).....	58
Figure 34: Equivalent circuit for TIG-FGMOSFET transistor.....	59
Figure 35: Layouts for TIG-FGMOSFET transistors: (a) TIG-FGMOSFET with W_{min} , (b) TIG-FGMOSFET with W equals to 3, and (c) folding transistor technique.....	60
Figure 36: Circuit area estimation.....	61
Figure 37: Transistor models: (a) 1-control gate, (b) 2-control gate, and (c) 3-control gate.....	63
Figure 38: RC network and timing arcs application: (a) RC network obtained from SG-FinFET logic gate, and (b) RC network obtained from IG-FinFET logic gate.....	68
Figure 39: Electric behavior of a transistor model based on Elmore delay modeling.....	69
Figure 40: Circuit representation for application of STA.....	70
Figure 41: Logic gate implementations using IG-FinFET devices: (a) inverter using IG-FinFET- HV_{th} in FGV, (b) inverter using IG-FinFET- HV_{th} in SIA, (c) inverter using IG-FinFET- LV_{th} in FGV, (d) inverter using IG-FinFET- LV_{th} in SIA, (e) NOR2 logic gate implemented from IG-FinFET- LV_{th} in FGV, and (f) NOR2 logic gate implemented from IG-FinFET- LV_{th} in SIA.....	72
Figure 42: Logic gate implementations using TIG-FGMOSFET devices: (a) inverter gate in FGV, (b) inverter gate in SIA, (c) NOR2 gate in FGV, and (d) NOR2 gate in SIA.....	72
Figure 43: Ripple-carry adder block diagram.....	74
Figure 44: 4-bits Carry Select Adder (CSelB).....	75
Figure 45: 16-bits CSelA implemented through 4-bits CSelB.....	76
Figure 46: 4-bits Carry Skip Adder block (CSkipB).....	77
Figure 47: 16-bits CSkipA implemented using 4-bits CSkipB.....	78
Figure 48: Basic block used to build parallel-prefix adder and carry lookahead adder.....	80
Figure 49: PPA structure: (a) generate-propagate operator, and (b) Brent-Kung arrangements for 16 bits.....	81
Figure 50: Ladner-Fischer arrangements for 16 bits.....	81
Figure 51: Manchester carry chain transistor network for group generate signals from indexes 0 to 3.....	82
Figure 52: Design flow applied to estimate the delay and area of the adder circuits evaluated.....	85
Figure 53: Topologies of XOR3 gate (sum signal) and MAJ3 gate (carry-out signal).....	86
Figure 54: FA designs chosen for each MIGFET technology: (a) IG-FinFET, (b) DG-SiNWFET, (c) TIG-SiNWFET and (d) TIG-FGMOSFET.....	90
Figure 55: Generate-propagate operator logic gate topologies: (a) IG-FinFET, (b) DG-SiNWFET, (c) TIG-SiNWFET and (d) TIG-FGMOSFET.....	94
Figure 56: Technology mapping stages.....	100
Figure 57: Generation of derived expressions: (a) assignments of variables and MIGFETs, and (b) structural representation of a derived expression.....	101
Figure 58: Generation of all possible Boolean function from a given derivate expression.....	102
Figure 59: Generation of switch networks from a given derivate expression.....	103
Figure 60: Logic gate built through of the proposed logic gate generator method.....	105
Figure 61: Design flow adopted to generate cell libraries.....	108

LIST OF TABLES

Table 1: Fictitious design rules adopted here for different MIGFET technologies	50
Table 2: Estimated area values, in nm ² , for different MIGFET with DBU from 1 to 6.....	61
Table 3: Values for area increasing factors, λ and Φ , for different MIGFET technologies.	62
Table 4: Estimated area and delay values for inverter and NOR2 logic gates implemented through IG-FinFET and TIG-FGMOSFET, normalized in relation to respective SG device of each MIGFET technology.....	73
Table 5: Estimated area and delay values for FA topologies using IG-FinFET according to Figure 53.	87
Table 6: Estimated area and delay values for FA topologies using DG-SiNWFET according to Figure 53.	88
Table 7: Estimated area and delay values for FA topologies using TIG-SiNWFET according to Figure 53.	88
Table 8: Estimated area and delay values for FA topologies using TIG-FGMOSFET according to Figure 53.	89
Table 9: The best XOR3 and MAJ3 gates for each MIGFET according to Figure 53.....	89
Table 10: Area and delay estimated for RCA implemented through the MIGFET devices, normalized in relation to respective SG device of each MIGFET technology.	91
Table 11: Gate sizing application on 8-bits RCA implemented using TIG-SiNWFET, area and delay values are normalized in relation to SG-SiNWFET.	93
Table 12: Estimated area and delay values for 64-bits PPA implementations considering each MIGFET technologies, area value is normalized in respect to BK PPA area, while delay value is normalized in respect to LF PPA delay.....	95
Table 13: Estimated area and delay values for 64-bits LF PAA, normalized in relation to SG version of the 64-bits LF PAA for each MIGFET technology.	96
Table 14: Tradeoff between area and delay values for 64-bits RCA and 64-bits LF PAA, delay and area values are normalized in relation to 64-bits RCA.....	97
Table 15: Direct and complementary MIGFET switches	104
Table 16: Maximum device area per MIGFET technology considering the benchmark circuits, the values are normalized in relation to mapped circuit using SG devices.	109
Table 17: Logic cells used in the mapped circuits.	110

LIST OF ABBREVIATIONS AND ACRONYMS

AIG	And Inverter Graph
BBDD	Biconditional binary decision diagram
CMOS	Complementary Metal-Oxide-Semiconductor
DG-SiNWFET	Double gate
FET	Field effect transistor
IG-FinFET	Independent-gate FinFET
ISOP	Irredundant sum-of-product
MIG	Majority inverter graph
MIGFET	Multiple-independent gate field effect transistor
MOSFET	Metal oxide semiconductor field effect transistor
POS	Product-of-sums
SOP	Sum-of-products
SIA	Single-input assignment
SFV	Short fixed version
SG	Single-gate
STA	Static timing analysis
RC	Resistance-capacitor
TIG-FGMOSFET	Tiple-independent gate floating gate MOSFET
TIG-SiNWFET	Triple-independent gate silicon nanowire FET

SUMMARY

ACKNOWLEDGEMENTS	11
RESUMO	12
LIST OF FIGURES	14
1 INTRODUCTION	20
1.1 Motivation	21
1.2 Objective	22
1.3 Text organization	23
2 BACKGROUND	24
2.1 Boolean function	24
2.2 Boolean expressions	24
2.2.1 SOP and POS expressions	24
2.2.2 ISOP expression	25
2.2.3 Factored form	25
2.3 Boolean difference	26
2.4 Classes of Boolean functions	26
2.5 Switch networks	27
2.6 Static CMOS logic gate	30
2.6 Elmore delay method	32
3 SWITCH-BASED DEVICES	33
3.1 Single-gate logic switch	33
3.2 Multiple-gates logic switches	33
3.2.1 IG-FINFET	34
3.2.2 DG-SINWFET	35
3.2.3 TIG-SINWFET	37
3.2.4 TIG-FGMOSFET	39
3.3 New challenges for MIGFET	41
3.3.1 Methods for MIGFET-based network generation	42
4 PHYSICAL AND ELETRICAL ISSUES	49
4.1 MOS and emerging transistor layouts	50
4.1 FINFET layout	51

4.2	SINWFET layout	54
4.3	FGMOSFET layout.....	58
4.4	Evaluation of MIGFET area.....	60
4.5	RC modeling of MIGFET	63
4.5.1	RC modeling.....	63
4.5.2	RC modelling of IG-FinFET	64
4.5.3	RC modelling of TIG-SiNWFET and DG-SiNWFET	65
4.6	Delay analysis of MIGFET logic gates	66
4.6.1	Elmore delay analysis of logic gates.....	67
4.6.2	Static timing analysis	69
4.6.3	Discussions and considerations	71
5	BINARY ADDERS DESIGN	74
5.1	Ripple-carry adder	74
5.2	Carry-select adder	75
5.3	Carry-skip adder	77
5.4	Parallel-prefix adder	79
5.5	Carry-lookahead adder	82
5.6	Binary adder design evaluation.....	83
5.6.1	Evaluation methodology	83
5.6.2	Ripple-Carry Adder design	85
5.6.3	Gate sizing approach.....	92
5.6.4	Parallel-Prefixed Adder design.....	93
6	MIGFET cell library and ASIC design	98
6.1	Technology mapping	98
6.2	Library generator	100
6.3	Benchmark circuit synthesis	106
6.3.1	Evaluation methodology	106
6.3.2	Experimental results	108
	CONCLUSIONS.....	112
	REFERENCES	114

1 INTRODUCTION

The miniaturization (scaling) of MOS transistor has been the key of semiconductor success in the last 30 years. Along these years, more compact and faster circuits have been manufactured as a result of the increasing high density of transistors in a single chip. However, with the current physical dimensions of MOS transistor reaching atomic dimensions of silicon (Si), several studies have indicated that MOS transistor scaling is reaching its limit (FRANK *et al.*, 2001) and (HAENSCH, 2006). In general, the main consequence of such a scaling approach is the fact that smaller devices tend to be more susceptible to electrical effects and parasitic elements. In this sense, several improvements have been performed over the MOSFET structure in order to maintain similar device performance while scaling down.

For instance, traditionally, SiO₂ has been adopted as the main insulator material in the manufacturing of MOS transistors, as depicted in Figure 1(a). In this context, according to the downscaling of transistor, the SiO₂ layer has become thinner. Such dimension reductions in the SiO₂ layer are needed in order to keep electrostatic control over the transistor channel. However, gate tunneling current increases as SiO₂ layer reduce, so impacting in the leakage current (GHANI *et al.*, 2000). In particular, it has been indicated the use of insulator materials with higher dielectric constant (k) such as HfO₂ (RIBES *et al.*, 2005) for technology nodes under of 45 nm in order to reduce the leakage current.

Another consequence of the downscaling of MOS transistor is the impact of the gate material on the electrostatic control of the inversion transistor channel. In general, polycrystalline silicon has been adopted to build the transistor gate. The polysilicon gate suffers from high resistance and depletion effects (YEO *et al.*, 2002). In depletion effect, the depleted region in the polysilicon tends to behave as an additional gate oxide that increases the oxide thickness layer and reduces the electrostatic control of the transistor channel. Such effects are more severe according to the technology node adopted in manufacturing. For instance, for technology node under 45 nm, the metal gate has replaced the polysilicon gate in order to reduce the gate resistance and to eliminate the depletion effects (ZHANG, 2016).

According to (ZHANG, 2016), the MOSFET inversion channel from the 22 nm node technology is quite short than the traditional MOSFET structure cannot provide sufficient electrostatic control of transistor channel. Alternative devices have been proposed and studied in order to keep the continuous downscaling strategy. Among the most promising device

structures, there are the silicon-on-insulator FET (SOI FET) (GNANI *et al.*, 2012), the fully depleted (FD) SOI FET (FD-SOI FET) (DOYLE *et al.*, 2003), the ultra-thin body and buried oxide FD SOI (UTBB-FD SOI) (NOEL *et al.*, 2011), the FinFET (ROSTAMI; MOHANRAM, 2011), and the gate-all-around (GAA) nanowires (ZHANG *et al.*, 2017).

In particular, the physical structure of FinFET and GAA devices tend to increase the electrostatic integrity and the scalability of transistor (RIEL *et al.*, 2014), because, in FinFET and GAA transistor structures, the control gate overlaps the transistor channel over more than one dimension (KUHN, 2012). As seen in Figure 1(b), notice that in the FinFET structure the gate overlaps the transistor channel over the sides and the top (BHOJ; JHA, 2013). On the other hand, in GAA structure, the gate surrounds transistor channel over all sides. According to (ZHANG, 2016), silicon nanowires (SiNW) are a further improvement on the MOSFET structure since the SiNWFET, illustrated in Figure 1(c), can provide significant improvements on the electrostatic control.

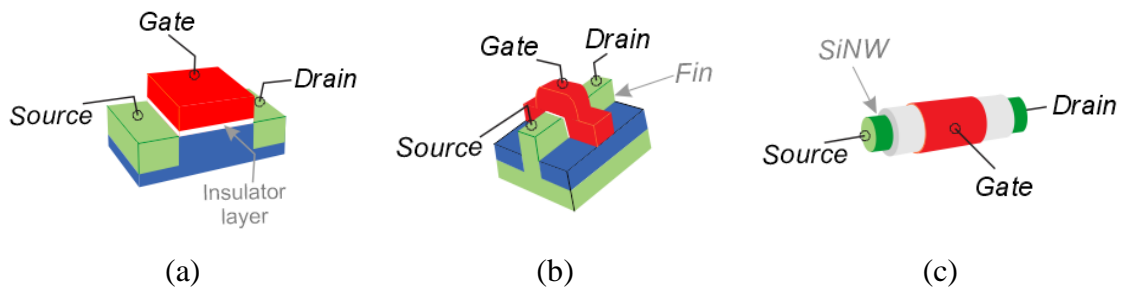


Figure 1: Transistor structures: (a) traditional MOSFET, (b) FinFET and (c) silicon nanowire FET. Source: (ZHANG, 2016).

1.1 Motivation

The adoption of a new technology impacts significantly in the manner as the integrated circuits (IC) are designed. In general, according to a given technology, different logic paradigms are possible. The logic paradigms define the way as the information is represented and handled through of a given technology (HAENSCH, 2006). In this context, much effort has been done in order to develop alternative methods to improve the digital circuit design based on emerging technologies. In such efforts, it is included modifications in the standard methods up to the creation of new algorithms and data structure (AMARÚ; GAILLARDON; DE MICHELI, 2014), (AMARÚ, GAILLARDON; DE MICHELI, 2016), (POSSANI *et al.*, 2016) and (ROSTAMI; MOHANRAM, 2011).

For instance, in CMOS technology the most basic logic element is the logic switch (POSSANI *et al.*, 2016). By combining several switches into a network, different Boolean

functions can be performed. Therefore, the optimization of switch networks is an important task to improve IC performance (POSSANI *et al.*, 2016). On the other hand, when multiple-independent-gate FETs (MIGFET) are adopted, more complex Boolean functions can be implemented through a single switch (AMARÚ *et al.*, 2015). Consequently, the optimal transistor arrangements considering MIGFET devices may be different when compared to conventional CMOS implementations.

1.2 Objective

Our goal is evaluated the impacts of using of MIGFET devices in digital IC design considering the tradeoff between improvements obtained at logic level and physical cost penalties. In particular, at logic level, MIGFETs can be used to merge logic switch arrangements in order to build more compact and efficient circuits. According to (POSSANI *et al.*, 2016), (DATTA *et al.*, 2007), (CHIANG *et al.*, 2006) and (ROSTAMI; MOHANRAM, 2011) it is expected that the reduction in transistor count at logic level compensates the area overhead provided by extra contacts needed for connecting the transistor gates. However, at physical level, the cost of more complex switches tends to penalize the area and signal delay propagation of the target circuit.

In this work, we performed two experiments considering a specific set of MIGFET devices. In the first one, we implement two distinct binary adder architectures, the ripple carry adder (RCA) and parallel prefix adder (PPA). The main goal is to evaluate if binary adder circuits can be optimized through the adoption of MIGFETs considering the tradeoff between logic optimizations and physical penalties. In the second experiment, we developed specific library cells to map a set of digital benchmark circuits in order to evaluate how the commercial tools treat the logic synthesis considering the MIGFET devices. Moreover, in this experiment, we aim to determine an area limit for each MIGFET device that ensures the improvements obtained at logic level.

1.3 Text organization

Chapter 2 presents the basic background needed for understanding the experiments and discussions performed in this work. Among the concepts presented are Boolean expression, logic switch, switch network and the building of static CMOS logic gate.

Chapter 3 offers an overview about MIGFET devices taken into account in this work, discussing the logic behavior of each transistor. In particular, in this chapter, we also aim to demonstrate the new possibilities and the challenges raised from using MIGFETs in building digital integrated circuits.

Chapter 4 presents some MIGFET layouts elaborated through fictitious design rules. In particular, the MIGFET layouts are used to extract some physical characteristic in order to create the RC modelling of devices.

Chapter 5 presents the binary adder architectures implementations discussed in this work. The estimated area and delay results for binary adders implementations are provided.

Chapter 6 presents the method used to build MIGFET libraries and discusses the logic mapping performed considering these libraries.

Finally, the Chapter 7 outlines the conclusions and final considerations of this work.

2 BACKGROUND

In this chapter, some definitions and basic notation are introduced. These concepts are helpful for understanding the present work.

2.1 Boolean function

An n -input Boolean function $f(X)$, defined by the variable support set $X = \{x_0, \dots, x_{n-1}\}$, can be represented as follows:

$$f(X): B^n \rightarrow B \quad (1)$$

where $B = \{0, 1\}$. A Boolean variable is defined in the domain of the function, and can assume the logic values 0 or 1, typically 0 = *false* and 1 = *true*. Moreover, a variable can be represented in positive or negative polarity. The positive polarity indicates the direct use of the variable, whereas the negative polarity indicate the use of the complemented (or negated) variable. An instance of a variable in its positive or negative polarity is called literal. The literals x_0 and x_1 are examples of positive literals. whereas $\neg x_0$ and $\neg x_1$ are negative literals.

2.2 Boolean expressions

A simple way to represent a Boolean function is using Boolean expression. A Boolean expression can be defined recursively as a constant (0 or 1), a variable (in positive or negative polarity), a product, a sum or a complement of Boolean expressions. In this work, the product operator is denoted as AND (\cdot), the sum operator is denoted as OR ($+$) and the complement is denoted as NOT (\neg or $\bar{}$).

2.2.1 SOP and POS expressions

A sum-of-product (SOP) is a Boolean expression representation where a given Boolean function is expressed as a sum of product terms of l literals, called *minterms*. On the other hand, a product-of-sum (POS) expression of a given Boolean function is expressed through a product of sum terms of l literals, called *maxterms* (DE MICHELI, 1994). The following Boolean expression:

$$f = (x_0 \cdot x_1 \cdot x_2) + (x_3 \cdot x_4) \quad (2)$$

is an example of SOP, whereas the following Boolean expression:

$$f = (x_0 + x_1 + x_2) \cdot (x_3 + x_4) \quad (3)$$

is an example of POS. Notice that both SOP and POS expressions represent two-level forms. In this approach, a given expression can be directly implemented through a circuit of logic depth equals to two.

2.2.2 ISOP expression

A cube (or product term) is a cube implicant of a given Boolean function f , if each assignment that makes this product to be evaluated to 1 also evaluates f to 1 (MURGAI; BRAYTON; SANGIOVANNI, 2012). In turn, a prime implicant of a given Boolean function f is a cube implicant such that removing any literal from such a cube results in a new cube that does not imply f . On the other hand, an irredundant sum-of-products (ISOP) expression is a SOP where each product term is prime implicant that cannot be removed without changing the Boolean function behavior by the expression.

2.2.3 Factored form

A factored form can be defined as a literal, a product, or a sum of factored forms (BRAYTON, 1987). The factored form is also defined as multi-level logic form. In multi-level logic, the Boolean expressions can have unbounded number of levels. However, not all multi-level expression are factored forms. For instance, the following expression:

$$f = !(!x_0 \cdot (!x_1 \cdot x_2)) \quad (4)$$

is described as a multi-level form but not a factored form. In turn, the Boolean expression presented in sequence:

$$f = !x_0 \cdot (!x_1 \cdot x_2) \quad (5)$$

is a factored form and also a multi-level form. According to (GOLUMBIC; MINTZ, 1999), the objective of factorization methods is to represent a Boolean function in a logically equivalent factored form but with a minimum number of literal. For instance, the following Boolean expression:

$$f = (x_0 \cdot x_2) + (x_1 \cdot x_2) + (x_2 \cdot x_3 \cdot x_4) \quad (6)$$

can be factored into an equivalent form, such as follow:

$$f = x_2 (x_0 + x_1 + (x_3 \cdot x_4)) \quad (7)$$

Notice that, this factored expression is optimum in the number of literals, where literals count reduced from seven to five.

2.3 Boolean difference

Given a Boolean function $f(x_0, x_1, \dots, x_{n-1})$ the cofactor operation consist of assigning a logic constant value $c_i \in \{0, 1\}$ to an input variable $x_i \in \{x_0, x_1, \dots, x_{n-1}\}$. Such operation is denoted $f_{x_i = c_i}$ (BOOLE, 1854). Thus, the positive cofactor of f with respect to variable x_i is denoted $f_{x_i} = f(x_0, x_1, \dots, x_i = 1, \dots, x_{n-1})$. On the other hand, the negative cofactor of f with respect to variable x_i is $f_{\bar{x}_i} = f(x_0, x_1, \dots, x_i = 0, \dots, x_{n-1})$ (DE MICHELI, 1994). This way, the Boolean difference with regard to a variable x_i of a given Boolean function f is defined as the exclusive sum of its cofactors, as follows:

$$\frac{\partial f}{\partial x_i} = f_{\bar{x}_i} \oplus f_{x_i} \quad (8)$$

2.4 Classes of Boolean functions

Boolean functions can be classified into classes of functions. In general, Boolean functions can be grouped considering the complementation (negation) of its inputs (x), the permutation of its inputs (y) and/or inversion (negation) of its outputs (z), as shown in Figure 2. In particular, P-class represents the set of equivalent Boolean functions obtained by permuting the inputs (HINSBERGER; KOLLA, 1998).

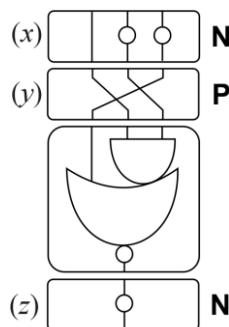


Figure 2: Boolean equivalences used to group functions in classes. Source: (HINSBERGER; KOLLA, 1998).

2.5 Switch networks

A logic switch can be represented through a device composed of one control terminal, gate (G), and two contact terminals, source (S) and drain (D). The control terminal determines if there is a connection between the contact terminals. Moreover, a logic switch can be classified as direct or complementary according to the polarity of the control terminal activation. In the direct logic switch the contact terminals are connected only when the logic value 1 is applied to the control terminal. In contrast, the complementary switch presents the connection between the terminal contacts only when the logic value 0 is defined in the control terminal.

In CMOS technology, the direct and complementary logic switches are represented by NMOS and PMOS transistors, respectively. Such devices implements 1-input Boolean function, *i.e.*, $f(x) = x$ or $f(x) = !x$ (POSSANI *et al.*, 2016). For convenience, in this work, devices or switches that presents such a logic behavior are named as single-gate (SG) one. Moreover, the definition of logic switch will be later extended to devices with more than one control terminals. Notice that the direct and complementary logic switches in CMOS technology are depicted in Figure 3.

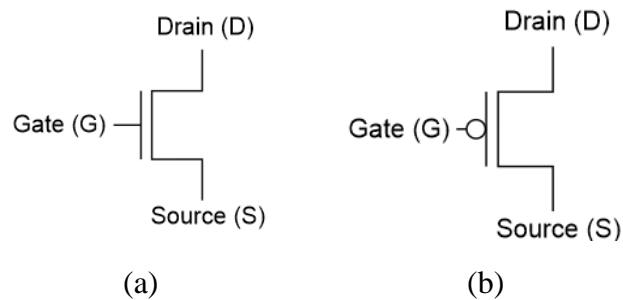


Figure 3: Logic switch representations: (a) direct or NMOS, and (b) complementary or PMOS.

Traditionally, switch networks are built through the arrangement of direct and/or complementary SG logic switches. In this way, a Boolean function can be represented through a switch network by defining the arrangement of logic switches that connects two external terminals (T1 and T2) of the logic network (POSSANI *et al.*, 2016). The most trivial logic switch arrangements are *series* and *parallel* associations. A series association ($x_0 \cdot x_1$) is shown in Figure 4(a), whereas a parallel association ($x_0 + x_1$) is shown in Figure 4(b). An arbitrary Boolean function can be implemented through a switch network, as illustrated in Figure 4(c).

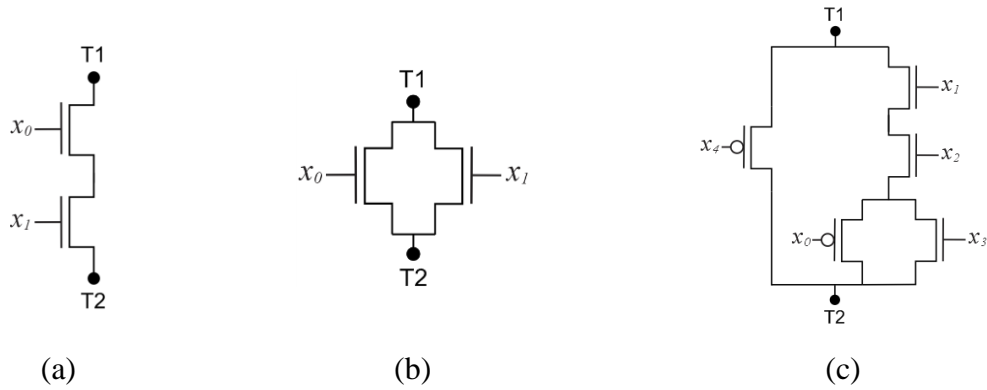


Figure 4: Switch networks: (a) series association $f = (x_0 \cdot x_1)$, (b) parallel association $f = (x_0 + x_1)$, and (c) arrangement corresponding to the function $f = !x_4 + ((x_1 \cdot x_2) \cdot (!x_0 + x_3))$.

In general, switch networks can be divided in series-parallel (SP) and non-series-parallel (NSP) networks. An SP switch network is composed by only series or/and parallel associations of logic switches. In turn, an NSP switch network, besides series and parallel associations, the logic switches can be associated through Delta or Wye logic arrangements (SHANNON, 1938), as shown in Figure 5.

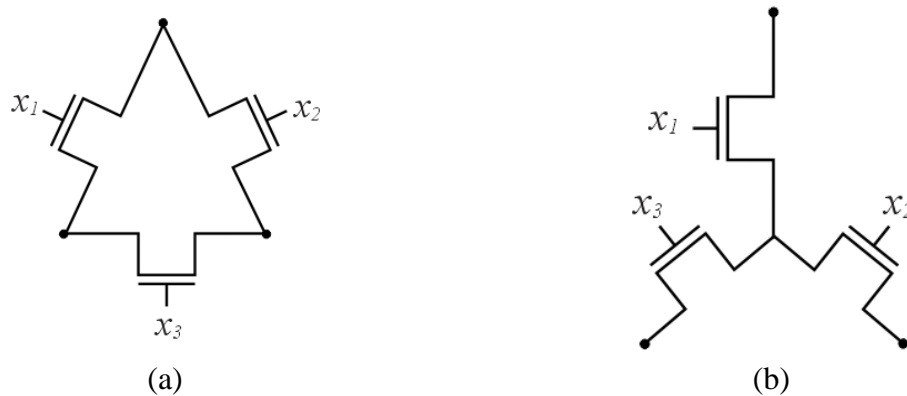


Figure 5: NSP logic switch associations: (a) Delta, and (b) Wye. Source: (SHANNON, 1938).

In general, NSP switch networks are more compact than SP ones. For instance, all switch networks presented in Figure 6 correspond to the following Boolean function:

$$f = (!x_4 \cdot !x_3) + (!x_4 \cdot x_0 \cdot x_2) + (x_1 \cdot x_2) + (x_1 \cdot x_0 \cdot !x_3) \quad (9)$$

In Figure 6(a), the switch network is obtained directly from a SOP expression. In Figure 6(b), the switch network is implemented from a factored form expression and, in Figure 6(c), such Boolean function is represented through an NSP switch network. The switch network shown in Figure 6(a) presents a network configuration known as *branch-based*. In this type of network, only arrangements of logic switches associated in series are used to connect the external terminals of the switch network (PIGUET, 1998). Notice that, among all possible

switch network representations shown in Figure 6, the NSP network is the most compact representation in the number of logic switches.

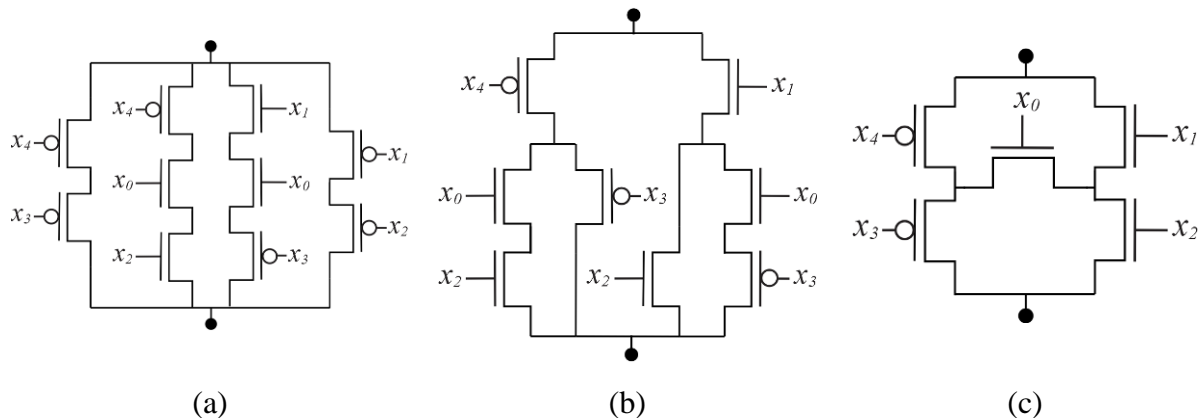


Figure 6: Switch networks corresponding to $f = (!x_4 \cdot !x_3) + (!x_4 \cdot x_0 \cdot x_2) + (x_1 \cdot x_2) + (x_1 \cdot x_0 \cdot !x_3)$: (a) branch-based, (b) factored form, and (c) NSP.

In the literature, the switch network generation methods are divided into factored-form-oriented methods and graph-oriented methods. In the methods based on factored form expressions, the switch network is built directly from a Boolean expression. In such approach, it is only possible to generate SP switch networks (BRAYTON, 1987), (MARTINS *et al.*, 2010), and (GOLUMBIC; MINTZ; ROTICS, 2008). In turn, in switch network generation methods based on graph structures, a given Boolean function is initially represented through a graph that can be later optimized and translated to a switch network. In such an approach, each edge of the graph represents a logic switch. This technique allows the generation of SP and NSP switch networks (ZHU; ABD-EL-BARR, 1993), (DA ROSA, MARQUES *et al.*, 2007), (KAGARIS; HANIOTAKIS, 2007), and (POSSANI *et al.*, 2016).

Another quite important property regarding to switch networks is the concept of complementarity. A given switch network is said logically complementary to another when only one of them is connecting respective contact terminals for a given configuration of the input vector. In turn, a given switch network is topologically complementary to another if such logic switch implements the dual plan of the other switch network (UEHARA; VANCLEEMPUT, 1979). In Figure 7, it is presented two switch networks that are logically complementary but are not topologically complementary. In turn, in Figure 8 it is shown two switch networks that are topologically complementary.

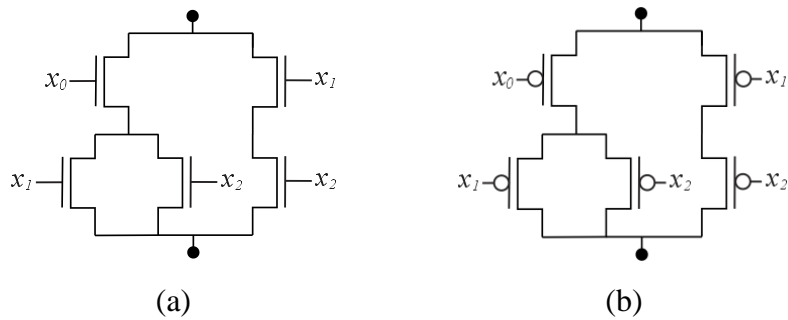


Figure 7: Logically complementary switch networks: (a) $f = (x_0 \cdot x_1) + (x_0 \cdot x_2) + (x_1 \cdot x_2)$ and (b) representation for $\neg f$.

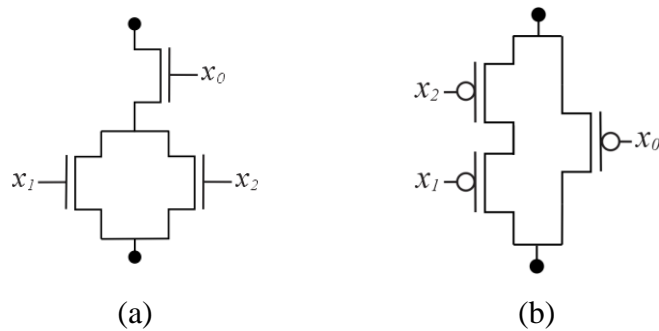


Figure 8: Topologically complementary switch networks (a) representation for $f = x_0 + (x_1 \cdot x_2)$, and (b) representation for $\neg f$.

2.6 Static CMOS logic gate

In conventional static CMOS logic topology, a logic gate is built through two complementary plans, called pull-up plan (PU) and pull-down plan (PD). The structure of the CMOS logic gate is illustrated in Figure 9. Notice that the PU plan is used to connect the output terminal to the constant logic value 1 (VDD terminal), while the PD plan is used to connect the output terminal to constant logic value 0 (GND terminal). This way, when the output of the gate logic is evaluated as 1, there is a logic path across the PU plan that connects the VDD terminal to output terminal. Otherwise, it exist a logic path across the PD plan that connects the output terminal to GND terminal. Notice that, in this context, a given Boolean function f is implemented in PD plan, whereas its complement is implemented in PU plan, such as it is shown in Figure 9.

