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

GERSON SCARTEZZINI

LOW-POWER DESIGN USING NETWORKS OF TRANSISTORS

Submitted in partial fulfillment of the requirements for the degree of Master in Computer Science with emphasis on Microelectronics.

Advisor: Prof. Dr. Ricardo Augusto da Luz Reis

Porto Alegre
2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Scartezzini, Gerson

Low-Power Design Using Networks of Transistors [manuscrito] / Gerson Scartezzini. – 2014.

84 f.:il.

Orientador: Ricardo Augusto da Luz Reis.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2014.

1. Computer-Aided Design. 2. Logical Mapping 3. Low-Power Systems. 4. Networks of Transistors. 5. Microelectronics I. Reis, Ricardo Augusto da Luz. II. III. Título.

ACKNOWLEDGMENTS

My time at UFRGS has been tremendously valuable for me, both professionally as well as personally. I reckon there are very few institutions where one is surrounded by so many people with brilliant minds and unwavering passion for what they do.

First of all, I thank God for giving me health, strength and persistence to complete this work.

My parents, Sérgio Antônio Scartezzini and Teresinha De David Scartezzini, who were the first investors in the enterprise of my education, and without their perseverance none of this could have been possible. Also, I would like to thank my brothers, Gilson and Jackson Scartezzini, for their support and companionship.

I give my special thanks to my wife, my guardian angel and my best friend, Elaine Pereira Lima Scartezzini. Without her support, caring, understanding, comforts, and love, I would not be where I am today.

I thank my colleagues and friends from NSCAD and PHYDEG groups for their unrestricted support, enthusiasm and generosity. I am grateful that I shared the lab with great people.

Finally, I sincerely thank my advisors, Ricardo Augusto da Luz Reis. Your advice and support has made this work possible. Thank you for giving me this research opportunity.

“Persistence is the shortest path to success”

— Charles Chaplin

ABSTRACT

In complex integrated circuits, power and performance have moved in opposite directions making the design of low-power devices a highly costly task. Traditionally, integrated circuit design companies adopt many techniques to ensure power requirements, however, techniques based on cell library has become a bottleneck for the development process. As the design complexity and density increase, greater will be the power dissipated, and thus its reduction becomes more important. Seeking to increase the power reduction capability, designers have applied different techniques for each level of the design flow abstraction. At the physical level, so as to bypass the limits of cell libraries, the development of specifically designed cells has become a routine for designs with large power constraints.

Observing this requirement, this work aims to investigate the implementation and optimization of digital static CMOS (Complementary Metal-Oxide-Semiconductor) cell at transistors level, and the use of library free design methodology as a resource for designing low power systems. In general, fewer transistors are desirable to reduce power dissipation, however, long chains of transistors, necessary for implementing specific logical functions, leads to the increase of the transition time, and hence greater energy dissipation. In order to avoid this effect, we constructed a mapping function, based on transistor size, in order to avoid slow transition time and minimize the number of transistors.

The use of this method has proven effective for fine adjustment low power circuits, resulting in an average reduction of 6.35% in dynamic power and 8.26% in static power as compared with the cell library based methodology. As further work, an automated flow set is presented for the logical mapping able to generate specific networks of transistors for each design, making possible their use in traditional design tools.

Keywords: Computer-Aided Design, Logical Mapping, Low-Power Systems, Networks of Transistors, Microelectronics.

REDES DE TRANSISTORES PARA O DESENVOLVIMENTO DE PROJETOS DE BAIXO CONSUMO

RESUMO

Em circuitos integrados complexos, potência e desempenho têm caminhado em direções opostas tornando o desenvolvimento de dispositivos de baixo consumo uma tarefa altamente custosa. Tradicionalmente, empresas de desenvolvimento de circuitos integrados utilizam variadas técnicas para garantir os requisitos de potência, no entanto, técnicas baseadas em biblioteca de células tem se tornado um gargalo para o processo de desenvolvimento. À medida que os projetos aumentam de complexidade e densidade, maior tende a ser a potência dissipada por estes dispositivos, e assim, mais importante torna-se sua redução. Buscando aumentar a capacidade de redução de potência, projetistas tem aplicado diferentes técnicas para cada nível de abstração do fluxo de projeto. No nível físico, de maneira a contornar os limites das bibliotecas de células, o desenvolvimento de células especificamente projetadas tem se tornado uma rotina em projetos com grandes restrições de potência. Observando este requisito, este trabalho visa pesquisar a implementação e otimização de células digitais CMOS *(Complementary Metal-Oxide-Semiconductor)* estática em nível de transistores, e o emprego de metodologia de projeto livre de biblioteca como um recurso para a concepção de sistemas de baixa potência. De um modo geral, menos transistores são desejáveis para reduzir a dissipação de potência, no entanto, longas cadeias de transistores, necessários para implementar funções lógicas específicas, conduz ao aumento do tempo de transição, e, portanto, maior dissipação de energia. A fim de evitar este efeito, construímos uma função de mapeamento, com base no tamanho dos transistores, de forma a evitar um tempo de transição lento e minimizar o número de transistores. O uso deste método demonstrou ser eficaz para o ajuste fino de circuitos de baixa potência, resultando em uma redução média de 6.35% no consumo dinâmico e de 8.26% no consumo estático em comparação com a metodologia baseada em biblioteca de células. Como trabalho adicional, é apresentado um fluxo automatizado de mapeamento lógico e capaz de gerar redes de transistores específicas para cada projeto, tornando possível sua utilização em ferramentas de desenvolvimento tradicionais.

Palavras-chaves: EDA, Mapeamento Lógico, Projetos de Baixo Consumo, Redes de Transistores, Microeletrônica.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASIC	Application-Specific Integrated Circuit
AVFS	Adaptative Voltage and Frequency Scaling
BDD	Binary Decision Diagram
BJT	Bipolar Junction Transistor
BSIM	Berkeley Short-channel IGFET Model
BTBT	Band-To-Band Tunneling
CAD	Computer-aided design
CG	Clock Gating
CMOS	Complementary Metal-Oxide-Semiconductor
DAG	Directed Acyclic Graph
DRC	Design Rule Check
DVFS	Dynamic Voltage and Frequency Scaling
ECB	Electron tunneling in the Conduction Band
EDA	Electronic Design Automation
EVB	Electron tunneling in the Valence Band
FPGA	Field-Programmable Gate Array
GIDL	Gate-Induced Drain Leakage
HDL	Hardware Description Languages
HVB	Hole tunneling in the Valence Band
IR-Drop	Voltage Drop
ITRS	International Technology Roadmap for Semiconductors
LUT	Look-Up Table
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MSV	Multi Supply Voltage
NMOS	N-type Metal-Oxide-Semiconductor
OTR	Odd-level Transistor Replacement
PMOS	P-type Metal-Oxide-Semiconductor
PTL	Pass Transistor Logic
PVT	Process, Voltage and Temperature
RTL	Register Transfer Level
SOC	System-On-Chip
SOI	Silicon-On-Insulator
V_{th}	Threshold Voltage

TABLE OF CONTENTS

INTRODUCTION

Along last decades, we observed a huge progress on microelectronics technology. Integrated devices became essential when applied in different areas and for different purposes. Performance requirements moved the progress. Requirements to process more information in the least amount of time resulted in faster components. At the same time, devices became denser, with more and more components placed closer in the same area, as predicted by Moore's Law [MOORE, 1965].

In order to achieve the requirements, designers developed different design methodologies. Not so far, companies applied full-custom design techniques [CHEN, 2000] to build handcrafted designs. For each component a set of masks were drawn and verified, assembling the whole circuit layout. Due to the handmade nature of the method, area, speed and power [MICHELI, 1994] could be managed and fine-tuned, in detriment of a long development time.

Aiming to minimize the time to market, companies recognized that many blocks repeated several times in the project and started to build sets of functional blocks to increase the project reusing. Reusing functional blocks, hereinafter called cells, ramp up a new design methodology based on pre-designed libraries, i.e. standard cell methodology [FISHER, 1996].

In a typical standard cell methodology, the flow starts with a high-level description and several steps runs in order to obtain a manufacturable physical layout. The main advantage of this methodology comes from the fact each cell is fully characterized, resulting in a set of accurate information about the behavior of the cells, e.g. delay, power and area. According to these characteristics, designers could evaluate the behavior of the system early in the design flow.

Current, integrated circuit design based on standard cells allows the design of denser systems; nevertheless, on novel process technologies such integration turns the development flow a challenge. Phases like design, synthesis, verification and test become more complex, and aspects like power and variability stands as a challenge on modern system designs.

Despite the development of different methods for power minimization, on last technologies power became a critical issue faced for the process scaling. Problems involving heat removal and cooling are worsening stimulated for the power dissipated on those systems.

Nowadays, a big challenge is to minimize the amount of energy consumed by devices keeping the same performance.

Along years, many techniques were developed aiming to reduce power dissipation [GONZALEZ, 1997] [DE-SHIUAN, 2006] [JEONG, 2006], however most efficient ones imposes limits to the performance. According to the International Technology Roadmap for Semiconductors 2011 (ITRS 2011) there was an important improvement related to the power in last years, as illustrated in Figure 1, however, to achieve the expected goals for low-power designs there is still a large gap to the desirable target.

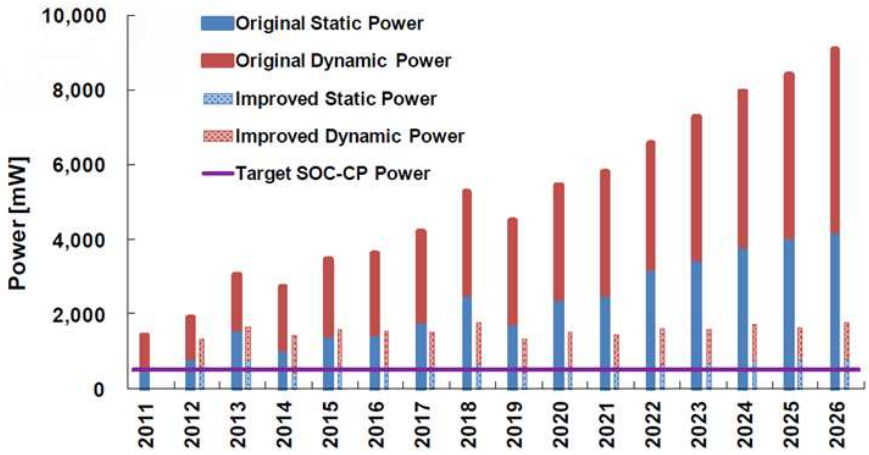

Figure 1: Impact of Low-Power Design Technology on Portable SOC Power Consumption (ITRS Design ITWG 2011).

In order to improve low-power capability, designers should apply power optimization in all abstraction levels of the design flow [CHANDRAKAZAN, 1995]. This optimization involves since the algorithm used to implements the circuit, up to the architecture, design and the technological node used for manufacture.

At algorithm level, the designer explores the temporal characteristics of the circuits to implement sequential or parallel architectures in order to minimize the number of operators, e.g. adders and multipliers. As consequence, area can be reduced, which leads to lower static and dynamic power dissipation [CHANDRAKAZAN, 1995]. Architectural level optimization aims to implement specific algorithm with parallel and/or pipelined structures to achieve same performance in reduced supply voltages or frequency [BENINI, 2001]. In design level, goal is to optimize both physical and logical design. Logical means to minimize the structure required to implement the modeled architecture, i.e. make optimizations in a technology-independent logical representation, using Boolean and algebraic techniques, and in the technology mapping phase [PEDRAN, 2005]. Physical, on the other hand, represent

optimizations in place and route stages, transistor sizing [PEDRAN, 1997] or primarily in library, through the development of digital cells optimized to speed, area and power. In technology level, manufacture features are the main factor, e.g. the use of high-K or metal gates [KAPUR, 2004], dimensions and concentration of the components as oxide thickness and substrate implantation, and device structure, as Silicon-On-Insulator (SOI).

According [BRATTACHARYA, 2002] digital design automation tools and methodologies have failed to keep pace with the increased ability of fabrication processes to pack tens, or even hundreds of millions of devices on a die. Consequently, the quality of designs created using cell-based automated tools has fallen far behind the quality of handcrafted designs [CHINNERY, 2000]. As an example, this gap is most visible in the area of performance, and actually in power. While the handcrafted digital designs are approaching up to 5 GHz clock speed mark, getting the standard-cell based synthesized designs to work at 500 MHz at state-of-the-art 90 nm and 65 nm process nodes, remains a serious challenge. Heavy manual intervention to achieve lower delay has become commonplace, and manual creation of design-specific tactical cells has become virtually routine for cell based designs.

1.1 MOTIVATION AND ORGANIZATION

In the previous statements we verified a huge challenge for the development of state-of-art designs. In complex integrated circuits, power and performance tends to be in opposite directions, and comply with the power specifications are a complex task.

Companies use to apply several techniques to ensure the power requirements without costly redesign; however, traditional design techniques are inefficient. As designs become denser and complex, greater tends to be the power dissipated, and most importantly is the reduction, in some cases making design-specific cells a routine on state-of-art cell based designs. Observing this requirements, this work addresses the logical implementation and optimization of digital design-specific static CMOS cells at transistor level, and the employment of library-free design methodology [BERKELAAR, 1988] [REIS, 1997] [MARQUES, 2007] [ZIESEMER, 2007] [MARQUES, 2008] [REIS, 2008] [REIS, 2011] as a powerful resource for design of low power systems.

Throughout this work, we studied the influence on the number of transistors allied to the size of the transistors in power dissipation. In general, fewer transistors are desirable to reduce power dissipation; however, long chains of transistors leads to slow transition time, and hence more power dissipation due short-circuit current. In order to prevent this fact, we

build a mapping function, based on the size of the transistors, to avoid slow transition time and minimize the number of transistors.

The use of this method demonstrated to be effective, resulting in a lower power dissipation compared to pre-defined standard cells approach. As an incremental work, we developed an automated flow for logical mapping capable of generating sized design-specific networks of transistor, leading to the use of these in traditional commercial place and routing tools.

The remaining of this work is organized as follow. In section 2, fundamentals aspects of power consumption and design flow are explained. Section 3, present a review of the literature and related works. Section 4 present preliminary studies. In section 5 is presented the methodology adopted for this work. In section 6, a set of experiments in submicron technologies are presented comparing the proposed method to the standard cells approach, finally, in section 7, conclusions are summarized.

2 BASIC CONCEPTS

2.1 DIGITAL DESIGN FLOWS

The rapid progress in design brings the need of agile methodologies and procedures to enable the development of designs in short time to market; the adoption of rigid methodologies and project strategies has become essential for the process design automation of integrated circuits.

As methodologies, innumerous approaches were adopted along the years. These approaches range from high performance, handcrafted design to fully programmable medium-to-low performance designs [RABAEY, 2002]. In Figure 2, based on [MICHELI, 1994], is possible to visualize the different methodologies for digital integrated circuit design. In order to clarify some terms used on this work, two approaches are explained in details: Cell Library and Library-Free based methodologies.

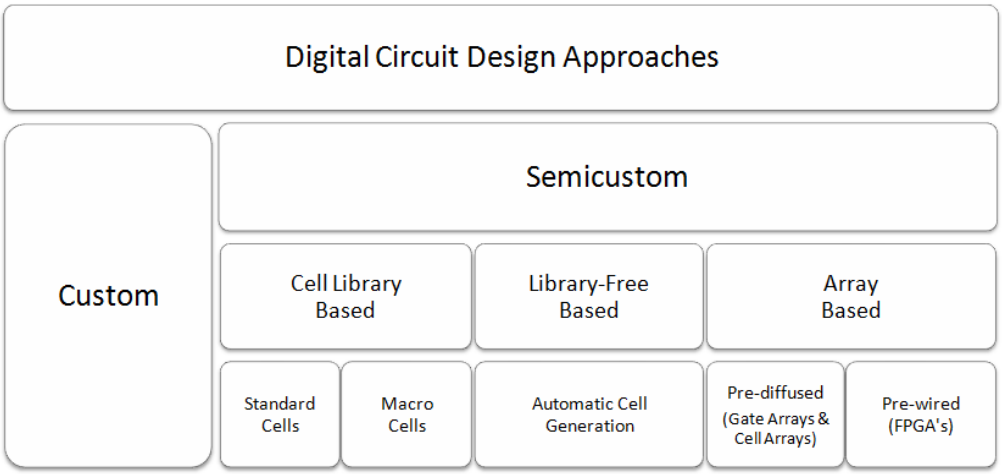

Figure 2: Overview of design methodologies for digital integrated circuits.

2.1.1 Cells Library Based Design

The reason for the use of cell-based design is to reduce the implementation effort by reusing a discrete library of cells, called standard cells. As an advantage of this methodology, the cells are designed and characterized only once for a determined technology node, making possible its reuse for many times, decreasing the design cost. As disadvantage, the constrained nature of the library reduces the opportunity of fine-tuning the design [RABAEY, 2002]. Typical cell libraries do not contain more than 150 different functions. However, the possible space for logical functions that can be mapped with a specific number of stacked transistors is much larger, as can be seen in Table 1.

Table 1: Number of possible different functions
using a limited number of stacked transistors [DETJENS, 1987]

Number of stacked NMOS transistors	Number of stacked PMOS transistors: 1	2	3	4	5
1	1	2	3	4	5
2	2	7	18	42	90
3	3	18	87	396	1677
4	4	42	396	3503	28435
5	5	90	1677	28435	125803

2.1.1.1 Standard Cells

For a given technology, a set of standardized cells are created containing a limited collection of logical gates over a range of fan-in and fan-outs, complementary physical gates like fillers and diodes are available too. These set of cells are known as standard cells, and are the primitives for the cell libraries, widely used in cell based designs. In a general library, besides the basic logical function, such as inverters, and/nand, or/nor, xor/xnor and registers, also contain complex functions, such and-or-inverter, muxes, adders, decoder and encoders [WESTE, 2005].

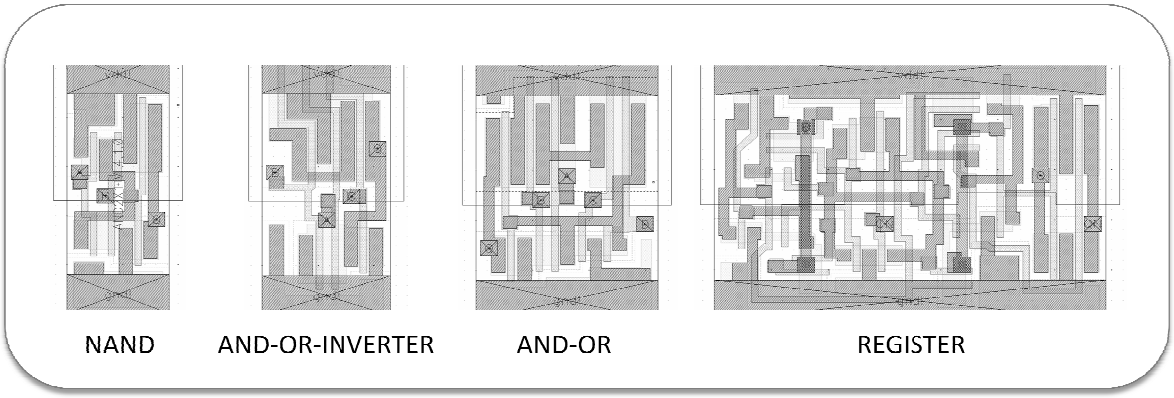

Figure 3: Example of standard cell layout.

Traditional standard cells have a fixed height, as illustrated in Figure 3, with power and ground pins routed in fixed sides, normally on top and bottom, fact that allows the cells to be easily abutted to the power rails [WESTE, 2005]. Usually, these cells try to use minimum-width transistors to reduce capacitance, and the widest transistor that fit within the height defined for the standard library. In cases where larger transistors are necessary, i.e. to

develop cells with larger current capability, transistor folding [HER, 1993] is required to ensure the standard height, resulting increased cell width.

According [RABAEY, 2002] to determine the composition of the library is a non-trivial task. It is difficult to measure in advance what is better: to create small libraries in which most cells have a limited fan-in, or to build a large one, with many versions of the same gate, i.e. cells with two, three or four inputs of the same logical function and different sizes for each one. Common practice uses the second option, ensuring a vast range of cells, with different sizes and characterization corner to be used in logical synthesis. While this practice turns design process more reliable, also may cause a detrimental effect, increasing the complexity of the technology mapping process.

2.1.1.2 Design Flow

Standard cells flow, as illustrated in Figure 4, is the most implemented methodology for commercial EDA companies. The mainly motivation for its use is the flexible and reliable flow based on a set of pre-designed and verified cells, i.e. standard cells library, distributed for foundries or third-party companies, as [XFAB] and [FARADAY].

Typical standard cell flow can be divided by three categories: (i) Test, (ii) Verification and (iii) Implementation. The first two, respectively, aims to validate the post-silicon prototype and verify the behavior of the implemented design, according to the functional and structural specification. Both uses specific methodologies and EDA tools, however, to explain each one in details is not the focus of this work. At this point, special attentions will be given for the implementation.

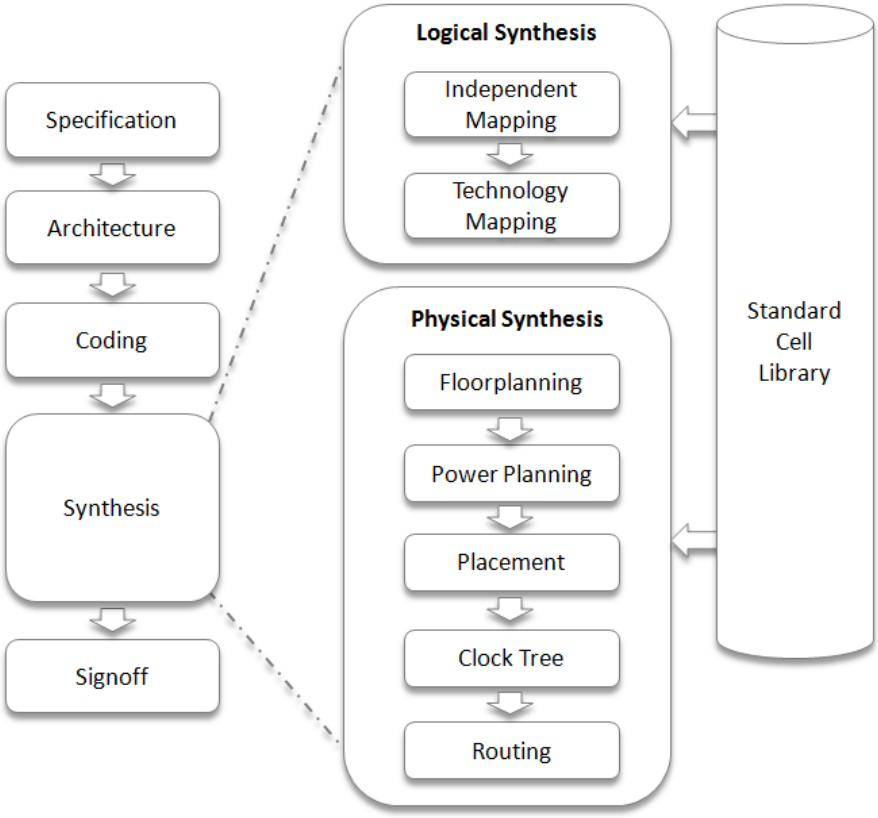

Figure 4: Traditional implementation cell-based design flow.

Implementation comprises all the steps from the specification to the layout, and it is subdivided into several well-defined steps, which are: specification, architecture, coding, logical synthesis, physical synthesis and sign-off.

- **Specification:** Technical leader defines the functionality of the system, in concordance with the customer, development team, marketing department and other stakeholders.
- **Architecture:** Use simulation environment to "prototype" or validate the system operation. By modeling architectures, designers perform numerous simulations of the system before building itself to ensure that it achieves the goals detailed in the specification.
- **Coding:** Aims to describe the functionality of the module or system architecture using a Hardware Description Language (HDL), abstracting the details regarding the technological node.
- **Logical synthesis:** Consists in change a Register Transfer Level (RTL), designed in a hardware description language, into generic cells, i.e. independent mapping, and/or cells from a specific technology library, i.e. technology mapping.

- **Independent Mapping:** Performs optimizations as speculation, carry-save arithmetic, resource sharing, arithmetic optimizations, sharing of sub-expressions, logic minimization and remove of unnecessary logic.
- **Technology Mapping:** Involves the mapping of logic equations to the cells available in the cell library to meet the requirements specified in the constraints file, aiming to get the better utilization in terms of area and/or power.

- **Physical synthesis**: Run thought a mapped netlist, from logical synthesis, and comprises all the steps up to the final layout, which are: floorplaning, power planning, placement, clock tree synthesis and routing.
- **Sign-off:** Ensure that the project achieved all specified constraints, in extremes circumstances. At this stage, stringent checks are carried out through the project ensuring minimum conditions to the circuit to be manufactured with an acceptable margin of confidence.

2.1.2 Library-Free Based Design

Unlike library based design flow, where the whole library was previously ready for use; in a network of transistor based, also called library-free and virtual library methodology, each logical element from design is described as a network of transistors, allowing unrestricted logical combinations, as continuous drive capability.

As advantage, the use of a lower abstraction level design allows higher flexibility in logical synthesis phase, making possible fine-tuned optimizations. Even an extensive cell library has the disadvantage of being discrete, which means that the number of design options is limited. When targeting performance or power, customized cells are attractive [RABAEY, 2002]. Furthermore, with the increased impact of interconnect load on novel process, to provide cells with adjusted driver sizes is essential from both, performance and power perspective [SYLVESTER, 1998].

As drawback, the use of transistor makes the design process costliest. Once the flow run in a transistor level, all elements must be characterized and its layout must be generated along the design, increasing the development time.

2.1.2.1 Design Flow

Network of transistor flow, illustrated in Figure 5, is an alternative to standard cell flow, and mostly adopted for fine-tune optimizations. Instead of using a couple of pre-designed cells, all the logical functions can be generated along the design as required.

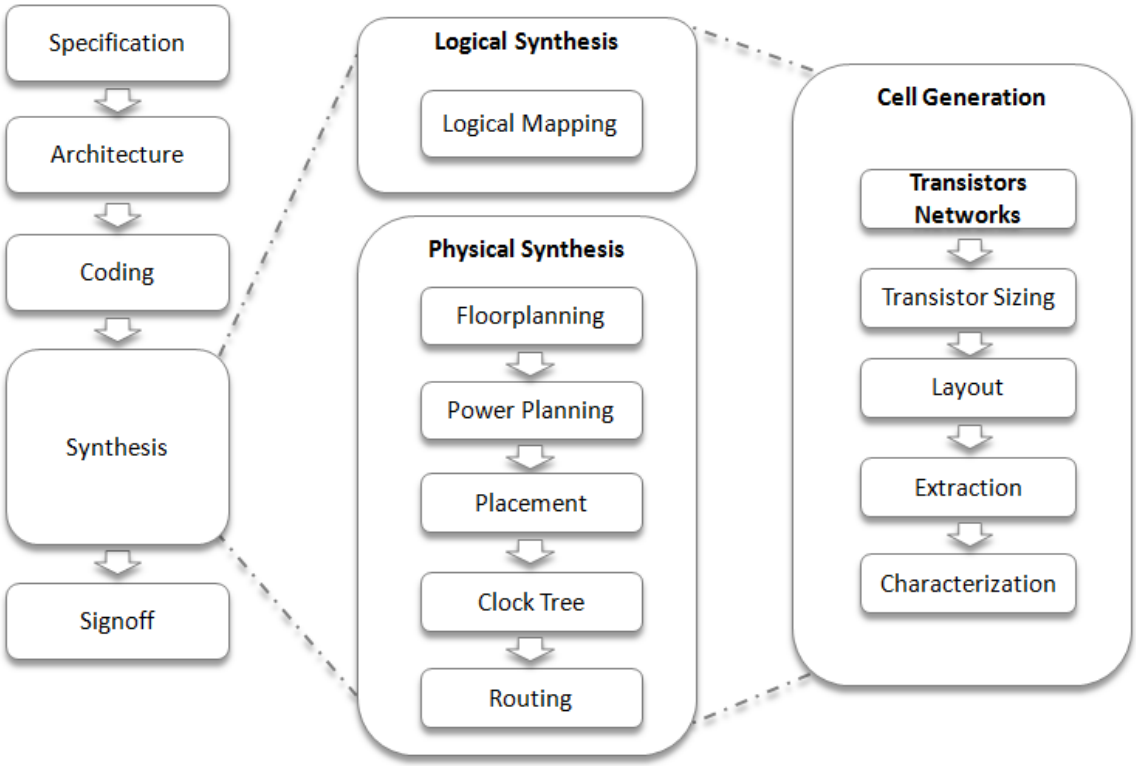

Figure 5: Network of transistors based design flow.

On this approach, the flow phases are very close to the traditional standard cell flow, presented in Figure 4. The main difference is regarding to the technological mapping process, now called logical mapping, and the creation of new phases required for implement the customized cell, i.e. cell generation phase. Instead of mapping the design in a restrictive set of technological functions, a new mapping tool is required to represent the system behavior into unrestricted logical functions. Regarding to the phases required to implement the customized cell, the process can be divided into four steps: transistor sizing, layout, extraction and characterization.

- **Transistor sizing:** as mapping process, transistor sizing is one of the most important phases on circuit design. Low sizing devices may minimize area and internal power; however when connected to a high fanout, have it load capability minimized, increasing delay. On the other hand, high sizing devices may increase

performance, however, also increase power consumption and area. To identify the design requirements, and evaluate the better sizing for the networks of transistors is a huge problem, and to understand the effects of the transistor sizing on power and delay are mandatory to design fine-tuned systems. As a relevant component, transistor sizing research is finding in [SAPATNEKAR, 1993] [POSSER, 2011].

- **Layout:** Process of physically draw the rectangles that will express the features of the process, and will be used to manufacture the logical function. In general, this step is fully handcrafted, using EDA tools to validate the design. However, it can be automated with the use of layout or cell generators [HILL, 1985] [BRATTACHARYA, 2002] [ZIESEMER, 2007].

As a local reference, we describe the academic spice-to-layout tool ASTRAN, developed in [ZIESEMER, 2007]. The tool automatically generates layouts, as illustrated in Figure 6, on linear array and is capable of supporting cells with different networks of transistors, observing the transistor sizing specified by the designer.

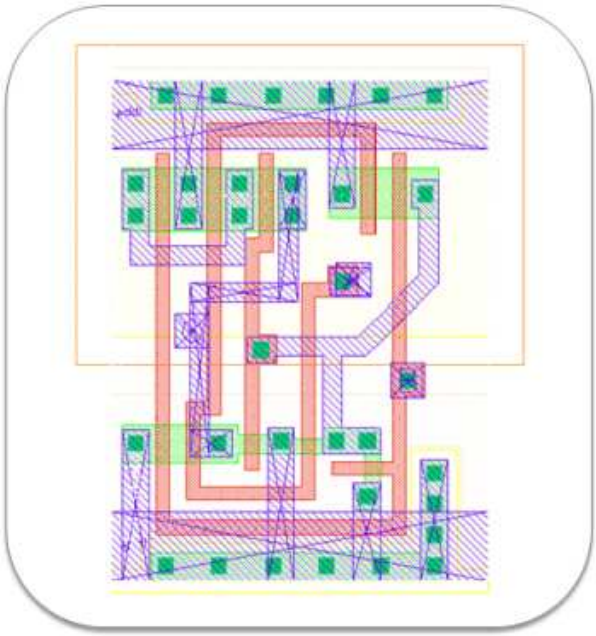

Figure 6: Example of automated generated layout using ASTRAN tool.

For the layout generation, ASTRAN takes as input three files: a spice-like netlist containing the description of each network; a configuration file, which defines the topology of the layout and several parameters for the generator; and the technology file, which contains the design rules. In the netlist, a list of networks of

transistors is defined, stating the length and width for each component, and their input and output pins. The generator receives this file as input and generates the layout of each network of transistor according to the technology rules specified.

- **Extraction:** Evaluate the circuit searching for parasites components that normally change the system behavior. Once the layout is ready, extraction can be done in search for components as capacitors, resistors, diodes and BJT transistor. As output, the extractor creates a new netlist containing the parasites, used in characterization.
- **Characterization:** Aims to determine the behavior of the circuits when applied a set of parameters over there, some common parameters are: Input slope, Output load and PVT (Process, Voltage and Temperature). According to the combination of those parameters, i.e. running several transistor-level simulations, delay, transition time and power characteristics may be acquired, creating high-level models used on logical and physical synthesis.

2.2 POWER DISSIPATION MECHANISMS

By construction, static CMOS technology use to be a very power-efficient device because dissipates almost zero power while in steady-state mode. In order to achieve higher performance and integration, in each new technology generation, devices have scaled down. Nevertheless, as transistor count and clock frequencies increased, power dissipation also increased. Actually, minimize power is a big challenge and the understanding of its components is essential to reach this goal. Power dissipation in static CMOS circuits is mainly defined by two mechanisms, dynamic and static power, as described below:

$$P_{total} = P_{dynamic} + P_{static}$$

2.2.1 Dynamic Power

Dynamic power is characterized by the sum of two factors, switching power plus short-circuits power:

$$P_{dynamic} = P_{switching} + P_{short-circuit}$$

2.2.1.1 Switching Power

Primary switching power is dissipated when charging or discharging internal and/or external nodes and net capacitances and is modeled as follow:

$$P_{switching} = \alpha \cdot f \cdot C_{eff} \cdot V_{dd}^2$$

Where α is the switching activity; f is the switching frequency; C_{eff} is the effective capacitance, which means: $C_{net} + C_{pin} + C_{internal}$ (Figure 7) and V_{dd} is the nominal supply voltage.

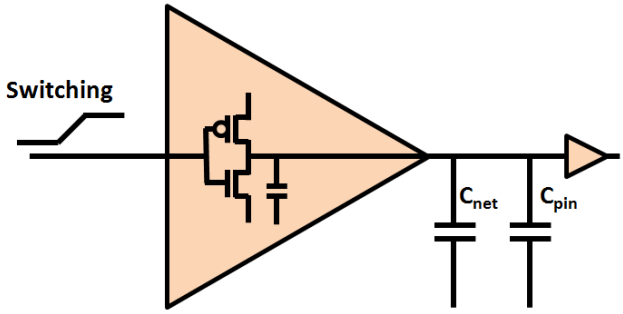

Figure 7: Dynamic switching power in static CMOS design.

2.2.1.2 Short-circuit Current

For all designs, a load capacitance and resistance is related to the path. This fact turn the assumption of zero rise and fall input transition an untruth. For each switching, a simultaneous conduction in both PMOS and NMOS transistor will occur, leading to a direct current between power and ground supply.

Short-circuit power is defined as the power dissipated by an instantaneous short circuit connection between the supply voltage and the ground at the time the gate switches. This component is defined by the following expression:

$$P_{short-circuit} = I_{short} \cdot f \cdot V_{dd}$$

Where I_{short} is the short-circuit current during switching (Figure 8), V_{dd} is the supply voltage and f is the switching frequency.

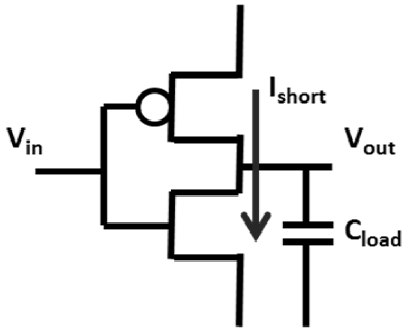

Figure 8: Short-circuit current in static CMOS design.

2.2.2 Static Power

Ideally, the static current of CMOS devices is equal to zero, as the PMOS and NMOS devices are never 'ON' simultaneously in steady state operation [RABAEY, 2002]. However, it cannot be a correct affirmation because uncontrollable currents flow throws the transistor even in steady-state operation, dissipating power even in the absent of switching.

Static power dissipation is defined by the following expression:

$$P_{static} = I_{static} \cdot V_{dd}$$

Where I_{static} is the current flowing in steady state and V_{dd} is the supply voltage. On the other hand, static current is a function mainly of three components: switching voltage threshold V_{th}, transistor sizing and transistor count. Figure 9 illustrates the main sources of static current [KEATING, 2007]; follow is presented a short description of each component.