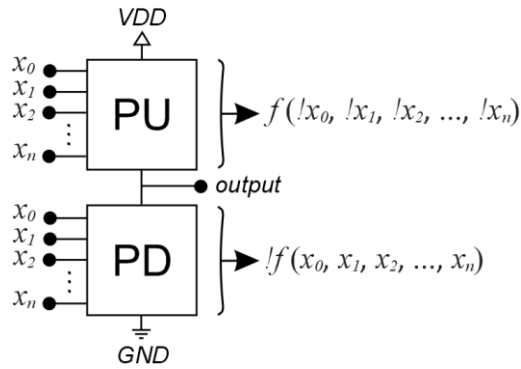


Figure 9: Generic n -inputs CMOS logic gate.

The PU plan is implemented through a switch network built exclusively by PMOS transistors. Additionally, the PD plan is implemented through a switch network comprising only NMOS transistors. Some static CMOS logic gates are shown in Figure 10. Notice that, to ensure the complementarity between the PU and PD plans, the switch network implemented in PU plan can be obtained through the dual switch network implemented in PD plan.

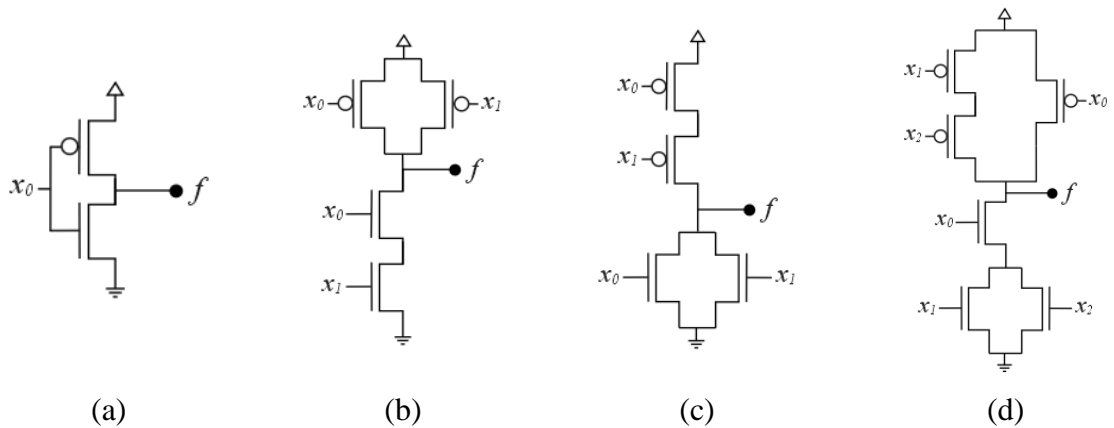


Figure 10: Examples of static CMOS logic gates: (a) inverter, (b) NAND2, (c) NOR2, and (d) or-and (OA).

In general, the static CMOS logic gates present low power dissipation when compared to others CMOS logic topology and families, since PU and PD plans are complementary, so only one of them is *on* (RABAHEY; CHANDRAKASAN; NIKOLIC, 2002). Moreover, the static CMOS logic topology presents high noise margin since due to the complementary structure, a static logic path restores the correct logic state in the presence of noise. Furthermore, the static CMOS topology presents a high robustness against power voltage downscaling (ANIS; ALLAM; ELMASRY, 2002). However, CMOS logic gates requires $2n$ transistors to implements an n -input Boolean function. Moreover, the load capacitance is quite significant since each input signal drives at least two devices per logic gate (RABAHEY; CHANDRAKASAN; NIKOLIC, 2002).

2.6 Elmore delay method

The Elmore delay modeling is used to obtain an approximated delay for a given input circuit (ELMORE, 1948). In general, the results obtained through Elmore delay method are not very accurate since the slope of input signal is not taken into account. However, reducing the Elmore delay tends to lead to a delay reduction in real circuit.

In the Elmore delay method, the evaluated circuit is modelled through resistive (R) and capacitive (C) elements in an RC network. This way, the circuit is represented through resistor and capacitance associations. Additionally, the estimated delay for a given RC network can be computed as follows (CONG; KWOK-SHING, 1995):

$$D = \sum_{i=0}^{all_nodes} R_i * C_i \quad (10)$$

where R_i represents the resistance between the source node and the node i , whereas C_i represents the capacitance at the node i . In order to exemplify the application of Elmore delay method, in Figure 11, it is illustrated an RC network implemented from a series association of resistors. In this case, the delay (D) of the circuit can be computed as follows:

$$D = C_1 * R_1 + C_2 * (R_1 + R_2) + \dots C_i * (R_1 + R_2 + \dots + R_i) \quad (11)$$

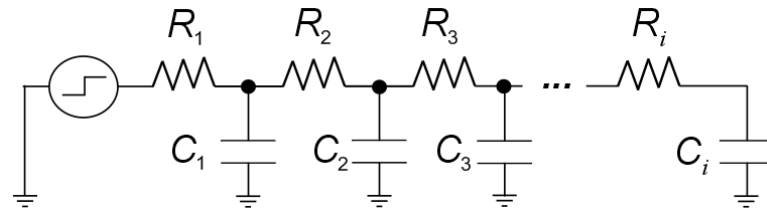


Figure 11: Example of RC network.

The Elmore delay model corresponds to the linear delay modeling (LDM) to compute the signal delay propagation across logic gates. In such a situation, the estimated delay value of a given logic gate is defined as a linear function in respect to its output capacitance (JU; HANDSCHIN; KARLSSON, 1996). However, the propagation of a given signal in a logic gate do not varies linearly with its output capacitance. Such a variation depends also on the input signal slope. Moreover, the resistances associated to the wires used to connect distinct logic gates tend to be more significant when the technology node is scaled down. In this sense, the non-linear delay model (NLDM), composite current source model (CCSM) (SYNOPTSYS, 2017) and effective current source model (ECSM) (CADENCE, 2017) present more accurate results (MARTINS *et al.*, 2015).

3 SWITCH-BASED DEVICES

In this chapter, the multiple-independent-gate field effect transistors (MIGFETs) are introduced. Similar to traditional MOSFET transistor, MIGFETs can be adopted in the construction of switch networks. However, due to higher logic capability of MIGFETs, the traditional methods for switch network generation may not be the most indicated for these transistors. In this sense, we discussed the main challenges and possibilities regarding to switch network generation taking into account a particular set of MIGFETs. The concepts presented herein are helpful for the understanding of experiments performed in this work.

3.1 Single-gate logic switch

In FinFET technology (ROSTAMI; MOHANRAM, 2011), the direct and complementary logic switches can be represented through of the shorted-gate FinFET, also referenced as single-gate (SG) FinFET. This device is a transistor controlled by a single gate. Similar to traditional MOSFET device, SG-FinFET can only implement 1-input Boolean function. Its physical structure consists of a thin silicon body wrapped by gate electrodes (CUI *et al.*, 2014), such as illustrated in Figure 12(a). Similarly to MOSFET devices, SG-FinFET can be manufactured in N-type and P-type versions. Therefore, SG-FinFET device corresponds to direct logic switch (N-type) or complementary logic switch (P-type). In Figure 12(b) and in Figure 12(c), the notations for N-type and P-type SG-FinFET logic switches are represented, respectively.

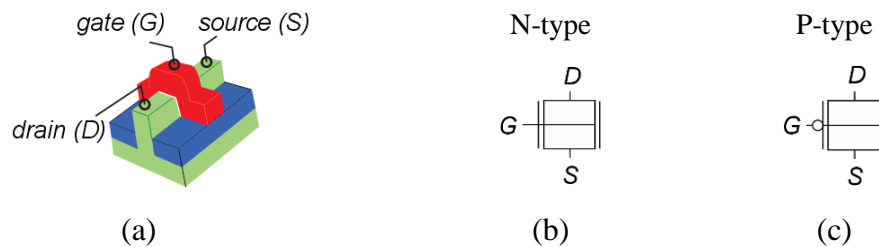


Figure 12: SG-FinFET: (a) physical structure, (b) direct or N-type symbol, and (c) complementary or P-type symbol.

3.2 Multiple-gates logic switches

Multiple-independent-gate field effect transistors (MIGFET) allow more logic capability into a single device. In particular, a MIGFET logic switch can implement logic functions with more than one variable into a single device, as illustrated in Figure 13. Thus,

the switch networks implemented using MIGFET tend to be more compact in the number of logic switches when compared to traditional SG approaches (AMARÚ *et al.*, 2015). However, the circuit area and the signal delay propagated through the devices tend to be larger than in circuits based on SG devices (ALIOTO, 2011). Indeed, there is a tradeoff between the increased logic capability of a given MIGFET and its area/performance cost. Consequently, the optimal MIGFET-based network arrangements may be different from conventional static CMOS implementations (POSSANI *et al.*, 2016).

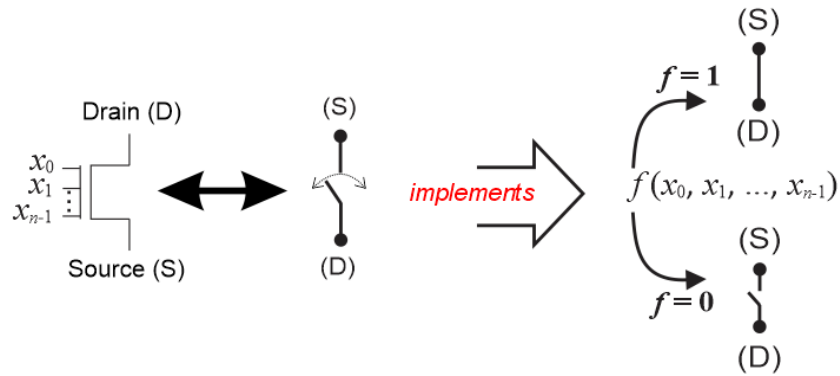


Figure 13: MIGFET logic switch behavior.

MIGFET devices have demonstrated a great potential for digital integrated circuit design. Several works have explored the use of such a kind of devices to improve mainly area and power consumption of digital ICs (AMARÚ *et al.*, 2015), (CHIANG, *et al.*, 2005), (BENJAMAA; MOHANRAM; DE MICHELI, 2011), (HSIEH, *et al.*, 2012) and (BHOJ; JHA, 2013). Among the most promising MIGFET alternatives discussed in the literature are the independent-gate FinFET (IG-FinFET) (RIEL *et al.*, 2014), the double-gate silicon nanowire FET (DG-SiNWFET) (AMARÚ; GAILLARDON; DE MICHELI, 2013), the triple independent-gate silicon nanowire FET (TIG-SiNWFET) (ZHANG *et al.*, 2017) and the triple independent-gate floating-gate MOS transistor (FGMOSFET) (SHIBATA; OHMI, 1992). Hereafter, the physical structure, graphic representation and logic behavior those MIGFETs are presented.

3.2.1 IG-FINFET

IG-FinFET is a transistor controlled by two independent gates, the front gate (FG) and the back gate (BG), as shown in Figure 14(a). Such a device implements 2-input (AND2) or 2-input OR (OR2) operation, which can be associated to series and parallel arrangements of two SG-transistors, respectively. AND2 and OR2 configuration are shown in Figure 14(b), and in Figure 14(c), respectively. The OR2 operation is performed through IG-FinFET-LV_{th}.

In such case, if at least one of the independent gates is active, the transistor is turned on (ROSTAMI; MOHANRAM, 2011). On the other hand, the AND2 operation is obtained by using IG-FinFET-HV_{th}, in which, both control gates must be active to create the conducting transistor channel (ROSTAMI; MOHANRAM, 2011).

Both IG-FinFET-LV_{th} and IG-FinFET-HV_{th} admits P-type and N-type versions. All the possible variations those devices are shown in Figure 14. Notice that the P-type IG-FinFET-LV_{th} implements a 2-input NAND (NAND2) operation, which can be related to two P-type SG-transistor associated in parallel. In turn, the P-type IG-FinFET-HV_{th} implements the 2-input NOR (NOR2) operation, which can be related to two P-type SG-transistors associated in series. NAND2 and NOR2 configurations are shown in Figure 14(d) and in Figure 14(e), respectively.

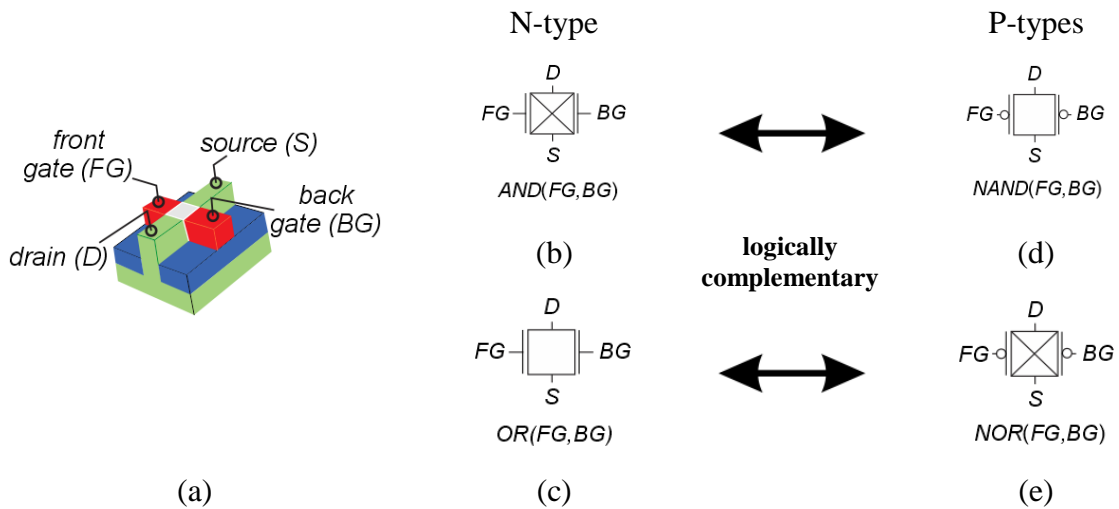


Figure 14: IG-FinFET: (a) physical structure, (b) N-type IG-FinFET-HV_{th} symbol, (c) N-type IG-FinFET-LV_{th} symbol, (d) P-type IG-FinFET-LV_{th} symbol and (e) P-type IG-FinFET-HV_{th} symbol.

3.2.2 DG-SiNWFET

DG-SiNWFET, as shown in Figure 15(a), is an ambipolar transistor. Therefore, the polarity (N-type or P-type) of the device is configurable via back gate, called polarity-gate (PG) (AMARÚ; GAILLARDON; DE MICHELI, 2013). If the signal on the PG is 0 the device acts as a P-type transistor, whereas if this signal is 1 the device acts as an N-type transistor. N-type and P-type configurations are shown in Figure 15(b) and in Figure 15(c), respectively. The other device gate in DG-SiNWFET is the control-gate (CG), which defines the connectivity between the source and drain terminals. Such a device implements a 2-input exclusive-NOR (XNOR2) operation, as illustrated in Figure 15(d). Thus, a DG-SiNWFET can represent the arrangement of SG-transistors that corresponds to a XNOR2 Boolean function.

However, due to the signal electrical degradation, it is necessary to use two devices to build the XNOR2 logic gate (DE MARCHI *et al.*, 2012).

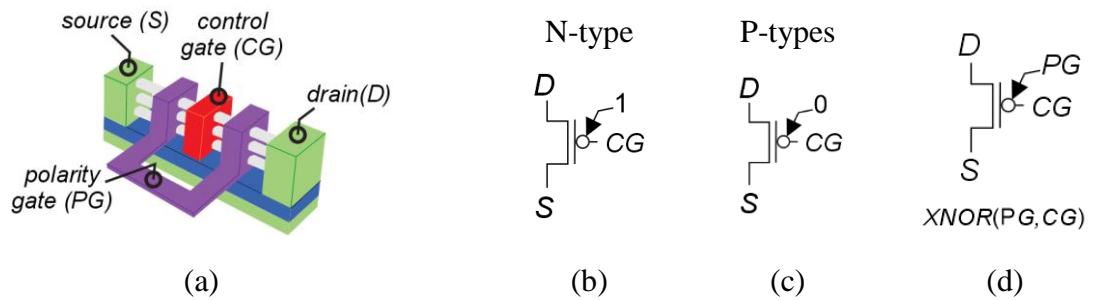


Figure 15: DG-SiNWFET and possible configurations: (a) physical structure, (b) N-type device, (c) P-type device and (d) XNOR operation.

Similar to MOSFET device, the N-type DG-SiNWFET conduces a weak logic value 1, and the P-type DG-SiNWFET conduces a weak logic value 0. In this context, according to polarization of the DG-SiNWFET, its use can cause degradation of the electrical signal that pass through the transistor. Notice that, for the P-type transistor, presented in Figure 16(a), the logic value that passes through the transistor is at least $GND + |V_{tp}|$, where the V_{tp} is defined as the threshold value of P-type transistor. In turn, in N-type transistor, presented in Figure 16(a), the output value is at most $VDD - V_{tn}$, where V_{tn} is the threshold value of the N-type transistor. For both situations, the output signal is degraded due to the polarization of the device. In order to guarantee the correct output signal, the transmission gate configuration is needed. In this structure, P-type and N-type devices are used in parallel to ensure that one of the two transistors restores the electrical signal level in all cases (BEN-JAMAA; MOHANRAM; DE MICHELI, 2011). In Figure 16(b), a 2-input exclusive-OR2 (XOR2) logic gate illustrates the use of transmission gate approach. Notice that, in both the PU and PD plans, the transistors are duplicated in order to ensure the correct output signal level.

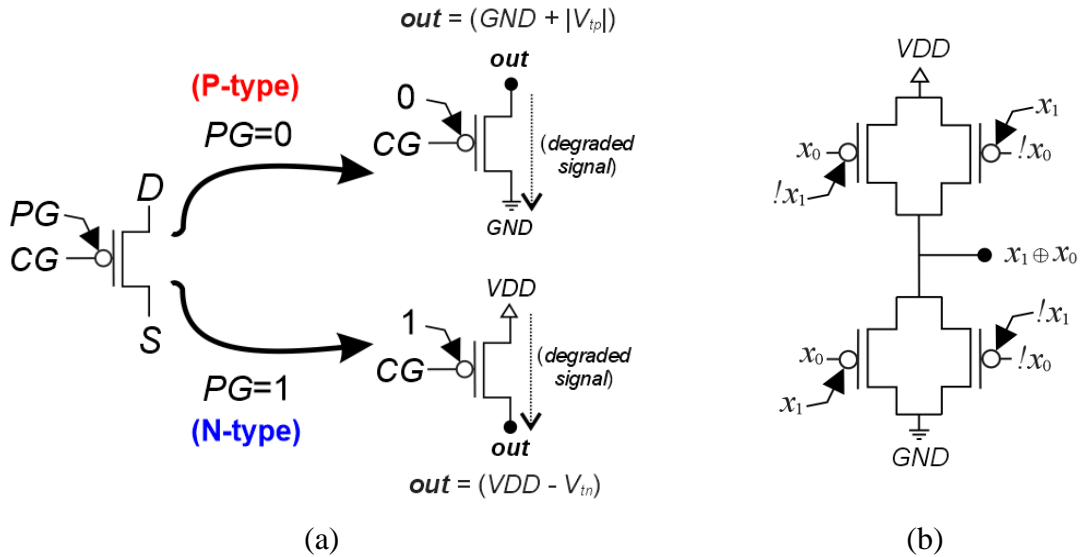


Figure 16: Signal degradation in SiNWFET: (a) degradation of signal in the N-type and P-type transistor, and (b) logic gate using transmission gate structure to restore the output signal.

3.2.3 TIG-SiNWFET

TIG-SiNWFET is an ambipolar device controlled by three independent gates: the control gate (CG), the polarity gate at source (PGS) and the polarity gate at drain (PGD) (ZHANG *et al.*, 2017). The physical structure of TIG-SiNWFET is depicted in Figure 17(a). The connection between drain and source terminals exists only when all three independent-gate present the same signal value (AMARÚ *et al.*, 2015). Therefore, the logic behavior of this device corresponds to the gamble function, in the following:

$$f = (x_0 \cdot x_1 \cdot x_2) + (!x_0 \cdot !x_1 \cdot !x_2) \quad (12)$$

Notice that, in this case, the transistor is on if the same logic level is applied to CG, PGS and PGD. Otherwise, the transistor is off. Similar to DG-SiNWFET, TIG-SiNWFET can be configured to act as an N-type or a P-type device according to its polarization (ZHANG *et al.*, 2017). A P-type device can be obtained by fixing PGS and PGD in 0. In this case, the transistor is on only when the CG assumes the logic value 0. In turn, an N-type device can be obtained by fixing PGS and PGD in 1. This way, the device is active only when the logic value 1 is applied to CG. N-type and P-type devices are shown in Figure 17(c) and in Figure 17(d), respectively.

Two others possible configurations for TIG-SiNWFET are the series SG transistors merging (ZHANG *et al.*, 2017). In this context, when only the PGD is fixed to 1, TIG-SiNWFET acts as an N-type device allowing the implementation of AND2 operation, as shown in Figure 17(e). On the other hand, when only the PGS is fixed to 0, it is possible to

obtain a P-type device that allows the implementation of NOR2 operation, as illustrated in Figure 17(f).

Additionally, TIG-SiNWFET can acts as DG-SiNWFET by applying the same logic level to PGS and PGD, as shown in Figure 17(h) (ZHANG; GAILLARDON; DE MICHELI, 2013). However, as discussed previously, this configuration may present degradation of the electrical signal that flows across the transistor. Similarly, the signal degradation problem may occur in gamble configuration, as illustrated in Figure 18(a). In both cases, the transmission gate structure may be adopted to ensure the correct signal. Figure 18(b) shows a logic gate that implements the gamble function using TIG-SiNWFET. Notice that the transmission gate structure is used in PU plan to ensure the correct signal transmission.

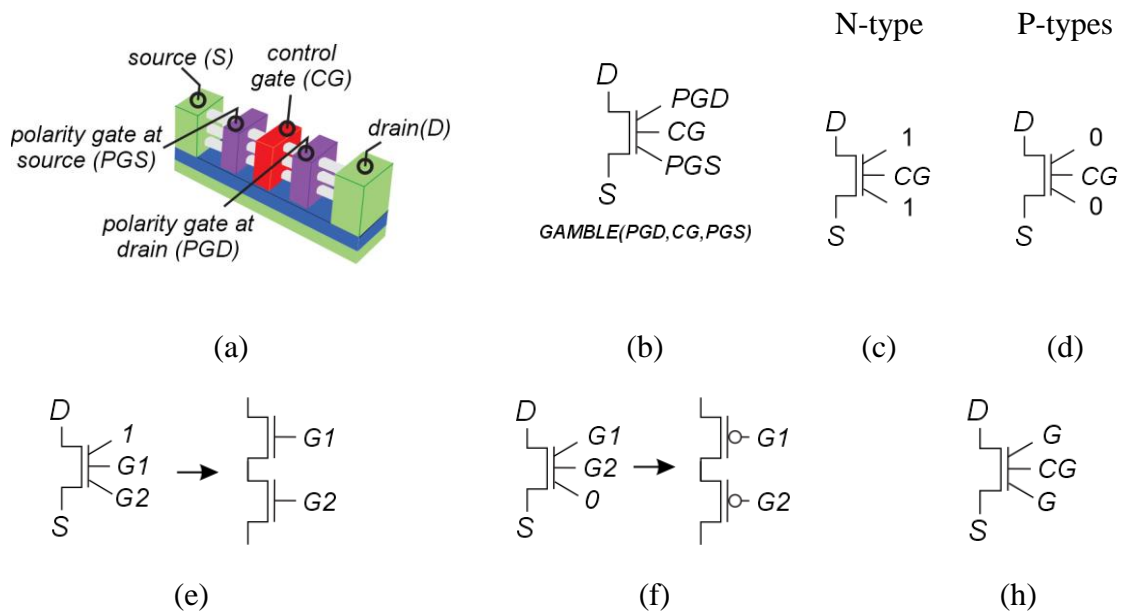


Figure 17: TIG-SiNWFET and possible configurations (a) physical structure (b) GAMBLE operation, (c) N-type device, (d) P-type device, (e) series nFET, (f) series pFET, and (h) XNOR configuration.

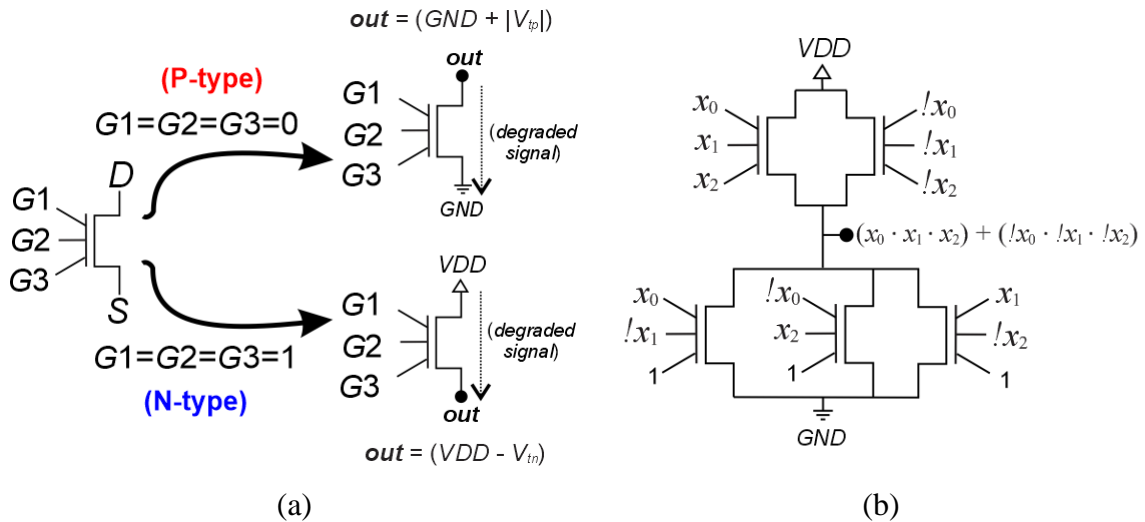


Figure 18: Signal degradation in TIG-SiNWFET: (a) degradation of signal in N-type and P-type transistors, and (b) logic gate using transmission gate structure to restore the output signal.

3.2.4 TIG-FGMOSFET

The last device considered in this work is the TIG-FGMOSFET, which is a transistor electrically controlled by a floating-gate, as shown in Figure 19(a). A general FGMOSFET transistor has i -inputs that are capacitively coupled to the floating gate (SHIBATA; OHMI, 1992). Such a device allows the computation of threshold logic functions (NEUTZLING *et al.*, 2015). However, in general case, the number of inputs of the device increases according to the number and weight assigned to input variables of the target threshold function. In this work, it is adopted a kind of TIG-FGMOSFET transistor similar to the one presented in (AMARÚ *et al.*, 2015).

The N-type version of TIG-FGMOS implements a 3-input majority (MAJ3) operation, as illustrated Figure 19(b). In this sense, such a transistor is only activated when at least two of its inputs are 1. Additionally, the N-type TIG-FGMOSFET can be configured to implement AND2 and OR2 operations. The AND2 configuration is attained by fixing one of the inputs to 0 value, as illustrated in Figure 19(c). In turn, the OR2 configuration is possible by fixing one input to 1 value, as shown in Figure 19(d). On the other hand, a P-type TIG-FGMOSFET implements a 3-input minority (MIN3) operation, as illustrated in Figure 19(e). Therefore, this device is active only when at least two inputs are 0. Moreover, the P-type device can implements the NAND2 operation by fixing one input to 0 value. Additionally, the NOR2 operation is implemented also through P-type transistor but by fixing one input to 1 value. The NAND2 and NOR2 configurations are depicted in Figure 19(f) and in Figure 19(g), respectively.

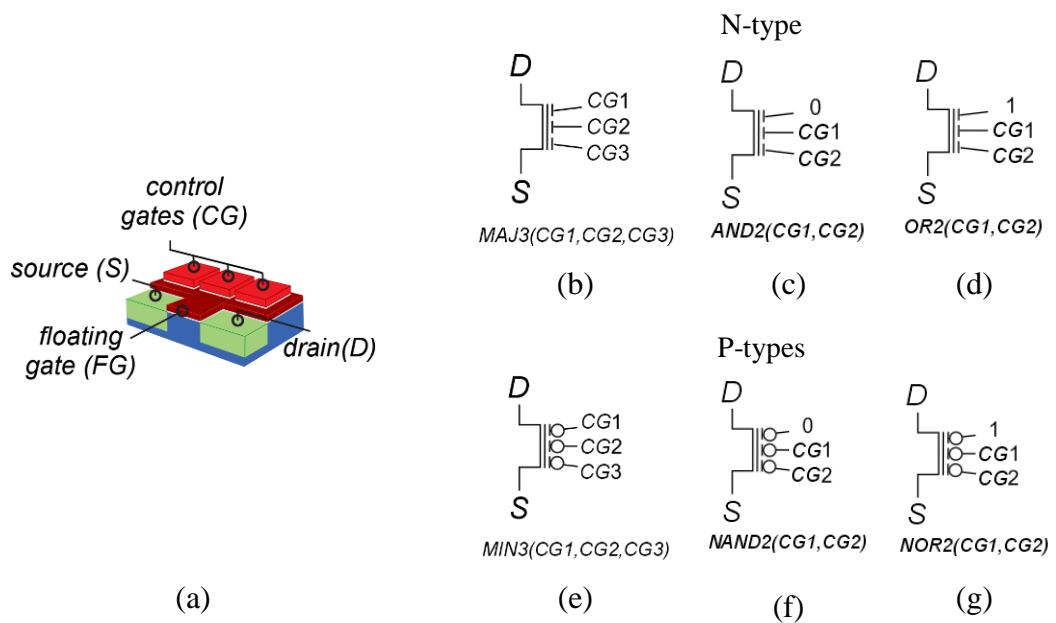


Figure 19: TIG-FGMOSFET and possible configurations: (a) physical structure (b) MAJ3 operation, (c) N-type device, (d) P-type device. Source: (DAVILA-SALDIVAR, *et al.*, 2014).

3.3 New challenges for MIGFET

As discussed before, MIGFET have the ability to implement into a single device n -input Boolean functions, such as AND2, OR2, MAJ3, XNOR2, among others. Thus, the adoption of MIGFETs to design digital integrated circuits introduces new logic paradigms and challenges that must be considered in order to obtain optimized solutions. For instance, the most conventional SG transistor network generation methods explore the unitary relationship between a literal and a logic switch. However, for MIGFET devices, such a relationship may not be the best alternative, since more than one literal can be mapped into a single transistor (POSSANI *et al.*, 2016).

In order to exemplify this concept, it is shown in Figure 20(a) a switch network that implements the following Boolean function:

$$f = ((x_0 \oplus x_1) \cdot (x_2 + x_3 + x_4)) + x_5 \cdot x_6 \quad (13)$$

Notice that nine SG logic switches are needed to implement such a switch network. In this case, each literal of the logic expression is mapped directly to a logic switch. In turn, in Figure 20(b)-(e), MIGFET switches are used to build alternative networks for the logic equation (13). In all these cases, the SG switch merging performed by using MIGFET devices results in more compact network when compared to conventional single gate switch based implementation. Notice also that the number of switch merging performed through a given MIGFET depends on the target Boolean function and the type of MIGFET used to generate the switch network. For instance, the most compact network implementations for logic equation (13) are reached through TIG-FGMOSFET and IG-FinFET switches, since series and parallel switch merging are better exploited through these MIGFETs. In this case, each switch network comprises only five logic switches. In contrast, the switch networks obtained through DG-SiNWFET and TIG-SiNWFET comprise seven and six switches, respectively. In the case of DG-SiNWFET, only XOR2 merges can be explored, whereas TIG-SiNWFET allows performing series and XOR2 switch merges.

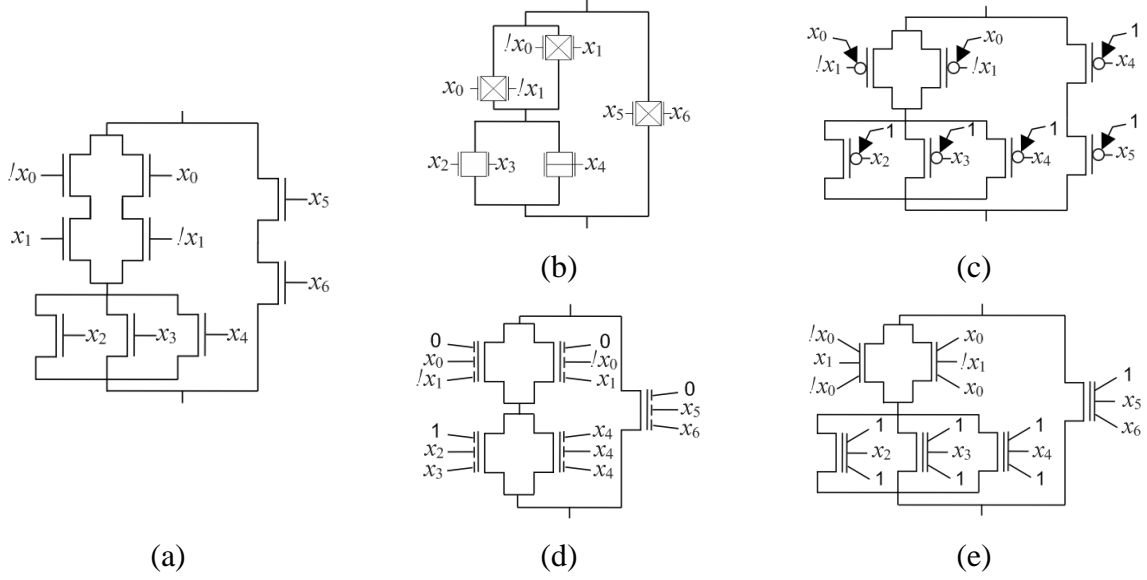


Figure 20: Switch networks corresponding to equation (13): (a) SG devices, (b) IG-FinFET, (c) TIG-FGMOSFET, (d) DG-SiNWFET, and (e) TIG-SiNWFET.