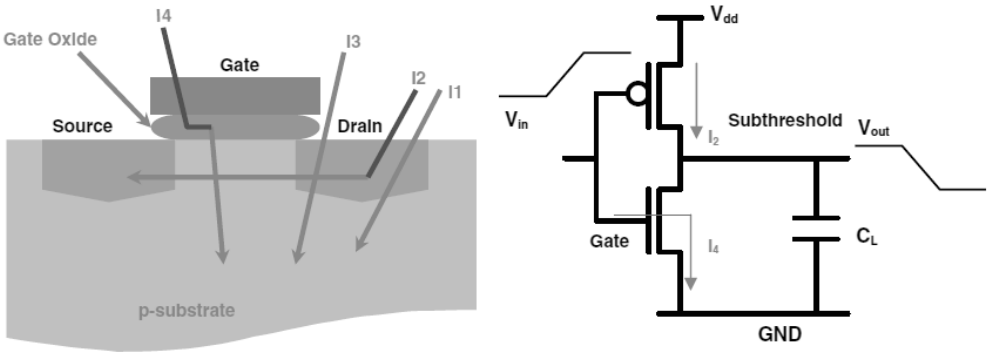

Figure 9: Static power sources in Static CMOS design. [KEATING, 2007]

- I1: Reverse Bias Junction Leakage
- I2: Sub-threshold current
- I3: Gate-induced drain leakage
- I4: Gate oxide leakage

2.2.2.1 Reverse Bias Junction Leakage

Parasitic diodes formed between the diffusion region of the transistor and substrate consumes power in the form of reverse bias current which is drawn from the power supply. Diode reverse bias current, results from minority carrier diffusion and drift near the edge of depletion regions, and also from generation of electron hole pairs in the depletion regions of reverse-bias junctions. When both "N" and "P" regions are heavily doped, as is the case for some advanced MOSFETs, there will also be junction leakage due to band-to-band

tunneling (BTBT), i.e., electron tunneling from valence band of the p-side to the conduction band of the n-side. In an inverter when the input is high, the NMOS transistor is ON and output voltage is discharged to ground. Now between drain and the n-well a reverse potential difference of VDD is established which causes diode leakage through the drain junction. The n-well region of the PMOS is also reverse biased. This also leads to leakage current at the N-well junction [PEDRAN, 2007].

2.2.2.2 Sub-threshold current

As devices become smaller, also power supply is scaled in order to keep dynamic power under control. However, to keep performance, voltage threshold has to be scaled down. Sub-threshold current flows from source to drain even if the gate to source voltage is lesser than the threshold voltage of the device. This happens due to the carrier diffusion between the source and drain regions of the CMOS transistor in weak inversion. When gate to source voltage is smaller than but very close to threshold voltage of the device then sub-threshold current becomes significant.

Sub-threshold current I_{sub}, which occurs when gate voltage is below threshold voltage V_{th}, is a main part of leakage current [PEDRAN, 2007]. I_{sub} depends on different effects and voltages. From BSIM transistor model [SHEU, 1987], the sub-threshold leakage current can be expressed as:

$$I_{sub} = I_0 \cdot e^{\frac{V_{gs}-V_{th}}{nV_t}} \cdot \left[1 - e^{-\frac{V_{ds}}{V_T}}\right]$$

$$I_0 = \frac{W \cdot \mu_0 \cdot C_{ox} \cdot V_T^2 \cdot e^{1.8}}{L}$$

$$V_T = \frac{KT}{q}$$

Where V_T is the thermal voltage; V_{th} is the threshold voltage; V_{ds} and V_{gs} are, respectively, the drain-to-source and gate-to-source voltage; W and L are the transistor width and length, respectively; C_{ox} is the gate oxide capacitance; μ_0 is the carrier mobility and n is the sub-threshold swing coefficient.

2.2.2.3 Gate-induced drain leakage

Gate-induced drain leakage (GIDL) is caused by high field effect in the drain junction of MOS transistors. In an NMOS transistor, when the gate is biased to form accumulation layer in the silicon surface under the gate, the silicon surface has almost the same potential as the p-type substrate, and the surface acts like a p region more heavily doped than the substrate. When the gate is at zero or negative voltage and the drain are at the supply voltage level, there can be a dramatic increase of effects like avalanche multiplication and band-to-band tunneling. Minority carriers underneath the gate are swept to the substrate, completing the GIDL path. Higher supply voltage and thinner oxide increase GIDL [PEDRAN, 2007].

2.2.2.4 Gate oxide leakage

When there is a high electric field across a thin gate oxide layer, tunneling of electrons over the gate oxide can result in leakage. Electrons may tunnel into the conduction band of the oxide layer; this is called Fowler-Nordheim tunneling. There can also be direct tunneling through the silicon oxide layer if it is less than 3–4 nm thick. Mechanisms for direct tunneling include electron tunneling in the conduction band (ECB), electron tunneling in the valence band (EVB) and hole tunneling in the valence band (HVB). The dominant source of leakage here is the direct tunneling of electrons through gate oxide. This current depends exponentially on the oxide thickness and the VDD [PEDRAN, 2007].

Of the following leakage components, sub-threshold leakage is dominant [LIU, 2007]. While dynamic power is dissipated only when switching, static power due to leakage current is continuous, and must be dealt with by using design techniques.

2.3 DISSIPATION REDUCTION TECHNIQUES

This chapter aims to present some common power management techniques for reducing power. In order to simplify the reading, the techniques were divided according to the goals, as follow: Only dynamic power reduction, only static power and both dynamic and static power reduction.

2.3.1 Only Dynamic Reduction

Only dynamic reduction techniques are mainly applied to minimize transition time and capacitive factor of the nets/pins. As those values are direct related to short-circuit and switching power, its improvement imply directly in dynamic power.

2.3.1.1 Clock Gating

Clock gating is a common technique used on synchronous circuits to reduce dynamic power dissipation. The main goal of this technique is to insert extra logic to avoid unnecessary switching on clock tree. On this kind of circuits, clock trees are the largest source of dynamic power moved for two factors, clock tree signal present the larger capacitive load and, normally, switch at the maximum rate. When inserted a logic structure on the clock tree, Figure 10, portions of the tree can be disabled, avoiding unnecessary switching in the gated branches, and decreasing the capacitive load in the non-gated tree.

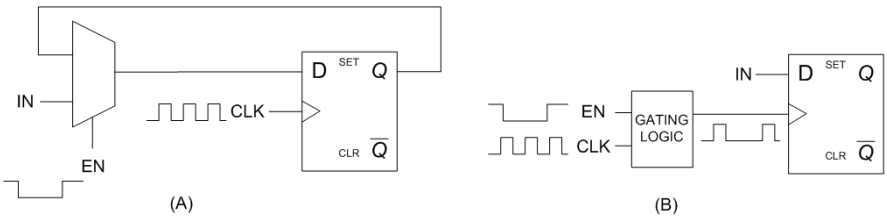

Figure 10: Register with clock gating (B) and without clock gating (A).

2.3.1.2 Operand Isolation

In a normal operation system, datapath elements are sampled only periodically. For instance, a controlled counter operation is required a few times over hundreds of clock transitions, Figure 11-A. Which means, even when the operation is not required, the inputs of the operator will be switching, i.e. charging and discarding the load capacitance, resulting in power dissipation.

Operand isolation aims to break the switching of the operator inputs, Figure 11-B. This action leads to aggressive reduction in the effective capacitance, as result, the datapath connected to operator, with higher capacitance, will not switch, saving dynamic power.

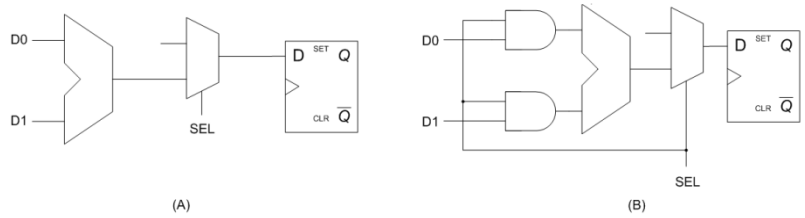

Figure 11: Example of datapath, in (A) a direct path to operator, in (B) an isolated operator.

2.3.1.3 Logic Restructuring

When working with dynamic power, an important component is the switching. As much a net or cell pin switch, more tends to be accumulated dynamic power. The basic idea behind logic restructuring is to move high frequency switching operations up in the logic cone, and low frequency switching operations back in the logic cone. This change in design minimize output frequency switching, i.e. requirement of charge and discharge load capacitance, without modifying the output logic; e.g., reduce two stages, Figure 12-A, to three stages through logic equivalence transformation, Figure 12-B, so the output has lower switching activity.

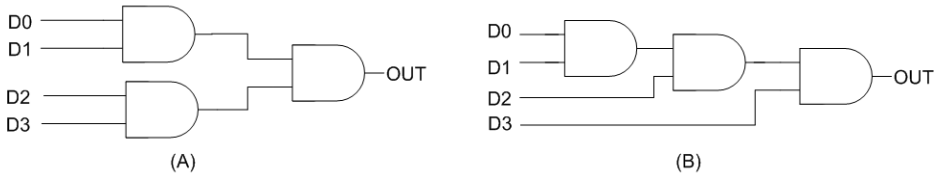

Figure 12: Example of datapath, in (A) a two stages logic structure, in (B) the logic restructured to reduce switching activity.

2.3.1.4 Transition Rate Buffering

Extended transition time is the main aggravator of short-circuit current. When transition is too long, due high capacitance, power dissipated becomes greater. In order to minimize transition time, and leads to lower dynamic power, buffer manipulation is a practical approach, illustrated in Figure 13.

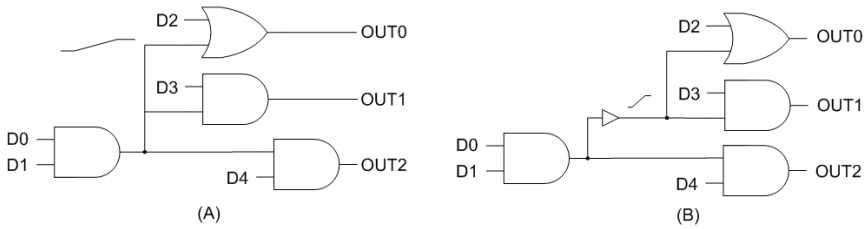

Figure 13: Buffering of datapath to reduce transition time, in (A) no buffering, high transition time, in (B) buffered logic, low transition time.

2.3.1.5 Pin swapping

The main goal of swapping gate pins is to ensure that fast switching occurs at gates/pins with lower capacitive loads. The pins are swapped so most frequently switching occurs at the pins with lower capacitive load. For instance, since the capacitive load of pin D1 is lower, lower will be the power dissipation when switching, Figure 14.

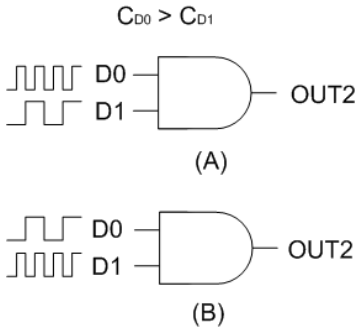

Figure 14: Pin Swapping, in (A) High frequency in most capacitive pin, in (B) High frequency in less capacitive pin.

2.3.2 Only Static Reduction

Only static reduction techniques are applied to reduce mainly the subthreshold current. As subthreshold is a mandatory component of static power, its improvement imply direct in the reduction of the static power.

2.3.2.1 Multi-V_{th}

Most library vendors provide libraries that have cells with different switching thresholds. This is due to the different features of the transistors, i.e. in a same library there is gates using low thresholds transistors (faster with higher leakage) or high thresholds transistors (slower with lower leakage).

Multi-V_{th} optimization takes into count the gates with different thresholds, as illustrated in Figure 15, to optimize the design for power, timing, and area constraints. Good synthesis tools for low-power applications are able to mix available multi-threshold library cells to meet timing and area constraints with the lowest power dissipation.

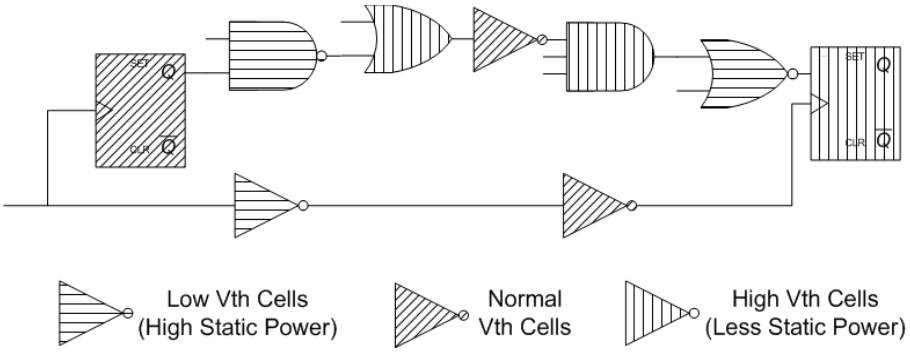

Figure 15: Illustration of design using Multi-V_{th} cells.

2.3.2.2 Substrate biasing

With this technique, the substrate or the appropriate well is biased to modify the thresholds voltage of the transistor, since leakage currents are a function of device V_{th}, it leads to reduced leakage. In PMOS, the body of transistor is biased to a voltage higher than power voltage. In NMOS, the body of transistor is biased to a voltage lower than ground voltage. Since raising V_{th} also effects performance, some advanced techniques allows the bias to be applied dynamically, so during an active mode of operation the reverse bias is small, while in standby the reverse bias is stronger.

Substrate bias returns are diminishing at smaller processes in advanced technologies. At 65nm and below, the body-bias effect decreases, reducing the leakage control benefits. TSMC *(Taiwan Semiconductor Manufacturing Company)* has published information pointing to a factor of 4x reduction at 90nm, and only 2x moving to 65nm. Consequently, substrate biasing is predicted to be overshadowed by power shut-off.

2.3.2.3 Power shut-off

Power shut-off, also called power gating, is one of the most effective techniques to minimize static power. Power gating uses low-leakage PMOS transistors as header switches to turn-off power supplies to parts of a design. NMOS footer switches can also be used as sleep transistors [SHI, 2007].

Inserting sleep transistors splits the chip's power network into a permanent power network connected to the power supply and a virtual power network that drives the cells and can be turned off, as can be seen in Figure 16 [SHI, 2007]. According [SHI, 2007] this technique is increasingly being used in the industry and can eliminate up to 96 percent of the leakage current.

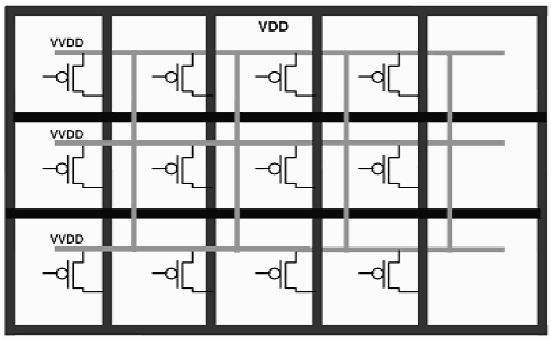

Figure 16: Grid implementation of sleep transistors.

2.3.2.1 Memory splitting

Normally, systems adopt memory size looking for the peak usage, which means, the maximum memory required for an application. However, during normal system operation, only a portion of that memory is used at any given time.

In many cases, it is possible to divide the memory into several sections, and selectively power down unused parts of the memory. Once, those parts are basically an amount of transistor, cutting down the power leads to static power minimization. With increasing SOC memory capacity, reducing the power consumed by memories is gradually more important.

2.3.3 Dynamic and Static Power Reduction

Dynamic and static reduction techniques take common characteristics between the two types of consumption, to minimize both dynamic and static dissipation. Supply voltage and cell/transistor size are examples of features that affect both, thus, managing at least one of those features can lead to improvements in power dissipation.

2.3.3.1 Multi-Supply Voltage

Running at a lower voltage reduces power consumption, at the expense of speed. In some applications, different supply voltages for different parts of the chip can be applied based on their performance requirements. The goal of this technique is to operate different blocks at different voltages. Lowering voltage has a squared effect on switching power consumption, and a linear effect in static power consumption. Multi-supply Voltage is an essential technique for low-power designs. Instead of the benefits, MSV technique require attention to be implemented, once need the use of level shifters on signals that go from one voltage level to another.

2.3.3.2 Dynamic Voltage and Frequency Scaling

Dynamic voltage and Frequency Scaling techniques provide ways to reduce power consumption of chips scaling down the voltage supply, and the frequency on-the-fly, based on the targeted performance requirements of the application. Since DVFS optimizes both, frequency and voltage, this technique is very effective on both dynamic and static power reduction.

2.3.3.3 Dynamic voltage scaling

Dynamic voltage scaling is a subset of Dynamic Voltage and Frequency Scaling that dynamically scales down the voltage based on the performance requirements.

2.3.3.4 Adaptive Voltage and Frequency Scaling

Adaptive voltage and frequency scaling is an extension of Dynamic Voltage and Frequency Scaling. In DVFS, the voltage levels of the targeted power domains are scaled in fixed discrete voltage steps. Frequency-based voltage tables typically determine the voltage levels. It implies large margins in the steps, and therefore the power reduction is not optimal. On the other hand, AVFS deploys closed-loop voltage scaling and is compensated for variations in temperature, process, and IR drop using dedicated circuitry (typically analog in nature) that constantly monitors performance and provides active feedback. Although the control is more complex, the payoff in terms of power reduction is higher.

2.3.3.5 Logic resizing (Transistor Resizing)

Logic resizing is a commonplace technique applied for power reduction. As described on previews chapters, both dynamic and static power are direct related to the sizing of the components. On dynamic power, capacitance and transition times are the key components to control power consumption. Once cell or transistor upsizing improves slew times, minimizing short-circuit current, downsizing reduces load capacitance on previews cell, leading to a lower effective capacitance, hence lower switching power. On the other hand, for static power, as sub-threshold current are directed dependent of the transistor width, minimize it can help to decrease static power. Other opportunity is to regroup equivalent logical cones in order to minimize the overall system transistor count and reduce static power dissipation.

3 RELATED WORKS REVIEW

This section aims to review the main approaches found in literature to handle with implementation and optimization of combinational logical, both on standard cells and library-free methodology. In sequence, we present studies associated to transistor sizing and its use on system design.

3.1 LOGICAL MAPPING REVIEW

3.1.1 Cells Library Based Methodologies

3.1.1.1 Gregory - 1986

The first automated process to map combination logic was introduced in [GREGORY, 1986]. In his work was presented Socrates, a set of programs, illustrated in Figure 17, which synthesize and optimize combinational logic circuits from Boolean equations. Socrates optimizes logic using Boolean and algebraic minimization techniques, and it optimizes circuits derived from this logic in a user-defined technology with a rule-based expert system.

Socrates also presents a circuit optimizer to improve measurable circuit characteristics by iteratively replacing and rearranging small portions of a circuit. The program uses a library which describes alternative circuit implementations in a rule (if antecedent then consequent) form. New rules can be generated automatically from netlist.

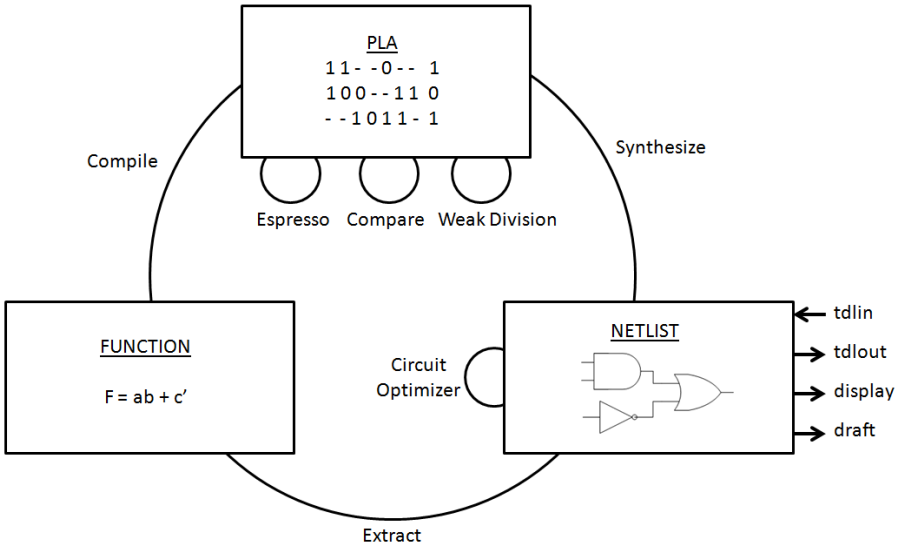

Figure 17: Socrates system diagram. [GREGORY, 1986]

A constraint specification allows designers to describe the desired characteristics of their circuits. Designers can specify when signals arrive at inputs, and the drive factor associated with them. Also it is allowed to specify the maximum propagation delays to individual outputs, and the loads that must be driven at those outputs. Socrates does not accept area constraints; it automatically tries to synthesize the smallest circuit it can under the timing constraints specified.

3.1.1.2 Keutzer – 1988

In [KEUTZER, 1988] was presented the process of mapping a technology independent description of a circuit into a particular technology. In this work, the author found a relation between the technologies mapping problem with those that have been encountered in the field of programming language compilers.

More specifically, verifying that the matching between graph-like descriptions of a technology independent circuit against a library of patterns in a technology, such as a standard cell library, is similar to matching a graph-like intermediate representation of a computer program against the patterns of an instruction set of a given machine.

In this implementation a tree manipulator is used for constructing code generators for programming language compilers, and to build an optimizing technology mapping tool. The result is DAGON, which is capable of optimizing for time, area or a function of both. DAGON rests on a firm algorithmic foundation, and is able to guarantee locally optimal matches against a set of over three thousand patterns.

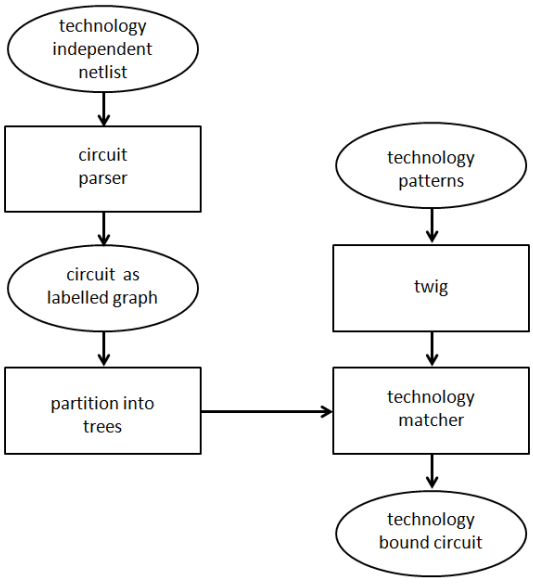

Figure 18: DAGON approach. [KEUTZER, 1987]

DAGON takes a canonical technology-independent description of a combinational circuit and a list of patterns describing both the cells in the technology and local transformations. DAGON creates a technology bound circuit by partitioning the circuit into a forest of trees, then using a tree pattern matching automation to match the individual trees. The tree matcher that is used in DAGON is based on twig, a tree matching generator tool that is generally used for constructing code generators for programming language compilers. In Figure 18 is given an overview of this approach.

3.1.1.3 Crastes – 1991

[CRASTES, 1991] deals with the logic mapping phase and takes as main optimization criterion the depth of the standard cell network, where the depth is defined as the number of standard cells on the longest path of the network. The second optimization criterion is the gate area. Differently from early approaches, [CRASTES, 1991] method does not allow extra factorization (such as a.b + a.c transformed in a.(b+c)), because there is several constraints on the variable ordering induced by the lexicographical factorization. The cells of the considered library are represented using a concise and simplified model, so-called, pattern. The technology mapping phase starts from the initial factorized trees and is based on a pattern matching phase leading to pattern trees. This matching phase is followed by a definitive mapping leading to the standard cell network.

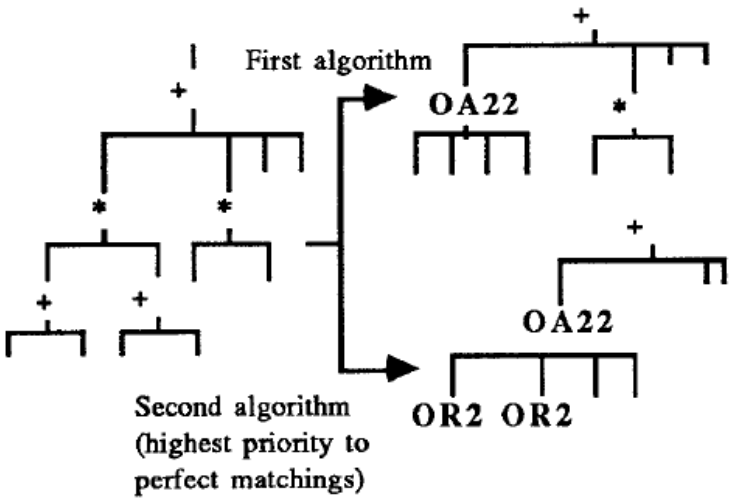

Figure 19: Matching algorithms. [CRASTES, 1991]

The technology mapping consists in transforming the set of pattern trees obtained after the covering phase by a standard cell network. According to the author definitions, two algorithms for the optimal covering of a due acted tree have been implemented. The first one

starts from the root of the directed tree and performs perfect and semi perfect 2-level matching as soon as possible. The second algorithm gives absolute priority to the 2-level perfect matching as they minimize the depth of the tree. In Figure 19 is presented the difference between both algorithms.

3.1.1.4 Tiwari – 1993

[TIWARI, 1993] focuses on the problem of mapping a technology independent circuit to a technology specific one, with power as the optimization metric. This algorithm for low power uses de same idea of technology mapping for area and delay as described early. Specifically, a canonical representation (called the subject DAG) is created for the Boolean functions to be mapped through two-input NAND/NOR gates and inverters structures.

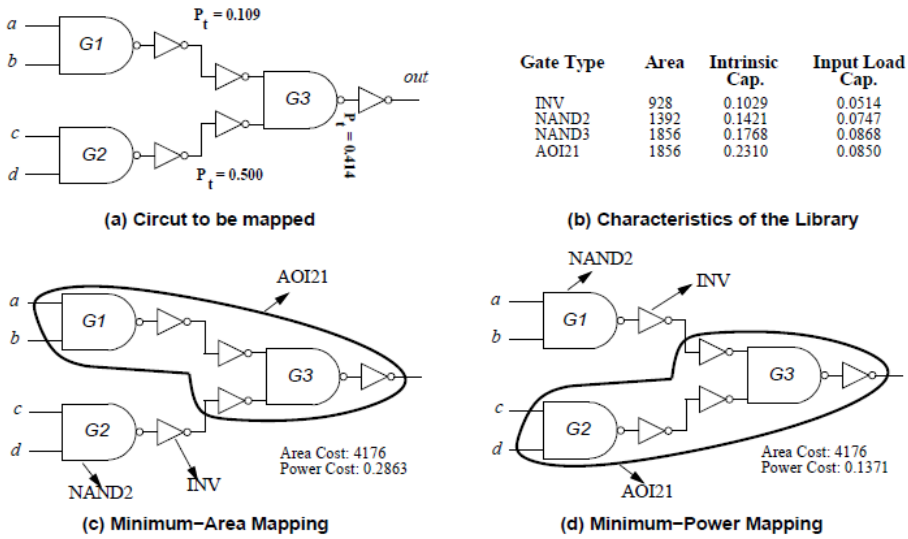

Figure 20: Illustration of the difference between technology mapping for power and area.

Canonical representations (called patterns) are also obtained for each of the gates in the library using the same basic functions. The technology mapping problem is then formulated as one covering the nodes in the subject DAG with patterns, so that cost function modeling power dissipation is minimized under the constraint that the inputs to a match are available as the outputs of other matches.

3.1.1.5 Kukimoto – 1998

[KUKIMOTO, 1998] consider the minimum-delay technology mapping problem for library-based designs where a subject graph is a DAG. Careful analysis shows that the

basic dynamic programming approach is not specific to FPGA mapping and can be easily adapted to library-based mapping. This leads to a linear time algorithm for minimum-delay DAG covering under load-independent delay models. As far as know, this is the first result that shows that the minimum-delay technology mapping problem for DAGs can be solved optimally in polynomial time. This implies that tree decomposition in performance-oriented mapping is not necessary; a given subject DAG can be directly mapped optimally. [KUKIMOTO, 1998] experimentally confirmed that the additional solution space explored by this direct approach finds significantly faster mappings especially under a rich library.

3.1.1.6 Mishchenko – 2005

[MISHCHENKO, 2005] makes three main contributions: The first is a new locally-Boolean matching framework for ASICs which combines exhaustive cut function enumeration with Boolean matching ensuring optimal phase assignment. Second, shows how this framework can naturally accommodate super gates without any change to the core algorithm using library pre-processing. Third, extends the matching algorithm to include choices in the mapping graph, and present a new way of generating choices leveraging ideas from combinational equivalence checking. The objective of this work is to minimize the delay of the longest path in the mapped netlist. [MISHCHENKO, 2005] assumes a load independent delay model for the gates in accordance with a gain based logic synthesis flow. As shown by [KUKIMOTO, 1998], this problem can be optimally solved using dynamic programming for DAG-covering. The key difference of this method with the conventional approaches based on DAG-covering lies in the matching step, where [MISHCHENKO, 2005] used Boolean matching instead of structural one.

3.1.2 Library-Free Based Methodologies

3.1.2.1 Berkelaar – 1988

In [BERKELAAR, 1988] was presented a new approach to technology mapping based on a library-free methodology, in his work was presented the process of making a standard-cell integrated circuit implementation from a previously optimized and decomposed set of Boolean functions. As defined in literature, the problem of mapping Boolean functions onto a random library of standard-cells is known to be NP-complete. So, in [BERKELAAR, 1988] is not solved the problem for random libraries, but rather for those libraries that can be produced by cell generators. Another approach introduced is the use of Boolean expression

denoted by a prefix notation; all expressions are assumed non-redundant and factorized. About the representation, each Boolean expression is associated with a directed graph, as illustrate in Figure 21.

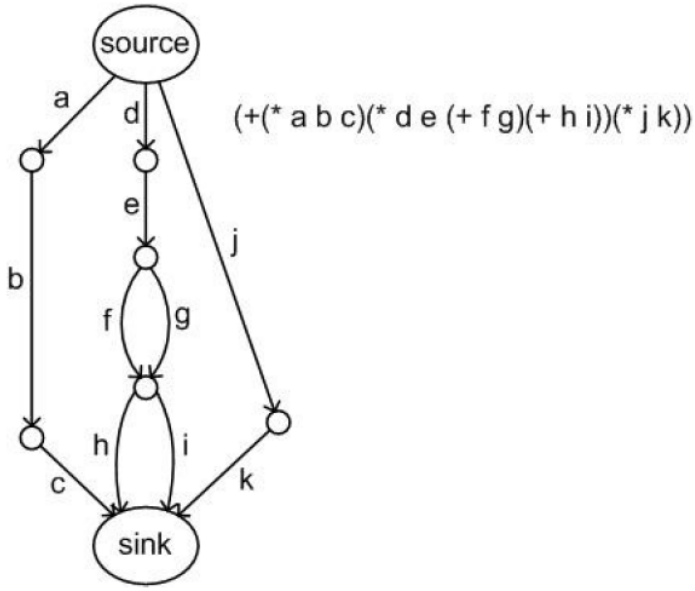

Figure 21: Graph expression representation.

The approach used to split up the functions is partly heuristically and partly analytical. Functions are taken one at a time and the algorithm does not look across their boundaries. The mapping process will be top-down, it will start the mapping operation from the source node and at each moment the process is limited in a certain expression to 1 level and solved the problem of mapping this expression on a tree of standard-cells completely optimal, in an analytical way. According to the author the result of his algorithm will generate an optimal Boolean network.

3.1.2.2 Jiang – 2001

Two new techniques for mapping circuits are proposed in [JIANG, 2001]. The first method, called the odd-level transistor replacement (OTR) method, has a goal that is similar to that of technology mapping, but without the restriction of a fixed library size, and maps a circuit to a virtual library of complex static CMOS gates. The second technique, the Static CMOS/PTL method, uses a mix of static CMOS and pass transistor logic (PTL) to realize the circuit, and utilizes the relation between PTL and binary decision diagrams. The main goal was to a dynamic programming framework for technology mapping for a library-less environment and for PTL. [JIANG, 2001] works on library-less mapping focuses on the

formation of gates, a procedure they called gate collapsing, which collapses smaller gates in a decomposed circuit into more complex gates.

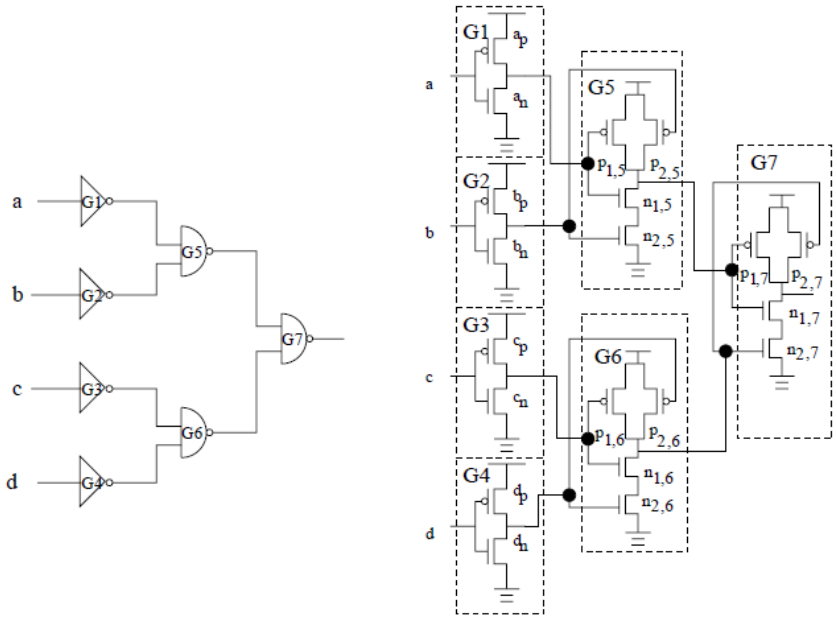

Figure 22: An example of gate collapsing.

For PTL mapping, his approach uses dynamic programming techniques to partition the circuit into PTL segments separated by static CMOS gates. The basic unit in a PTL circuit is a multiplexor, and there is a close relationship between the BDD representation of a circuit and its PTL implementation. [JIANG, 2001] method dynamically builds BDD's of logic functions and finds an optimal mapping, under the constraint that the number of PTL transistors in series must be constrained never to exceed a user-specified number.

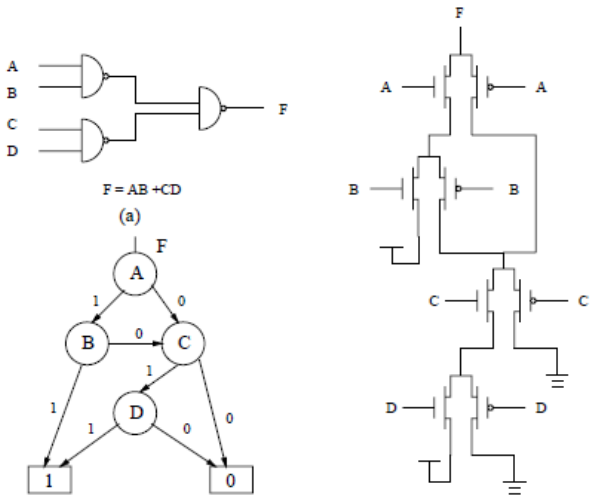

Figure 23: Transformation of a circuit from BDD representation to PTL.

3.1.2.3 Correia – 2004

[CORREIA, 2004] explore several AND/OR circuit decompositions by using a n-ary tree representation of the circuit taking into account criteria that optimize the logic depth of the nodes before gathering them into complex gates. The mapping procedure acts over tree representations of Boolean expressions, which can be achieved by partitioning heuristics. In his implementation, each combinational portion of a given circuit is initially represented as a DAG. Later, each multiple fanout node is taken as the root of a partitioned sub-tree. By applying this transformation, it is possible to get a disjoint forest of mappable trees.

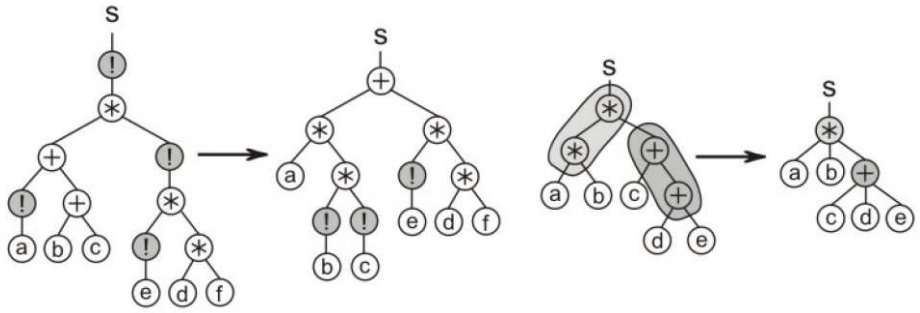

Figure 24: Transformation of a circuit into n-ary tree.