As demonstrated in the example above, at logic level, MIGFETs can be used to minimize the number of logic switches in a given network, so resulting in a more compact circuit (POSSANI *et al.*, 2016). However, at physical level, MIGFET devices tend to be penalized due to extra contact gates, as will be discussed in Chapter 4 (AMARÚ *et al.*, 2015). However, it is expected that the transistor merging performed at logic level compensates the additional area and delay costs when using MIGFET devices (ROSTAMI; MOHANRAM, 2011), (DATTA *et al.*, 2007) and (CHIANG *et al.*, 2006).

3.3.1 Methods for MIGFET-based network generation

In the literature, several works have discussed the use of MIGFET devices to obtain more compact transistor networks (AMARÚ *et al.*, 2015), (ROSTAMI; MOHANRAM, 2011) and (POSSANI *et al.*, 2016). In general, such methods aim to minimize the number of switches of a given network by exploiting the transistor merging performed over a specific type of MIGFET.

In (ROSTAMI; MOHANRAM, 2011), a defactorization technique is adopted to maximize the number of transistor merges performed by IG-FinFETs. In general, the defactorization is applied over a factored expression in order to expand the number of pairs of literals. For instance, by applying the defactorization technique over the following factored equation:

$$f = x_0 \cdot ((x_1 \cdot (x_2 + x_3)) + (x_4 \cdot (x_5 + x_6))) \quad (14)$$

it is possible to obtain the following defactored expression:

$$f = (x_0 \cdot x_1) \cdot (x_2 + x_3) + (x_0 \cdot x_4) \cdot (x_5 + x_6) \quad (15)$$

Notice that this expression has one more literal than the expression shown in (14). However, such an expression benefits the use of IG-FinFET, since the literals are grouping two-by-two through series and parallel associations. In this sense, using SG-FinFET switches, the network implemented directly from the equation (14) comprises five switches, as presented in Figure 21(a). In turn, the switch network obtained from the defactored expression comprises four switches, as depicted in Figure 21(b). Notice that a factored form with a minimum number of literal does not necessarily result in the most compact FinFET-based network (POSSANI *et al.*, 2016).

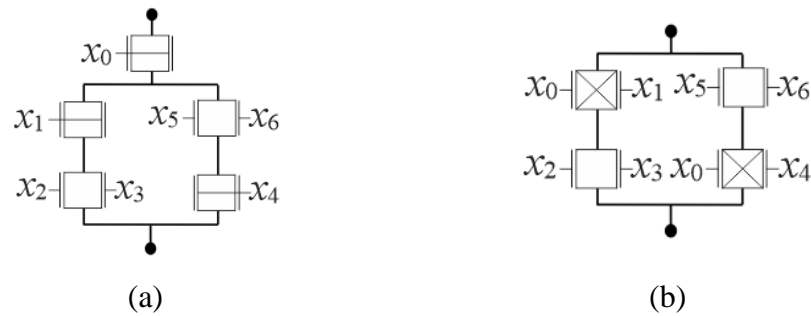


Figure 21: Defactorization technique: (a) original network and (b) defactored one.

The defactorization method proposed in (ROSTAMI; MOHANRAM, 2011) can be used to maximize the number of transistor merges performed over SG-FinFETs. However, such an approach does not ensure the most compact network implementation (POSSANI *et al.*, 2016). This situation can be observed in Figure 22, where series and parallel defactorizations are applied over arbitrary switch networks. A series defactorization occurs when a literal is defactored resulting in a series association of IG-FinFET switches, as illustrated in Figure 22(a). On the other hand, when the defactorization results in a parallel association of IG-FinFET, such an operation is called parallel defactorization, as shown in Figure 22(b). As observed in Figure 22, the networks obtained by defactoring Boolean expressions may result in misleading replication of literals. Notice that more compact networks can be reached by grouping the logic switches in series and parallel logic arrangement. Therefore, in some cases, the use of defactorization technique must be carefully evaluated in order to obtain more compact solutions (POSSANI *et al.*, 2016).

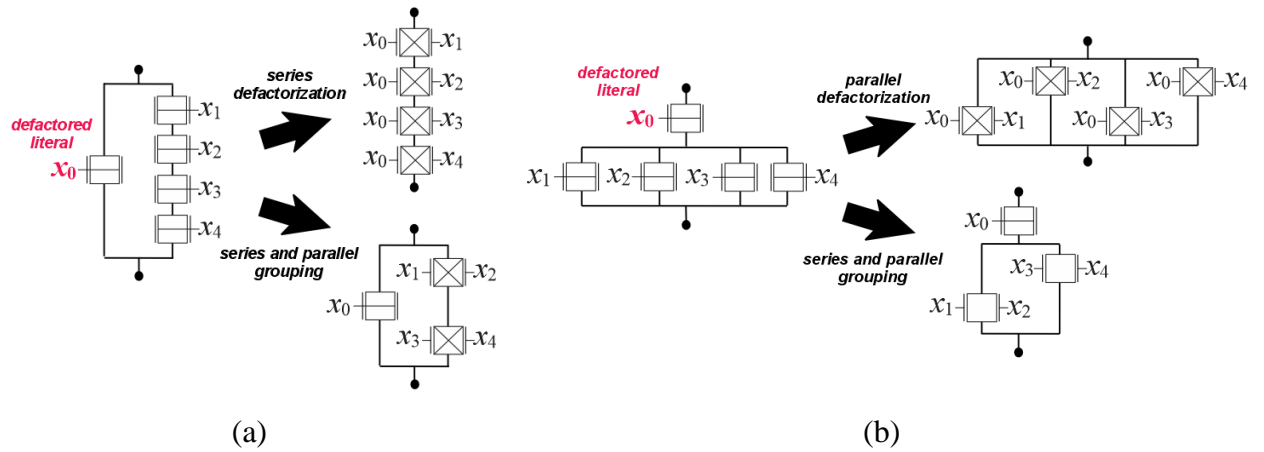


Figure 22 Series and parallel defactorizations: (a) switch network obtained from $f = x_0 + (x_1 \cdot x_2 \cdot x_3 \cdot x_4)$, and (b) switch network obtained from $f = x_0 \cdot (x_1 + x_2 + x_3 + x_4)$
Source: (POSSANI *et al.*, 2016).

In this way, (POSSANI *et al.*, 2014) proposes an alternative defactorization method that enables to merge literals into IG-FinFET switches while avoiding unnecessary replication of literals. In this method, a given factored logic expression is translated into a logic tree that is collapsed in order to maximize the number of pairs of literals. Basically, the method consists of traveling the logic tree, looking for candidate literals to be defactored. This process is based on pattern matching check where, for each operator node of the logic tree, it is evaluated the occurrence of the general pattern presented in Figure 23(a). As it is possible to notice, the pattern comprises four levels, so for each iteration of the method a pattern matching is performed in a portion of the logic tree. This pattern allows to determine when it is appropriate to perform series and parallel defactorizations of literals. The series pattern occurs when the node X is equals to ‘*’ and the node Y is equals to ‘+’. On the other hand, the parallel pattern is defined when the node X is equals to ‘+’ and the node Y is equals to ‘*’. After the pattern matching process, a switch network can be obtained from the resulting logic tree, where pairs of literals are mapped into a series or parallel IG-FinFET, whereas the isolated literals are mapped into SG-FinFET. A simple example is shown in Figure 23(b), where this method is applied over the following logic expression:

$$f = x_0 \cdot (x_1 + (x_2 + x_3 + x_4)) + x_5 \cdot (x_6 + x_7 + x_8) + x_9 \quad (16)$$

In order to collapse the logic tree, the literals are grouped in the leaves nodes of the logic tree. Moreover, in this example, a series and a parallel pattern matching are found. Notice that, after performing the defactorization, all literals are grouped in pairs. In this stage, the logic tree can be mapped into a compact switch network.

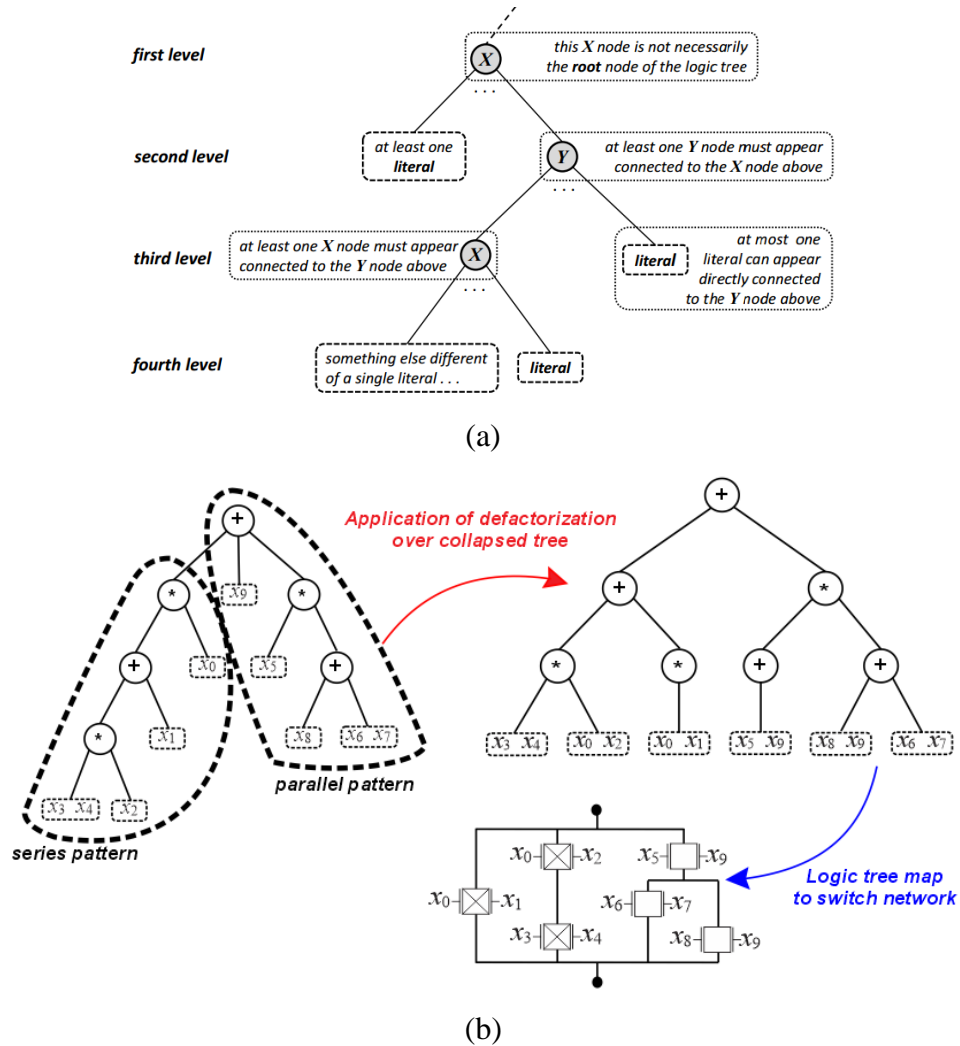


Figure 23: Defactorization method proposed in (POSSANI *et al.*, 2016): (a) series and parallel patterns, and (b) use of defactorization technique over the logic expression $f = x_0 \cdot (x_1 + (x_2 + x_3 + x_4)) + x_5 \cdot (x_6 + x_7 + x_8) + x_9$. Source: (POSSANI *et al.*, 2016).

Additionally, in (POSSANI *et al.*, 2016), it is also proposed an IG-FinFET transistor network generation method. This method receives an ISOP expression as input, which represents a given target Boolean function. In the next, this ISOP is mapped to a set of graph-based structure called SP-kernel (POSSANI *et al.*, 2016). Through the SP-kernel it is possible to find out promisor merges of series and parallel arrangements of literals. Each arrangement found is stored as a local solution. In the final stage, the method aims to group these local solutions in order to avoid redundancies and to perform logic sharing between literals whenever possible. As output the method produces an optimized switch network, where the use of IG-FinFET is maximized without performing defactorization.

The methods proposed in (ROSTAMI; MOHANRAM, 2011) and (POSSANI *et al.*, 2014) are specific for IG-FinFET. Therefore, in both cases, only series and parallel arrangements of transistors can be explored. In this sense, a more general synthesis method

for MIGFET devices is proposed in (AMARÚ *et al.*, 2015). This method can be applied to all MIGFET devices discussed herein. In this approach, a given circuit description and a k -input MIGFET, with its respective Boolean functionality (f_M), are used as input. In the sequence, the circuit is mapped into k -LUTs, where k is defined according to the number of variables of f_M . Such a restriction ensures that the size of each LUT is equal to or larger than the number of inputs of the MIGFET. In the next stage, each LUT is mapped to a logic gate through of a match/decompose strategy. An important observation for this method is that a logic gate is defined only through the PD plan, since the PU plan is built from an auxiliary device, similar to pseudo-NMOS logic style (SUTHERLAND; SPROULL; HARRIS, 1999).

A biconditional binary decision diagram (BBDD) is employed to perform the logic matching between the Boolean function implemented into a given LUT, and the f_M (plus NPN variations of f_M) (AMARÚ; GAILLARDON; DE MICHELI, 2014). Therefore, always that a LUT is mapped into a logic gate, firstly, it is performed a logic check in order to verify if the Boolean function implemented into the LUT can be implemented by the chosen MIGFET. This logic matching is performed from a simple pointer comparison in BBDD (AMARÚ *et al.*, 2015). When the result of the matching is negative, the Boolean function implemented by LUT is decomposed iteratively until the obtained cofactors can be built through the chosen MIGFET. The decomposition strategy is based on a generalized expansion and the Shannon's expansion (DE MICHELI, 1994). An example of logic gate obtained from the application of this method is presented in Figure 24. Notice that, in this case, the MIGFET used in the example is a TIG-FGMOSFET ($f_M = \text{MAJ3}$), and the Boolean function implemented by the logic gate is the following:

$$f = (x_0 \cdot x_1 \cdot !x_2) + (x_1 \cdot x_2 \cdot !x_3) + (x_0 \cdot x_2 \cdot !x_3) \quad (17)$$

Notice that the given Boolean function cannot be implemented into a single MIGFET and then the decomposition is needed. The cofactors obtained from this process are $\{\bar{f}_{f_M} = x_2 \cdot x_3; \bar{f}_{\bar{f}_M} = 1\}$. Notice that these cofactors result from the generalized expansion of f in relation to f_M . The positive basis is defined as $x_2 \cdot x_3$, which can be implemented through TIG-FGMOSFET just by setting one of the inputs to 0. On the other hand, the negative basis, which is defined as 1, just requires a direct connection to GND terminal.

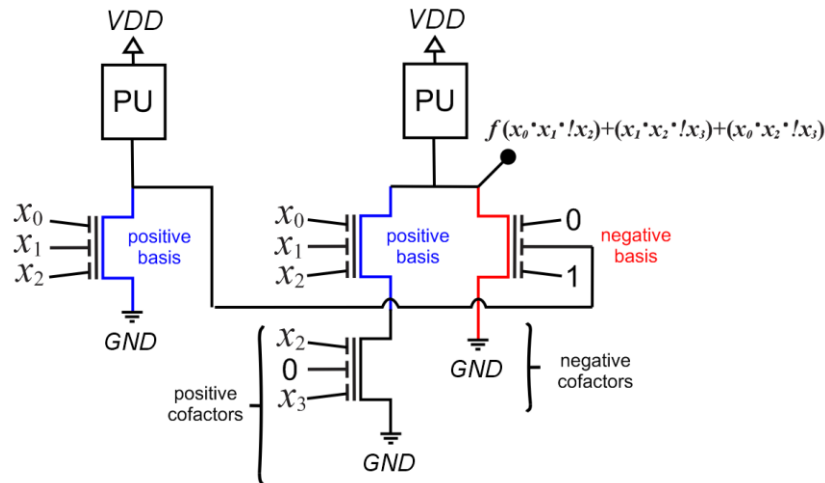


Figure 24: Logic gate implementation for $f = (x_0 \cdot x_1 \cdot !x_2) + (x_1 \cdot x_2 \cdot !x_3) + (x_0 \cdot x_2 \cdot !x_3)$ using TIG-FGMOSFET, where $f_M = (x_0 \cdot x_1) + (x_0 \cdot x_2) + (x_1 \cdot x_2)$. Source: (AMARÚ *et al.*, 2015).

Through the method proposed in (AMARÚ *et al.*, 2015), it is not possible to ensure the most optimized logic switch implementation for a given MIGFET type. However, the method is robust enough to ensure the correct synthesis for a great set of MIGFET devices. This method was implemented into a framework (LSI, 2017), in which it is possible to obtain estimations of area, delay and power consumption in respect to the mapped circuit.

Besides the synthesis methods for MIGFET devices, discussed before, new logic structures can offer advantages when used together with a given type of MIGFET device, as the case of BBDD. In particular, as presented in (GAILLARDON *et al.*, 2014), BBDD directly supports the behavior of DG-SiNWFET. Such a logic structure has demonstrated powerful properties when used in one-pass synthesis (OPS). In OPS flow, the logic optimizations and technology mapping task are carried on a common data structure. This data structure can efficiently represent the logic behavior and physical implementation of the target circuit (TENACE *et al.*, 2015). Therefore, BBDD can bring advantages to logic synthesis of DG-SiNWFET since that it can be translated directly to a one-pass circuit composed by DG-SiNWFET. Hence, all optimizations performed at BBDD level can be mapped directly to the circuit level. Such a concept is illustrated in Figure 25, where it is possible to observe the direct relationship between a BBDD node and a transistor network implemented using DG-SiNWFET devices.

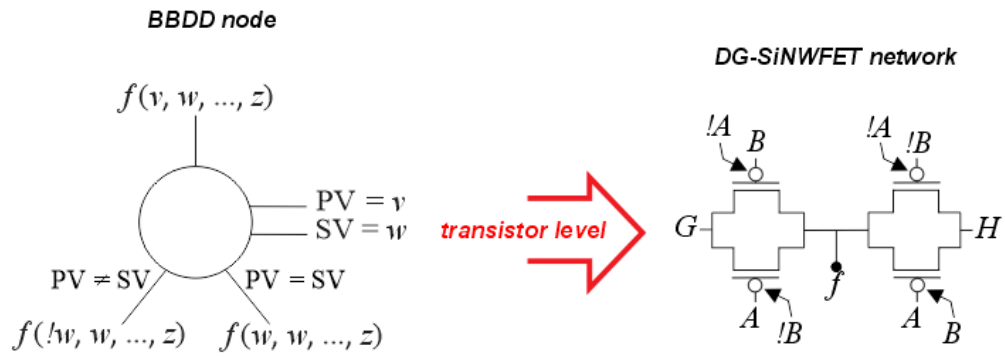


Figure 25: Direct relationship between BBDD node and transistor network composed by DG-SiNWFET. Source: (GAILLARDON *et al.*, 2014).

4 PHYSICAL AND ELETRICAL ISSUES

As discussed in the previous chapter, the higher logic capability into a single MIGFET may reduce the number of SG devices to perform equivalent logic behavior. However, the signal delay propagation and area (size) of a single MIGFET device tends to be larger than of SG devices. Area overhead is consequence of more complex signals routing, since the number of control gates in a given MIGFET increases according to the Boolean function implemented on such a transistor. As presented in (AMARÚ *et al.*, 2015), the physical costs of a MIGFET tends to increase according to the number of independent gates in the device. In this sense, there is a tradeoff between the number of SG transistors merging performed into a given MIGFET and its area/delay cost. Thus, for a given target circuit, if there are not enough simplifications in logic level to compensate the use of a more complex switch, the adoption of MIGFET devices may lead to design losses and penalties.

In order to demonstrate the physical impacts of MIGFETs, in particular the area overhead, in this section is presented a brief study about MIGFET layout. As some MIGFET technologies are more mature than others, we assume a general set of design rules, inspired from (ALIOTO, 2011), (BOBBA *et al.*, 2012) and (LOPEZ-MARTIN *et al.*, 2013). In Table 1, it is shown the design rules used for each MIGFET. Notice that a common technology node is considered for all devices.

Through the MIGFET layouts produced herein, we define evaluation methods to estimate the area and delay of circuits implemented using these devices. Such evaluation approaches are adopted in the experiments discussed in the next chapters in order to characterize the target benchmark circuits.

Another important observation is that design rules adopted in this work cannot be used in real fabrication of devices discussed herein. However, such a design rules give an insight about MIGFET physical impacts in the design of digital integrated circuits. In particular, the layouts drawn demonstrate how the area overhead and signal routing complexity increase according to the logic capability of a given MIGFET.

Table 1: Fictitious design rules adopted here for different MIGFET technologies (45 nm)

Design rules description		Values (nm)
General rules	Contact size	60
	Minimum metal length	70
	Minimum polysilicon length	45
	Minimum contact enclosure by poly	12
	Minimum contact enclosure by metal	5
	Minimum contact enclosure by diffusion	15
	Minimum poly-to-poly spacing	45
	Minimum metal-to-metal spacing	70
SG-FinFET and DG-FinFET	Minimum SG-FinFET fin pitch	120
	Minimum DG-FinFET fin pitch	174
	Minimum poly-to-fin spacing	30
	Minimum silicon thickness	30
DG-SiNWFET and TIG-SiNWFET	Minimum SiNW pitch	65.6
	Minimum silicon thickness	25.1
	Minimum poly-to-poly spacing	20
TIG-FMOS	Minimum poly2 length	135
	Minimum enclosure poly2 by poly	12
	Minimum spacing poly2-to-poly2	10

4.1 MOS and emerging transistor layouts

The dimensions of MOSFET are defined according to width (W) and length (L) of the transistor. In general, W and L values depend on the technology node adopted. In particular, L value is defined according to length of the polysilicon gate, whereas W value is defined according to electrical current value required in the transistor (drive strength). Moreover, as noticed in Figure 26(a), in the layout of MOSFET, W and L values can be defined of continuous way.

On the other hand, in emerging transistors, as FinFET and SiNWFET, the drive strength of the transistor is defined according to the number of fins (N_{fin}) or the number of silicon nanowires (N_{SiNW}) adopted in the transistor structure, as illustrated in Figure 26(b). In this sense, for these technologies the drive strength of a given transistor is defined in a discrete way. According to (ALIOTO, 2011), the relationship between N_{fin} and W of a FinFET device can be defined through the following relationship, being similar to SiNWs:

$$N_{\text{fin}} = \frac{W}{W_{\text{min}}} \quad (18)$$

where the W_{min} is the minimum channel width of the transistor.

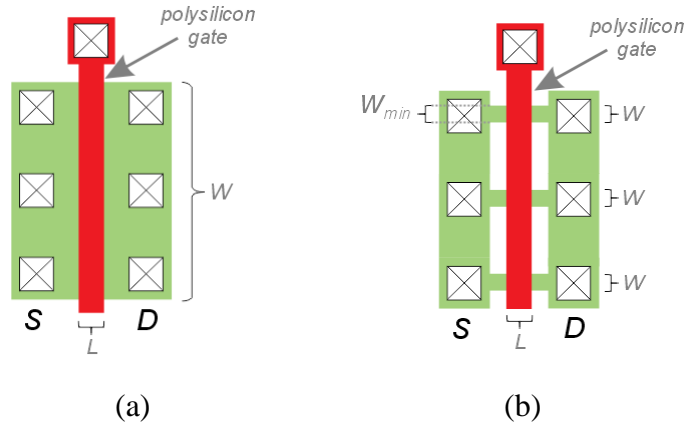


Figure 26: Transistor layout: (a) MOSFET, and (b) FinFET and SiNWFET

4.1 FINFET layout

As discussed previously, the SG-FinFET structure is very similar to IG-FinFET structure, since both devices are built through a vertical fin that connects drain (D) and source (S) terminals. The physical dimensions of the fin are the height (h_{fin}), length (L_{fin}) and silicon thickness (T_{si}) (ALIOTO, 2011).

In particular, an IG-FinFET can be ideally obtained from SG-FinFET by cutting the upper part of the gate (G) of the SG-FinFET, splitting the gate into a front gate (FG) and a back gate (BG) parts, as depicted in Figure 27. Due to this feature, both SG-FinFET and DG-FinFET devices can be implemented on the same die (LIU *et al.*, 2007), (TAWFIK; KURSUN, 2008), (CUI *et al.*, 2015) and (ALIOTO, 2011).

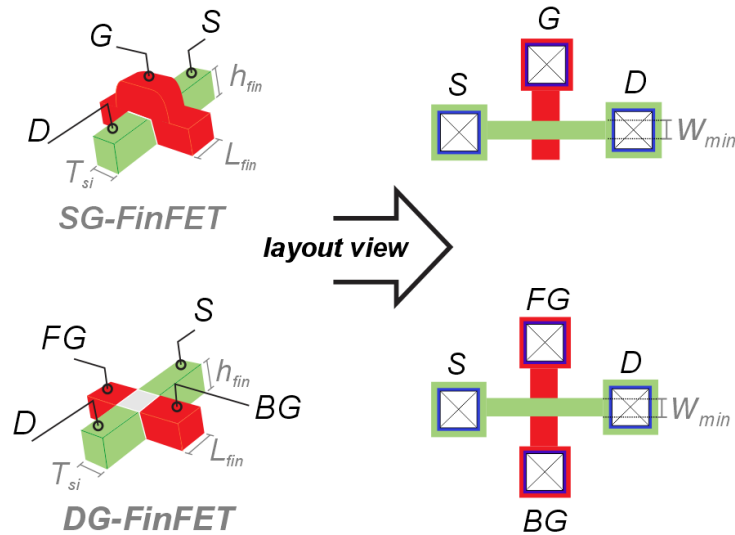


Figure 27: Dimension parameters for SG-FinFET and DG-FinFET and layout representation.

Similar to MOS transistor, the channel length of a FinFET device is defined according to L_{fin} . In turn, W_{min} is twice the L_{fin} (HUANG *et al.*, 1999). Therefore, a FinFET device with a single fin represents a transistor with W_{min} . Additionally, the values for the transistor channel width (W) can be increased according to the number of fins (N_{fin}) connected in parallel (COLINGE, 2008). In such a configuration, the lateral diffusions of fins associated in parallel are shared (ANIL; HENSON; BIESEMANS, 2003), as illustrated in Figure 28. Notice that the minimum distance between two different fins is called *fin pitch*.

In Figure 28(a) and in Figure 28(b) are presented the layout of a SG-FinFET and an IG-FinFET with N_{fin} equals to 3. Notice that, in IG-FinFET, each fin is controlled by two signal gates (FG and BG) whereas, in SG-FinFET, all fins are controlled through a single signal gate (G). Moreover, as shown in Figure 28, the area of an IG-FinFET tends to be larger than a SG-FinFET. In IG-FinFET, it is necessary to place contact gates between parallel fins, so expanding the fin spacing and resulting in more complex routing and larger transistors (CHIANG *et al.*, 2005).

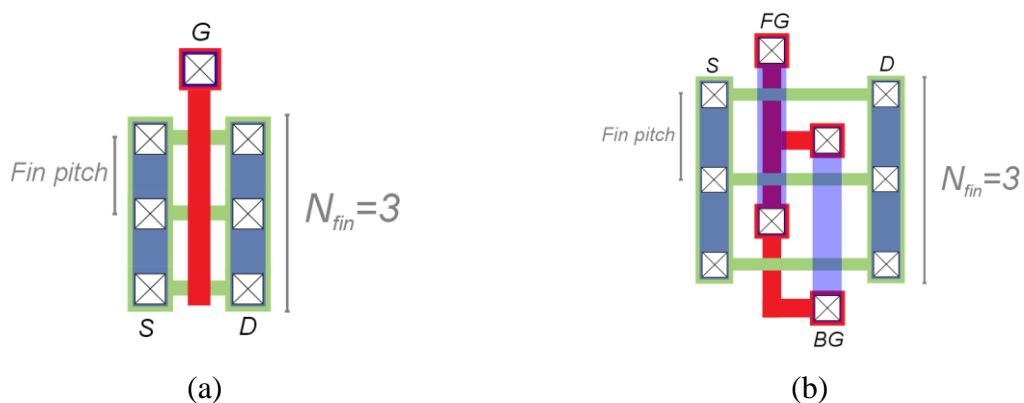


Figure 28: Layouts for FinFET devices with N_{fin} equals to 3: (a) SG-FinFET and (b) IG-FinFET.

In order to demonstrate as the multiples contact gates may increase the routing complexity and the transistor area, folding technique was applied on the transistors shown in Figure 29. The folding technique consists in breaking a large transistor into smaller ones (legs) that are connected in parallel in order to reduce the height of the transistor and to increase the width (CORTADELLA, 2013). Both SG-FinFET and IG-FinFET shown in Figure 29(a) and Figure 29(b), respectively, have width equals to 6. However, due to the application of folding, only three fins are necessary to build each transistor. Notice that, in SG-FinFET, the extra area is consequence of the number of poly fingers used. In turn, in DG-SiNWFET, besides the number of poly fingers, the additional area is associated to the number of contact gates in the transistor. Notice that the metal lines used to route the contact gates between parallel fins increase the spacing between poly fingers, resulting in a larger IG-FinFET.

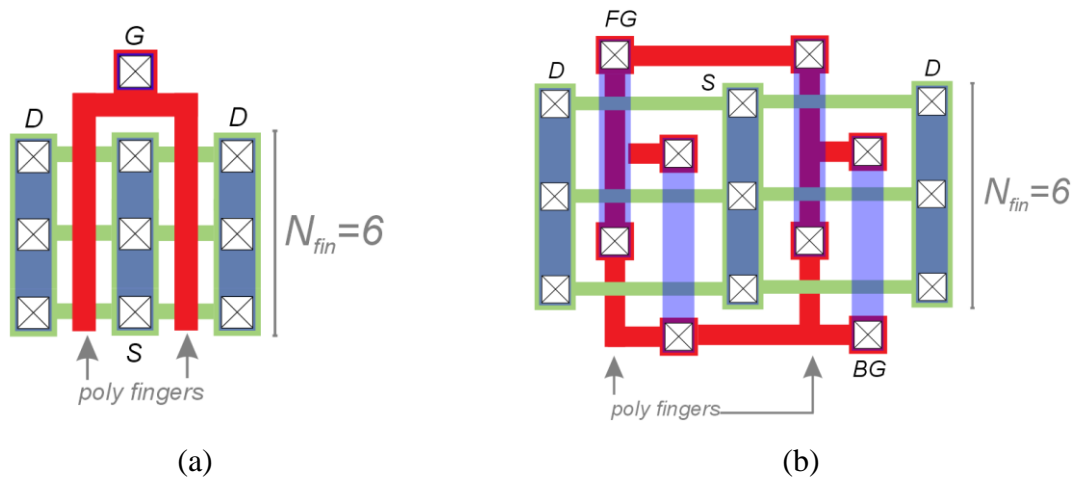


Figure 29: Example of application of folding technique on two-finger FinFETs: (a) SG-FinFET, and (b) DG-FinFET.

The IG-FinFET layout pattern presented in Figure 28(b) and in Figure 29(b) can be used in the manufacturing of IG-FinFET-LV_{th} and IG-FinFET-HV_{th} (AGOSTINELLI; ALIOTO; SELMI, 2010). Another possibility adopted in the literature is the use of the device in back-gate biasing mode, where the transistor threshold voltage is scaled by setting the BG voltage (CUI *et al.*, 2015), (BHOJ; JHA, 2013) and (HSIEH *et al.*, 2012). For instance, in static back-gate biasing mode, the BG of IG-FinFET is biased reverse to VDD (P-type) or GND (N-type) voltage. This technique allows to increase the threshold voltage and to reduce the leakage current at the expense of an increasing delay (AGOSTINELLI; ALIOTO; SELMI, 2010).

4.2 SiNWFET layout

Controllable polarity transistors can be fabricated by using materials like carbon nanotubes (KOZIOL *et al.*, 2007), graphene (SORDAN; TRAVERSI; RUSSO, 2009) and silicon nanowires (SiNW) (DE MARCHI *et al.*, 2012). Among these possibilities, SiNW offers better controllability of polarity and enables the fabrication of transistors in vertical stacks (AMARÚ; GAILLARDON; DE MICHELI, 2013). The vertical stack transistor contributes with the circuit area reduction since larger transistors impact in the height of the chip.

SiNW are used to build DG-SiNWFET and TIG-SiNWFET. For both transistor structures, SiNW are divided in three parts that are polarized by gate-all-around regions as presented in Figure 30. The center gate region controls the transistor channel. On the other hand, the side gate regions define the transistor polarity, N- or P-type (BOBBA *et al.*, 2012). The physical dimensions of SiNW are the radius of the nanowire (R_{nw}), the oxide thickness (T_{ox}), the length of control gate (L_{CG}), the length of polarity gate at source (L_{PGS}), the length of polarity gate at drain (L_{PGD}), and the length of spacer (L_{cp}) (MOHAMMADI; GAILLARDON; DE MICHELI, 2015). Similarly to FinFET device, where the width of the transistor is defined according to number of fins in parallel, larger DG-SiNWFET and TIG-SiNWFET are obtained by increasing the number of SiNW (N_{SiNW}) in parallel. In Figure 30, it is possible to see a DG-SiNWFET and a TIG-SiNWFET with W_{min} , whereas in Figure 31 are shown DG-SiNWFET and TIG-SiNWFET with N_{SiNW} equals to 6 (BOBBA *et al.*, 2015).