The subject tree is mandatorily modified by the following rules. First, each and every inverter node is propagated to the input variables, in the leaf boundary of the tree. Afterward, every connected pair of nodes with the same operator label is gathered into a single node, as illustrated in Figure 24.

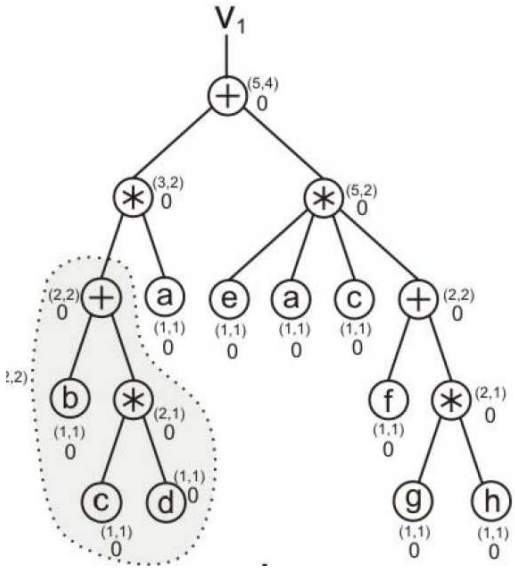

Figure 25: N-ary tree with the initial costs.

After initial transformations, each node has associated (s,p) costs and a logic depth (l), Figure 25. All logic depths are calculated starting from the primary inputs of the entire Boolean network. If a node is found having the (s,p) costs in the maximum value allowed, it is marked as an ideal cut, illustrated in Figure 26. The sub-tree rooted at this node is then cut and directly associated to a CMOS complex gate.

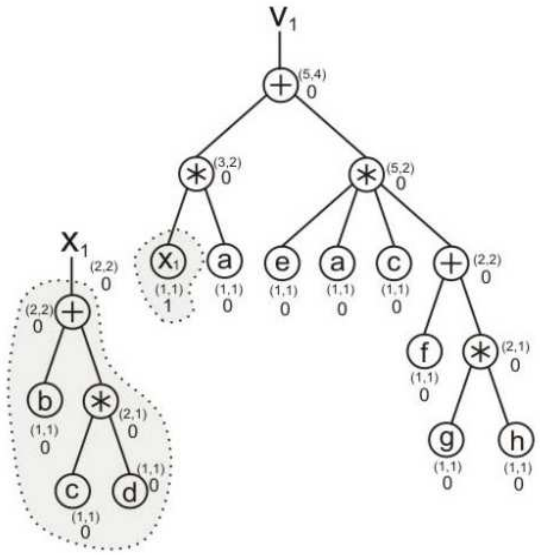

Figure 26: N-ary tree with a cut.

According [BERKELAAR, 1988], the advantage of this method is that it considers dynamically several decompositions of the subject description at low or no cost and covers it optimally in a dynamic programming manner, following a bottom-up strategy.

3.1.2.4 Marques – 2008

The approach presented in [MARQUES, 2008] introduces two library-free technology mapping tools: VIRMA-WF and VIRMA-K. His technology mapping tool puts together two concepts: library-free methodology and the use of CMOS gates with minimum transistor stack. In both is defined as object function the delay of the circuit. Once library-free methodology does not present the characteristics of the cells in advance, [MARQUES, 2008] used topological metrics to estimate delay costs.

To analyze delay, the number of serial transistors in the longest pull-up/pull-down path was used as metric. In his topological model, each cell has its own cost for the pull-up plane and pull-down plane. They represent the cost from the longest path from a primary input to the output net of the cell and the object function is calculated by the sum of all the serial transistors in the pull-up (pull-down) plane along the path.

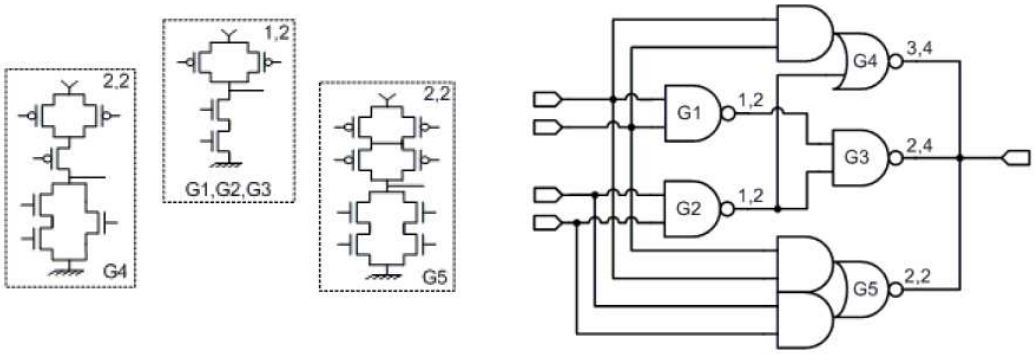

Figure 27: Cost function presented in [MARQUES, 2008].

In his works the circuit is represented by a DAG containing only nodes with two outgoing edges, and to provide more freedom for the matching algorithm is assumed that the whole initial circuit is decomposed in 2-Input AND/OR gates.

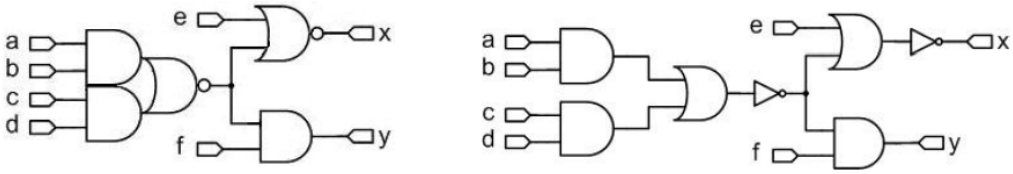

Figure 28: Original and decomposed circuits.

3.1.3 Summary

Table 2: Features of each author and its methods.

Author	Year	Logical Structure	Matching Type	Covering Type	Library-Free Method
Gregory	1986	Boolean expression	Boolean or algebraic	Bottom-up	No
Keutzer	1987	Binary tree	Structural	Bottom-up	No
Berkelaar	1988	Boolean expression	Topological	Top-down	Yes
Crastes	1991	Tree	Topological	Bottom-up	No
Tiwari	1993	DAG	Topological	Bottom-up	No
Kukimoto	1998	DAG	Structural	Bottom-up	No
Jiang	2001	Electrical diagram	Topological	Bottom-up	Yes
Correia	2004	N-ary tree	Topological	Bottom-up	Yes
Mishchenko	2005	And-Inverter graph	Structural or boolean	Bottom-up	No
Marques	2008	N-ary tree	Topological	Bottom-up	Yes

3.2 TRANSISTOR SIZING REVIEW

3.2.1 Fishburn– 1985

In this work was presented TILOS (Timed Logic Synthesizer), a program that adjust transistor sizes and connectivity within logical gates to meet user's requirements for area and/or time. It was one of the first algorithm for transistor sizing using Elmore delay model (ELMORE, 1948) and couples synchronous analysis with convex optimization techniques in order to optimize the minimum clock period at which circuit operate and/or the sum of transistor sizes. The author present that the second factor is positively correlated with a number of attributes, including silicon area, capacitance-discharge power, short-circuit power, and probability of a device failure within a chip.

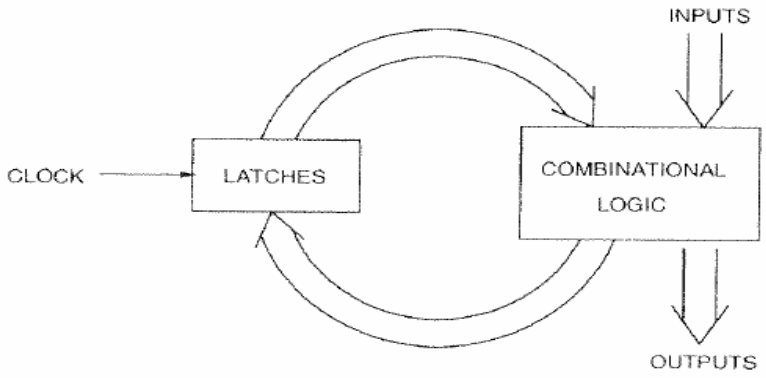

Figure 29: A path begins at the output of a lath or an input,
and ends at the input of a lath or an output.

TILOS proceeds as follows: Starting with minimum sizes for all transistors and a static timing analysis is performed on the circuit, recognizing even latches, Figure 29. From each path output that fails to meet timing goals, TILOS walks backward along the failing path. For each critical path uses heuristics to reduce delay along the path, calculating the sensitivity of each transistor. The size of the transistor with the largest sensitivity is increased, and the process is repeated interactively. Sizing process terminates when either the constraints are met, or when the circuit has passed its absolute minimum and is getting slower instead of faster.

3.2.2 Borah – 1996

A direct approach to transistor sizing for minimizing the power consumption of a CMOS circuit under a delay constraint is presented. The author presents that static power consumption is a convex function related to the active area of the transistor. Analytical

formulation for the power dissipation of a circuit in terms of the transistor size is derived which includes both the capacitive and the short circuit power dissipation. Also the relationship between short circuit power dissipation and transition time is described along the paper, Figure 30.

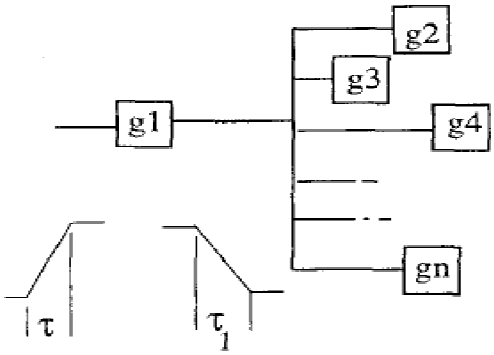

Figure 30: Fanout representation

Circuit simulation results are presented to confirm the validity of the formulation. Analytical methods for transistor sizing to minimize the power consumption of a CMOS circuit subject to a given delay constraint is presented with experimental results to show its applicability to actual circuits.

3.2.3 Santos – 2003

In this work is proposed a method for performing delay optimization of combinational blocks that are generated by automatic layout generator, as illustrated on Figure 31.

Given a maximum delay imposed to the circuit, the method begins by identifying a set of critical paths that violate such delay. This is performed by a timing analysis tool that is able to guarantee that the longest sensitizable paths belong to this set. Then, a gate belonging to these paths is selected to be sized. Transistors of the selected gate are sized in a discrete manner, by using the transistor folding technique. The area and delay of the sized circuit is estimated and, if necessary, a new gate is selected to be sized. This procedure is repeated until the delay constraint is satisfied or there is no gate to be resized.

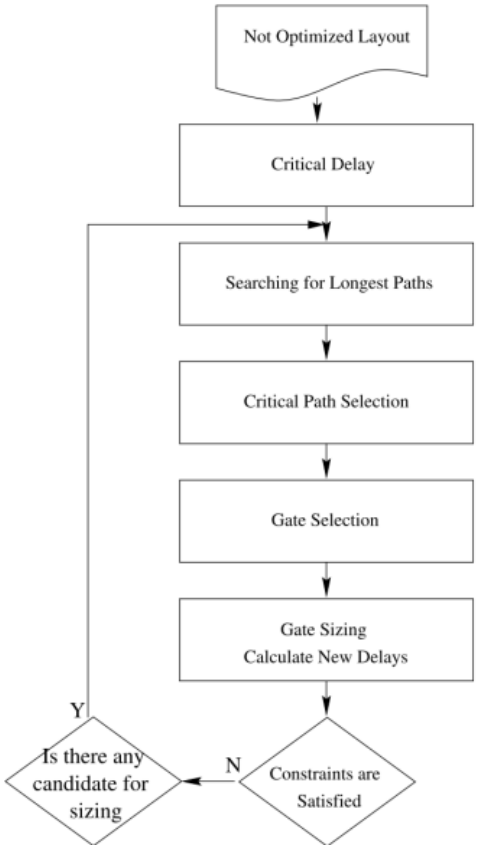

Figure 31: Sizing method presented on [SANTOS, 03]

3.2.4 Beece – 2010

This work presents a novel transistor sizing approach which takes yield considerations into account. It describe an extension to EinsTuner, an existing state-of-the-art tuner, and uses statistical timing to model the behavior of a circuit in terms of probability distributions, thereby enabling the device sizing problem to depend upon the statistical correlations exhibited by process variability.

The resulted tuner make the tool's capability to exploit the statistical ability of EinsTLT and EinsTimer to implement a statistical objective function which optimizes parametric yield for a given slack, or optimizes chip slack for a given yield, Figure 32.

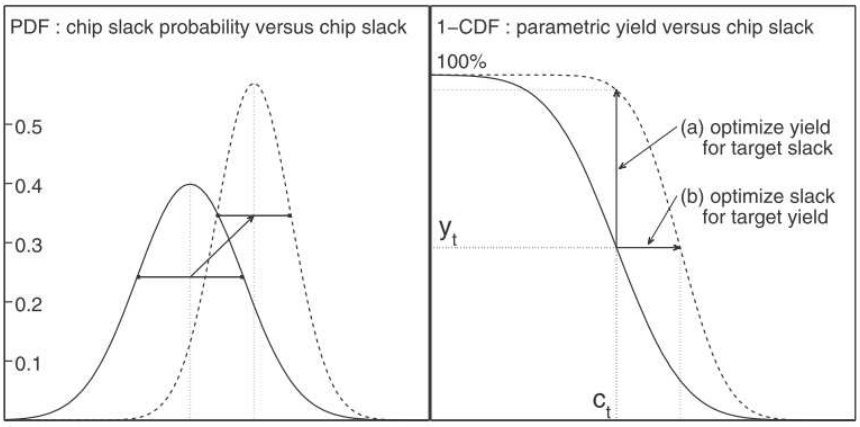

Figure 32: Chip slack (left) and yield (right).

Along this work are presented some experiments showing that for circuits which are dependent upon process variation. The statistical tuner can find better solutions than deterministic tuning, making it an attractive alternative to other sizing methods, such a Monte Carlo.

4 PRELIMINARY STUDIES ON NETWORKS OF TRANSISTORS

Unlike traditional standard cell flow, the employment of lower abstraction level to design integrated circuits enables an advanced freedom in terms of optimization, allowing each component to be tuned as required [KAGARIS, 2007]. Consequently, to understand the benefits and drawbacks on designing networks of transistors are essential to find conditions to work with library-free methodology and obtain maximum advantages of this methodology.

When designing networks of transistors, mainly two factors allow the advanced freedom in terms of optimization: First the increased capability of topological optimizations, i.e., once networks of transistors accept any topological connection, several logical functions can be easily implemented using the lower number of components. Second the freedom of sizing the components as required. Both, minimize the number of components, as build well sized networks could contribute to optimize factors as power, area and / or timing and will be deeply evaluated in this chapter.

4.1 NETWORKS OF TRANSISTORS VERSUS STANDARD CELL

As presented on previous sections the constrained nature of standard cell libraries reduces the opportunity for fine-tuning the design [RABAEY, 2002]. Traditional cell libraries do not contain more than 150 different functions. However, the possible space for logical functions that can be logically mapped with a specific number of stacked transistors is much larger, as presented in Table 1 [DETJENS, 1987].

As case study, we compared sets of logical function implemented using an assembled couple of cells available in commercial libraries, and the same functions implemented as networks of transistors, as shown in Figure 33.

To start we selected a set of functions from two to nine inputs, and one output. All these logical functions were described using spice language, in two different abstraction levels, gate and transistor level. For gate level, each logical functions was described though the use of primitive functions (nand2, nand3, nor2, nor3 and inverter), and on transistor level, same logical function was described using pure transistors structures. All these cells were characterized using commercial cell characterizer, and results were collected as lookup tables. For each logical function we obtained up to twenty values of dynamic power and delay, one for each combination of input slope and load capacitance.

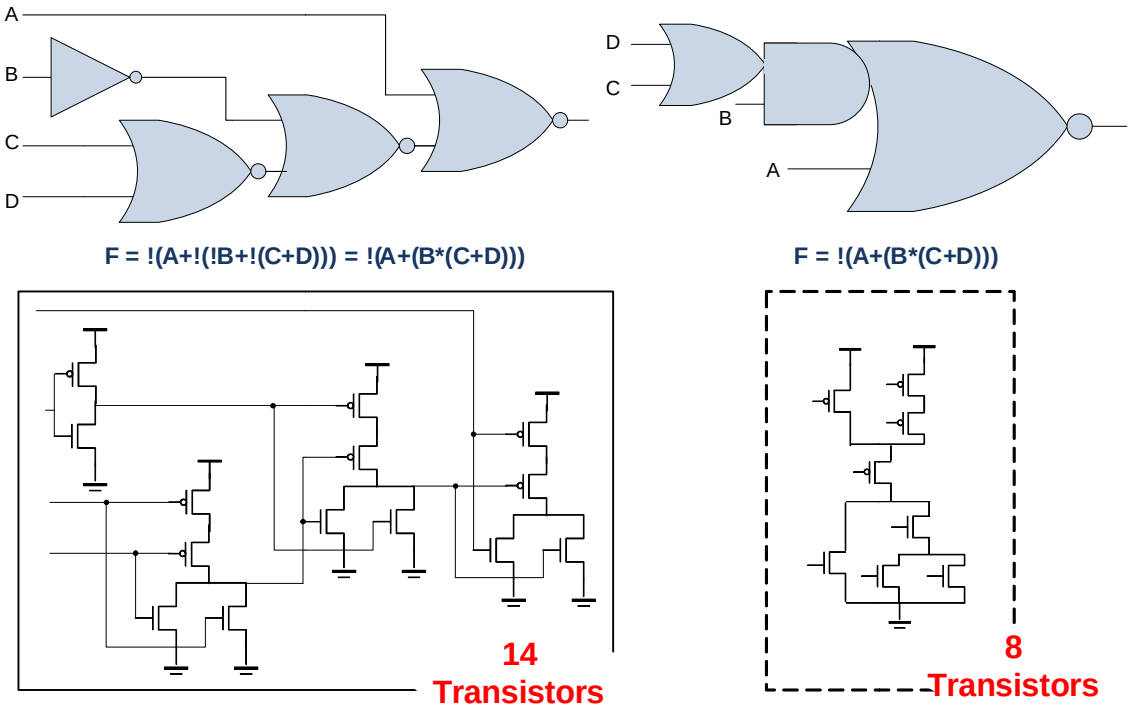

Figure 33: Different options for the physical design of a same function.

BSIM models from STMicroelectronics 65 nm technological process were used for electrical characterization. From the average of each parameter (leakage, dynamic power, transition time and delay) collected in each logical function, we could create the graph presented in Figure 34. The ordinate axis represents the percent reduction of each parameter.

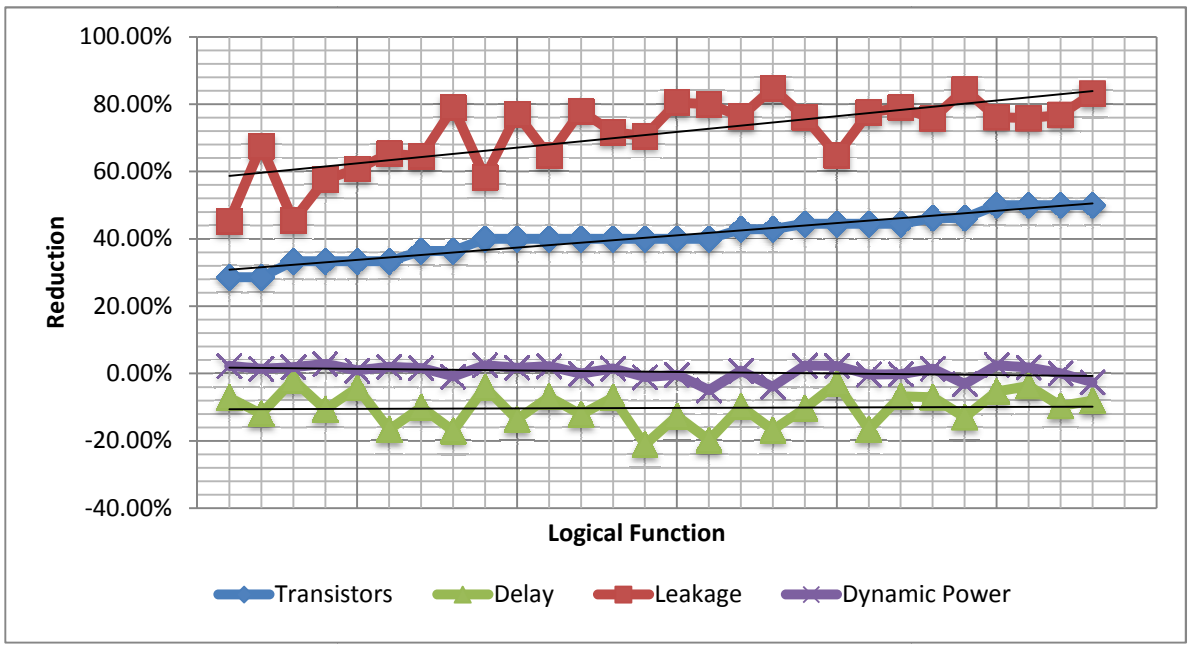

Figure 34: Comparison between standard cells
and networks of transistor implementation (Appendix A).

According Figure 34 and Appendix A it is possible to evaluate the relationship between the reduction of transistors, power consumption (leakage and dynamic power), delay and transition time. The graph presents that while increasing the reduction in the number of transistors also increases the reduction of leakage, almost as a linear tendency. For dynamic power, reduction was not so expressive, represented by an almost constant tendency line.

As expected, delay and transition time had the worst results on this experiment; once no sizing was performed on this case. This behavior can be explained by the high resistive paths created by the series of transistors in the pull-up or pull-down network, thus, in order to evaluate this behavior a specific study case is presented in next sections. The average of results presented on Figure 34 is summarized in Table 3.

Table 3: Average reduction for each parameter related to the standard cells implementation.

Average Reduction	
Transistor	40,68%
Delay	-10,25%
Transition Time	-39,05%
Leakage Power	71,30%
Dynamic Power	0,53%

4.2 TRANSITION TIME VERSUS POWER

In [VENDRICK, 84] was presented the behavior of the short circuit power consumption of a CMOS inverter and showed that the short circuit power consumption is proportional to the input transition time. Since the change in the output transition time of a gate affects the power consumption of the next stage gates of the circuit, it is essential to evaluate the power consumption related to rise/fall transition time.

In this study we evaluate, quantitatively, the relation between transition time and power consumption. For this analysis we build a scenario, as illustrated in Figure 35, and run several transistor level simulations collecting the power consumption related to the stimulus applied in the input of the device under analysis.

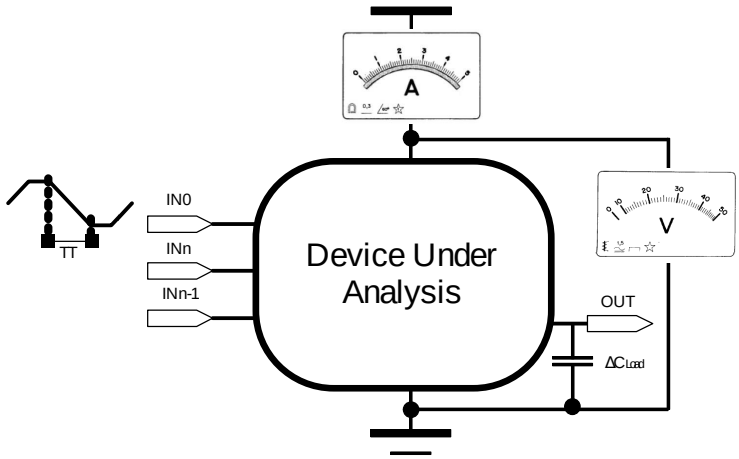

Figure 35: Scenario used to evaluate the relation between transition time and power consumption.

The device under analysis chosen for this study is an arbitrary logical function with at least two transistors in series in pull-up and pull-down network, presented in Figure 36. The most resistive input path was chosen to receive the stimulus. All the others were tied to non controlling values in order to allow the output to follow the stimulus. As stimulus we applied a square wave, varying the rise/fall transition time, with a constant load capacitance in the output.

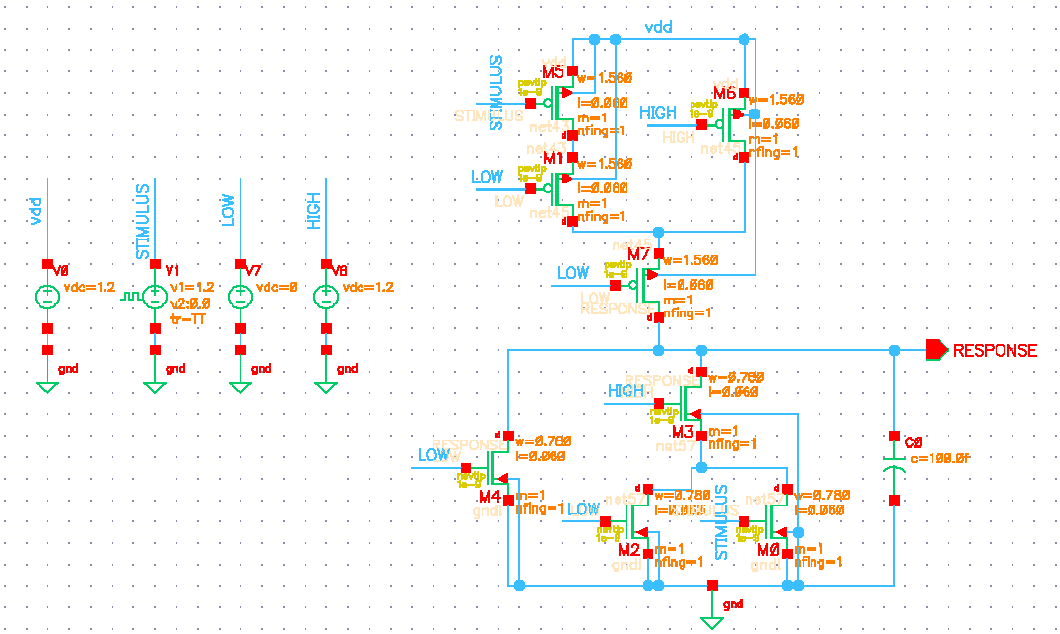

Figure 36: Device under analysis represented for equation: ! (((A + B) ∗ C)+D)

According to the input stimulus, we could build several graphs, Figure 37, which represent the voltage output response related to the input stimulus. Due to the resistive path, output response amplifies the transition time of the stimulus.

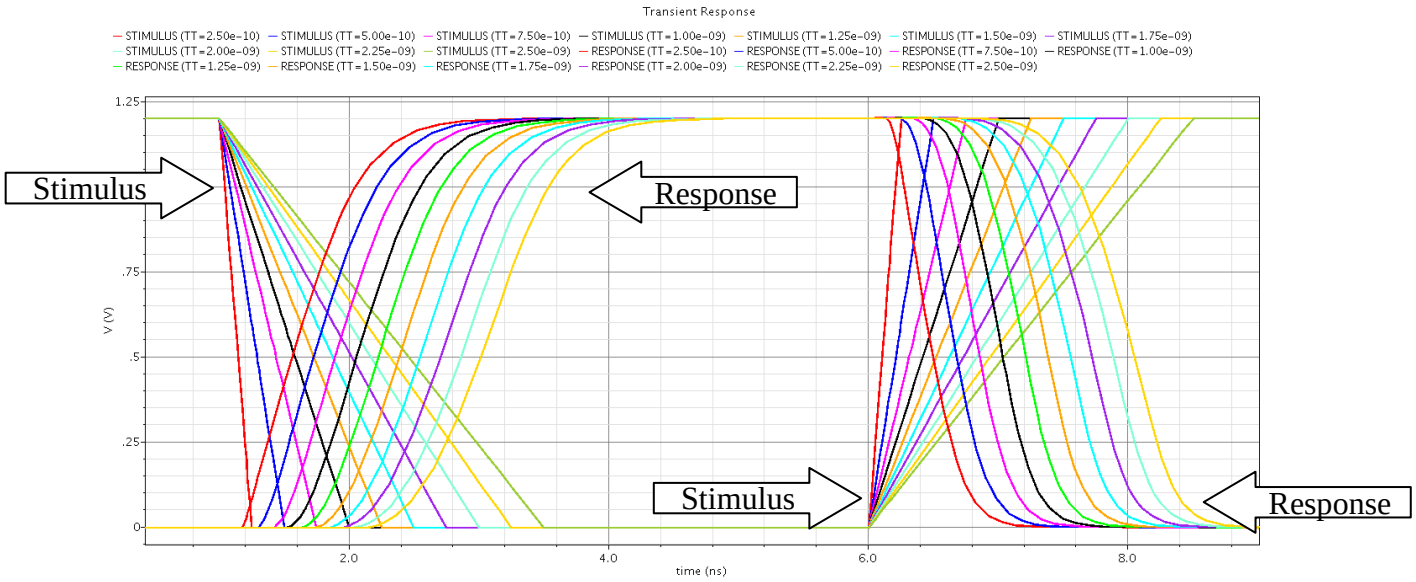

Figure 37: Voltage response related to the stimulus applied to the device.
[V(stimulus) x Time] and [V(response) x Time]

Figure 38 illustrate the behavior of the supply (VDD) current according to the variation of transition time (TT) in the stimulated input. As transition time increases, most time PMOS and NMOS transistor keep in “ON” state, rising short-circuit current, as consequence, supply current.

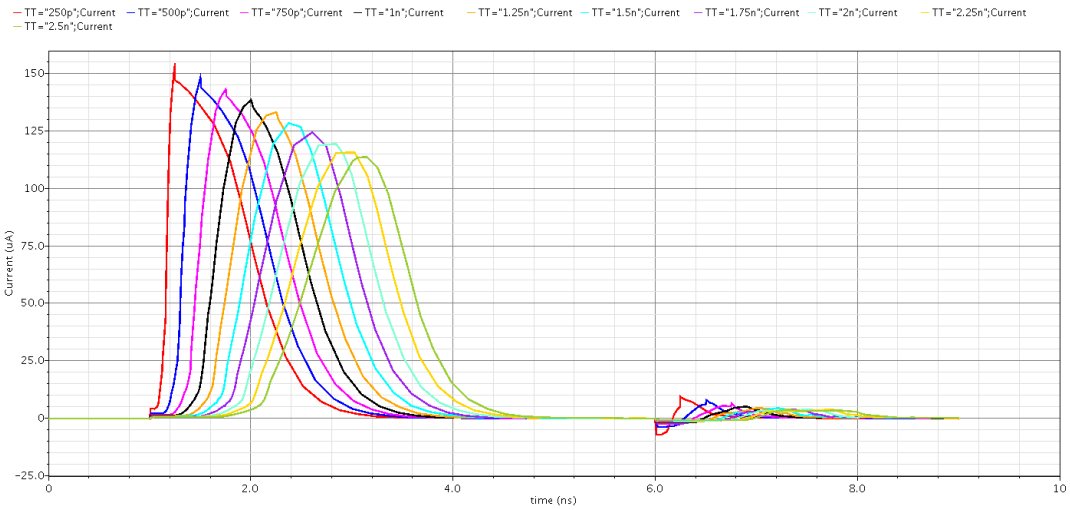

Figure 38: Current response related to the stimulus
applied to the device. [A(vdd) x Time]

In order to evaluate the behavior of the power consumption according to the transition time, for each input stimulus the total power was plotted, creating the graph presented in Figure 39. As expected, total power rises as transition time stimulus increases. Once same scenario was used for all cases, i.e. same voltage and capacitances, it is realistic to

say static and switching power keep equivalent, thus, demonstrating the relation between the transition time and power related to the short-circuit current.

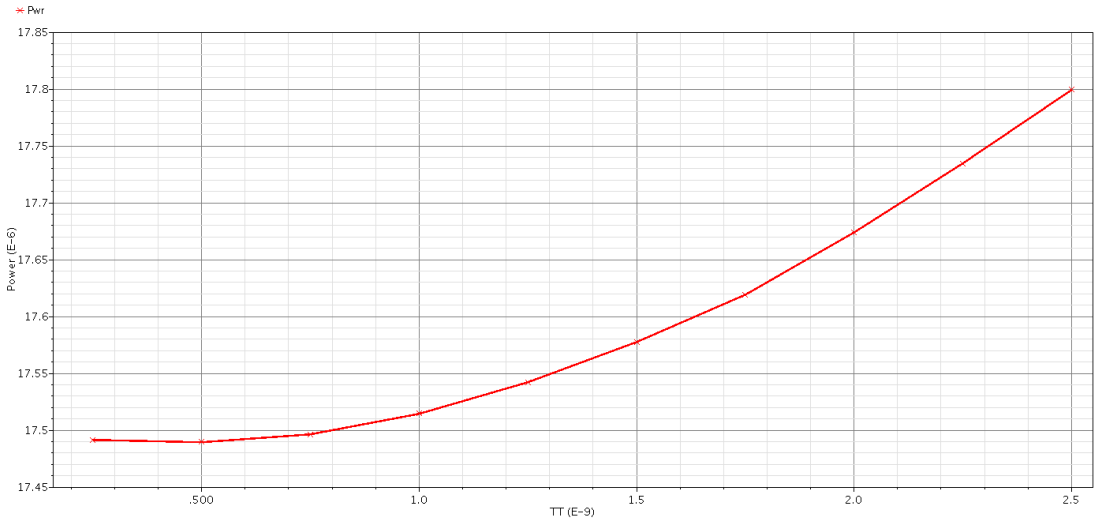

Figure 39: Power measure related to the transition time (TT) applied to the device.

4.3 NETWORKS OF TRANSISTORS (WITH SIZING VERSUS WITHOUT SIZING)

The proposal of sizing method consist in determine the best transistor sizing to find good delay or power according to the available area restriction. As presented in the literature review, there are many methods for transistor sizing, each with different proposes, some trying to optimize delay, power consumption or area.

Evaluating the previous analysis, transistor sizing became important in this work to increase the current capability of long network of transistors and handle with transition time problem. In order to model the behavior of the transistor and its interconnections, traditional works as [ELMORE, 48] and [FISHBURN, 85] were used to describe the device as a distributed RC network. With RC models, the delay through any series of transistors configuration can be expressed in terms of the transistor sizes and routing parasitic. Figure 40 shows the two-input NOR gate and its equivalent RC switch level model.

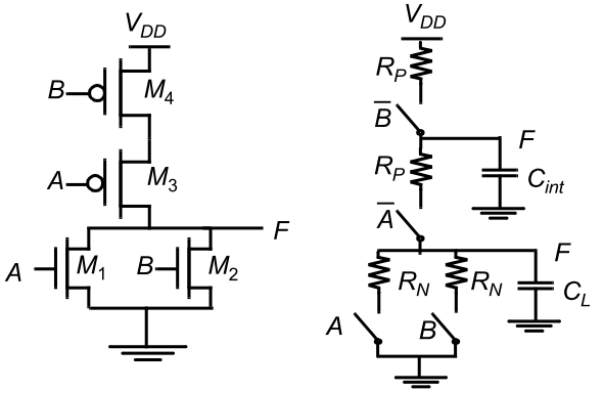

Figure 40: Equivalent RC model for a 2-input NOR [RABAEY, 2002].

In order to increase the current capability, lower must be the equivalent resistance from the load to supply on the pull-up and/or pull-down network, or higher the conductance of the path. In this work the resistances of the transistors networks are sized to get the same driving capability of a minimum inverter. So, to evaluate the equivalent RC networks two assumptions are following: (i) on parallel transistor, equivalent transistor is the one with biggest width and (ii) on series transistor, equivalent is the inverse of the inverted width sum.

Trying to evaluate just the impact of transistor sizing on transition time, scenario illustrated on Figure 36 was duplicated, and transistor sizing applied on original one, as illustrated on Figure 41.