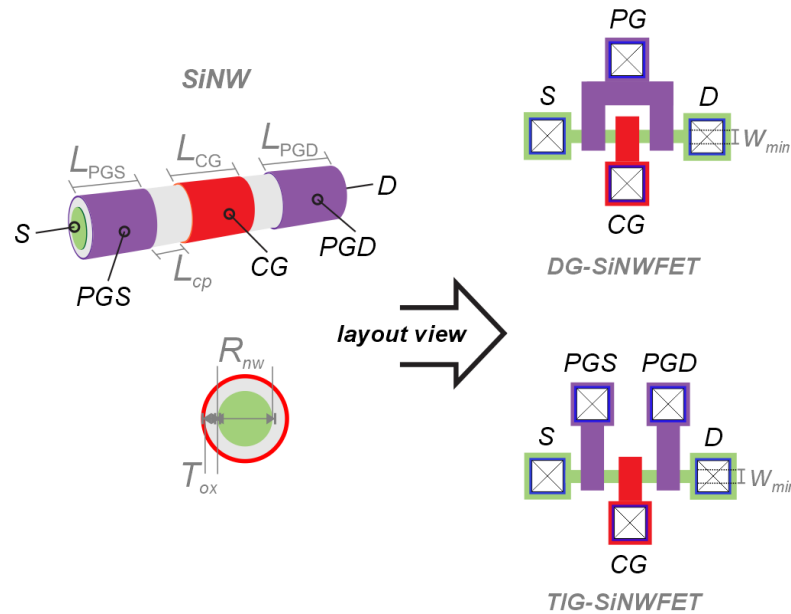


Figure 30: Physical dimensions for DG-SiNWFET and TIG-SiNWFET, and layout representation considering minimum W .

DG-SiNWFET and TIG-SiNWFET present quite similar physical structures, since the main difference between both transistors is the use of the polarity gate (PG), which can be built to merge the PGS and PGD. Such a structural similarity is also presented at layout level, as illustrated in Figure 31. Notice that, differently from IG-FinFET, the extra contact gates used to obtain more logic capability are not inserted between the SiNW spacing. In this way, the signal routing complexity for DG-SiNWFET and TIG-SiNWFET tends to be lower than for IG-FinFET.

When the folding technique is applied on DG-SiNWFET, as shown in Figure 32(a), it is necessary to replicate only the lines of poly in order to parallelize the transistors. In turn, when TIG-SiNWFET is parallelized, as illustrated in Figure 32, additional lines of metal are required to connect the respective PGS and PGD poly fingers. In this case, due to the design rules, the spacing penalty leads to the transistor area duplication. Notice that, due to folding technique, both transistors shown in Figure 32 are built from three parallel SiNW instead of six ones.

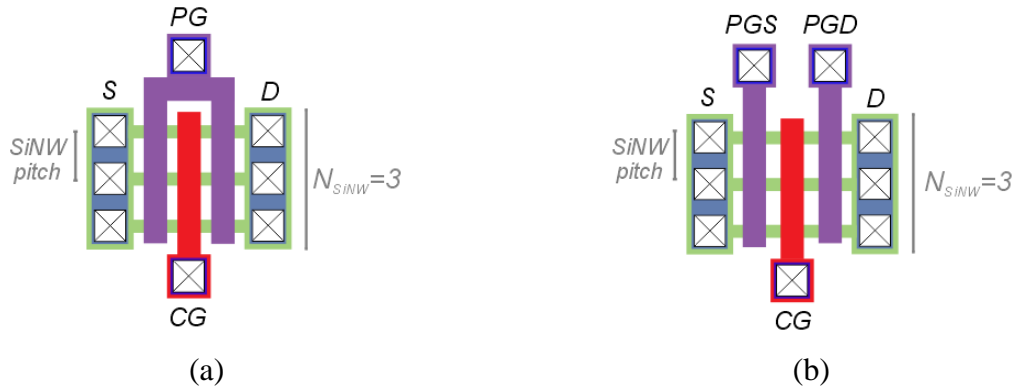


Figure 31: Layouts of SiNWFETs with N_{SiNW} equals to 3: (a) DG-SiNWFET, and (b) TIG-SiNWFET.

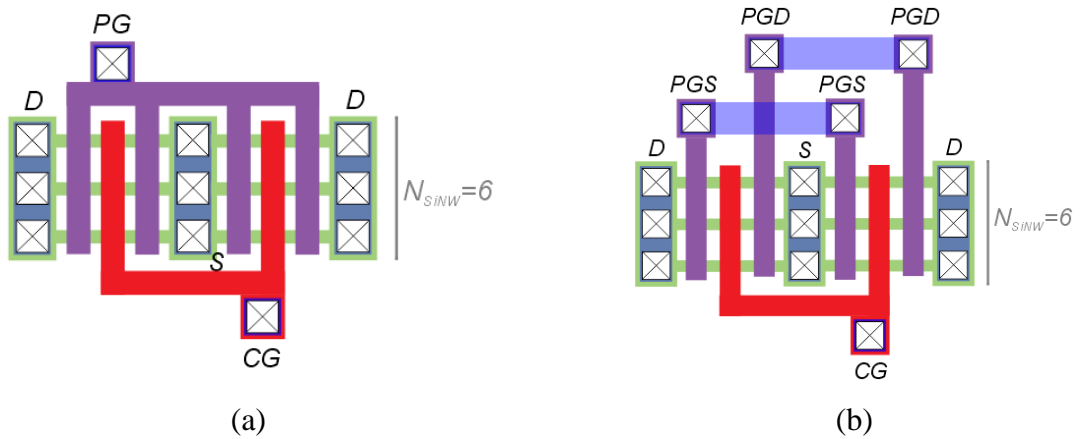
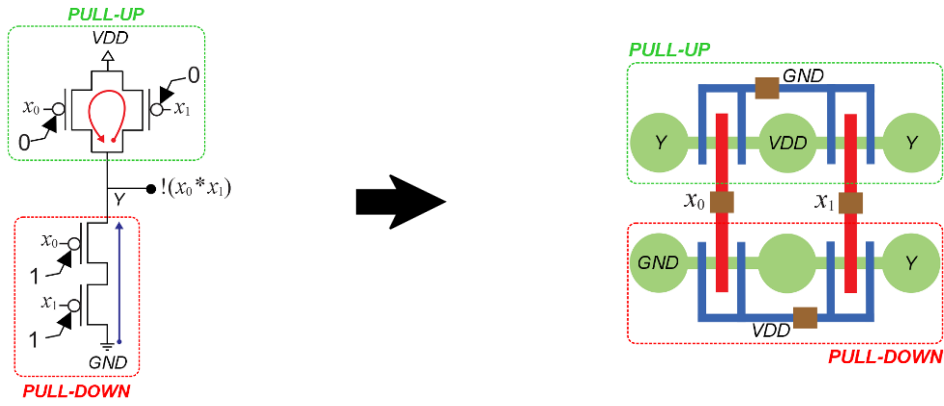


Figure 32: Example of application of folding technique on two-finger SiNWFET devices: (a) DG-SiNWFET, and (b) TIG-SiNWFET.

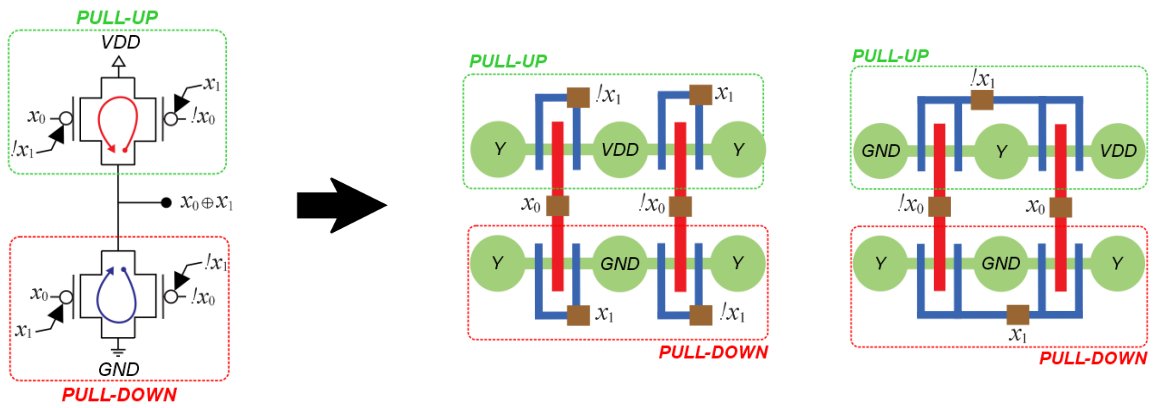
As discussed previously, both DG-SiNWFET and TIG-SiNWFET are ambipolar devices, and so can be configured to implement P-type and N-type transistors. It is worth to notice that those configurations do not bring additional cost for the layouts, because P-type and N-type transistors are obtained through the configuration of the contact gates signals. For instance the DG-SiNWFET is configured as a P-type (N-type) transistor by fixing the PG to GND (VDD) voltage, whereas a P-type (N-type) TIG-SiNWFET is configured by fixing the PGS and PDS to GND (VDD) voltage. Notice that different Boolean functions and transistor configurations can be reached by DG-SiNWFET and TIG-SiNWFET by just configuring the input signals in the transistor contact gates.

As discussed in the literature, DG-SiNWFET and TIG-SiNWFET enable the building of layouts with high regularity (BOBBA *et al.*, 2015), (ZHANG *et al.*, 2017), (GAILLARDON *et al.*, 2014). In general, the regular layout fabrics have the advantage of higher yield as they maximize layout manufacturability (JHAVERI *et al.*, 2006). For instance, in (BOBBA *et al.*, 2015), proposed a new design methodology to explore the regularity

offered by ambipolar devices, such as DG-SiNWFET and TIG-SiNWFET. Such a methodology is called sea-of-tiles (SoT) and allows that Boolean functions can be mapped into an array of logic tiles, which are uniformly spread across the chip. From this approach, the transistors are paired according to their polarity in order to obtain an efficient ambipolar circuit. An example of this technique is illustrated in Figure 33, where the layout of a NAND2 gate is shown in Figure 33 (a) whereas, in Figure 33 (b), two possible implementations of XNOR2 gate are presented. Notice that in these examples, each transistor represents a DG-SiNWFET. Notice also that the layouts presented in Figure 33 are based on dumbbell-stick diagram (DSD) (BOBBA *et al.*, 2012). In DSD, the size of the transistors is not taken into account but just the topology of the interconnection. DSD is a simplified layout abstraction used to study the circuit-routing complexity, quite similar to CMOS stick diagram (MEAD; CONWAY, 1980).



(a)



(b)

Figure 33: Examples of DSD for DG-SiNWFET devices: (a) NAND2 gate layout, and (b) possible layouts for XOR2 gate. Source: (BOBBA *et al.*, 2015).

4.3 FGMOSFET layout

TIG-FGMOSFET is built through three independent control gates (CG_1 , CG_2 , CG_3), which are capacitively coupled over a floating gate (FG). Each control gate forms a poly-to-poly capacitor with the FG, as shown in Figure 34. Due to electrical charge conservation, the FG voltage is defined as follows:

$$V_{FG} = \frac{1}{C_T} (C_{CG1} V_{CG1} + C_{CG2} V_{CG2} + C_{CG3} V_{CG3} + C_S V_S + C_D V_D + C_B V_B + Q_0) \quad (19)$$

where the $C_T = (C_{CG1} + C_{CG2} + C_{CG3} + C_S + C_D + C_B)$ and Q_0 represents the initial charge in the FG produced during the fabrication process of the transistor (LOPEZ-MARTIN; CARVAJAL, 2008). Q_0 has direct impact in the electrical behavior of TIG-FGMOSFET since the connection between drain (D) and source (S) terminals of the transistor is determined as a function of the weighted sum of voltages applied to CG_1 , CG_2 and CG_3 (MONDRAGÓN-

TORRES; SCHNEIDER; SÁNCHEZ-SINENCIO, 2002). In this sense, (RODRIGUEZ-VILLEGAS; BARNES, 2003) proposes a technique to avoid such an initial charge Q_0 , where a dummy stacked contact is created on the FG. This dummy stacked contact includes all the contact layers, and is used to discharge the initial charge Q_0 (LOPEZ-MARTIN *et al.*, 2013). The layout of TIG-FGMOSFET with the dummy stacked contact is shown in Figure 35.

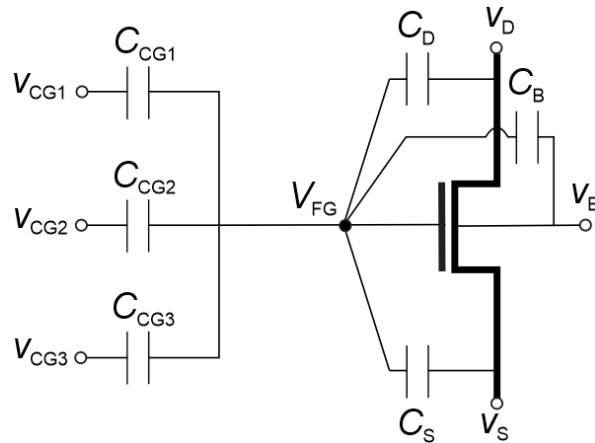


Figure 34: Equivalent circuit for TIG-FGMOSFET transistor.

Similar to other transistors, previously discussed, the electric current capacity of a TIG-FGMOSFET transistor is defined according to its width (W). However, differently from FinFET and SiNWFET devices, the minimum width (W_{\min}) of TIG-FGMOSFET is not defined from the width of a fin or a SiNW. In this case, the minimum width is evaluated through of the width of the transistor substrate. In general, this value is defined according to the specifications of the design. In the layout shown in Figure 35(a), W_{\min} is defined as the size of a contact plus the enclosure of contact to the substrate. Consequently, larger values of W can be reached to increase the number of contacts in parallel on the diffusion regions, as shown in Figure 35(b). Notice that the transistor presented in Figure 35(c) is obtained from the application of folding technique on a TIG-FGMOSFET transistor with W equals to 6. Notice also that the folding technique has low impact on the final area of the TIG-FGMOSFET since the most part of this area is used by FG and the control gates, which are not affected by transistor folding.

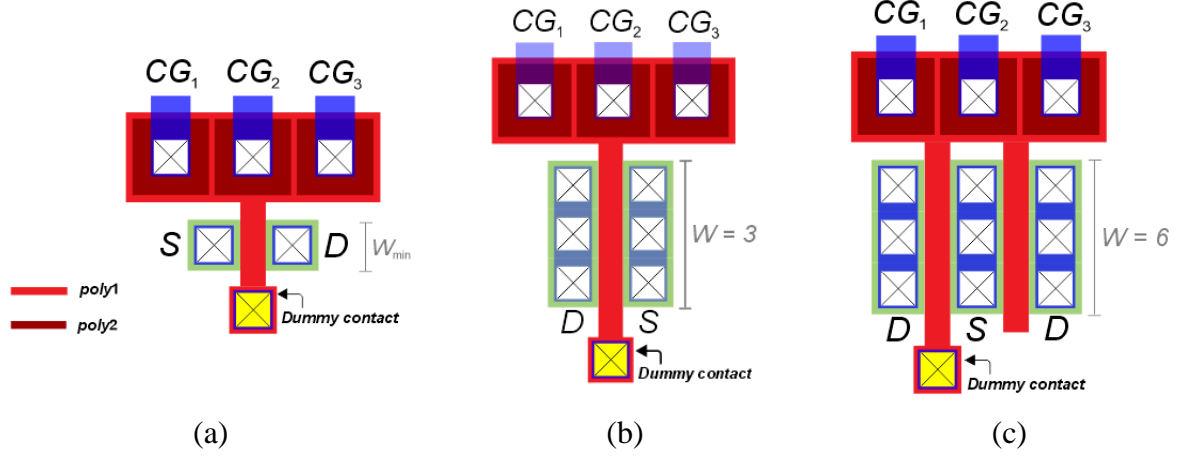


Figure 35: Layouts for TIG-FGMOSFET transistors: (a) TIG-FGMOSFET with W_{\min} , (b) TIG-FGMOSFET with W equals to 3, and (c) folding transistor technique.

Differentially from other MIGFET technologies, presented in this section, FGMOSFET has few applications in digital circuits. In general, it is used to build storage cores, such as flash memories (BEZ *et al.*, 2003). On the other hand, in analog circuits, FGMOSFET is better explored in several applications, like D/A converts (YIN; SANCHEZ-SINENCIO, 1997), multiple input amplifier (YANG; ANDREOU, 1993), low-voltage operations (MINCH, 2000) and electronic programming (HASLER; MINCH; DIORIO, 1999).

4.4 Evaluation of MIGFET area

In order to estimate the physical area value for each MIGFET adopted, a particular evaluation has been developed. In this context, the area of a given MIGFET is estimated according to its respective layout pattern. In such an area analysis approach, the folding transistors are not being considered. Moreover, we compute the area of a given MIGFET in respect to its respective SG version. This way, we are assuming that, for each MIGFET technology is possible to provide a SG transistor version. In the adopted area evaluation, each MIGFET layout is represented as a circuit. This way, the estimated area value is computed as the product between the height and length of the circuit layout, such as illustrated in Figure 36. In particular, according to the transistor technology, the area of a given transistor can be estimated in terms of W , N_{fin} or N_{SiNW} . Thus, in order to define a common unit to represent such parameters, in our area analysis, larger transistors are obtained by increasing the device basic unit (DBU), as indicated in Figure 36. In Table 2, it is shown the estimated area value for different MIGFET considering distinct DBU values.

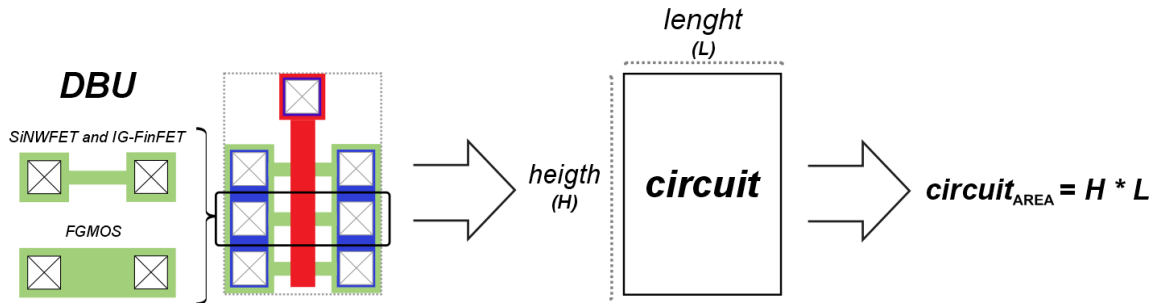


Figure 36: Circuit area estimation.

Table 2: Estimated area values, in nm², for different MIGFET with DBU from 1 to 6.

Technologies	DBU					
	1	2	3	4	5	6
SG-FinFET	74034.675	119594.5	165154.3	210714.1	256273.9	301833.7
IG-FINFET	108204.525	206917.4	305630.3	404343.2	503056.1	601769.0
SG-SiNWFET	74034.675	108204.5	142374.4	176544.2	210714.1	244883.9
DG-SiNWFET	90601.875	126842.6	163083.4	199324.1	235564.9	271805.6
TIG-SiNWFET	93190.5	130466.7	167742.9	205019.1	242295.3	279571.5
SG-FGMOSFET	107686.8	132537.6	157388.4	182239.2	207090.0	231940.8
TIG-FGMOSFET	179478	220896.0	262314.0	303732.0	345150.0	386568.0

From the estimated area values presented in Table 2, it is possible to defined two approximated area increasing factors, λ and Φ . The λ factor represents the proportion of extra area that a MIGFET device presents in respect to its SG version. In turn, the Φ factor represents the extra area provide to increase the DBU of a given MIGFET. The values of λ and Φ factors for different MIGFET technologies are presented in Table 3. Notice that the λ value for a given MIGFET is normalized in respect to its SG version. Additionally, λ and Φ factors can be used to compute the estimated area for any MIGFET ($MIGFET_A$) with arbitrary DBU value, in relation to the area value of its SG version (SG_A), which presents minimum area ($DEVICE_{min}$), as follows:

$$MIGFET_A(DEVICE_{min}) = \lambda * SG_A(DEVICE_{min}) \quad (20)$$

$$MIGFET_A(DBU) = MIGFET_A(DBU_{min}) + (\Phi * (DBU - 1)) \quad (21)$$

Table 3: Values for area increasing factors, λ and Φ , for different MIGFET technologies.

Technologies	Φ	Λ
SG-FinFET	0.51	1.00
IG-FINFET	0.76	1.82
SG-SiNWFET	0.38	1.00
DG-SINWFET	0.33	1.15
TIG-SINWFET	0.33	1.18
SG-FGMOSFET	0.19	1.00
TIG-FGMOSFET	0.19	1.67

4.5 RC modeling of MIGFET

In order to determine a common way to estimate the performance of MIGFET devices, we have adopted a simplified RC modeling. Similarly to area analysis, the delay estimation of a given MIGFET is based on the characteristics of a SG transistor version at the same MIGFET technology. In this sense, such SG version is used as reference in delay analysis.

4.5.1 RC modeling

To model a transistor as an RC network, the transistor is represented through a resistance (R_{on}), which connects the source (S) and drain (D) contact terminals, when the device is *on*. However, each contact terminal presents capacitors (C_S and C_D) in the transistor model. In particular, in the modeling adopted in this work, equal capacitance values are defined for C_S and C_D . This way, these capacitances will be denoted as diffusion capacitance (D_C). Moreover, input capacitors (C_{in_i}) are associated to control gates (CG) of the transistor. In Figure 37, they are shown different transistor models used to represent the MIGFETs discussed in this work. In particular, the transistor model shown in Figure 37(a) is used to represent a SG transistor. In turn, the transistor model shown in Figure 37(b) is adopted to represents IG-FinFET and DG-SiNWFET. Finally, the transistor model presented in Figure 37(c) is used to represent TIG-SiNWFET and TIG-FGMOSFET.

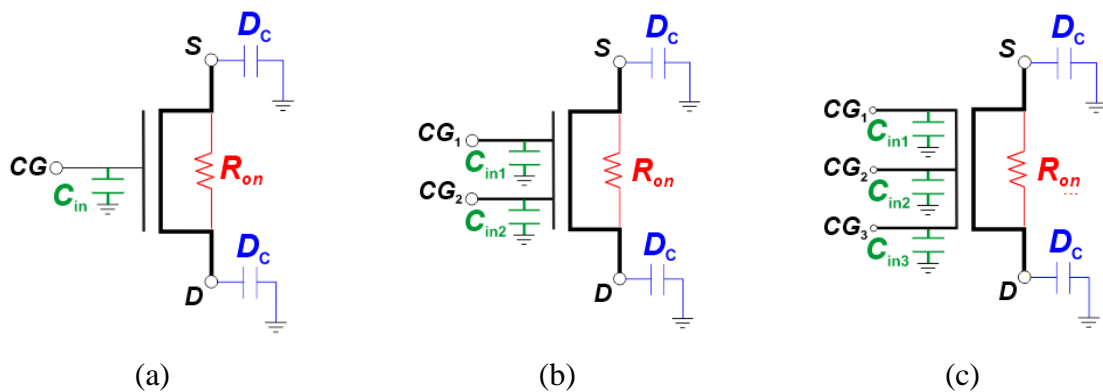


Figure 37: Transistor models: (a) 1-control gate, (b) 2-control gate, and (c) 3-control gate.

In order to estimate the performance of a given MIGFET, R_{on} , C_{in_i} and D_C values must be defined. In particular, the capacitance value defined for D_C of a given MIGFET is equivalent to D_C value assigned to the SG version of respective MIGFET. On the other hand, R_{on} and C_{in_i} values applied at each transistor model are proportional to the number of control gates that must be active for transistor conduction. In some cases, the threshold value of the

transistor must be also considered to determine R_{on} and C_{in_i} . Therefore, distinct values of R_{on} and C_{in_i} can be adopted according to the particular MIGFET evaluated. R_{on} and C_{in_i} values for each MIGFET device modeling will be discussed in more detail hereafter. Notice that the capacitances and resistances on the transistors modeling are susceptible to the DBU variations. This way, according to a specific DBU value, the R_{on} , C_{in_i} and D_C of a given transistor can be defined as follows:

$$R_{on}(DBU) = \frac{R_{on}}{DBU} \quad (22)$$

$$C_{in_i}(DBU) = C_{in_i} * DBU \quad (23)$$

$$D_C(DBU) = D_C * DBU \quad (24)$$

4.5.2 RC modelling of IG-FinFET

As presented previously, IG-FinFET can implements AND2, NAND2, OR2 or NOR2 operations, according to IG-FinFET technology, high- V_{th} or low- V_{th} , and the type of device, P- or N-type. However, notice that these operations are obtained through the same layout pattern. In particular, in IG-FinFET-HV_{th}, the inversion channel is formed only when the two control gates of the transistor are active. In turn, in IG-FinFET-LV_{th}, one control gate active ensures the transistor inversion channel.

Thus, in the IG-FinFET-LV_{th} modeling, the R_{on} parameter varies according to control gate configuration. When only a control gate is active, the R_{on} value is similar to the one in SG-FinFET. However, when the two control gates are active R_{on} is the half of the resistance of a SG-FinFET, since the threshold value to active the transistor is reached faster. In contrast, the R_{on} value in the IG-FinFET-HV_{th} modeling is twice larger than the resistance of a SG-FinFET due to the higher threshold value of the transistor.

Additionally, the C_{in_i} value is defined according to the height and length of a transistor fin. In (ALIOTO, 2011), the height of a fin can be defined as twice the fin length. Moreover, the contact area between the fin of a SG-FinFET and its control gate terminal tends to be larger than in IG-FinFET, since this device is similar to SG-FinFET by cutting upper part of the control gate of the transistor (LIU *et al.*, 2007). Therefore, we assumed that the C_{in_i} value for an IG-FinFET is lower than the input capacitance value of a SG-FinFET. In

particular, when $CG_1 \neq CG_2$, the R_{on} in the IG-FinFET-LV_{th} is defined according to following equation:

$$R_{on} = R_{on}(SG) \quad (25)$$

In turn, when the $CG_1 = CG_2 = 1$, the R_{on} in the IG-FinFET-LV_{th} is obtained as follows:

$$R_{on} = \frac{R_{on}(SG)}{2} \quad (26)$$

In the case of the IG-FinFET-HV_{th}, the R_{on} is defined as follows:

$$R_{on} = 2 * R_{on}(SG) \quad (27)$$

On the other hand, the C_{in_i} value in the IG-FinFET-HV_{th} and in IG-FinFET-LV_{th}, is obtained as follows:

$$C_{in_i} = 2 * \frac{C_{in}(SG)}{5} \quad (28)$$

4.5.3 RC modelling of TIG-SiNWFET and DG-SiNWFET

According to control gates configuration, uncertain states can occurs in TIG-SiNWFET (ZHANG *et al.*, 2017). In particular, such uncertain states occur when the value applied to the PGD terminal is different from the value assigned to the PGS one. Such an uncertain states must be prevented due to possible signal degradation problems (ZHANG *et al.*, 2017). In this sense, for the allowed states, always that the transistor is *on*, the three control gates must be active. Therefore, we are assuming that the activation of TIG-SiNWFET requires three times more resistance and charge than a SG-SiNWFET. This situation is similar to DG-SiNWFET, since the Boolean functions implemented through this transistor are a sub-set of the functions implemented through TIG-SiNWFET. Moreover, the physical structure of both transistors is quite similar, as previously discussed. Notice that the transistor model used to represent DG-SiNWFET has one control gate less than in the transistor modeling adopted for TIG-SiNWFET. However, the R_{on} and C_{in_i} values for DG-SiNWFET are similar to the values defined for TIG-SiNWFET, as follows:

$$R_{on} = 3 * R_{on}(SG) \quad (29)$$

$$C_{in_i} = C_{in}(SG) \quad (30)$$

4.5.4 RC modelling of TIG-FGMOSFET

Similar to IG-FinFET, TIG-FGMOSFET presents two possible operation states. When the three gates are active, the resistance in the transistor modeling is three times larger than the one in SG-FGMOSFET model. The R_{on} value was defined considering that TIG-FGMOSFET presents physically two extra gate in respect with the SG version. In turn, when only two control gates are active, the R_{on} in the transistor model is penalized. In this case, such penalization is proportional to one third of the resistance of transistor when the device is conducting with the three gates active. In this configuration, with a gate disabled it is necessary more time to reach the transistor threshold value. Similar to DG-SiNWFET and TIG-SiNWFET, we assume that the input capacitance of the TIG-FGMOSFET is proportional to the number of control gates used in the transistor. In particular, when the $\sum_{i=1}^3 CG_i = 2$, the R_{on} in the TIG-FGMOSFET is defined according to following equation:

$$R_{on} = 4 * R_{on}(SG) \quad (31)$$

In turn, when the $\sum_{i=1}^3 CG_i = 3$, the R_{on} in the TIG-FGMOSFET is obtained as follows:

$$R_{on} = 3 * R_{on}(SG) \quad (32)$$

On the other hand, the C_{in_i} value in the TIG-FGMOSFET is obtained as follows:

$$C_{in_i} = C_{in}(SG) \quad (33)$$

4.6 Delay analysis of MIGFET logic gates

The circuits implemented in this work are built through logic gates implemented using MIGFET devices. Each logic gate is composed by a pull-up plan (PU) and a pull-down plan (PD). In order to compute the delay of a given logic gate, each plan is modeled as an RC network, in which the Elmore delay method is applied. In this process, for each input of logic gate, a delay value is computed. This delay value is defined as intrinsic delay (T_d), and it is computed in respect to the output capacitance of the logic gate.

In order to estimate the delay in more complex circuits, static timing analysis (STA) is adopted. In this approach, the delay value is estimated according to the primary inputs and

outputs of the evaluated circuit. Both the Elmore delay analysis and the STA approach are discussed in more detail hereafter.

4.6.1 Elmore delay analysis of logic gates

A timing arc is defined as an input vector configuration, such that, the transition of the evaluated input causes a transition at the output of the logic gate. The number of timing arcs computed for a given input depends on the Boolean function implemented in the logic gate. This way, all timing arcs for a given logic gate input are considered in order to define the worst delay value. Basically, the timing arcs are produced through the Boolean difference with respect to a given input.

The input vectors produced through this process are used to determine the active transistors in the PU and PD plans. In particular, each transistor is represented through a RC equivalent circuit. Transistors on *off* state are represented through an open circuit. This way, the delay value obtained in a given plan is defined by apply the Elmore delay method on the logic paths created through active transistors. Therefore, low and high input delay propagation signal of a logic gate are computed from a given input condition. When there is more than one input condition the worst delay value is considered.

In order to exemplify this procedure, using SG-FinFET and IG-FinFET devices, two distinct logic gates representing the same Boolean function are shown in Figure 38. Notice that, for both cases, the RC network representations are extracted and a single timing arc is evaluated. Notice also that, the resistance value associated to a resistor that represents a given IG-FinFET depends of the number of control gates active and of the Boolean function implemented through of the device.

Furthermore, both the RC networks can be translated to a graph structure, where each edge represents a resistor and each node represents a capacitor. In this case, the capacitance value associated to each node (C_{NODE}) is defined as follow:

$$C_{\text{NODE}} = C_D * \text{degree}(\text{NODE}) \quad (34)$$

where the *degree* is the number of edges incident on the evaluated node.

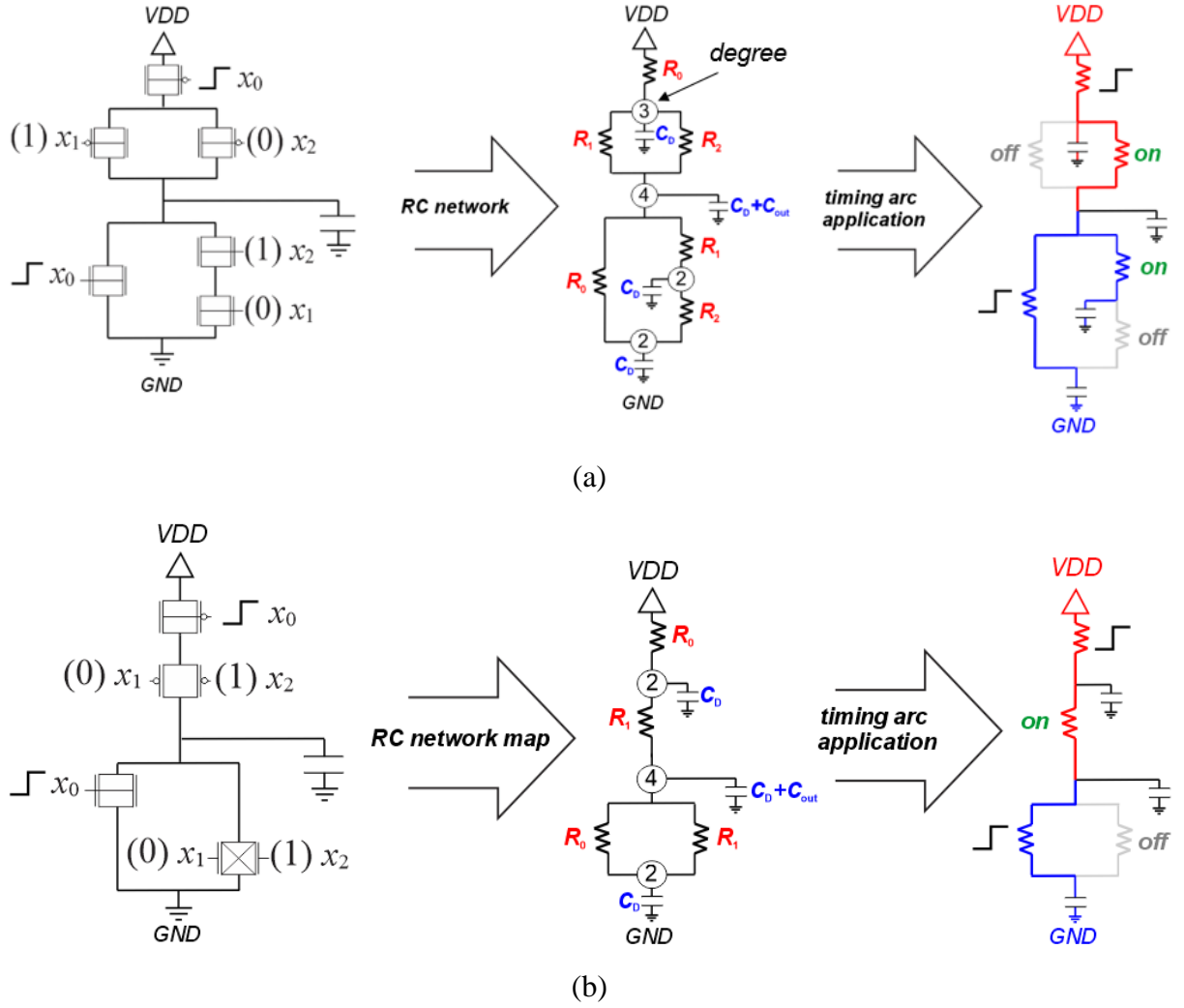


Figure 38: RC network and timing arcs application: (a) RC network obtained from SG-FinFET logic gate, and (b) RC network obtained from IG-FinFET logic gate.