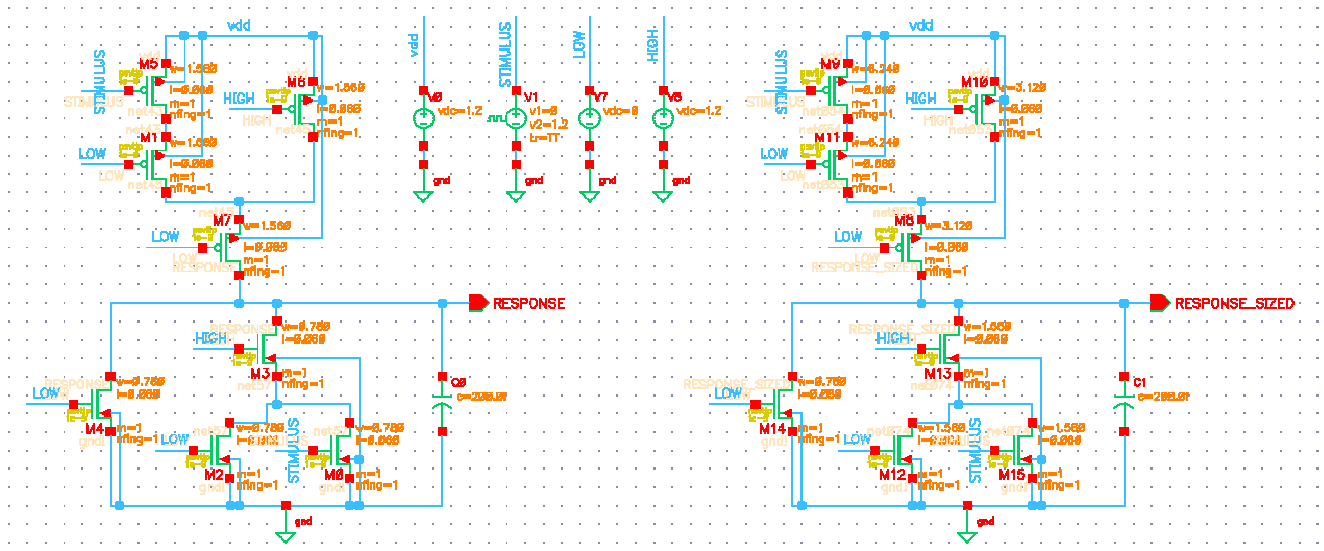

Figure 41: Device under analysis represented for equation: ! $(((A + B) * C)$+D).
On left side no sized network, on right side sized network.

In order to evaluate the worst case scenario for transition time, the most resistive path of each network of transistor was stimulated i.e. the input stimulus was applied to the port connected in the transistor nearest to the supply, on the higher chain of transistors. All the others input ports were tied to power or ground to be transparent to the analysis. For simulation, same stimulus and load capacitance were configured on both devices and output voltage plotted along the time. For better analysis, response related to pull-up and pull-down networks was splitter in two graphs.

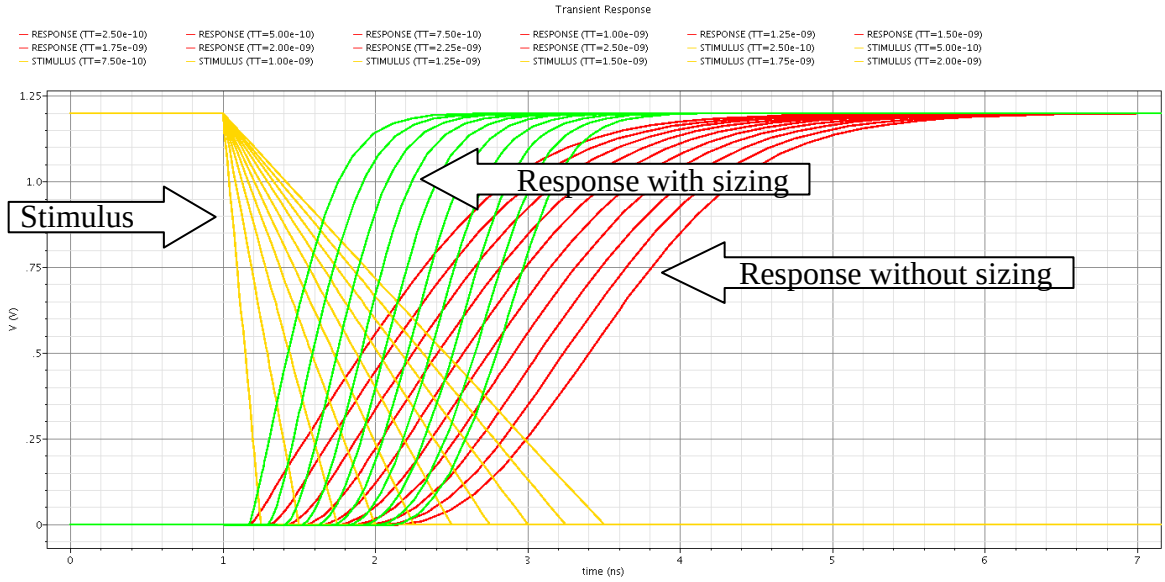

Figure 42: Voltage response related to the pull-up network.
[V(stimulus) x Time] and [V(response) x Time]

When a stimulus is applied, the current flow through the circuit loading internal and external capacitances. According to the resistance along the path, more or less time is expended to fully load the capacitor. As illustrated in Figures 42 and 44, no sized network presents more resistance to the current flow, i.e. have a lower current capability, related to the sized network. The simulation become clear when plotted the measured transition time of the outputs, Figures 43 and 45, for each input transition variation (TT).

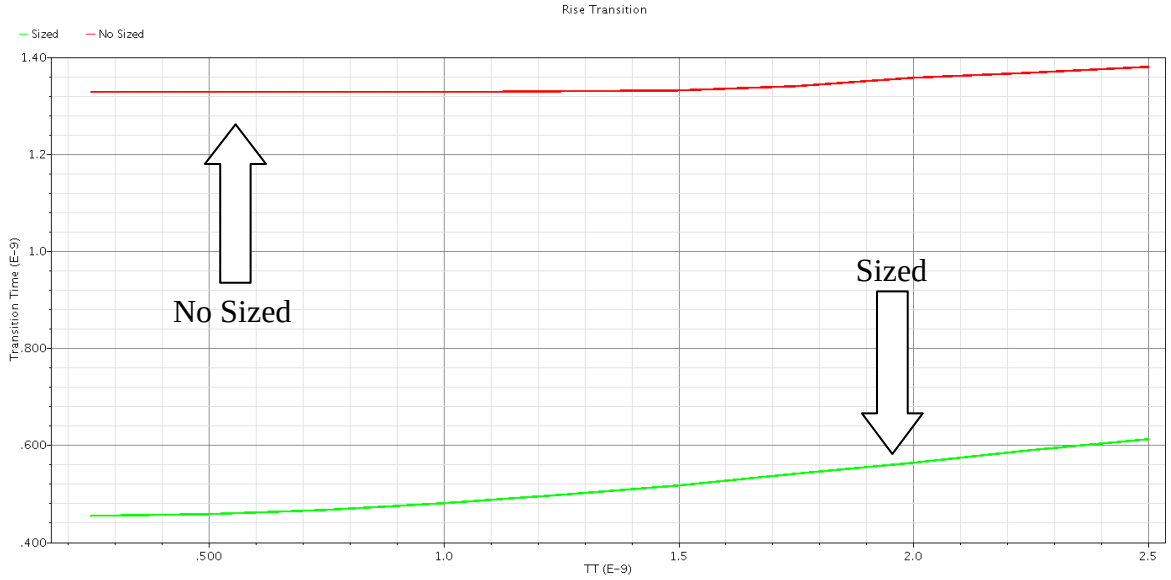

Figure 43: Transition time (Rise transition) measure related to the transition time stimulus. [Transition Time (No sized, Sized) x TT]

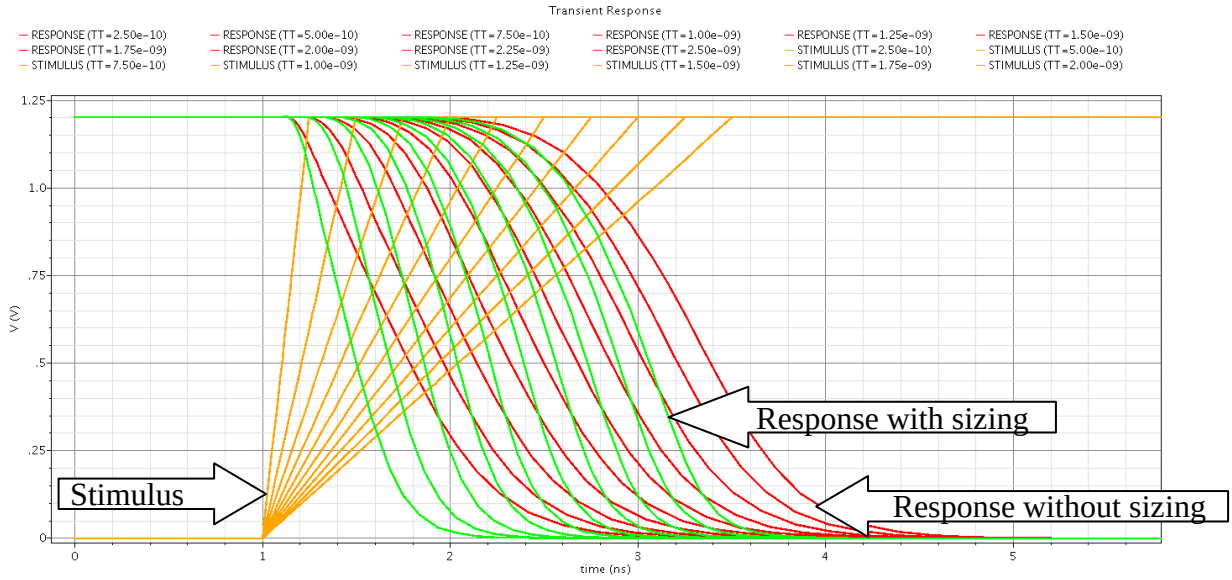

Figure 44: Voltage response related to the pull-down network. [V(stimulus) x Time] and [V(response) x Time]

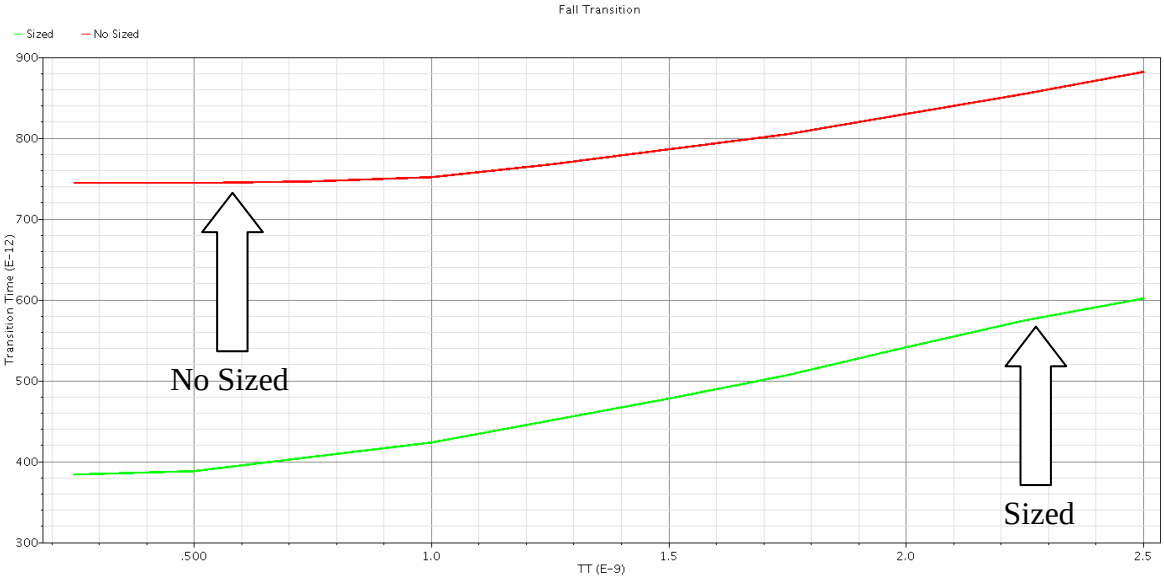

Figure 45: Transition time (Fall transition) measure related to the transition time stimulus. [Transition Time (No sized, Sized) x TT]

5 NETWORKS BASED ON SIZED TRANSISTORS

As stated in the literature review section, there are many different works related to logical mapping, some of them are related to the use of static predefined set of cells, and others leads to the use of library-free methodology, aiming to increase the fine-tuning. In both cases, the main goal of mapping process is restricted to the troubles involving timing and area optimization, leaving aside the power dissipation challenges.

Faced to the integration and the growth in mobile computing, power became a critical issue for the newest electronic development. Problems involving heat removal and cooling are worsening, stimulated for the average power dissipated for those systems. So, minimization of the amount of energy consumed is an import part of a design. To improve this capability, power optimization should be done in all abstraction levels of the design flow [CHANDRAKASAN, 1995].

Throughout this work, we evaluate the influence of the number of transistors and transistors sizing in power dissipation. In general, fewer transistors are desirable to reduce power dissipation, however, long chains of transistors required to implement specific logical functions, leads to slow transition time, and hence more power dissipation due short-circuit current. In order to prevent this fact, we build a mapping function, based on the size of transistors, to avoid slow transition time and minimize the number of transistors.

This work puts together two key issues: first the use of library-free mapping, by using networks of transistors, to optimize transistor count and second to control the transition time effects by sizing the networks of transistors and controlling the transistor gate area of the optimized design. A third issue is also import in this work, which is the use of meta heuristic techniques, as simulated annealing, to optimize the global solution, however, this is not the central component of this work.

5.1 DEFINING THE MAPPING FUNCTION

In a mapping algorithms based on standard cell, the information about the behavior of the cells is available in advance; it includes delay, area and power. With this information the algorithms for timing and power analysis are simplified, making possible to get a realistic analysis for mapping process. However, as our mapping process is based on library-free methodology, which means, no previews information about the cells, the same strategy implemented in [MARQUES, 2008] will be adopted for this work.

In Marques, the cost function, also described as regulatory or object function, uses topological metrics to estimate the delay costs. For this work, we adopted same approach, using topological metrics trying to reduce the average power. Basically, the metric is defined as the sun of the transistor gate area of each transistor, for all networks of transistors on the design.

Analytical object function formulation:

$$MF_{circuit} = \sum_{1}^{x=T_{tn}} \sum_{1}^{y=T_t[x]} L[x][y] * W[x][y]$$

Where, $MF_{circuit}$ represents the mapping function; T_{tn} is the total number of networks of transistors in the design; $T_t[x]$ is the total number of transistor in a given network of transistor; $L[x][y]$ is the length and $W[x][y]$ is the width of a given transistor. Figure 46 illustrates a function computation, $MF_{circuit}$ = 21 area units.

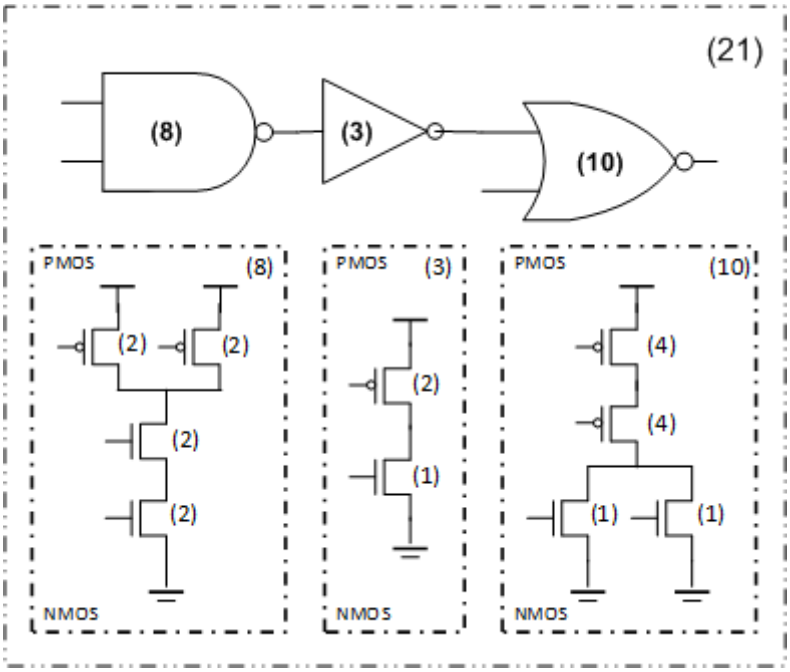

Figure 46: Computation of cost function based on W_n=1, W_p=2*W_n and L=1.

The goal in the choice of transistor gate area component is associated to the closed relation between this component with the power consumption, both dynamic and static power.

5.1.1 Transistor gate area on dynamic power

As presented in section two, the essential portion of the dynamic power is related to switching power, which means, a direct relation with the effective capacitance, as defined in equation below:

$$P_{switching} = \alpha \cdot f \cdot C_{eff} \cdot V_{dd}^2$$

As the main component of the effective capacitance is related to the gate area, i.e. pin capacitance, illustrated in Figure 8, its reduction may contribute straight to the switching power minimization. Furthermore, our approach ensure that all networks of transistor will have balanced rise and fall transition time, thus, a reduction in short-circuit is made possible in some cases.

5.1.2 Transistor gate area on static power

As the main component of static power is the sub-threshold leakage power, lots of methods can be used to minimize its impact in the design. Some example is related to the power supply and the factor "transistor width over transistor length (W/L)". In a visual analysis of the equation presented in [SHEU, 1987], and replicated below, the sub-threshold current is directed related to this factor.

$$I_{sub} = I_0 \cdot e^{\frac{V_{gs}-V_{th}}{nV_t}} \cdot \left[1 - e^{-\frac{V_{ds}}{V_T}}\right]$$

$$I_0 = \frac{W \cdot \mu_0 \cdot C_{ox} \cdot V_T^2 \cdot e^{1.8}}{L}$$

$$V_T = \frac{KT}{q}$$

This means that, as minimized is the transistor width, as increased is the transistor length leads to the reduction of static power. Once, the proposed logical mapping method uses the same length for transistors, and controls the area by sizing width, static power reduction is a direct resultant.

5.2 DEFINING THE SYSTEM DESCRIPTION

On this work we use Directed Acyclic Graph (DAG) to describe the whole system, as illustrated in Figure 47. Logic function, inputs and outputs are described as nodes of the graph, connected through arrows that represent the wires between the nodes. Special input and outputs were created to implement sequential logical, as flip-flops and latches.

Once, owner proposes is the optimization of combinational logic, all the sequential logic must be removed, that is why, the need for special nodes.