To apply the Elmore analysis on the PU network, it is computed the intrinsic delay value to connect the VDD terminal to the output terminal (T_{dUP}). In turn, when the Elmore analysis is applied on the PD network, it is computed the intrinsic delay value to connect the GND terminal to the output terminal (T_{dDOWN}). Therefore, the intrinsic delay (T_d) of a logic gate in relation to a given input i is computed as follows:

$$T_{d_i} = \frac{(T_{dUPi} + T_{dDOWNi})}{2} \quad (35)$$

As Elmore analysis is based on linear delay modeling, the total delay (D) of a given logic gate related to a given input i is defined as function of the output capacitance (C_{out}) connected to the output terminal. This way, the total delay can be computed as follows:

$$D_i = T_{d_i} + (\alpha * C_{out}) \quad (36)$$

where α represents the slope of the line that defines the delay behavior of a given logic gate, as illustrated in Figure 39.

In turn, the output capacitance of a given logic gate LG_i is defined as the sum of all input capacitances of the logic gates connected to the output terminal of such LG_i . Notice that, the total delay variation in relation to output capacitance is defined through a constant α . Thus, in order to determine the α value, the intrinsic delay for a given input of a logic gate is computed twice. At the first time, the intrinsic delay is computed considering that the output capacitance of the logic gate is zero. In the second time, it is assumed an arbitrary value C for the output capacitance of the logic gate. In this sense, the factor α in relation to a given input i can be computed as follow:

$$\alpha_i = \frac{Tdc_i - Td_i}{C} \quad (37)$$

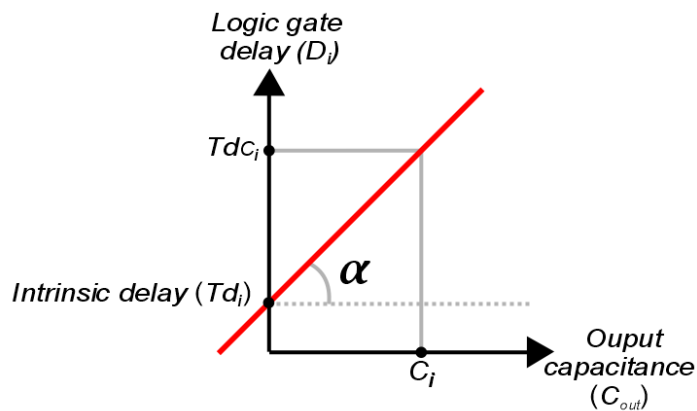


Figure 39: Electric behavior of a transistor model based on Elmore delay modeling.

4.6.2 Static timing analysis

In static timing analysis (STA), the delay value of a given circuit, considering a specific output, is obtained through the time that a signal need to go from a primary input to this output. In this way, as a given circuit can present several outputs, for each output a delay value is computed, and the worst delay is defined as the final estimated delay value of the circuit. Normally, STA is implemented recursively. In this approach, the STA algorithm begins from a specific output and performs recursive calls until to find out a primary input.

When the Elmore delay analysis is performed, the intrinsic delay and α factor are computed for each input of the logic gate. Such parameters are required in the STA procedure to estimate the delay of each circuit according to the output capacitance for such a logic gate. This capacitance value is only evaluated in the STA task since, in this step, the input

capacitance of a given logic gate becomes the output capacitance of other one according to the connections defined in the evaluated circuit. During the recursive call performed in the STA, each logic gate is visited and its delay table is updated since, at this moment, it is possible to determine the output capacitance for each logic gate according to the input capacitances of the logic gates ahead.

In this sense, when the STA procedure returns from the recursion, it is possible to evaluate the arrival time to reach a given logic gate from primary inputs, as illustrated in Figure 40. In particular, the estimated delay value for a specific output of the circuit (Td_{out_i}) is defined by the *worst* arrival time to reach such an output from a given primary input of the circuit. In turn, the highest Td_{out_i} value is defined as the estimated delay of the evaluated circuit. The Td_{out_i} is computed as follows:

$$Td_{out_i} = worst(Td_{cell} + max(Td_j)) \quad (38)$$

where, the Td_{cell} is the average between the low and high delay propagation value for a given input of the evaluated logic gate, and the $max(Td_j)$ is defined as the highest arrival time with respect to inputs of the evaluated logic gate.

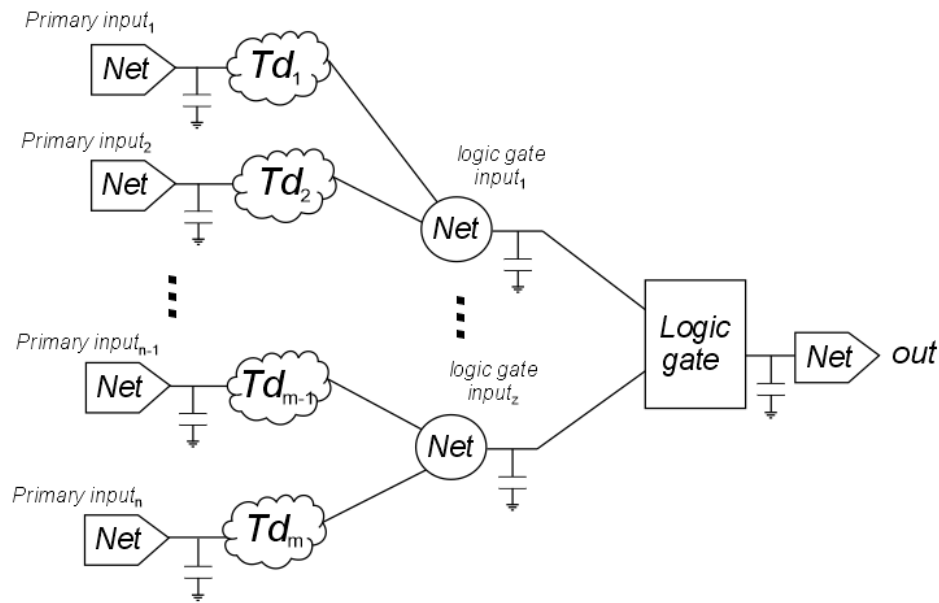


Figure 40: Circuit representation for application of STA.

4.6.3 Discussions and considerations

In this work, we are not taking into account the use of different MIGFET devices into a single circuit design. Therefore, the circuits presented herein are implemented through a single MIGFET technology.

In this sense, in some cases, as occur for the IG-FinFET and TIG-FGMOSFET, the SG switches associations that cannot be merged into a single device (non-merged transistor) are implemented through of two distinct approaches. In the first case, the non-merged transistors are implemented through single-input assignment (SIA) of the respective MIGFET. This way, all multiple-independent gates of the MIGFET are connected together in order to produce a SG version. On the other hand, in the second approach, the non-merged transistors are built fixing specific multiple-independent gates of the MIGFET in constant logic values, as 0 or 1, such an approach is denoted as fixed-gate version (FGV). In contrast, for ambipolar devices, like DG-SiNWFET and TIG-SiNWFET, the non-merged SG switches must be configured of a specific way, as discussed in Chapter 3. In the case of the DG-SiNWFET, the non-merged transistor is built fixing the PG to the logic value 0 or 1, according to type of the used MIGFET, P- or N-type. On the other hand, for TIG-SiNWFET, the non-merged SG switches are built connecting PGS and PGD to the same logic value, 0 or 1.

In order to demonstrate the area and delay analysis, discussed before, we performed a simple experiment, where an inverter and a NOR2 gates were evaluated for each approach of non-merged switches considering IG-FinFET and TIG-FGMOSFET. Both logic gates are implemented using only non-merged SG transistors. From the inverter gate, we evaluate the impacts of the SIA and FGV MOSFET representation on a single logic gate.

In contrast, NOR2 gate was designed taking into account a NAND2 logic gate with its inputs complemented. Such an implementation allows us to evaluate the SIA and FGV representations considering the connection between logic gates. The gate topologies are shown in Figure 41 for IG-FinFET implementations and in Figure 42 for TIG-FGMOSFET implementations.

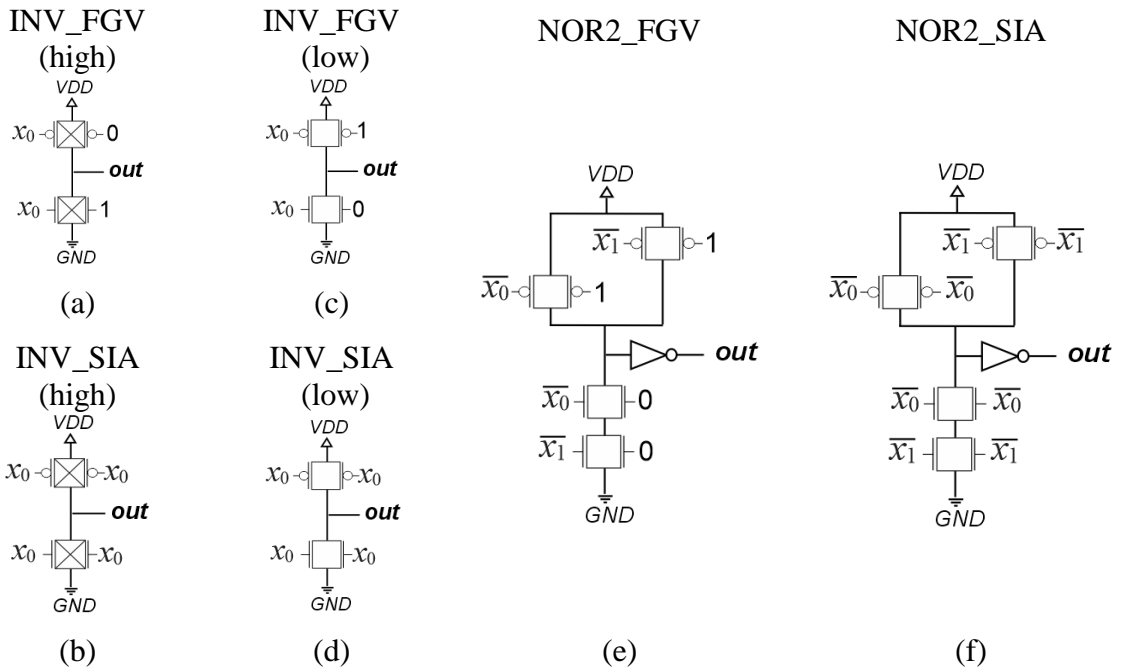


Figure 41: Logic gate implementations using IG-FinFET devices: (a) inverter using IG-FinFET-HV_{th} in FGV, (b) inverter using IG-FinFET-HV_{th} in SIA, (c) inverter using IG-FinFET-LV_{th} in FGV, (d) inverter using IG-FinFET-LV_{th} in SIA, (e) NOR2 logic gate implemented from IG-FinFET-LV_{th} in FGV, and (f) NOR2 logic gate implemented from IG-FinFET-LV_{th} in SIA.

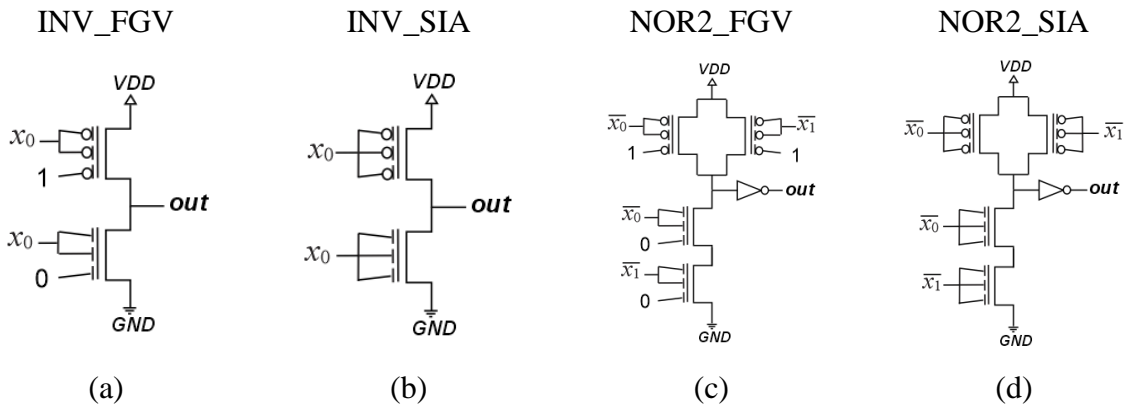


Figure 42: Logic gate implementations using TIG-FGMOSFET devices: (a) inverter gate in FGV, (b) inverter gate in SIA, (c) NOR2 gate in FGV, and (d) NOR2 gate in SIA.

When an input of a given logic gate is connected directly to 0 or 1 logic value, the input capacitance value of this logic gate tends to decrease. This situation benefits all the logic gates connected to such a capacitance, since the loading time decreases. However, to fix one physical gate of the transistor in a particular logic value can lead to the increasing in the resistance value in the transistor RC model. For instance, in the case of IG-FinFET-LV_{th} and TIG-FGMOSFET, the resistance value is defined according to the configuration of the input signals of the transistor. In both cases, the smallest resistance value is reached when all the

control gates of the transistor are active. As discussed previously, these transistors can be active through other input configurations, such as by fixing one of the inputs in the logic value 0. However, in this situation, the resistance value associated to this transistor will be higher. Such a behavior can be seen in Table 4, where the delay and area values are estimated for each logic gate shown in Figure 41 and in Figure 42.

Table 4: Estimated area and delay values for inverter and NOR2 logic gates implemented through IG-FinFET and TIG-FGMOSFET, normalized in relation to respective SG device of each MIGFET technology

Technology	Logic gate	Delay	Area
IG-FinFET	INV_FGV (high)	4.00	3.64
	INV_SIA (high)	4.00	
	INV_FGV (low)	2.00	
	INV_SIA (low)	1.00	
	NOR2_FGV	8.70	18.20
	NOR2_SIA	4.95	
TIG-FGMOSFET	INV_FGV	8.00	3.34
	INV_SIA	6.00	
	NOR2_FGV	54.00	16.70
	NOR2_SIA	49.50	

Notice in Table 4, that the inverter logic gates built through IG-FinFET-HV_{th} present larger delay values than inverters built using IG-FinFET-LV_{th}, as expected. Notice also that, the inverter implemented through IG-FinFET-LV_{th} in SIA approach shows the best result in relation to the delay value than another inverter IG-FinFET implementations. As discussed previously, in this situation, the resistance value associated to transistors is minimum, since the two independent gate are used to active the device. A similar situation occurs in the inverter gates built through TIG-FGMOSFET, since that the resistance value of the transistor model is penalized according to the configuration of the input signals assigned to multiple-independent gates of the transistor.

It is possible to notice in Figure 42 and in Figure 41, the NOR2 gate is built through a NAND2 and three inverters. Therefore, when the FGV is adopted, the input capacitances of the logic gates that compose the NOR2 gate decrease. However, the internal resistance values of the logic gate increase. In turn, when the SIA is adopted, the opposite situation occurs. However, as shown in Table 4, for IG-FinFET and TIG-FGMOSFET, the NOR2 logic gate implemented through SIA tends to present the best results, even considering the increasing in the input capacitances of the logic gates that compose the NOR2 gate.

5 BINARY ADDERS DESIGN

For many emerging technologies, such as MIGFET devices, there is still a lack of knowledge regarding the design of arithmetic circuits. In this section, we evaluate whether binary adder circuits can be optimized through the use of MIGFETs. Moreover, the binary adders implemented herein are used to demonstrate the logic and physical impacts of the MIGFET devices in the synthesis of digital integrated circuits.

In this sense, two distinct adder architectures are explored: (i) ripple-carry adder (RCA), and (ii) parallel-prefix adder (PPA). Such architectures represent the two extreme situations terms of area saving and high performance for binary adder designs. In particular, RCA is the best approach for area saving, whereas PPA is the best option for high performance, since in PPA the parallel signal processing is widely exploited. Thus, the trade-off between area and performance of binary adders designs are discussed herein.

5.1 Ripple-carry adder

Ripple-carry adder (RCA) is the simplest and the most intuitive adder built by chaining full-adder (FA) circuits. In particular, each FA represents a stage of the adder. Thus, an m -bit RCA is implemented through m stages, as shown in Figure 43. In RCA, the sum (sum_i) and carry-out ($cout_i$) signals at each FA can be computed as follows:

$$sum_i = a_i \oplus b_i \oplus cin_i \quad (39)$$

$$cout_i = MAJ(a_i, b_i, cin_i) = (a_i \cdot b_i) + (a_i \cdot cin_i) + (b_i \cdot cin_i) \quad (40)$$

Notice that the sum-bit signal can be generated by exploiting the same majority gate used to provide the carry-out bit:

$$sum_i = !cout_i \cdot (a_i + b_i + cin_i) + a_i + b_i + cin_i \quad (41)$$

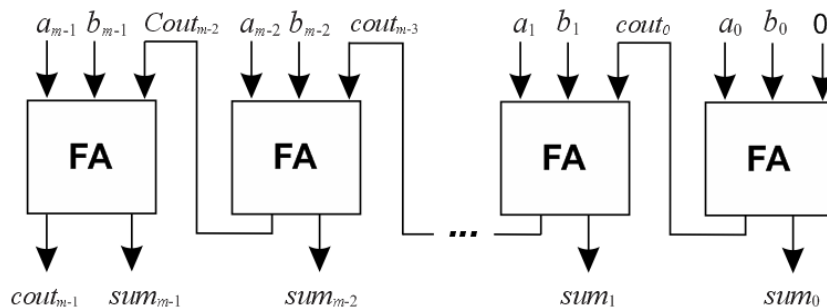


Figure 43: Ripple-carry adder block diagram.

Among the adder designs, RCA is considered the most compact one in terms of area but it is also the slowest one due to the carry signal propagation, which is computed gradually along of m stages of FA. The worst signal delay propagation occurs when the carry bit is generate in the first stage and propagated until the last one. Thus, the delay of RCA is linearly proportional to its number of stages. Similarly, as each stage of RCA is implemented through FA circuit, the total area is linearity proportional to number of replicated FAs. In this sense, the area (A_{RCA}) and delay (D_{RCA}) can be defined as follows:

$$A_{RCA} = A_{FA} * N_{FA} \quad (42)$$

$$D_{RCA} = D_{FA} * N_{FA} \quad (43)$$

where the FA area is denoted as A_{FA} , the FA delay is denoted as D_{FA} , and the number of FA adopted is denoted as N_{FA} .

5.2 Carry-select adder

Carry-select adder (CSelA) was idealized to reduce the propagation time of the carry signal on the traditional RCA by exploiting hardware duplication. Basically, the carry-select architecture is composed of two RCAs in parallel and multiplexer circuits (MOHANTY; PATEL, 2014) and (RAMKUMAR; KITTUR, 2012). These multiplexers are responsible for selecting the correct sum and carry-out signals, as illustrated in the 4-bit carry-select block (CSelB) presented in Figure 44. Notice that in the upper RCA, shown in Figure 44, the carry-in input is 0, whereas at the bottom RCA the carry-in input is 1. This way, the carry-out and sum bits in the upper RCA and at the bottom RCA are computed in parallel. Thus, the right signals are defined according to the carry-in signal used as selector of the multiplexers.

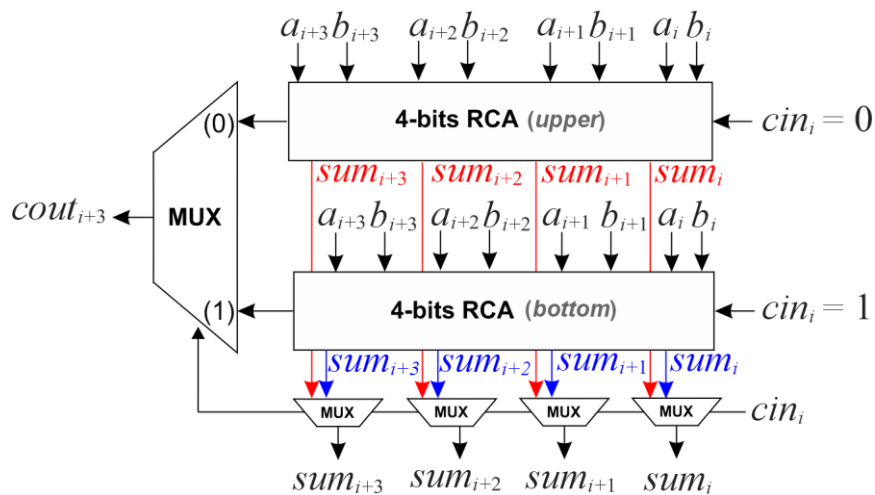


Figure 44: 4-bits Carry Select Adder block (CSelB).

The parallelism of CSelA is better explored for larger input vectors, where the CSelB is replicated in order to divide the carry chain and so optimize the carry delay of the adder. For instance, in 16-bits CSelA, shown in Figure 45, the carry chain of the adder is broken uniformly through 4-bits CSelB. This way, the sum bits and the carry-out signal of each 4-bits CSelB are all computed at the same time. Thus, the total propagation carry-out delay is computed as the time of carry propagation through the first CSelB plus the delay of the multiplexers used to propagate the carry-out signals along of the adder. Notice that, in the first stage, it is not necessary to replicate the 4-bits RCA, since the carry-in signal is known for this stage.

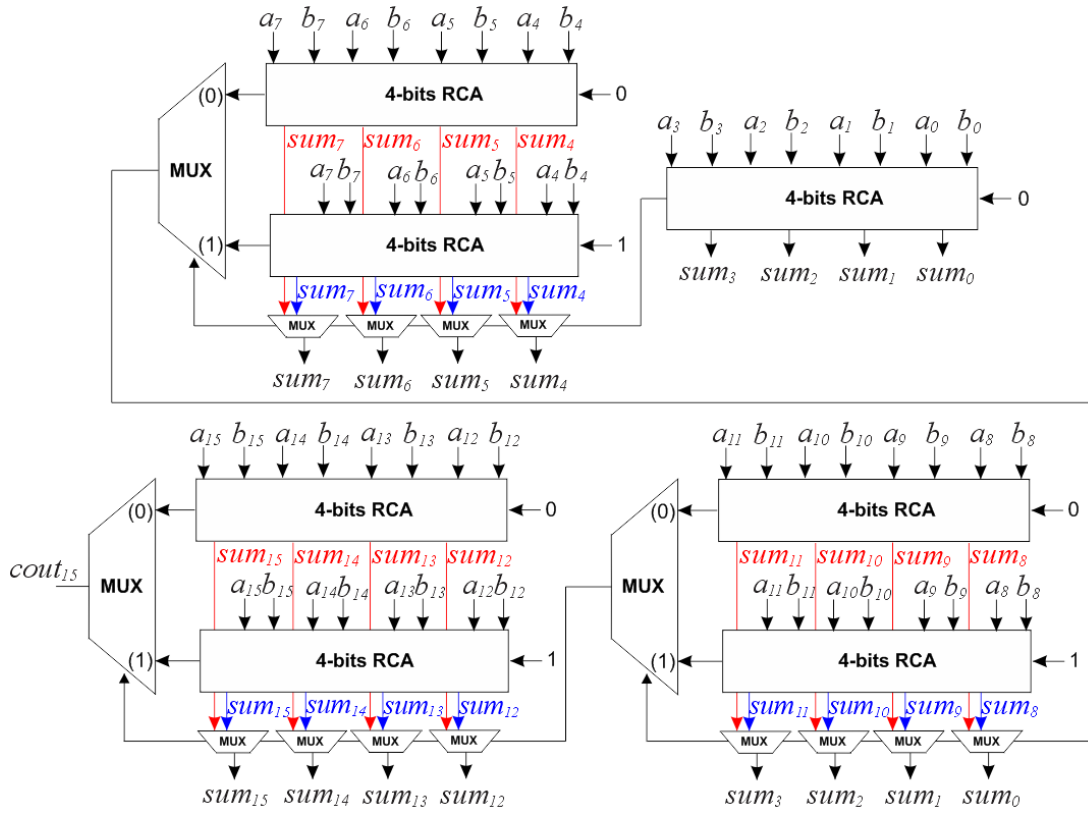


Figure 45: 16-bits CSelA implemented through 4-bits CSelB.

Indeed, CSelA is more effective than traditional RCA. However, the performance gains are penalized by area overhead, since the parallel carry computation requires approximately the duplication of the hardware. In general, the total area of the CSelA is proportional to the number of CSelB and multiplexers used along the adder. Thus, CSelA area (A_{CSelA}) and delay (D_{CSelA}) can be defined as follows:

$$A_{CSelA} = ((2 * A_{RCA}) * N_{CSelB}) + (A_{MUX} * N_{MUX}) + A_{RCA} \quad (44)$$

$$D_{CSelA} = D_{RCA} + (D_{MUX} * (N_{STAGES} - 1)) \quad (45)$$

where A_{RCA} is the area of an m -bits RCA used in the CSelB and A_{MUX} is the area of a multiplexer. In turn, N_{CSelB} and N_{MUX} are number of CSelB and multiplexers adopted along the adder, respectively. On the other hand, D_{RCA} is the delay of an m -bits RCA, D_{MUX} is the delay of a multiplexer and N_{STAGES} is the number of stages of the adder.

5.3 Carry-skip adder

Carry-skip adder (CSkipA) is quite similar to CSelA, since in both approaches the carry chain of an RCA is divided in stages. However, for each stage of the CSkipA, a specific and small circuit is adopted to anticipate the carry-in signal of the next stage (GUYOT; HOCHET; MULLER, 1987) and (ALIOTO; PALUMBO, 2003). In particular, the CSkipA is implemented through carry-skip blocks (CSkipBs), which are built through a sum/generate block and a propagate block (PB). The sum/generate block performs the sum-bits computation and indicates if a bit position i can produce a carry-out bit. In general, the sum/generate block is implemented using conventional m -bit RCA. In turn, the PB indicates if it is possible to propagate the carry-in bit of a given CSkipB to the next one. Both the sum/generate block and the PB are illustrated in the 4-bits CSkipB shown in Figure 46.

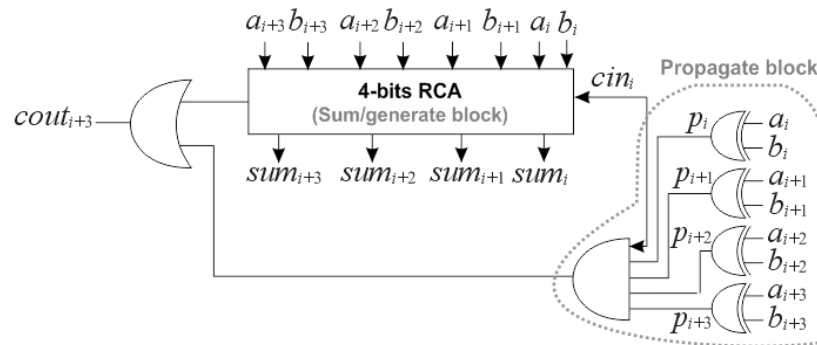


Figure 46: 4-bits Carry Skip Adder block (CSkipB).

Notice that in PB, XOR2 operations are applied over the inputs vectors of the adder. In particular, each XOR2 operation is applied to a bit position pair (a_i, b_i) . In turn, the XOR2 operations are associated through an AND operation. In the case of the CSkipB presented in Figure 46, four XOR2 operations are performed, which are associated together to carry-in signal through a five-input AND (AND5) operation. Notice that this circuit is faster than 4-bits RCA implemented in the sum/generate block. Therefore, always that a carry-in signal is propagated to the next CSkipB from the PB, the total delay time of the carry propagation of the adder is reduced. Notice also that, in CSkipB, for larger input vectors, all the propagate

signals are computed in parallel, as shown in Figure 47, where a 16-bits CSkipA is implemented using 4-bits CSkipB.

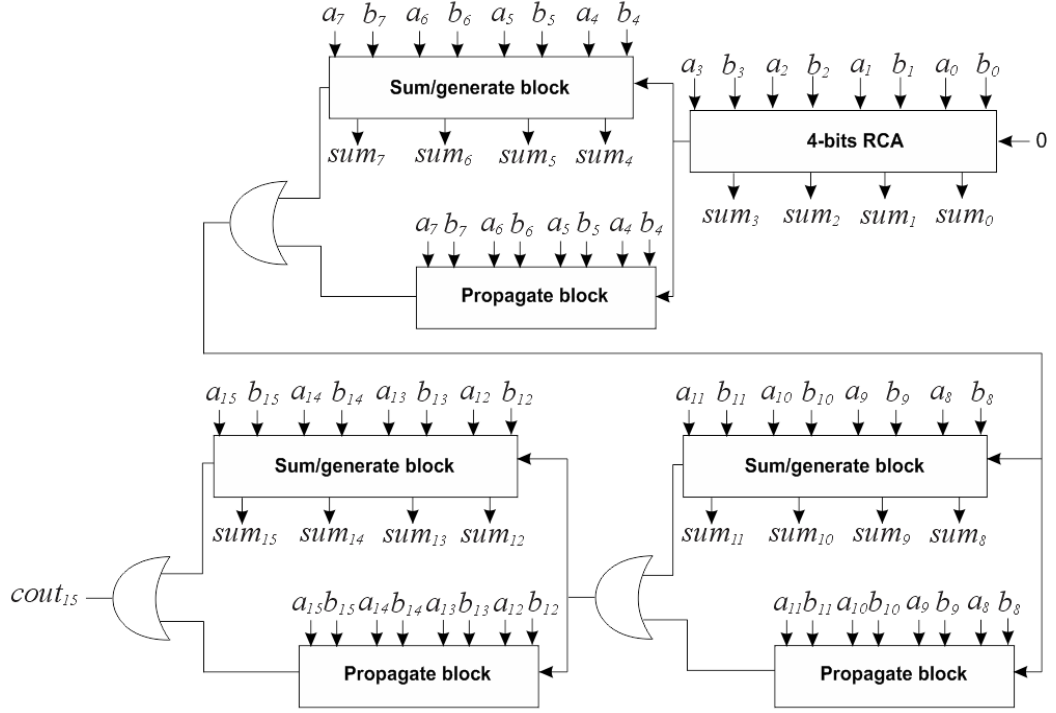


Figure 47: 16-bits CSkipA implemented using 4-bits CSkipB.

The great advantage of CSkipA is that the carry bit of a given CSkipB can be directly propagated to the next block, decreasing the propagation time of the carry chain of the adder. However, the skip of the carry signal is only performed when the propagate signal is 1. Instead, the carry signal is computed through the carry chain implemented by sum/generate block. When, for all stages of the adder, it is not possible to perform carry propagation, the total carry propagation delay in the CSkipA is similar to the delay of a CSelA. In turn, the area overhead of the CSkipA tends to be smaller than the one related to the CSelA, since the propagation block is more compact than the RCA replicated in CSelA. Moreover, in CSkipA multiplexers to select the correct sum bits are not used. CSkipA area (A_{Skip}) and CSkipA delay (D_{Skip}) are defined as follows:

$$A_{\text{Skip}} = \left(\sum_{i=1}^{n-1} A_{\text{PB}_i} + A_{\text{RCA}_i} \right) + A_{\text{RCA}_0} \quad (46)$$

$$D_{\text{Skip}} = \left(\sum_{i=1}^{n-1} \begin{cases} \text{cin}_i = 1, D_{\text{PB}_i} \\ \text{cin}_i = 0, D_{\text{RCA}_i} \end{cases} \right) + D_{\text{RCA}_0} \quad (47)$$

where, A_{PB} is the area of a PB and the A_{RCA} is the area of an m -bits RCA. In turn, D_{PB} is the delay of a PB and D_{RCA} is the delay of an m -bits RCA.

5.4 Parallel-prefix adder

Parallel-prefix adder (PPA) presents better performance among the adder architectures because, in this approach, the carry signals computation is fully performed in parallel. The main structure used in PPA design comprises four steps, as shown in Figure 48. The procedure begins with the computation of individual generate signal, as follows:

$$g_i = (a_i \cdot b_i) \quad (48)$$

and propagate signal, as follows:

$$p_i = (a_i \oplus b_i) \quad (49)$$

The generate operation indicates if a bit position i produces a carry-out bit, regardless the carry-in signal. In turn, the propagate operation indicates if this bit can propagate internal carry bit to the next position, similarly to CSkipA. In the second step, generate and propagate operations are implemented for groups of bits. These operations are denoted as group generate ($gg_{i,j}$) and group propagate ($gp_{i,j}$), where the $i > j$. The computation of the groups of bits, from index 0 to i , is performed by a specific arrangement of generate-propagate operators. Each generate-propagate operator, shown in Figure 49(a), computes the group generate and group propagate signals of two adjacent bits or two adjacent groups, as follows:

$$gg_{i,j} = g_i + (p_i \cdot g_j) \quad (50)$$

$$gp_{i,j} = p_i \cdot p_j \quad (51)$$

In the third step, the carry generator block computes all carry-inputs required to the sum operations considering the group generate and group propagate signals from indexes i to 0, and the eventual carry-input signal (cin_0), as follows:

$$cin_{i+1} = gg_{i,0} + (gp_{i,0} \cdot cin_0) \quad (52)$$

In the last step, the sum bits are then computed according to equation (39). Notice that the computation of the sum bits can also be performed using the propagate signals generate in the first step of the PPA procedure, as follows:

$$sum_i = (p_i \oplus cin_i) \quad (53)$$

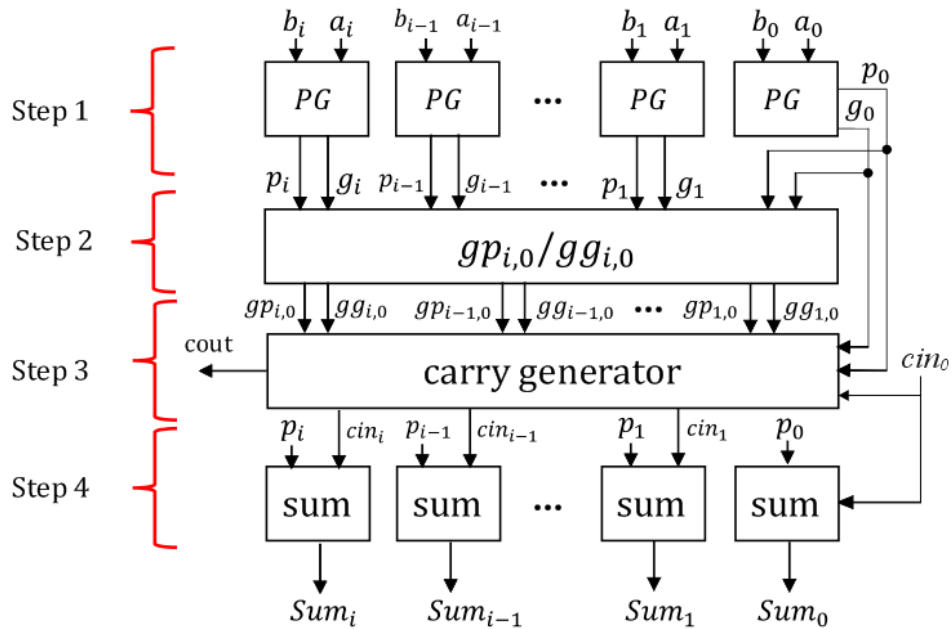


Figure 48: Basic block used to build parallel-prefix adder and carry lookahead adder.

There are many variations of PPA topologies in the literature named as Kogge-Stone: (KS) (KOGGE; STONE, 1973), Brent-Kung (BK) (BRENT; KUNG, 1982), Ladner-Fischer (LF) (LADNER; FISCHER, 1980) and Han-Carlson (HC) (HAN; CARLSON 1987), among others. For each approach, different arrangements of generate-propagate operators are defined in the second step of the PPA procedure. These arrangements impact in the area, performance and power dissipation of the final adder circuit. In Figure 49(b), it is illustrated the Brent-Kung arrangement of generate-propagate operators for an input vector of 16 bits. This arrangement ensures the minimum number of generate-propagate operators but it presents the largest logic depth (signal path) among PPA variations. On the other hand, the 16-bit Ladner-Fischer arrangement, illustrated in Figure 49(c), provides the shortest logic depth but increases the fanout (number of output connections) of the generate-propagate operators.

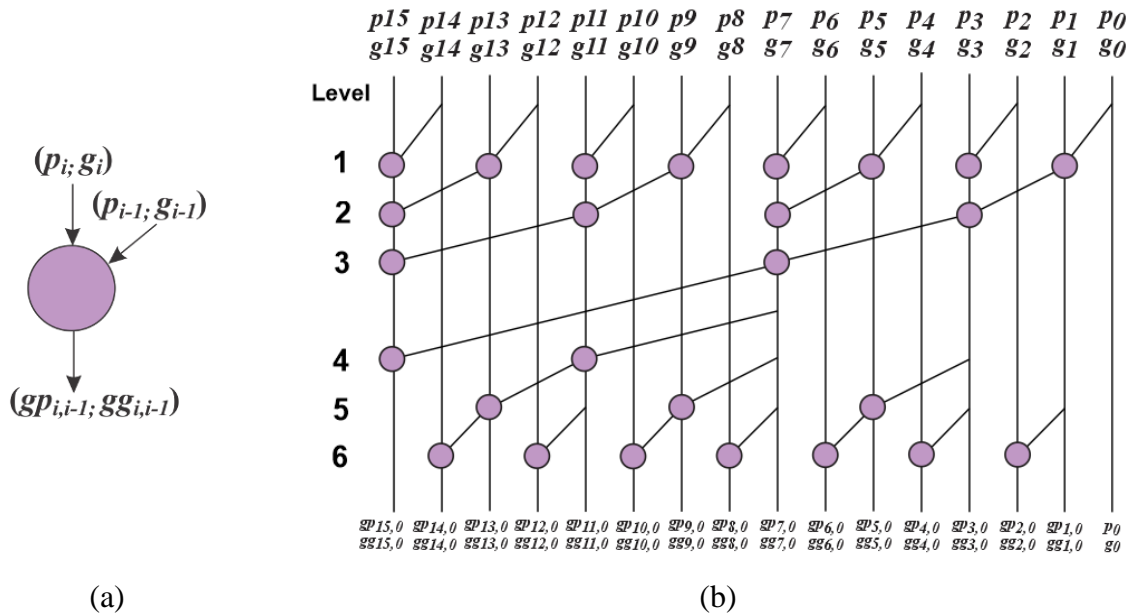


Figure 49: PPA structure: (a) generate-propagate operator, and (b) Brent-Kung arrangements for 16 bits.

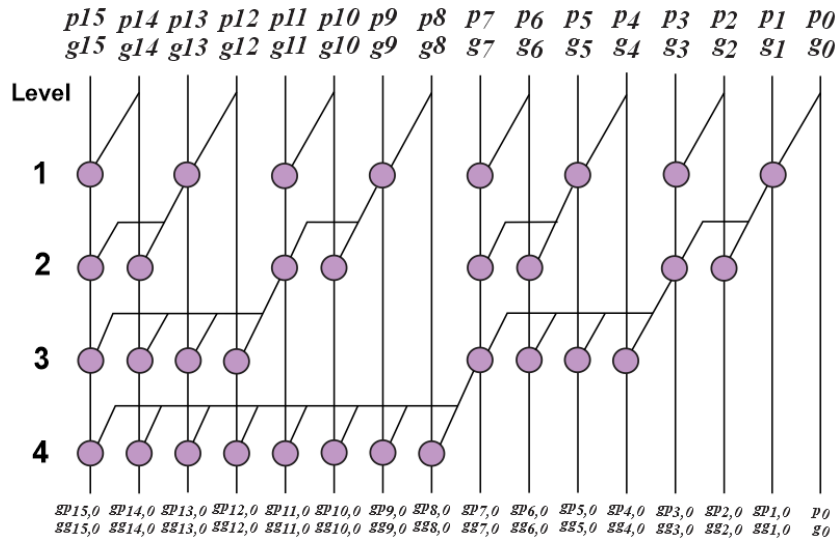


Figure 50: Ladner-Fischer arrangements for 16 bits

The PPA algorithm offers a high parallelization in the computation of the carry signals, since all the carry-in signals required to perform the sum operations are produced at the same time. Moreover, all the sum-bits are computed in parallel immediately after the computation of the carry-in signals. In this sense, due to the high parallelization, the PPA approach presents the best performance among binary adders. On the other hand, PPA also presents the highest area overhead. In general, the area penalizations occur due to the replication of generate-propagate operators necessary to implement logic arrangements defined in the second step of the PPA algorithm.

5.5 Carry-lookahead adder

The carry-lookahead adder (CLA) is a particular PPA architecture optimized to the current CMOS technology. In CLA design, the group generate signals are implemented through a specific transistor network topology known as Manchester carry chain (UNWALA; SWARTZLANDER, 1993) and (RUIZ, 1998). As observed in Figure 51, the Manchester chain is a multiple-output network in which each output implements a group generated as follows:

$$gg_{i+1,i} = g_i + (p_{i+1} \cdot g_i) \quad (54)$$

$$gg_{i+2,i} = g_{i+2} + (p_{i+2} \cdot g_{i+1}) + (p_{i+2} \cdot p_{i+1} \cdot g_i) \quad (55)$$

$$gg_{i+3,i} = g_{i+3} + (p_{i+3} \cdot g_{i+2}) + (p_{i+3} \cdot p_{i+2} \cdot g_{i+1}) + (p_{i+3} \cdot p_{i+2} \cdot p_{i+1} \cdot g_i) \quad (56)$$

As the Manchester chain provides only the group generate, the group propagate is implemented through series arrangement, as follows:

$$gp_{i+1,i} = (p_{i+1} \cdot p_i) \quad (57)$$

$$gp_{i+2,i} = (p_{i+2} \cdot p_{i+1} \cdot p_i) \quad (58)$$

$$gp_{i+3,i} = (p_{i+3} \cdot p_{i+2} \cdot p_{i+1} \cdot p_i) \quad (59)$$

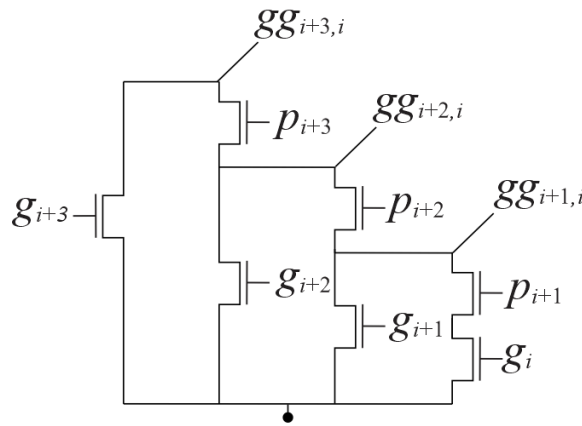


Figure 51: Manchester carry chain transistor network for group generate signals from indexes 0 to 3.

Thus, in CLA design, the group generate and group propagate signals are computed through a specific 4-bits block, which is composed by Manchester chain and the propagate series arrangements. Such 4-bits block replaces the generate-propagate operator adopted in other PPA approaches. Notice that, through one single 4-bits block, it is possible to compute all the group generate and group propagate signals from indexes i to $(i + 3)$. Notice also that the Manchester chain is a general structure, which can be implemented for unlimited number

of bits. However, in CMOS technology the maximum practical stacked transistor allowed is four, since larger stacks impact signal delay propagation (RABAHEY; CHANDRAKASAN; NIKOLIC, 2002). Therefore, the 4-bits limitation adopted in CLA design is defined due to the CMOS technology restrictions.

Similar to other PPAs, presented previously, CLA design is implemented through the algorithm shown in Figure 48. However, the second step of the procedure is modified in the CLA design, since the input vectors are divided in 4-bits blocks. Just as for other PPA approaches, in the literature are proposed optimizations and particular arrangements of 4-bits blocks for CLA (LYNCH; SWARTZLANDER, 1992) and (PAI; CHEN, 2004). Such 4-bits arrangements offer great parallelization in the carry signals computation. However, such as discussed previously, in PPA designs, these arrangements also impact in area overhead.

5.6 Binary adder design evaluation

As it is possible to notice from adder architectures discussed before, the main approach adopted to reduce the signal delay of binary adder designs is the parallelization of the carry chain computation. Such a strategy exploits the replication or insertion of new logic blocks in order to accelerate the carry signals. However, the performance gains are penalized by area overhead provided by extra logic blocks.

For many emerging technologies, such as MIGAET devices, there is still a lack of knowledge regarding the design of arithmetic circuits. Therefore, in this section, we evaluate the trade-off between area overhead and performance improvement of the binary adder circuits considering the use of MIGAET devices. In particular, we adopted the RCA and PPA architectures as case studies, since such architectures focus on area saving, in one case, and in high performance computation, in the other case.

5.6.1 Evaluation methodology

Basically, the circuit analysis presented herein is performed in two steps: (i) the logic design definition, and (ii) the delay and area estimation. Firstly, the adder architectures are designed through switch networks. By using the MIGAET logic switches, we performed optimizations in the switch networks through transistor merging. These optimizations aim to obtain the smallest network in terms of device count. Moreover, such optimizations can also reduce the transistor stacking in the network (maximum number of transistors in series). In secondly, the adders are represented through transistor netlists. Hence, area and timing

characteristics of the device are considered according to the transistor models discussed in previous chapters. In this analysis, input and output inverters are also taken into account. In order to evaluate the adder circuit built using MIGFET devices, the conventional SG adder designs are adopted as reference. This way, we can evaluate the impact of using MIGFETs in relation to the traditional adder architectures.

The design flow adopted in the experiments performed herein is shown in Figure 52. In general, the adder circuits can be built through the replication of some basic blocks, as in the case of RCA design implemented by replicating FA blocks. In this sense, in order to reach general implementations of the studied adders, only basic blocks of each adder approach are used as input. Notice that such basic blocks are represented through a library of blocks. In turn, each basic block is represented as a logic gate optimized considering a given MIGFET technology. In the next, an automatic procedure has been developed in order to connect these logic gates in order to produce the target adder architecture according to the number of bits, to the adder architecture and to the MIGFET device chosen for the design.

In the next stage, the resulting adder circuit is submitted to timing and area analysis. As seen in Figure 52, some technology parameters must be known in this stage. These parameters are defined according to the chosen MIGFET to build the evaluated adder, since that each MIGFET is represented through a specific transistor model. The circuit characterization stage has as goal to estimate the area and delay of each logic gate presented in the evaluated circuit. In this stage, besides of the area value, also is computed the delay and input capacitance for each logic gate. For each output of the logic gate, a delay table and a capacitance table are computed. These table storage the delay values and input capacitances for each input variable of the logic gate. Consequently, such an information is used in the static time analysis (STA) stage in order to extract the estimated delay value of the evaluated circuit.

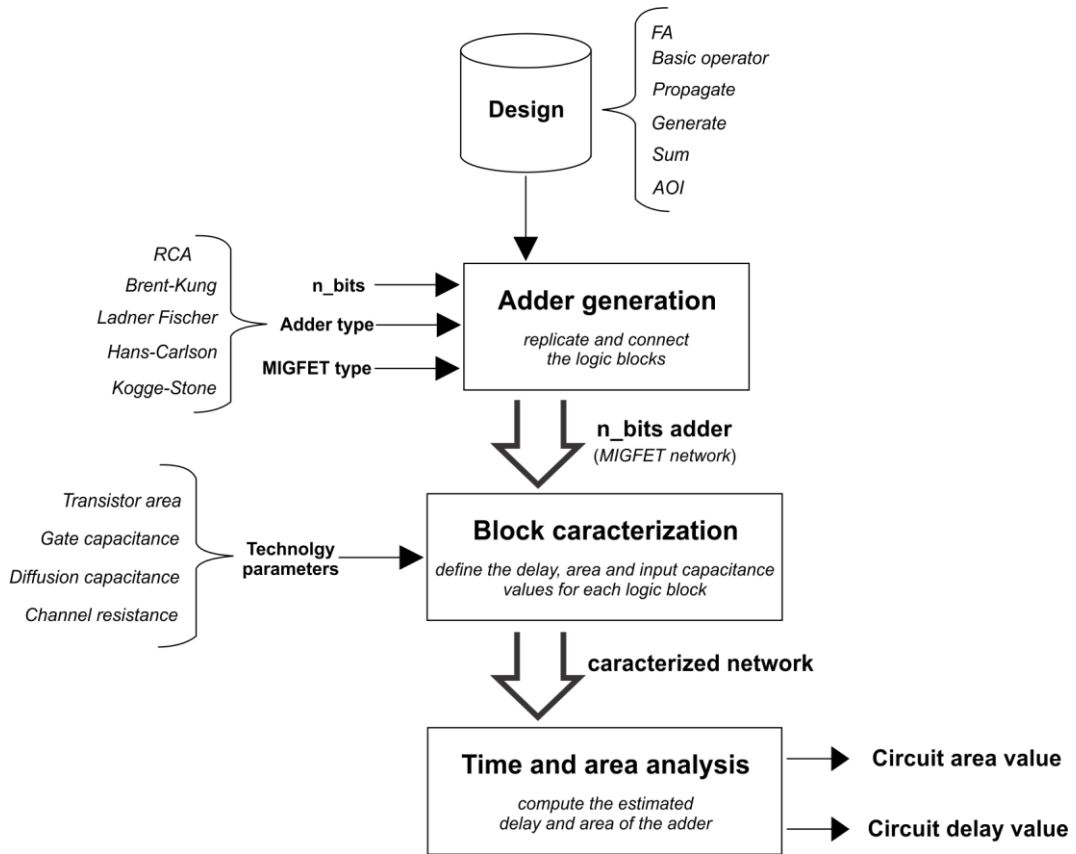


Figure 52: Design flow applied to estimate the delay and area of the adder circuits evaluated.

5.6.2 Ripple-Carry Adder design

In this experiment, we aim to investigate the most appropriate full-adder (FA) topology for each MIGFET based design. Such an evaluation considers different implementations of XOR3 gate (sum signal) and MAJ3 gate (carry-out signal). The topologies evaluated are illustrated in Figure 53. Notice that, the XOR3 and MAJ3 functions are *self-dual*. In this sense, only the transistor network of the pull-down plan is shown. It is said that a given Boolean function g is dual of f if $f(x_0, x_1, \dots, x_k) = \overline{g(\overline{x_0}, \overline{x_1}, \dots, \overline{x_k})}$. This way, for a given self-dual Boolean function f , the dual function g can be obtained just by complementing the input variables of f . In the static CMOS logic gates, due to the complementarity of the pull-up (PU) and the pull-down (PD) plans, both PU and PD network arrangement are similar.

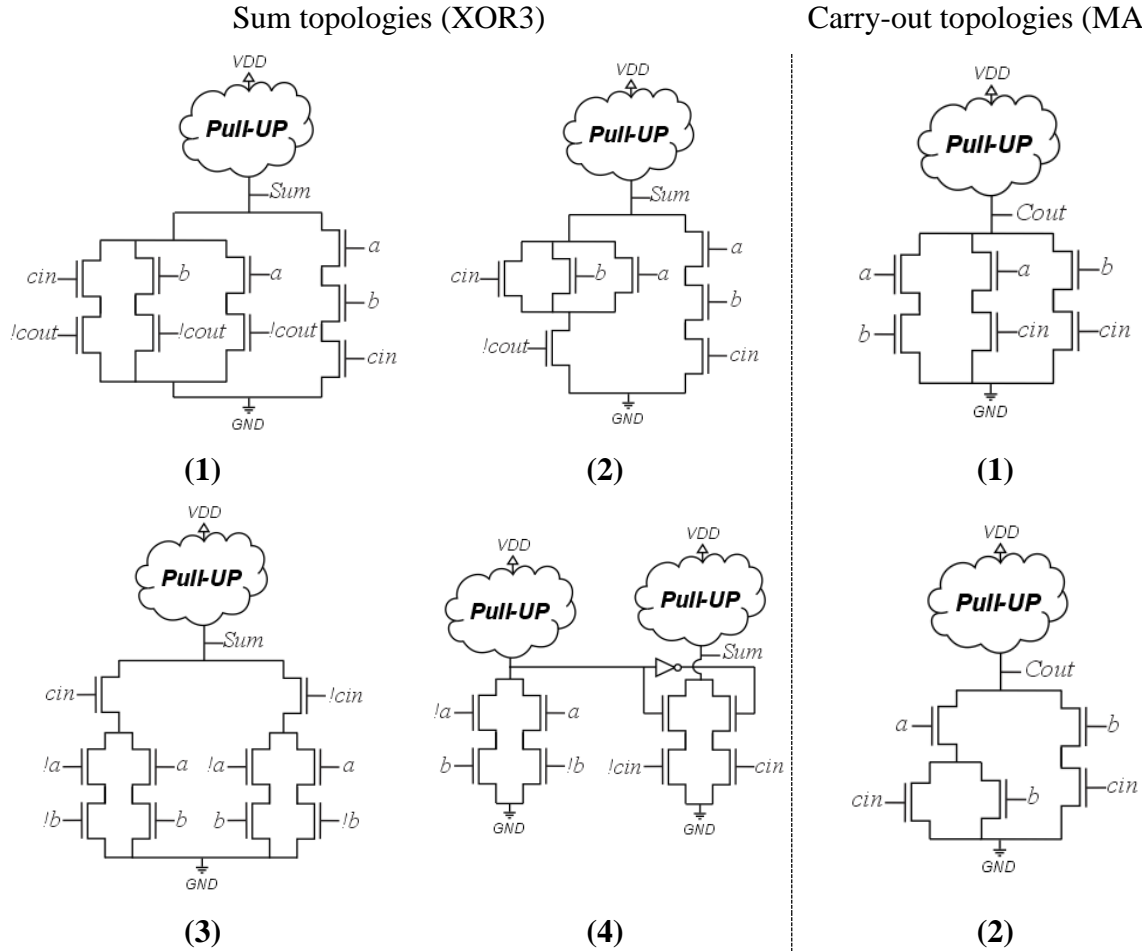


Figure 53: Topologies of XOR3 gate (sum signal) and MAJ3 gate (carry-out signal).

Considering XOR3 and MAJ3 gates shown in Figure 53, eight FA designs were investigated. All possible FA designs were evaluated considering the use of each MIGFET in order to choose the most appropriated FA implementation. In particular, the FA design for each MIGFET was defined through the delay-area product as figure-of-merit. The optimizations in such logic gates were performed manually, aiming to maximize the number of the SG transistor merges. Since the performance of RCA is defined through the propagation of the carry signal, only the delay of the carry-out signal has been taken into account for each FA implementation.

Considering XOR3 and MAJ3 gates shown in Figure 53, eight FA designs were investigated. All possible FA designs were evaluated considering the use of each MIGFET in order to choose the most appropriated FA implementation. In particular, the FA design for each MIGFET was defined through the delay-area product as figure-of-merit. The optimizations in such logic gates were performed manually, aiming to maximize the number of the SG transistor merges. Since the performance of RCA is defined through the

propagation of the carry signal, only the delay of the carry-out signal has been taken into account for each FA implementation.

The estimated area and delay values for each FA design implemented using the MIGFET devices are presented in Table 5 to Table 8. Notice that for some devices the same delay-area product is reached by different FA designs, as presented in Table 5. In this case, the chosen FA topology is defined through the worst delay of the FA design, where are considered the XOR3 gate and the MAJ3 gate delay.

The relationship of chosen XOR3 and MAJ3 gates used to build the most appropriated FA design for each MIGFET technology is presented in Table 9. Notice that, this relationship is defined according to logic gates illustrated in Figure 53. In turn, the resulting FA topologies for each MIGFET technology are shown in Figure 54. In the case of the TIG-FGMOSFET, the MAJ3 gate can be implemented through a single device at each plane.

Table 5: Estimated area and delay values for FA topologies using IG-FinFET according to Figure 53.

FA topology (sum_carry)	Area (A)	Delay carry (D)	A * D	FA Delay
1_1	36.40	15.60	567.84	39.90
1_2	36.40	11.60	422.24	35.90
2_1	36.40	15.60	567.84	26.00
2_2	36.40	11.60	422.24	22.00
3_1	47.32	15.60	738.19	15.60
3_2	47.32	11.60	548.91	13.50
4_1	43.68	15.60	681.41	15.90
4_2	43.68	11.60	506.69	15.90

Table 6: Estimated area and delay values for FA topologies using DG-SiNWFET according to Figure 53.

FA topology (sum_carry)	Area (A)	Delay carry (D)	A * D	FA Delay
1_1	39.10	66.00	2580.60	186.00
1_2	36.80	57.00	2097.60	177.00
2_1	34.50	66.00	2277.00	126.00
2_2	32.20	57.00	1835.40	117.00
3_1	36.80	66.00	2428.80	66.00
3_2	34.50	57.00	1966.50	57.00
4_1	34.50	66.00	2277.00	66.00
4_2	32.20	57.00	1835.40	57.00

Table 7: Estimated area and delay values for FA topologies using TIG-SiNWFET according to Figure 53.

FA topology (sum_carry)	Area (A)	Delay carry (D)	A * D	FA Delay
1_1	23.60	30.00	708.00	78.00
1_2	25.96	63.00	1635.48	129.00
2_1	25.96	30.00	778.80	108.00
2_2	28.32	63.00	1784.16	147.00
3_1	30.68	30.00	920.40	45.00
3_2	33.04	63.00	2081.52	63.00
4_1	28.32	30.00	849.60	36.00
4_2	30.68	63.00	1932.84	63.00

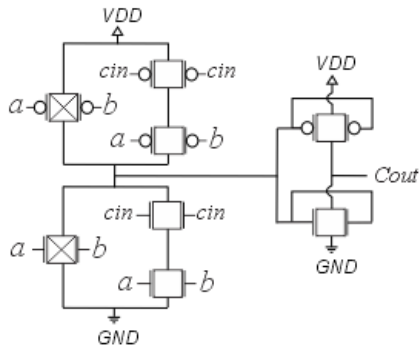
Table 8: Estimated area and delay values for FA topologies using TIG-FGMOSFET according to Figure 53.

FA topology (sum_carry)	Area (A)	Delay carry (D)	A * D	FA Delay
1_1	26.72	32.00	855.04	104.00
1_2	33.4	72.00	2404.80	162.00
2_1	27.1	32.00	867.20	140.00
2_2	33.78	72.00	2432.16	216.00
3_1	36.74	32.00	1175.68	49.00
3_2	43.42	72.00	3126.24	72.00
4_1	33.4	32.00	1068.80	46.00
4_2	40.08	72.00	2885.76	72.00

Table 9: The best XOR3 and MAJ3 gates for each MIGFET according to Figure 53.

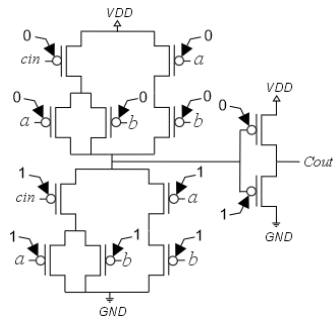
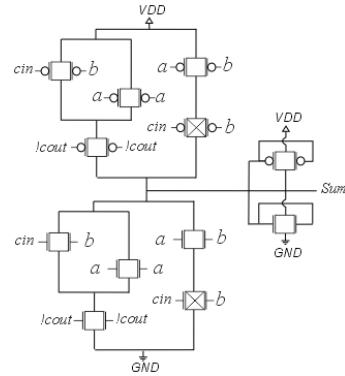
Technologies	Sum topology	Carry-out topology
IG-FinFET	2	2
DG-SiNWFET	4	2
TIG-SiNWFET	1	1
TIG-FGMOSFET	1	-

Carry-out topologies

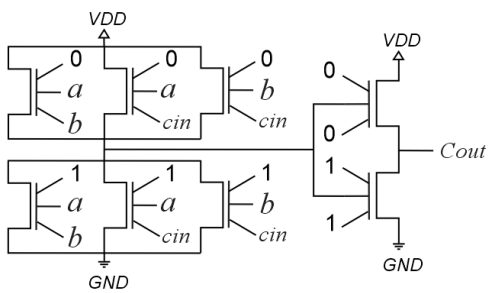
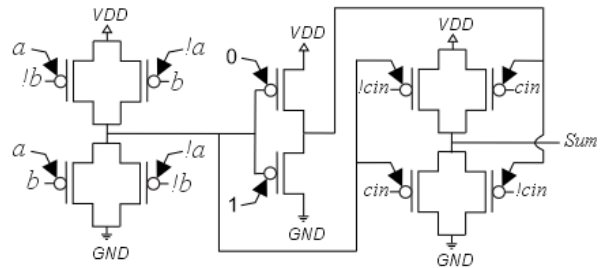


(a)

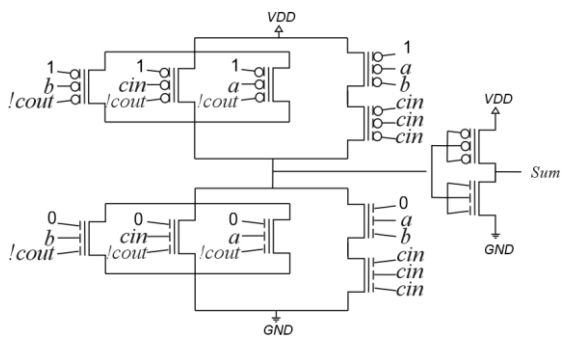
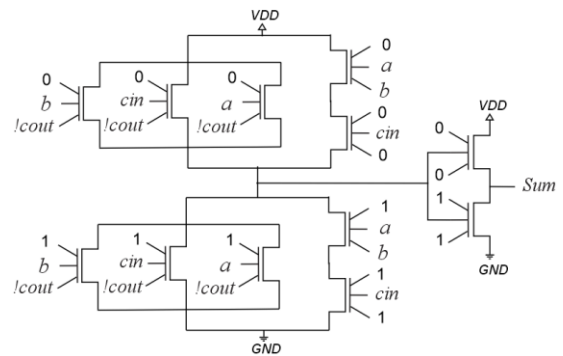
Sum topologies



(b)



(c)



(d)

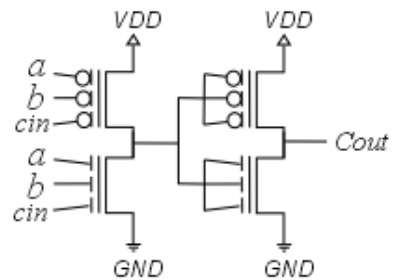


Figure 54: FA designs chosen for each MIGFET technology: (a) IG-FinFET, (b) DG-SiNWFET, (c) TIG-SiNWFET and (d) TIG-FGMOSFET.

Table 10 summarizes the estimated circuit area and delay propagation values for the FA designs chosen considering the MIGFET technologies. Notice that the area and delay values, for each MIGFET are normalized in relation to a FA implemented through the SG version of such device. Since for a RCA, the estimated area and delay are proportional to number of FA adopted, the values presented in Table 10 are used to estimate the delay and the area values of any n-bits RCA design.

Table 10: Area and delay estimated for RCA implemented through the MIGFET devices, normalized in relation to respective SG device of each MIGFET technology.

Technologies	Area (A)	Delay (D)	A*D
SG-FinFET	1.00	1.00	1.00
IG-FinFET	1.30	0.68	0.89
SG-SiNWFET	1.00	1.00	1.00
DG-SiNWFET	1.15	3.35	3.89
TIG-SiNWFET	0.84	1.76	1.49
SG-FGMOSFET	1.00	1.00	1.00
TIG-FGMOSFET	0.95	1.88	1.80

From Table 10, considering the area-delay product, only the FA implemented using IG-FinFET presents improvements in relation to SG FA versions. Moreover, to notice that for some cases, as TIG-SiNWFET and TIG-FGMOSFET, the transistor merging performed at logic level (switch networks) were enough to reduce the physical penalizations of using MIGFETs. Notice also that the area and delay gains for a given target circuit depend on the number of the transistor merges performed at logic level and of the physical cost of the device. In particular, the logic level improvements provided from a given MIGFET tend to vary according to the Boolean function implemented by such device and the target circuit topology.

Another important observation is that the IG-FinFET is the only technology in which the FA design presents a delay lower than respective SG FA design. This occurs due to the number of IG-FinFETs LV-th used in the FA circuits, besides the number of transistor merges performed using IG-FinFET device. As can be seen in Figure 54(a), such an optimization contributes to reduce the transistor stacking on the circuit when compared to the SG version.

On the other hand, the FA design illustrated in Figure 54(d) whose is implemented using TIG-FGMOSFET, is the most compact FA design. Notice that the FA implementation based on TIG-FGMOSFET is benefited from a very simple implementation for MAJ3 gate.

Notice also that the FA design implemented using DG-SiNWFET presents the largest difference of delay value in relation to respective SG FA implementation. Differently from the other MIGFET devices, DG-SiNWFET provides optimization only for the sum gate design. The carry-out gate, in turn, is built according to the conventional SG carry-out circuit topology. In this case, the performance penalty is consequence of the physical costs of implementing the carry-out gate using only SIA version of the DG-SiNWFET.

5.6.3 Gate sizing approach

As the MIGFET adopted in the binary adder circuits are modelled according to the width of the transistor channel, number of fins or number of silicon nanowire of the transistor (DBU), in timing and area analysis, the DBU value could be carefully adjusted in order to obtain better delay and area values for the evaluated circuit. In order to demonstrate such a possibility, a simple gate sizing strategy was adopted to improve the estimated delay of an 8-bits RCA design implemented through TIG-SiNWFET. In particular, the critical path of an RCA is the carry chain, and such chain is composed by a sequence of MAJ3 gates connected to inverters. In this sense, in our experiments, we increased gradually the DBU value of each transistor of the inverters connect to a MAJ3 gates. In Table 11 is shown the estimated area and delay values for the 8-bits RCA considering the DBU variation. Notice that the delay of the 8-bits RCA is improvement up to DBU reaches the value equals to 3. Such an experiment demonstrates that strategies of gate sizing could be investigated in order to improve the adder design. However, it represents a lot of work, maybe a new specific work. For this reason, it was not taken into account herein, being so future work for further improvements.

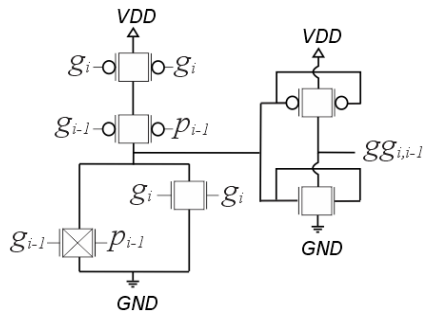
Table 11: Gate sizing application on 8-bits RCA implemented using TIG-SiNWFET, area and delay values are normalized in relation to SG-SiNWFET.

W	8-bits RCA (TIG-SiNWFET)	
	Area (A)	Delay (D)
1	226.60	366.00
2	231.80	351.00
3	237.10	378.00
4	242.40	415.50
5	247.70	457.20

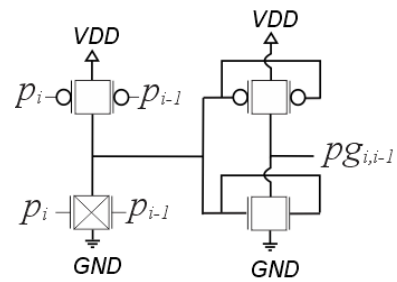
5.6.4 Parallel-Prefixed Adder design

For parallel-prefix adder (PPA) design evaluation, we have considered the Kogge-Stone (KS) (KOGGE; STONE, 1973), Brent-Kung (BK) (BRENT; KUNG, 1982), Ladner-Fischer (LF) (LADNER; FISCHER, 1980) and Han-Carlson (HC) (HAN; CARLSON 1987) architectures. As expected, the area and speed of each PPA design is mostly defined by the arrangement of the generate-propagate operators. Each generate-propagate operator is defined according to equations (50) and (51). In particular, equation (50) is translated to the group generate logic gate, and equation (51) is translated to group propagate logic gate, as shown in Figure 55. These logic gates are combined in order to produce the generate-propagate operator. Notice that in the generate-propagate operator versions presented in Figure 55, the main optimizations performed through the MIGFET devices are obtained from the merge of transistors associated in series and parallel. As observed, such optimizations can be performed by IG-FinFET, TIG-FGMOSFET and TIG-SiNWFET devices. On the other hand, DG-SiNWFET is not suitable to perform such a transistor merging since device does not implement AND2 or OR2 operations, as presented in Figure 55. However, DG-SiNWFET provides a more compact implementation of XOR2 operation, which is used to compute the propagate signal and the sum-bit signal.

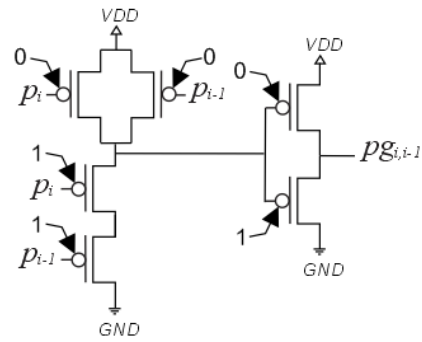
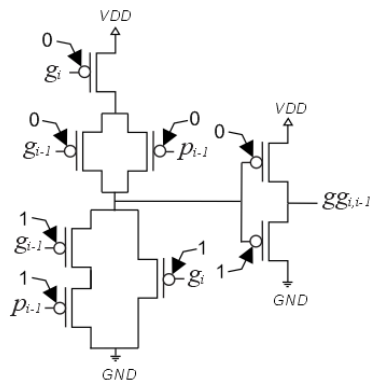
Group generate logic gates (50)



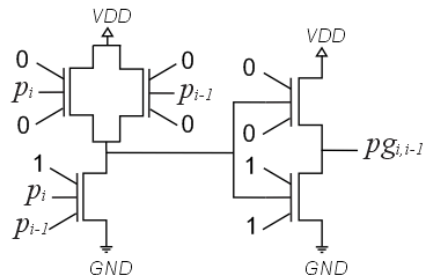
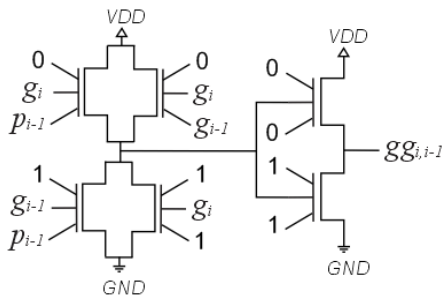
Group propagate logic gates (51)



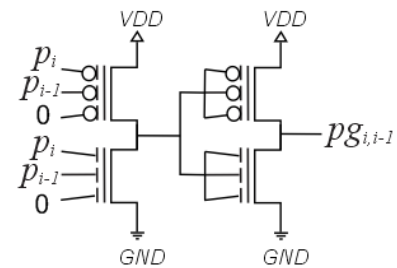
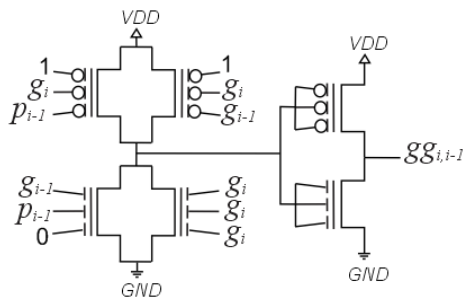
(a)



(b)



(c)



(d)

Figure 55: Generate-propagate operator logic gate topologies: (a) IG-FinFET, (b) DG-SiNWFET, (c) TIG-SiNWFET and (d) TIG-FGMOSFET.

We have estimated the area and delay values for all PPA architectures from 8 to 256 bits. In this experiment, we noticed that the circuit area and delay grow similarly in all the evaluated PPA architectures. Therefore, we choose the 64-bits version as case study, since the analysis can be extended from any word size. In general, the BK PPA architecture presents the best area for all MIGFET devices. In turn, in relation to the best delay, both LF PPA and KS PPA lead to quite similar results. In Table 12 is shown, the results estimated delay and area for all 64-bits evaluated PPA architectures. Notice that the delay values are normalized in relation to KS PPA estimated delay, while the estimated area is normalized in relation to BK PPA area. Similar to FA design evaluation, we defined the best PPA architecture for each MIGFET device according to the area-delay product. From this metric, the best PPA architecture among all MIGFET devices is the LF PPA, as presented in Table 12.

Table 12: Estimated area and delay values for 64-bits PPA implementations considering each MIGFET technologies, area value is normalized in respect to BK PPA area, while delay value is normalized in respect to LF PPA delay.

Technologies	Area (A)				Delay (D)				A * D			
	BK	HC	KS	LF	BK	HC	KS	LF	BK	HC	KS	LF
SG-FinFET	1.00	1.27	1.75	1.27	1.41	1.08	1.00	1.00	1.41	1.38	1.75	1.27
IG-FinFET	1.00	1.28	1.77	1.28	1.51	1.11	1.00	1.04	1.51	1.42	1.77	1.32
SG-SiNWFET	1.00	1.27	1.75	1.27	1.41	1.08	1.00	1.00	1.41	1.38	1.75	1.27
DG-SiNWFET	1.00	1.33	1.92	1.33	1.47	1.10	1.00	1.00	1.47	1.46	1.92	1.33
TIG-SiNWFET	1.00	1.29	1.81	1.29	1.49	1.12	1.00	1.02	1.49	1.44	1.81	1.32
SG-FGMOSFET	1.00	1.27	1.75	1.27	1.41	1.08	1.00	1.00	1.41	1.38	1.75	1.27
TIG-FGMOSFET	1.00	1.28	1.77	1.28	1.52	1.11	1.00	1.03	1.52	1.42	1.77	1.31

In turn, in Table 13, it is shown the estimated area and delay values for the LF PPA implementation considering each MIGFET technology. Notice that each column is normalized with respect to the SG LF PPA implementations. Notice also that, in general considering the area-delay product, the MIGFET optimizations performed at logic level in PPA designs are not enough to compensate the physical level penalizations.

Table 13: Estimated area and delay values for 64-bits LF PAA, normalized in relation to SG version of the 64-bits LF PAA for each MGFET technology.

Technologies	64-bit LF PPA		
	Area (A)	Delay (D)	A*D
SG-FinFET	1.00	1.00	1.00
IG-FinFET	1.28	0.77	0.99
SG-SiNWFET	1.00	1.00	1.00
DG-SiNWFET	1.11	2.60	2.88
TIG-SiNWFET	0.88	1.72	1.51
SG-FGMOSFET	1.00	1.00	1.00
TIG-FGMOSFET	1.17	2.25	2.65

As presented in Table 13, IG-FinFET presents the best tradeoff between area and delay for LF PPA design. Such a situation occurs due to the maximization of the number of IG-FinFET-LV_{th} used in the logic gates adopted in LF PPA implementation, which compensates the number of IG-FinFET-HV_{th} adopted. Moreover, all logic gates that compose the PPA design can be optimized through of series and parallel transistors merging.

As occurred in FA design, LF PPA implementation using DG-SiNWFET is quite similar to conventional SG LF PPA implementation. Excepting for propagate and sum-bit blocks, all other blocks of the PPA structure are built using SIA of the DG-SiNWFET. This way, as such device cannot be fully explored in PPA design, the physical cost of this device impacts negatively in the tradeoff between area and delay for LF PPA design.

In order to evaluate the area/delay tradeoff between RCA and PPA designs for different technologies, in Table 14, the 64-bits LF PPA implementation and 64-bits RCA design are compared. Notice that the estimated area and delay values are normalized in relation to LF PPA. Notice also that, such as in CMOS technology, the PPA designs present an area overhead in relation to RCA design. However, such overhead is compensated by high performance provide through of the PPA structure.

Table 14: Tradeoff between area and delay values for 64-bits RCA and 64-bits LF PAA, delay and area values are normalized in relation to 64-bits RCA.

Technologies	64-bit LF PPA		
	Area (A)	Delay (D)	A*D
SG-FinFET	2.31	0.02	0.05
IG-FinFET	2.00	0.07	0.15
SG-SiNWFET	2.31	0.02	0.05
DG-SiNWFET	2.54	0.06	0.14
TIG-SiNWFET	2.29	0.05	0.12
SG-FGMOSFET	2.31	0.02	0.05
TIG-FGMOSFET	2.60	0.07	0.17

6 MIGFET cell library and ASIC design

In this chapter, we present an automatic logic gate generator method used to build specific MIGFET cell libraries. In this sense, in the experiments performed herein we aim to define an area limit for each MIGFET device in order to obtain circuit area optimizations from transistor count reductions. In particular, the results presented in this chapter can be used as design restrictions in order to obtain more compact MIGFET layouts.

6.1 Technology mapping

Technology mapping is an important step of the logic synthesis that chooses the cells that will be used to implement a given circuit. In general, the cells are chosen in order to minimize a given cost function, as timing restrictions, power consumption or circuit area (KEUTZER, 1987). As shown in Figure 56, the technology mapping can be divided in three main stages: (i) logic decomposition, (ii) pattern matching and (iii) logic covering (CORREIA; REIS, 2004).

In the logic decomposition stage, a given input circuit description is translated to a directed acyclic graph (DAG). In this data structure, each node is represented by a binary operator, as AND2 or OR2 operator. Additionally, inverters can be associated to the edges of the DAG in order to complement the output of the nodes (MINATO, 2012). In particular, the main DAG structure adopted in the technology mapping is the and-inverter graph (AIG) (MISHCHENKO; BRAYTON, 2013). In an AIG, a given circuit is represented through AND2 and inverter operators. Indeed, the results from the technology mapping tend to be benefited by using of AIG due to increase of the granularity of representation of the circuit (MARQUES *et al.*, 2007). However, a higher granularity tends to increase the time to perform the technology mapping task (MARQUES *et al.*, 2007).

In particular, alternatives data structures could be evaluated in order to optimize the technology mapping when MIGFETs are adopted. For instance, a DAG composed of AND2 and OR2 operators could bring benefices to mapping of circuits build from IG-FinFETs since, each node of the DAG could be directly represented through an IG-FinFET. Similarly, the majority inverter graph (MIG) (AMARÚ; GAILLARDON; DE MICHELI, 2016) and BBDD (AMARÚ; GAILLARDON; DE MICHELI, 2014) could be adopted in order to benefice the mapping of circuits built from TIG-FGMOSFET and DG-SiNWFET, respectively.

In the second stage of the technology mapping, a pattern matching routine is adopted. The main goal of this stage is determine the possibilities of sub-function patterns in the AIG that can be covered by the cells available in a given cell library. Basically, a cell library can be defined as a set of logic gates, where each gate corresponds to a cell of the library. In general, in a cell library can have different versions of a same cell with distinct drive strengths and topologies.

Each sub-function pattern found out in the pattern matching stage is stored as a possible candidate to compose the final logic covering of the circuit. In general, pattern matching task can be classified in structural, functional or based on restrictions.

In the structural pattern matching approach, it is checked the existence of isomorphic graphs between the AIG and the graphs that represent the cells of the library. On the other hand, in the functional approach, the pattern matching is performed through a logic check between a part of the AIG and the Boolean functions implemented by the cells of the library. Finally, in the pattern matching based on restrictions, it is checked whether the stored sub-function patterns respect the input restrictions, as maximum number of inputs, or maximum number of transistors associated in series.

In the last stage of the technology mapping it is defined the logic covering of the circuit. In this case, are chosen the most promisor sub-function patterns in order to cover fully the circuit target. Moreover, the sub-function patterns employed in the target circuit are chosen respecting one or more cost functions, as timing, power, and area. Notice that the technology mapping allows to translate a circuit description at register transfer level (RTL) to circuit description at logic gate level. As presented in Figure 56, the output of the technology mapping is a netlist that contains all cell instances used in the mapped circuit and their interconnections.

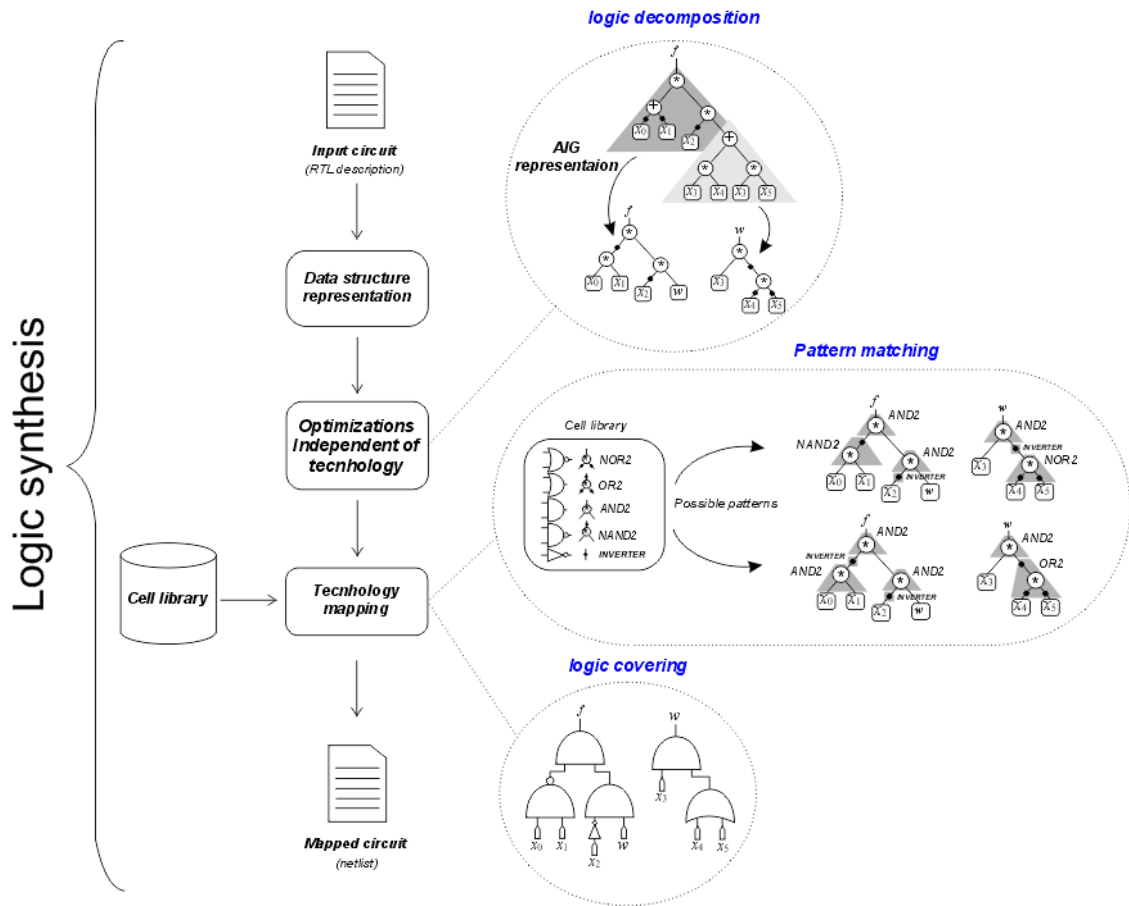


Figure 56: Technology mapping stages.

6.2 Library generator

In this section, we present an automatic logic gate generator method adopted to build MIGFET cell libraries. In this method, the logic gates are implemented using static CMOS style. Basically, the proposed method is divided in three main steps, (i) extraction of derived expressions, (ii) Boolean functions generation, and (iii) logic gate generation.

At first step, an input Boolean expression and a set of MIGFET devices are taken as input of the method, as illustrated in Figure 57. In a derivate expression, each literal is represented through an independent logic input (x_i). Notice that, a distinct logic input is generated for each control gate of a given MIGFET device. Therefore, the number of independent logic inputs is proportional to sum of the control gates of all MIGFETs used to represent a given derivate expression.

In order to generate derived expressions the variables of the support of a given input Boolean function are assigned to MIGFET device present in the input MIGFET set. This way, distinct variables can be assigned to a same MIGFET, as shown in Figure 57(a). The number

of possible assignments between the variables and MIGFETs is M^S , where M is the size of the MIGFET set and the S is the variables support size. In particular, after the assignments between devices and variables, each literal of the input Boolean expression is replaced by the Boolean function implemented through the MIGFET assigned to variable represented by such a literal.

Notice in Figure 57(a) that, the Boolean function f_2 and f_3 are P-class equivalent. In this case, the derivative expression of larger cost is discarded. This cost is computed according to the MIGFETs assigned to the literals of the expression. In particular, the cost associated to a given MIGFET depends on the number of control gates of the device. However, in the example shown in Figure 57(a), the same cost is associated to both derived expressions since the adopted MIGFETs present the same number of control gates. In such a situation, only the smallest function is stored. Each derived expression is represented through a data structure similar to a logic tree, where the leaves nodes represent MIGFET devices, as shown in Figure 57(b).

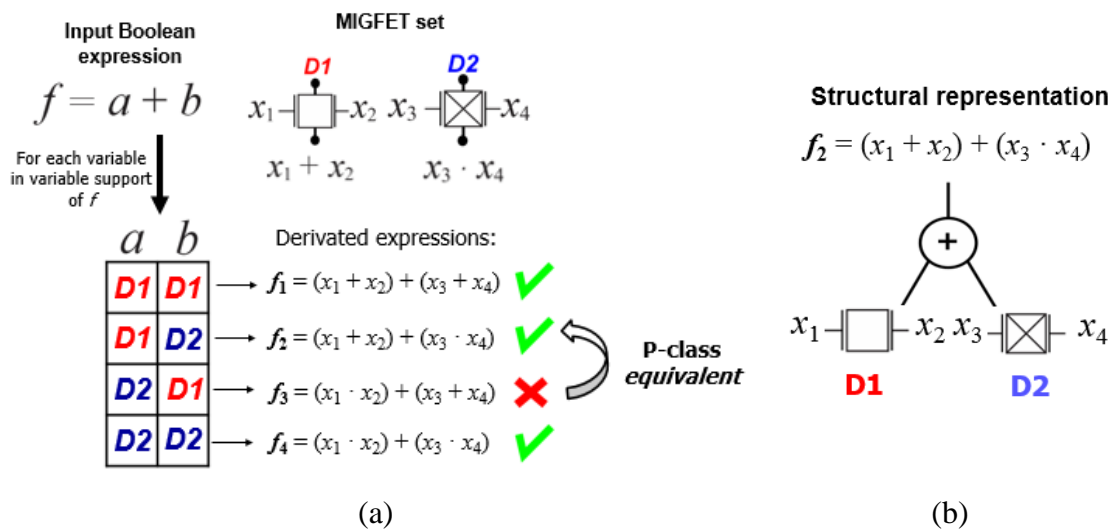


Figure 57: Generation of derived expressions: (a) assignments of variables and MIGFETs, and (b) structural representation of a derived expression.

In Figure 58, the derived expression that implements the Boolean function f_2 in Figure 57(b) is used as input to two distinct routines, the logic assignments generation and the Boolean functions generation. The logic assignments generation routine produces a set of logic values that are applied to a given derived expression. In particular, the logic values set is created considering the number of independent logic inputs of the evaluated derived expression and the logic constant 0 and 1, as shown in Figure 58. Notice that different logic values can be associated to a given independent logic input. This way, the size of the logic

value set is $(N + 2)^N$ where N is the number independent logic inputs of the evaluated derived expression.

On the other hand, the Boolean functions generation routine produces a set of Boolean functions from the derived expression and the logic value set. In particular, logic value assignments for which the evaluated derived expression results in the logic constant 0 or 1 are discarded. As observed in Figure 58, that a general P-signature filter is implemented in the Boolean functions generation routine (DEBNATH; SASAO, 2004). Such a filter ensures that the Boolean functions that belong to same P-class will be represented from a single structure (CHUANG *et al.*, 2013).

Notice that in the second stage of the method a given Boolean function can be produced from different derived expressions. When this situation occurs the logic trees adopted to represent each derived expression are compared. Thus, one of the Boolean functions is discarded according to the sum of the cost of the MIGFET associated to respective derived expression.

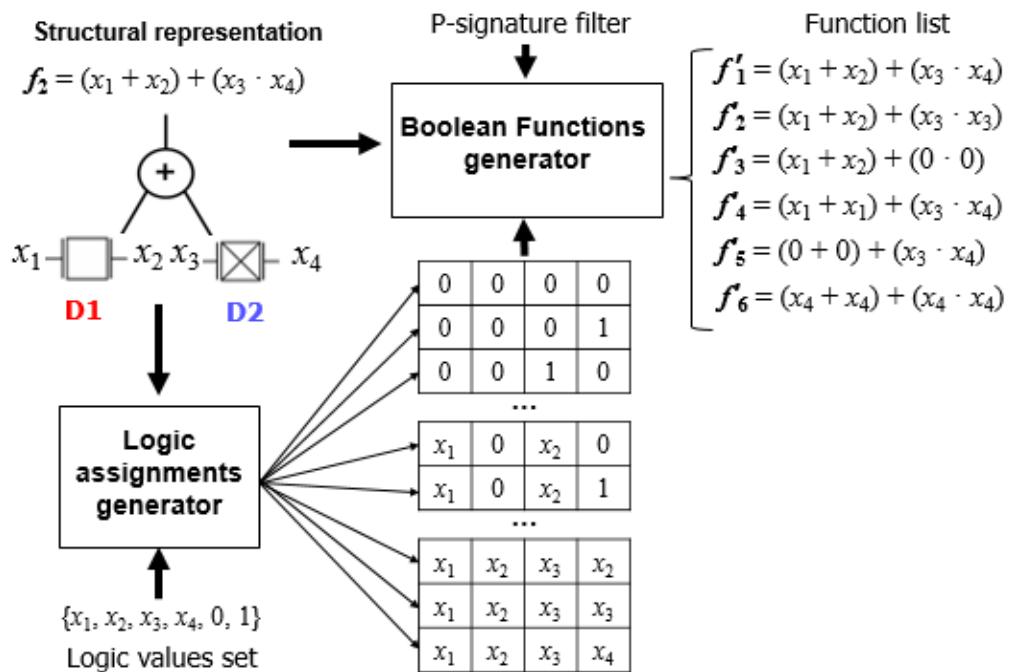


Figure 58: Generation of all possible Boolean function from a given derivate expression.

In the last stage of the proposed method, each Boolean function is translated to a switch network. Such switch network is used to implement the PD plan of the target logic gate. As shown in Figure 59, according to the adopted logic values assignment, different switch networks can be generated from a given derived expression. Notice that, the PD plan of a given logic gate is built directly from a Boolean expression. In this case, only SP switch

networks can be implemented by the method discussed herein. On the other hand, the PU plan is built through the dual expression obtained from the Boolean expression used to build the PD plan.

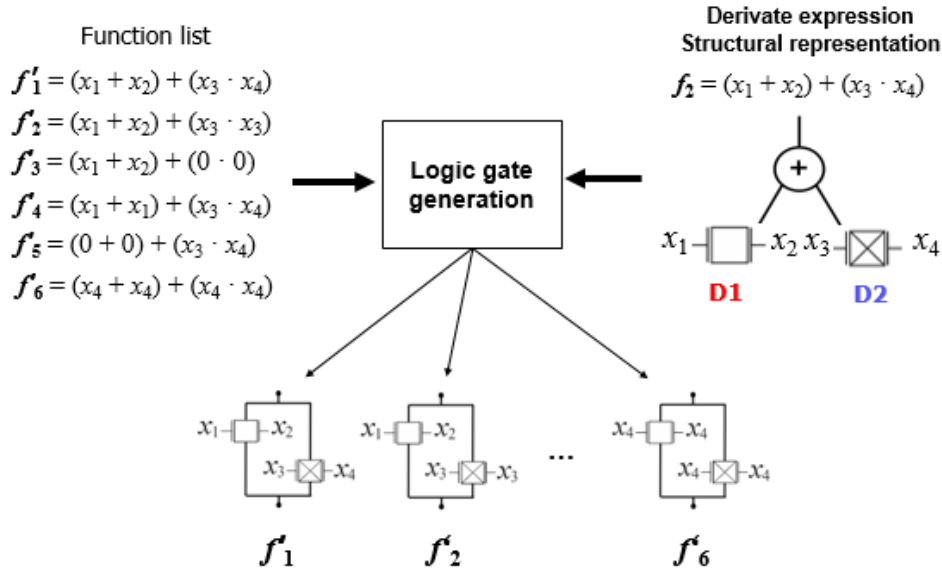




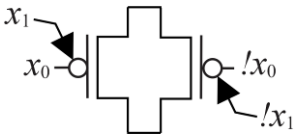
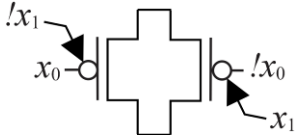
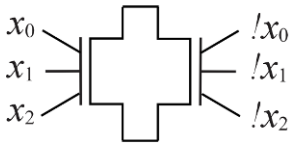
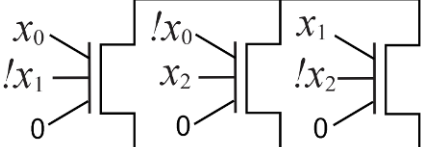
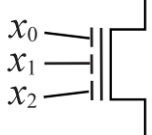
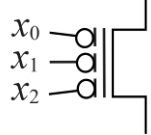


Figure 59: Generation of switch networks from a given derivate expression.

In order to generate the dual expression used in the PU plan, each logic operator OR in the logic tree adopted to represent a given derivate expression is replaced by an AND operator, whereas the AND operators are replaced by OR operators. Moreover, the leaves nodes are translated to a complementary device or a complementary logic arrangement of transistors, according to Table 15. In order to exemplify this situation, notice that the PU plan of the logic gate presented in Figure 60 can be built directly from the dual Boolean expression used to build the PD plan.

As discussed in Chapter 3, each MIGFET admits an N- and P-type version. In particular, P-type version of a given MIGFET device does not necessarily implements the complement of the Boolean function implemented by the N-type version. For instance, the N-type IG-FinFET-HVth implements the AND2 Boolean function, whereas its P-type version implements the NOR2 Boolean function. In this sense, it is assumed that for each MIGFET used in PD plan there is a correspondent MIGFET or transistors arrangement in the PU plan that implements the respective complementary Boolean function, as shown in Table 15. The additional configurations of each MIGFET discussed in chapter 3 also are taken into account in the method presented herein. Notice that for ambipolar devices, DG-SiNWFET and TIG-SiNWFET, additional devices are necessary to prevent signal degradation.

Table 15: Direct and complementary MIGFET switches

Technology	Direct MIGFET or transistors arrangement		Complementary MIGFET or transistors arrangement
IG-FinFET	x_0 —  — x_1 OR2 (x_0, x_1)	↔	x_0 —  — x_1 NOR2 (x_0, x_1)
	x_0 —  — x_1 AND2 (x_0, x_1)	↔	x_0 —  — x_1 NAND2 (x_0, x_1)
DG-SiNWFET	 XNOR2 (x_0, x_1)	↔	 XOR2 (x_0, x_1)
TIG-SiNWFET	 GAMBLE (x_0, x_1, x_2)	↔	 !GAMBLE (x_0, x_1, x_2)
TIG-FGMOSFET	 MAJ3 (x_0, x_1, x_2)	↔	 MIN3 (x_0, x_1, x_2)

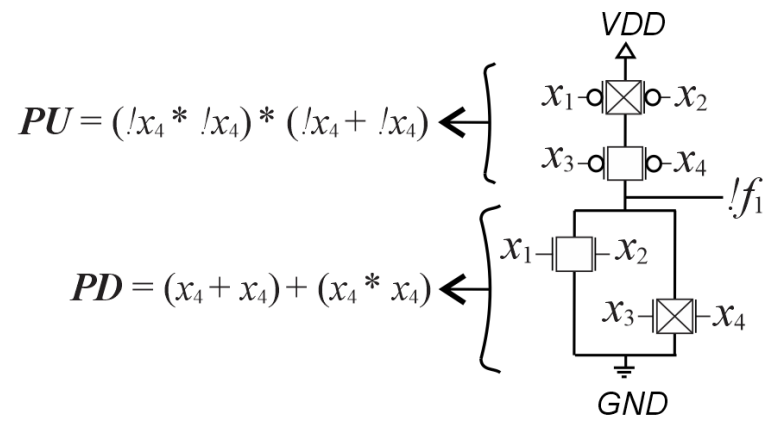


Figure 60: Logic gate built through of the proposed logic gate generator method.

6.3 Benchmark circuit synthesis

In the experiments discussed in this section cell libraries are adopted to evaluate the logic potential of the MIGFET devices. In particular, our goal is to evaluate the area improvements that can be reached in a given circuit. In total, 19 different circuits were analyzed, ten random logic circuits and nine arithmetic circuits. Such a circuits are obtained from the circuit benchmark set available in (LSI, 2017).

6.3.1 Evaluation methodology

Basically the experiment is divided in two main steps: (i) cell libraries generation and (ii) technologic mapping. The design flow adopted in this experiment is shown in Figure 61. Notice that two different types of cell libraries are used, SG and MIGFET libraries, both built from the Genlib library (GENLIB, 2017). In particular, the SG cell library is composed by all the Boolean functions from the Genlib with up to six variables. For each Boolean function added into SG cell library it is added also its complement. Similar to the automatic logic gate generator method presented before, logic gates built from SG devices are implemented directly through Boolean expressions.

On the other hand, cell libraries based on MIGFET devices are built through a reduced set of Boolean functions extracted from the Genlib. Such a set is used as input of the method discussed in previous section. As the automatic logic gate generator method is limited to six variables, for cell libraries built from MIGFETs with three control gates the support of variables of the Boolean functions used as input must be less or equals to two. For cell libraries based on MIGFET devices with two control gates, the input Boolean functions must have a support less or equals to three. In particular, a given MIGFET cell library is restricted to a specific MIGFET technology. Therefore, logic gates built using different MIGFET technologies are not allowed.

In order to balance the number of Boolean functions in the cell libraries, each Boolean function presents only in the SG cell library is implemented in the MIGFET libraries using the SIA version of the respective MIGFET. On the other hand, Boolean functions present only in the MIGFET cell libraries are implemented in SG cell library using SG devices. Such an approach ensures the same Boolean functions set for each cell library produced. In total, each cell library is composed of 396 logic cells.

As shown in Figure 61, each MIGFET cell library is submitted to a replication routine. This way, for each MIGFET technology 10 different versions of a given cell library are built. In particular, a new MIGFET library version is generated to increase the area of the MIGFET devices used in the logic gates of the library. For each new cell library version the area of the MIGFETs grows according to a k-area factor, which increases in 10% to each new library version generated. In particular, the replicated cell libraries are used to define the area limits for each MIGFET technology.

After the replication stage, the MIGFET cell libraries and the SG cell library are pre-characterized according to the delay and area analysis discussed in the Chapter 4. Notice that the characterization stage requires an output capacitance values set besides the transistor models parameters. These capacitance values are needed to generate the liberty files used to represent the cell libraries (SYNOPTIS, 2017). In the next, the cell libraries are submitted to technology mapping task. Therefore, the benchmark circuits are mapped considering each cell library built. Moreover, in technology mapping task we are not using time constraints, so the evaluated circuit is mapped aiming the minimum area. In particular, the RTL compiler (CADENCE, 2017) was used in the experiments in order to map each benchmark circuit.

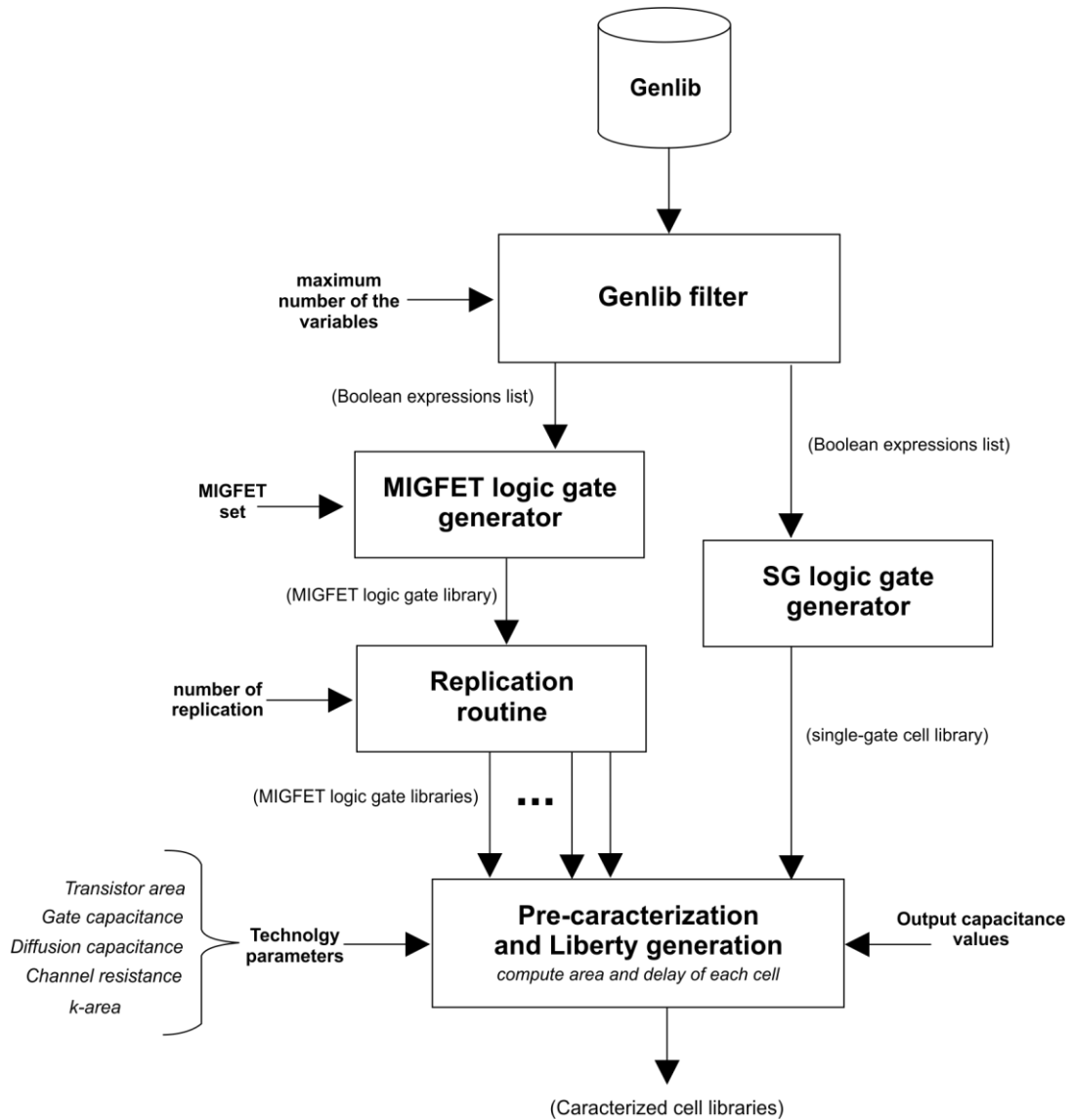


Figure 61: Design flow adopted to generate cell libraries.

6.3.2 Experimental results

In the experiment discussed herein, the area of each circuit mapped from a given MIGFET library is compared to area of an equivalent circuit mapped from a SG cell library. Initially, the benchmark circuits are mapped from MIGFET libraries where each MIGFET device presents the same area than a SG-device. In the next, each benchmark circuit is again mapped using different versions of the MIGFET cell library adopted in the initial mapping. Notice that these libraries are built considering different k-area values. This way, as the k-area factor increases in 10% for each new MIGFET cell library, it is possible to evaluate, in the mapped circuit, the area penalizations provided by increasing the dimensions of the used MIGFETs.

In general, when a circuit is mapped using a given MIGFET library, two different types of logic gate can be adopted. Basically, logic gates can be built using MIGFETs in SIA versions or through optimized arrangements of MIGFET devices. In particular, logic gates built from optimized logic arrangements will be denoted as optimized logic cell (COP).

In particular, the main metric used to evaluate a given mapped benchmark circuit is the area of the MIGFETs used in such a circuit. Notice that the device area can be adjusted according to number of COP present in the mapped circuit. This way, in Table 16 are shown the maximum extra area allows to each MIGFET technology in order to ensure the circuit area improvements provided by transistor count reductions. The values presented in Table 16 are obtained to compare the area of the benchmark circuits mapped from the MIGFET libraries to area of the equivalent circuit mapped through the SG cell library.

Table 16: Maximum device area per MIGFET technology considering the benchmark circuits, the values are normalized in relation to mapped circuit using SG devices.

Circuits	IG-FinFET	DG-SiNWFET	TIG-SiNWFET	TIG-FGMOSFET
Adder	1.60	1.20	1.20	1.90
Bar	2.00	1.10	1.20	1.80
Div	1.70	1.10	1.20	1.50
Log2	1.80	1.10	1.20	1.60
Max	1.80	1.00	1.10	1.90
Multiplier	1.70	1.10	1.20	1.60
Sin	1.80	1.10	1.20	1.60
Sqrt	1.90	1.00	1.20	1.80
Square	1.70	1.10	1.20	1.50
Arbiter	1.70	1.00	1.10	1.60
Cavlc	1.80	1.00	1.00	1.50
Ctrl	1.80	1.00	1.00	1.60
Dec	2.00	1.10	1.10	1.80
I2c	1.70	1.00	1.10	1.60
Int2float	1.80	1.00	1.10	1.50
Mem_ctrl	1.90	1.00	1.10	1.50
Priority	1.70	1.10	1.10	1.60
Router	1.70	1.00	1.10	1.60
Voter	1.70	1.10	1.20	1.80
Average	1.78	1.06	1.14	1.65

Considering the area limits presented in Table 16 is possible to notice that the design rules presented in the Chapter 3 must be adjusted in order to produce more compact MIGFET layouts. Moreover, MIGFET devices that implement AND2 and OR2 Boolean functions tend to present a superior area limit than other MIGFET technologies, as it is the case of the IG-FinFET and TIG-FGMOSFET. Another important observation is that in the case of the DG-SiNWFET and TIG-SiNWFET, the logic gate area increasing provided by using transmission gate structure tends to reduce the circuit area improvements when COP are adopted.

In Table 17 is shown the total of the logic cell CUSE used in each mapped benchmark circuit with respect to the 369 cells available in each library. Notice that among the CUSE it is highlighted the total of used COP. Moreover, in Table 17 is also shown the circuit area required by the COP instances (AOP). Notice that these values are presented through the percentage of area of the mapped circuit occupied by the COP.

Table 17: Logic cells used in the mapped circuits.

Circuits	IG-FinFET			DG-SiNWFET			TIG-SiNWFET			TIG-FGMOSFET		
	COP	CUSE	AOP	COP	CUSE	AOP	COP	CUSE	AOP	COP	CUSE	AOP
Adder	7	7	100.00	4	6	67.18	4	6	71.53	7	7	100.00
Bar	4	4	100.00	2	5	6.68	4	5	60.66	5	5	100.00
Div	42	42	100.00	9	69	48.73	16	54	74.12	25	25	100.00
Log2	46	46	100.00	14	81	45.11	20	61	83.95	27	37	99.39
Max	52	52	100.00	6	62	20.67	12	51	36.29	29	20	99.25
Multiplier	22	22	100.00	10	37	49.57	15	25	90.09	17	17	100.00
Sin	40	40	100.00	11	58	45.91	20	53	79.82	28	30	99.90
Sqrt	42	42	100.00	7	65	70.05	11	48	72.16	28	28	100.00
Square	43	43	100.00	9	51	54.34	15	50	73.01	24	24	100.00
Arbiter	64	64	100.00	6	93	30.39	15	83	59.73	24	25	99.87
Cavlc	37	37	100.00	6	68	36.61	10	43	55.03	20	20	100.00
Ctrl	13	13	100.00	7	17	48.54	7	14	39.58	13	13	100.00
Dec	5	5	100.00	5	5	100.00	4	4	100.00	5	5	100.00
I2c	43	43	100.00	9	64	33.24	13	41	64.22	24	24	100.00
Int2float	32	32	100.00	6	46	38.06	12	32	67.80	15	15	100.00
Mem_ctrl	87	87	100.00	21	200	32.87	24	151	64.85	28	28	100.00
Priority	38	38	100.00	4	49	26.61	8	43	41.12	24	27	97.23
Router	23	23	100.00	3	24	22.81	10	23	50.35	18	19	97.30
Voter	22	22	100.00	8	24	17.37	12	22	82.47	20	20	100.00
Average	35	35	100.00	8	54	41.83	12	43	66.67	20	20	99.63

Notice also that, for some MIGFET technologies the number of the COP used during the mapping is more expressive than for other ones. In general, the number of the COPs adopted in the mapped circuit tends to improve the circuit area, since that these cells present a reduced number of devices. Consequently, the maximum area limit per MIGFET devices tends to increase according to the number of the COP adopted in the mapped circuit, as it is possible to notice to compare the results shown in Table 17 to area limits presented in Table 16.

In general, the Boolean functions implemented through a given MIGFET have a significant impact on the logic functions set that will be implemented in a given cell library. For instance, any cell library based on DG-SINWFET tends to present in its composition XOR-based Boolean functions. Consequently, some MIGFET libraries tend to be more adequate to evaluated benchmark circuits due to the Boolean function set available in such a library.

CONCLUSIONS

MIGFET devices have demonstrated great potential for digital integrated circuit design. These devices tend to present more compact solutions than traditional SG switches at logic level due to its higher logic capability. However, the use of MIGFET may lead to losses and penalties when there are not enough simplifications in the number of switches to compensate the use of such a bigger devices. Basically, the logic improvements performed by a given MIGFET depends on the topology of the target circuit and the Boolean function set implemented through the device. On the other hand, at physical level, the effectiveness of MIGFETs depends exclusively on the physical dimensions of the device area defined from the design rules adopted in the layout drawing of each MIGFET.

In this work, we have presented an extensive evaluation about the impact of using different MIGFETs in the design of the digital integrated circuits. In particular, this work presented three main contributions.

Firstly, through a careful study about MIGFET layout, it was proposed a simple RC transistor model to estimate the performance of MIGFET devices and to evaluate qualitatively the delay of circuits implemented in MIGFET technologies. The main advantages of this model is that its simplicity can be easily extend to MIGFET devices with more than three independent gates. Moreover, this model can be adopted in a future study about gate and transistor sizing since, in the proposed RC transistor model, the performance of MIGFET devices varies according to number of fins, number of silicon nanowires or width of the transistor adopted.

Secondly, another contribution of this work is the MIGFET method used to generate the logic gates adopted in the standard cell experiments. Indeed, this method does not ensure the optimized logic gates implementation but can be used to generate MIGFET cell libraries. Moreover, future optimizations can be performed in such a method in order to select specific input vectors and build cell libraries focusing in low power and high performance. Furthermore, optimizations can be preformed in the logic gates to remove redundant MIGFET switches with all control gate connected to constant logic values 0 or 1.

Finally, from binary adders designed herein, it was demonstrated that the optimizations performed through a given MIGFET at logic level, in terms of the transistor count, may not be enough to ensure the area and performance improvements at physical level. Therefore, in this case, it is needed to improve the manufacturing process in order to obtain

more efficient binary adder implementations. In this context, it was identified area limits for each MIGFET devices studied in order to ensure that the transistor count reductions result in area optimizations. These area limits were defined from standard cell design flow experiments applied on specific circuit benchmark set. In particular, the area limits defined in the standard cell experiments can be used in future works in order to reach more compact MIGFET layouts. It is important to notice that, in this work, the standard cell experiments were adopted to evaluate the impacts of the MIGFET on the area of digital integrated circuits. However, the evaluated flow developed in this work is easily configurable from design parameters.

REFERENCES

- AGOSTINELLI, M.; ALIOTO, M.; ESSENI, D.; SELMI, L. Leakage–delay tradeoff in FinFET logic circuits: A comparative analysis with bulk technology. **IEEE transactions on very large scale integration (VLSI) systems**, v. 18, n. 2, p. 232-245, 2010.
- AKERS, S. B. A rectangular logic array. **IEEE Transactions on Computers**, v. 21, n. 8, p. 848-857, 1972.
- AKERS, S. B. Binary decision diagrams. **IEEE Transactions on Computers**, v. 27, n. 6, p. 509-516, 1978.
- ALIOTO, M.; BONGIOVANNI, S.; DJUKANOVIC, M.; SCOTTI, G.; TRIFILETTI, A. Effectiveness of leakage power analysis attacks on DPA-resistant logic styles under process variations. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 61, n. 2, p. 429-442, 2014.
- ALIOTO, M. Comparative evaluation of layout density in 3T, 4T, and MT FinFET standard cells. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 19, n. 5, p. 751-762, 2011.
- ALIOTO, M.; PALUMBO, G. A simple strategy for optimized design of one-level carry-skip adders. **IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications**, v. 50, n. 1, p. 141-148, 2003.
- AMARÚ, L.; HILLS, G.; GAILLARDON, P. E.; MITRA, S. Multiple Independent Gate FETs: How many gates do we need? In: **Design Automation Conference (ASP-DAC)**, p. 243-248, 2015.
- AMARÚ, L.; GAILLARDON, P. E.; DE MICHELI, G. Efficient arithmetic logic gates using double-gate silicon nanowire FETs. In: **New Circuits and Systems Conference (NEWCAS)**, 2013. 1-4.
- AMARÚ, L.; GAILLARDON, P. E.; DE MICHELI, G. Biconditional binary decision diagrams: A novel canonical logic representation form. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, v. 4, n. 4, p. 487-500, 2014.
- AMARÚ, L.; GAILLARDON, P. E.; DE MICHELI, G. Majority-Inverter Graph: A New Paradigm for Logic Optimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 35, n. 5, p. 806-819, 2016.
- ANIL, K. G.; HENSON, K.; BIESEMANS, S.; COLLAERT, N. Layout density analysis of FinFETs. In: **European Solid-State Device Research**, 2003. 139-142.
- ANIS, M.; ALLAM, M.; ELMASRY, M. Impact of technology scaling on CMOS logic styles. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, v. 49, n. 8, p. 577-588, 2002.

BATE, J. A.; MUZIO, J. C. Three cell structures for ternary cellular arrays. In: **Proceedings of the sixth international symposium on Multiple-valued logic**, 1976. 55-60.

BEN-JAMAA, M. H.; MOHANRAM, K.; DE MICHELI, G. An efficient gate library for ambipolar CNTFET logic. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 30, n. 2, p. 242-255, 2011.

BEZ, R.; CAMERLENGHI, E.; MODELLI, A.; VISCONTI, A. Introduction to flash memory. **Proceedings of the IEEE**, v. 91, n. 4, p. 489-502, 2003.

BHOJ, A. N.; JHA, N. K. Design of logic gates and flip-flops in high-performance FinFET technology. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 21, n. 11, p. 1975-1988, 2013.

BOBBA, S.; GAILLARDON, P. E.; ZHANG, J.; DE MARCHI, M.; SACCHETTO, D.; LEBLEBICI, Y.; DE MICHELI, G. Process/design co-optimization of regular logic tiles for double-gate silicon nanowire transistors. In: **Nanoscale Architectures (NANOARCH)**, 2012. 55-60.

BOBBA, S.; DE MICHELI, G. Layout Technique for Double-Gate Silicon Nanowire FETs With an Efficient Sea-of-Tiles Architecture. **IEEE Transactions on Large Scale Integration (VLSI) System**, v. 23, n. 10, p. 2103-2115, 2015.

BRAYTON, R. K. Factoring logic functions. **IBM Journal of Research and Development**, v. 31, n. 2, p. 187-198, 1987.

BRAYTON, R. K.; HACHTEL, G. D.; MCMULLEN, C.; SANGIOVANNI-VINCENTELLI, A. **Logic minimization algorithms for VLSI synthesis**. MA, USA.: Kluwer Academic Publishers, 1984.

BRENT, R. P.; KUNG, H. T. A regular layout for parallel adders. **IEEE Transactions on Computers**, v. 31, n. 3, p. 260-264, 1982.

BRYANT, R. E. Symbolic Boolean manipulation with ordered binary-decision diagrams. **ACM Computing Surveys (CSUR)**, v. 24, n. 3, p. 293-318, 1992.

CADENCE. ECSM Library Format. **Effective Current Source Model ECSM**, 2017. Disponivel em: <https://www.cadence.com/content/cadence-www/global/en_US/home/alliances/standards-and-languages/ecsm-library-format.html>. Acesso em: 2017.

CADENCE. Encouter RTL Compiler. **CADENCE**, 2017. Disponivel em: <https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/84441.html>. Acesso em: 12 mar. 2017.

CHIANG, M. H.; KIM, K., TRETZ, C.; CHUANG, C. T. Novel high-density low-power logic circuit techniques using DG devices. **IEEE Transactions on Electron Devices**, v. 52, n. 10, p. 2339-2342, 2005.

CHIANG, M. H.; KIM, C. T.; TRETZ, C. 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, 2006.

CHRZANOWSKA, J.; MALGORZATA, X. Y.; PERKOWSKI, M. Logic synthesis for a regular layout. **VLSI Design**, v. 10, n. 1, p. 35-55, 1999.

CHUANG, P.; LI, D.; SACHDEV, M. Constant delay logic style. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 21, n. 3, p. 554-565, 2013.

COLINGE, J. P. **FinFETs and other multi-gate transistors**. New York: Springer, 2008.

CONG, J. J.; KWOK-SHING, L. Optimal wiresizing under Elmore delay model. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 14, n. 3, p. 321-336, 1995.

CORREIA, V.; REIS, A. Advanced technology mapping for standard-cell generators. In: **symposium on Integrated circuits and system design**, p.254-259, 2004.

CORTADELLA, J. Area-optimal transistor folding for 1-D gridded cell design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 32, n. 11, p. 1708-1721, 2013.

COUDERT, O. Two-level logic minimization: an overview. **Integration, the VLSI journal**, v. 17, n. 2, p. 97-140 1994.

CUI, T.; XIE, Q.; WANG, Y.; NAZARIAN, S.; PEDRAM, M. 7nm FinFET standard cell layout characterization and power density prediction in near-and super-threshold voltage regimes. In: **Green Computing Conference (IGCC)**, 2014. 1-7.

CUI, T.; CHEN, B.; WANG, Y.; NAZARIAN, S.; PEDRAM, M. Layout Characterization and Power Density Analysis for Shorted-Gate and Independent-Gate 7nm FinFET Standard Cells. In: **Great Lakes Symposium on VLSI** , 2015. 33-38.

DA ROSA, L. S. J.; REIS, A. I.; RIBAS, R. P.; MARQUES, F. D. S.; SCHNEIDER, F. R. A comparative study of CMOS gates with minimum transistor stacks. In: **Proceedings of the 20th annual conference on Integrated circuits and systems design**, 2007. 93-98.

DATTA, A.; GOEL, A.; CAKICI, R. T.; MAHMOODI, H.; LEKSHMANAN, D.; ROY, K. 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, 2007.

DAVILA-SALDIVAR, C.; MEDINA-VAZQUEZ, A. S.; JIMENEZ-PEREZ, A.; GURROLA-NAVARRO, M. A. Extracting the floating gate voltage on the multiple-input FG MOS transistor. In: **Electrical Engineering, Computing Science and Automatic Control (CCE)**, 2014. 1-4.

DE MARCHI, M.; SACCHETTO, D.; FRACHE, S.; ZHANG, J.; GAILLARDON, P. E.; LEBLEBICI, Y.; DE MICHELI, G. Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs. In: **International Electron Devices Meeting (IEDM)**, 2012. 8-4.

DE MICHELI, G. **Synthesis and optimization of digital circuits**. McGraw-Hill Higher Education, 1994.

DOYLE, B. S.; DATTA, S.; DOCZY, M.; HARELAND, S.; JIN, B.; KAVALIEROS, J.; CHAU, R. High performance fully-depleted tri-gate CMOS transistors. **Electron Device Letters**, v. 24, n. 4, p. 263-265, 2003.

ELMORE, W. C. The transient response of damped linear networks with particular regard to wideband amplifiers. **Journal of applied physics**, v. 19, n. 1, p. 55-63, 1948.

FRANK, D. J.; DENNARD, R. H.; NOWAK, E.; SOLOMON, P. M.; TAUR, Y.; WONG, H. S. P. Device scaling limits of Si MOSFETs and their application dependencies. In: **Proceedings of the IEEE**, vol. 89, no. 3, Mar 2001. pp. 259-288.

GAILLARDON, P. E.; AMARÚ, L. G.; BOBBA, S.; DE MARCHI, M.; SACCHETTO, D.; DE MICHELI, G. Nanowire systems: Technology and design. **Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences**, v. 372, n. 2012, p. 20130102, 2014.

GANGE, G.; SONDERGAARD, H.; STUCKEY, P. J. Synthesizing Optimal Switching Lattices. **ACM Transactions on Design Automation of Electronic Systems**, 1 November 2014. 1 - 14.

GHANI, T.; MISTRY, K.; PACKAN, P.; THOMPSON, S.; STETTLER, M.; TYAGI, S.; BOHR, M. Scaling challenges and device design requirements for high performance sub-50 nm gate length planar CMOS transistors. **VLSI Technology Symposium**, 2000. 174-175.

GNANI, E.; GNUDI, A.; REGGIANI, S.; BACCARANI, G. Physical model of the junctionless UTB SOI-FET. **Transactions on Electron Devices**, v. 59, n. 4, p. 941-948, 2012.

GOLUMBIC, M. C.; MINTZ, A. Factoring logic functions using graph partitioning. In: **Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design**, 1999. 195-199.

GOLUMBIC, M. C.; MINTZ, A.; ROTICS, U. An improvement on the complexity of factoring read-once Boolean functions. In: **Discrete Applied Mathematics**, v. 156, n. 10, p. 1633-1636, 2008.

GUYOT, A.; HOCHET, B.; MULLER, J. M. A way to build efficient carry-skip adders. **IEEE Trans. Computers**, v. 36, n. 10, p. 1144-1152, 1987.

HAENSCH, W.; NOWAK, E. J.; DENNARD, R. H.; SOLOMON, P. M.; BRYANT, A.; DOKUMACI, O. H.; FISCHETTI, M. V.; Silicon CMOS devices beyond scaling. **IBM Journal of Research and Development**, v. 50, n. 4.5, p. 339 - 361, 2006.

HAN, T.; CARLSON, D. A. Fast Area-Efficient VLSI Adders. **IEEE Symposium on Computer Arithmetic**, 1987. 49-56.

HASLER, P.; MINCH, B. A.; DIORIO, C. Floating-gate devices: they are not just for digital memories any more. In: **ISCAS**, 1999. 388-391.

HINSBERGER, U.; KOLLA, R. Boolean matching for large libraries. **Design Automation Conference**, 1998. 206-211.

HSIEH, C. Y.; FAN, M. L.; HU, V. P. H.; SU, P.; CHUANG, C. T. Independently-controlled-gate FinFET Schmitt trigger sub-threshold SRAMs. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 20, n. 7, p. 1201-1210, 2012.

HUANG, X.; LEE, W. C.; KUO, C.; HISAMOTO, D.; CHANG, L.; KEDZIERSKI, J.; SUBRAMANIAN, V. Sub 50-nm finfet: Pmos. In: *Electron Devices Meeting*. p. 67-70. 1999.

JU, P.; HANDSCHIN, E.; KARLSSON, D. Nonlinear dynamic load modelling: model and parameter estimation. **IEEE Transactions on Power Systems**, v. 11, n. 4, p. 1689-1697, 1996.

KAGARIS, D.; HANIOTAKIS, T. A methodology for transistor-efficient supergate design. **IEEE transactions on very large scale integration (VLSI) systems**, v. 15, n. 4, p. 488-492, 2007.

KEUTZER, K. DAGON: technology binding and local optimization by DAG matching. In: **Design Automation Conference on IEEE**, 1987. 341-347.

KOGGE, P. M.; STONE, H. S. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. **IEEE Transactions on Computers**, v. C-22, n. 8, p. 786-793, 1973.

KOZIOL, K.; VILATELA, J.; MOISALA, A.; MOTTA, M.; CUNNIFF, P.; SENNETT, M.; WINDLE, A. High-performance carbon nanotube fiber. **Science**, v. 318, n. 5858, p. 1892-1895, 2007.

KUHN, K. J. Considerations for Ultimate CMOS Scaling. **Transactions on Electron Devices**, v. 59, n. 7, p. 1813-1828, 2012.

LADNER, R. E.; FISCHER, M. J. Parallel Prefix Computatoin. **Journal of the ACM**, v. 27, n. 4, p. 831-838, 1980.

LIU, Y.; MATSUKAWA, T.; ENDO, K.; MASAHARA, M.; ISHII, K.; YAMAUCHI, H.; SUZUKI, E. Cointegration of high-performance tied-gate three-terminal FinFETs and variable threshold-voltage independent-gate four-terminal FinFETs with asymmetric gate-oxide thicknesses. **Electron Device Letters**, v. 28, n. 6, p. 517-519, 2007.

LOPEZ-MARTIN, A. J.; RAMIREZ-ANGULO, J.; CARVAJAL, R. G.; ACOSTA, L. CMOS transconductors with continuous tuning using FGMOS balanced output current scaling. **IEEE Journal of Solid-State Circuits**, v. 43, n. 5, p. 1313-1323, 2008.

LOPEZ-MARTIN, A. J.; ALGUETA, J. M.; GARCIA-ALBERDI, C.; ACOSTA, L.; CARVAJAL, R. G.; RAMIREZ-ANGULO, J. Design of micropower class AB transconductors: A systematic approach. **Microelectronics Journal**, v. 44, n. 10, p. 920-929, 2013.

LSI. MIGFET Synthesis Package, 2017. Disponivel em: <<http://lsi.epfl.ch/MIGFET>>. Acesso em: 24 jan. 2017.

LYNCH, T.; SWARTZLANDER, E. E. A spanning tree carry lookahead adder. **Transactions on Computers**, v. 41, n. 8, p. 931-939, 1992.

MARQUES, F. S.; DA ROSA, J. L. S.; RIBAS, R. P.; SAPATNEKAR, S. S.; REIS, A. I. DAG based library-free technology mapping. **ACM Great Lakes symposium on VLSI**, 2007. 293-298.

MARTINS, M.; MATOS, J. M.; RIBAS, R. P.; REIS, A.; SCHILINKER, G.; RECH, L.; MICHELSEN, J. Open cell library in 15nm FreePDK technology. **International Symposium on Physical Design**, 2015. 171-178.

MARTINS, M. G.; DA ROSA, J. L. S.; RASMUSSEN, A. B.; RIBAS, R. P.; REIS, A. I. Boolean factoring with multi-objective goals. **Computer Design (ICCD), 2010 IEEE International Conference on. IEEE**, 2010. 229-234.

MEAD, C.; CONWAY, L. **Introduction to VLSI systems**. [S.l.]: Reading, MA: Addison-Wesley, v. Vol. 1080, 1980.

MINATO, S. I. **Binary decision diagrams and applications for VLSI CAD**. Springer Science & Business Media, Vol. 342, 2012.

MINCH, B. A. A folded floating-gate differential pair for low-voltage applications. In: **ISCAS 2000**, 2000. 253-256.

MISHCHENKO, A.; BRAYTON, R. Faster Logic Manipulation for Large Designs. **International Workshop on Logic and Synthesis (IWLS)**, 2013.

MOHAMMADI, H. G.; GAILLARDON, P. E.; DE MICHELI, G. From defect analysis to gate-level fault modeling of controllable-polarity silicon nanowires. **IEEE Transactions on Nanotechnology**, v. 14, n. 6, p. 1117-1126, 2015.

MOHANTY, B. K.; AND PATEL, S. K. Area-Delay-Power efficient carry-select adder. **IEEE transactions on circuits and systems II: express briefs**, v. 61, n. 6, p. 418-422, 2014.

MONDRAGÓN-TORRES, A. F.; SCHNEIDER, M. C.; SÁNCHEZ-SINENCIO, E. Well-driven floating gate transistors. **Electronics Letters**, v. 38, n. 11, p. 530-532, 2002.

MURGAI, R.; BRAYTON, R. K.; SANGIOVANNI, V. A. **Logic synthesis for field-programmable gate arrays**. [S.l.]: Springer Science & Business Media, 2012.

NEUTZLING, A.; MATOS, J. M.; REIS, A. I.; RIBAS, R. P.; MISHCHENKO, A. Threshold logic synthesis based on cut pruning. **Computer-Aided Design (ICCAD)**, 2-6 Nov. 2015. 494 - 499.

NOEL, J. P.; THOMAS, O.; JAUD, M. A.; WEBER, O.; POIROUX, T.; FENOUILLET-BARANGER, C.; ROZEAU, O. Multi-Vt UTBB FDSOI Device Architectures for Low-Power CMOS Circuit. **Transactions on Electron Devices**, v. 58, n. 8, p. 2473-2482, 2011.

PAI, Y. T.; CHEN, Y. K. The fastest carry lookahead adder. **Electronic Design, Test and Applications**, 2004. 434-436.

PIGUET, C. Design of low-power libraries. **Electronics, Circuits and Systems**, 1998. 175-180.

POSSANI, V. N., REIS, A. I.; RIBAS, R. P.; MARQUES, F. S.; DA ROSA, L. S. J. Exploring independent gates in FinFET-based transistor network generation. **Proceedings of the 27th Symposium on Integrated Circuits and Systems Design**, 2014. 41.

POSSANI, V. N., REIS, A. I.; RIBAS, R. P.; MARQUES, F. S.; DA ROSA, L. S. J. Transistor Count Optimization in IG FinFET Network Design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, 2016.

POSSANI, V. N., CALLEGARO, V.; REIS, A. I.; RIBAS, R. P.; MARQUES, F. S.; DA ROSA, L. S. J. Graph-based transistor network generation method for supergate design. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 24, n. 2, p. 692-705, 2016.

RABAEY, J. M.; CHANDRAKASAN, A. P.; NIKOLIC, B. **Digital integrated circuits**. Englewood Cliffs: Prentice hall, v. 2, 2002.

RAMKUMAR, B.; AND KITTUR, H. M. Low-power and area-efficient carry select adder. **IEEE transactions on very large scale integration (VLSI) systems**, v. 20, n. 2, p. 371-375, 2012.

RIBES, G.; MITARD, J.; DENAIS, M.; BRUYERE, S.; MONSIEUR, F.; PARTHASARATHY, C.; GHIBAUDO, G. Review on high-k dielectrics reliability issues. **IEEE Transactions on Device and Materials Reliability**, v. 5, n. 1, p. 5-19, 2005.

RIEL, H.; WERNERSSON, L. E.; HONG, M.; DEL ALAMO, J. A. III-V compound semiconductor transistors-from planar to nanowire structures. **Mrs Bulletin**, v. 39, n. 8, p. 668-677, 2014.

RODRIGUEZ-VILLEGAS, E.; BARNES, H. Solution to trapped charge in FGMOS transistors. **Electronics Letters**, v. 39, n. 19, p. 1416-1417, 2003.

ROSTAMI, M.; MOHANRAM, K. Dual-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, 2011.

RUIZ, A. G. Evaluation of three 32-bit CMOS adders in DCVS logic for self-timed circuits. **IEEE Journal of Solid-State Circuits**, v. 33, n. 4, p. 604 - 613, 1998.

SHANNON, C. E. A symbolic analysis of relay and switching circuits. **Electrical Engineering**, v. 57, n. 12, p. 713-723, 1938.

SHIBATA, T.; OHMI, T. A functional MOS transistor featuring gate-level weighted sum and threshold operations. **IEEE**, *IEEE Transactions on Electron devices*, v. 39, n. 6, p. 1444-1455, 1992.

SORDAN, R.; TRAVERSI, F.; AND RUSSO, V. Logic gates with a single graphene transistor. **Applied Physics Letters**, v. 94, n. 7, p. 51, 2009.

SUTHERLAND, I. E.; SPROULL, R. F.; HARRIS, D. F. **Logical Effort: Designing Fast CMOS Circuits**. Morgan Kaufmann, 1999.

SYNOPTYS. SYNOPTYS Newsroom. **Composite Current Source (CCS)**, 2017. Disponível em: <<http://news.synopsys.com/index.php?s=20295&item=122723>>. Acesso em: 2017.

TAWFIK, S. A.; KURSUN, V. Low-power and compact sequential circuits with independent-gate FinFETs. **IEEE Transactions on Electron Devices**, v. 55, n. 1, p. 60-70, 2008.

TENACE, V.; CALIMERA, A.; MACII, E.; PONCINO, M. One-pass logic synthesis for graphene-based Pass-XNOR logic circuits. In: **Design Automation Conference (DAC)**, 2015. 1-6.

UEHARA, T.; VANCLEEMPUT, W. M. Optimal layout of CMOS functional arrays. **Design Automation Conference**, 1979. 287-289.

UNWALA, I. H.; AND SWARTZLANDER, E. E. Superpipelined adder designs. **Circuits and Systems, ISCAS '93, International Symposium**, 1993. 1841 - 1844.

YANG, K.; ANDREOU, A. G. Multiple input floating-gate MOS differential amplifiers and applications for analog computation. **Midwest Symposium on Circuits and Systems**, 1993. 1212-1216.

YEO, Y. C.; KING, T. J.; HU, C. Effects of high- κ gate dielectric materials on metal and silicon gate workfunctions. **Electron Device Letters**, v. 23, p. 342-344, 2002.

YIN, L.; EMBABI, S. H.; SANCHEZ-SINENCIO, E. A floating-gate mOSFET d/A converter. In: **ISCAS**, 1997. 409-412.

ZHANG, J. **Multiple-Independent-Gate Field-Effect Transistors for High Computational Density and Low Power Consumption**, 2016. PHD thesis, École Polytechnique Fédérale De Lausanne, 2017.

ZHANG, J.; TANG, X.; GAILLARDON, P. E.; DE MICHELI, G. Configurable circuits featuring dual-threshold-voltage design with three-independent-gate silicon nanowire FETs. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 61, n. 10, p. 2851-2861, 2017.

ZHANG, J.; GAILLARDON, P. E.; DE MICHELI, G. Dual-threshold-voltage configurable circuits with three-independent-gate silicon nanowire FETs. In: **Circuits and Systems (ISCAS)**, 2013. 2111-2114.

ZHU, J.; ABD-EL-BARR, M. On the optimization of MOS circuits. **IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications**, v. 40, n. 6, 1993.

ZIMMERMANN, R.; FICHTNER, W. Low-power logic styles: CMOS versus pass-transistor logic. **IEEE journal of solid-state circuits**, v. 32, n. 7, p. 1079-1090, 1997.