Inside each node and wire structure, different components are defined. The main component of the node is the logical function; for this implementation tree different representations were used to build the system and minimize run timing: Binary Decision Diagram (BDD), Binary Trees and Boolean equation, illustrate in Figure 48. With these different representations, execution time is reduced, paid by the increased memory utilization.

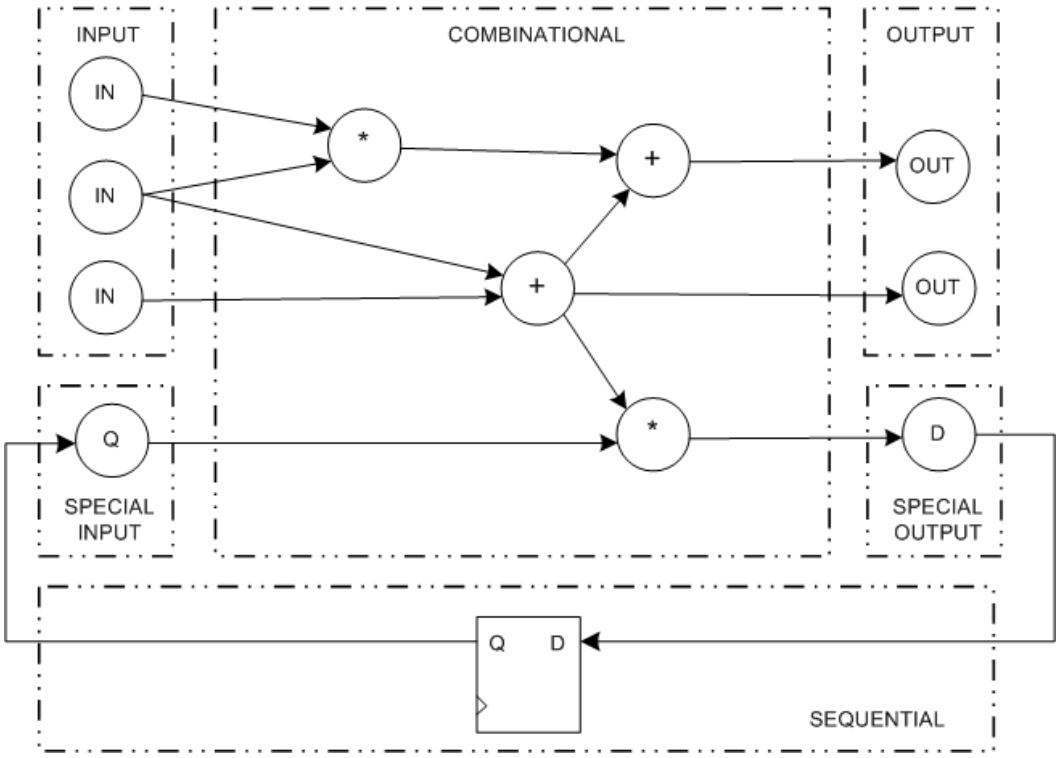

Figure 47: Example of system representation.

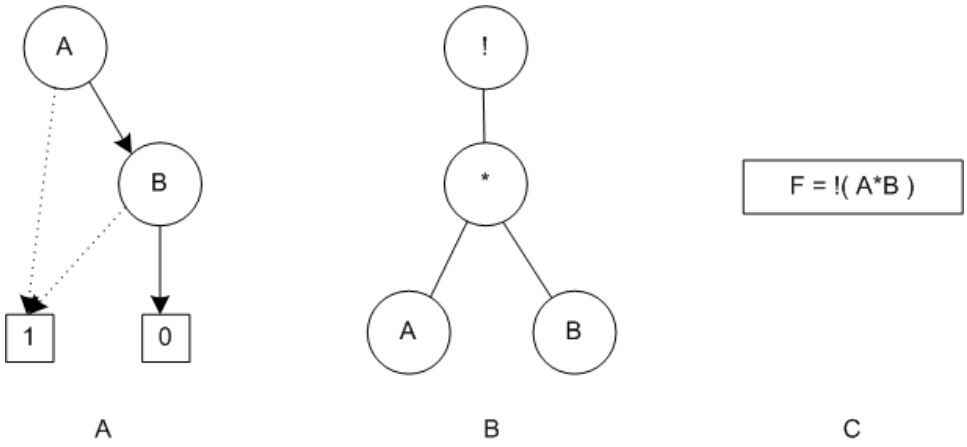

Figure 48: Different representations of the same logical function: (A) BDD, (B) Binary Tree, (C) Boolean equation.

5.3 PRE-PROCESSING PROCEDURES

As others mapping process, the present work require a pre-processing procedures before running the matching and covering steps. At first, the original circuit representation must be translated to the topology defined in section 5.2. In order to simplify the problem of describing a logical equation into a networks of transistor; all logical elements are described using inverted outputs. This procedure makes the translation to networks of transistors straightforward, once a NAND function can be directed described as two NMOS transistor in series (and its complementary pull-up network) with a unique operator.

About the literals of the logical function, is defined that all literals are non-inverted, which means, in case of requiring an inverted literal, e.g. !a + b , an inverter node must be placed in the graph, and connected to the required input arrow, Figure 49.

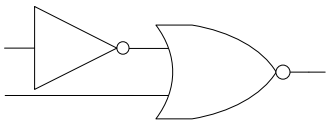

Figure 49: Graph representation of function: !a + b.

Another important characteristic is that our algorithm assumes that the initial circuit is completely decomposed in 2-input NAND/NOR gates and inverters, in order to increase the granularity of the circuit, and to give a higher freedom for the matching algorithm in creating the complex functions. According to the property described above, all logical function must be declared as inverted, so in case of requiring implementing a non-inverted function, an inverter node must be placed in the graph connected to the output arrow, as illustrated in Figure 50.

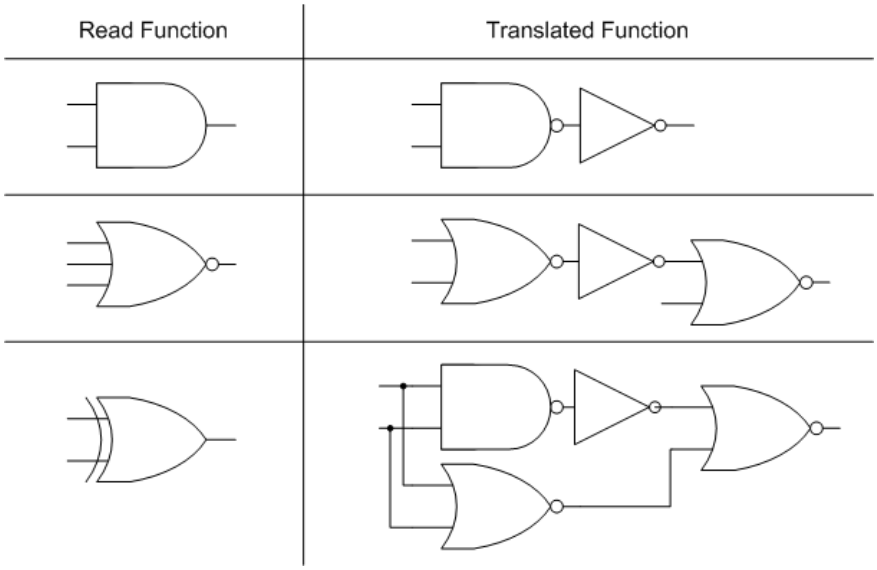

Figure 50: Example of decomposed AND/NOR/XOR gates.

5.4 LOGICAL MAPPING METHOD

In [LI, 2009] is presented a mapping method for sequential circuits in FPGA. A look-up table (LUT) based FPGA consists of an array of programmable logic blocks together with programmable logic blocks. The core of a programmable logic block is an n-input LUT that can implement any combinational logic with up to “n” inputs and a single output. According to the author, the technology mapping problem for LUT based FPGAs is to produce, for a given circuit, an equivalent circuit comprised of LUTs, obeying the minimum constraints.

The definitions presented in [LI, 2009] for FPGA is very close to the problem of mapping a generic logical structure into a library-free approach. The similarity lies in the fact that both n-input LUT, as the library-free based methodology are freedom to implement any combinational logic, according to requirements.

In order to solve the technology mapping problem for LUT based FPGA, most structural mapping algorithms are based on dynamic programming technique. However, in [LI, 2009] was applied, for the first time, simulated annealing algorithm to solve mapping problems. According to the author, simulated annealing algorithm can solve the contradiction between mapping features leading to better results. Due to the similarity of the problems simulated annealing algorithm will be adopted in this work too.

5.5 SIMULATED ANNEALING

Simulated annealing is a generic probabilistic metaheuristic to solve global optimization problem, namely locating a good approximation to the global minimum of a given function used to address discrete and, to a lesser extent, continuous optimization problems. The key feature of simulated annealing is to provide means to escape local optimal by allowing hill-climbing moves, i.e., moves that worsen the objective function value, in hopes of finding a global optimum [HENDERSON, 2003]. For certain problems, simulated annealing may be more effective than exhaustive enumeration, provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

The name and inspiration of this method comes from its analogy to the process of physical annealing with solids, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one.

By analogy with this physical process, at each iteration of the simulated annealing algorithm replaces the current solution by a random "nearby" solution, according to a perturbation function. Improved solution is always accepted, while a fraction of nom-improved solutions are accepted according to a temperature dependent probability that is typically decreased along the iterations of the algorithm.

5.5.1 Perturbation Functions

Perturbation functions are specified in order to create structural modification on system topology, leading to modifications on cost function. In our approach were defined three types of perturbation: invert forward, group forward and logical cut, as describe below.

5.5.1.1 Invert Forward

The invert forward perturbation function is created to minimize the inverters in the system. For each logical node, when applied this perturbation, its logical function is inverted, Figure 51, and all fanout connected to this node is also inverted, keeping the same logical structure. In case of the node under perturbation is an input, output, special input or special output, invert forward process are ceased, which means, the system is not perturbed.

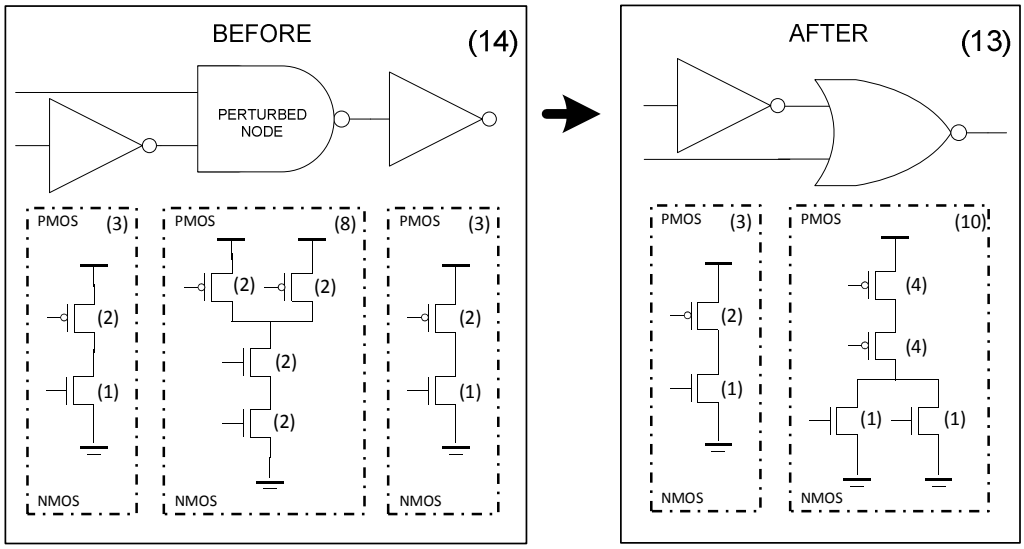

Figure 51: Example of invert forward perturbation applied in 2-inputs NAND gate.

5.5.1.2 Group Forward

The group forward perturbation function is created to optimize the logical structure by the creation of complex logical functions. A node is chosen, and its logic is used in the nodes connected to the fanout to create a composed complex structure. If the new logical function leads to an inverted literal, inverters must be placed in the respective input. In case of node under perturbation, or any fanouts of this node be an input node, output, special input or output node, group forward process are ceased, which means, the system is not perturbed.

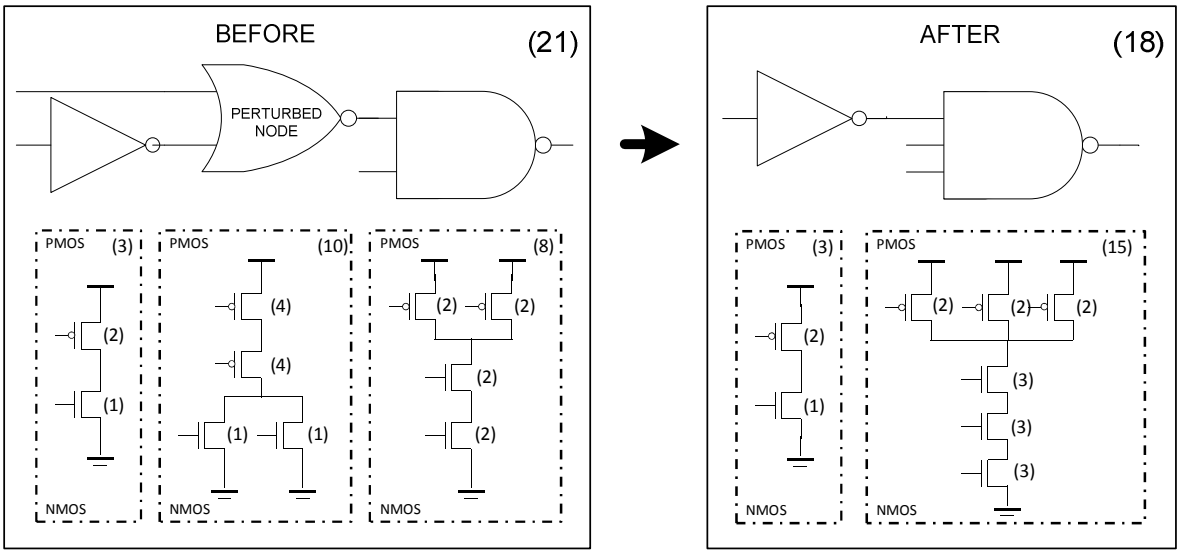

Figure 52: Example of group forward perturbation applied in 2-inputs NOR gate.

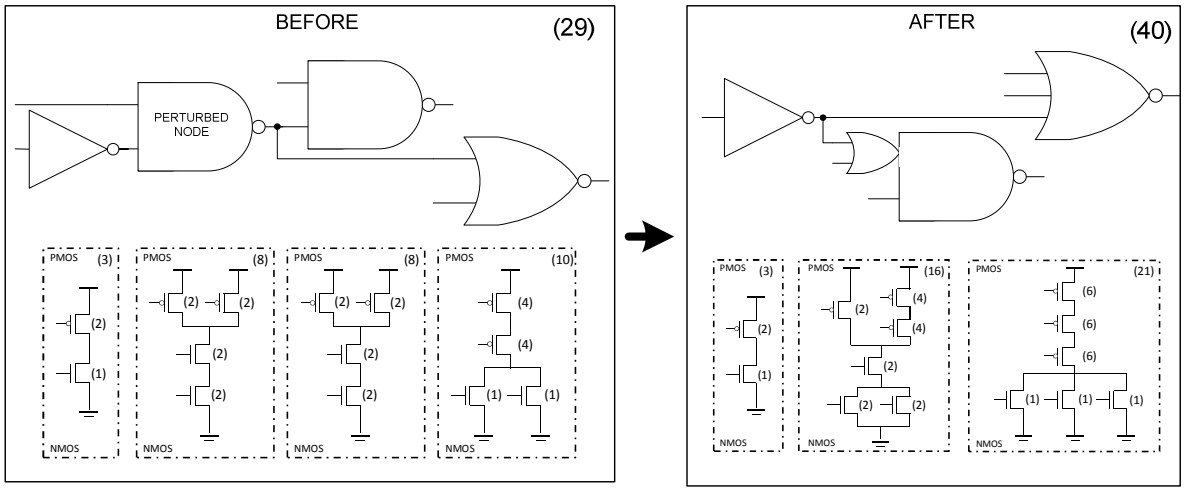

Figure 53: Example of group forward perturbation applied in 2-inputs NAND gate, with two branches.

5.5.1.3 Logical Cut

The logical cut perturbation function is created in order to break complex logical structures, increasing the granularity of the system, leading to an option of topology. A specific node is selected, and its logical structure are decomposed in 2-input NAND/NOR gates and inverters, keeping the same logical function. In case of node under perturbation is an input, output, special input or output, logical cut process are ceased, which means, the system is not perturbed.

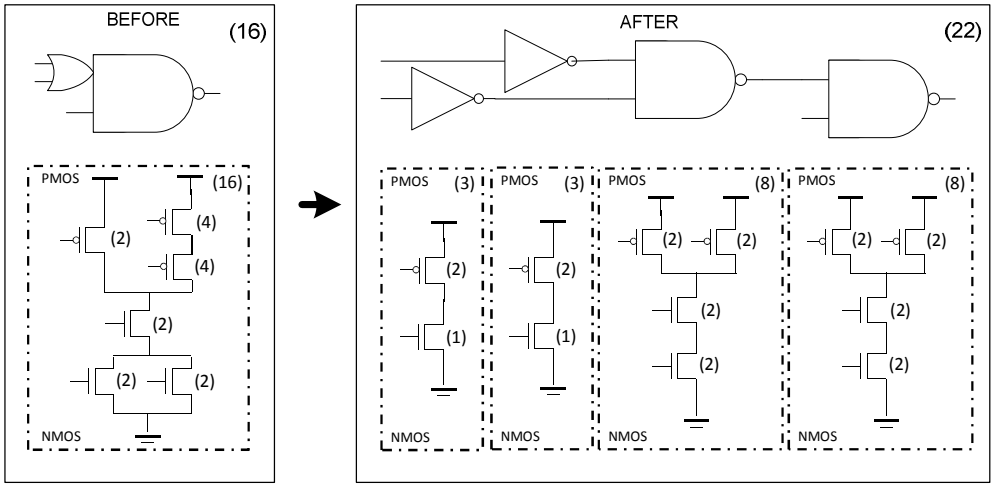

Figure 54: Example of logical cut perturbation applied in a complex gate.

5.5.2 Simulating Annealing Algorithm

In this section is presented the basic structure of the simulation annealing algorithm implemented in this work.

Table 4: Basic structure of simulating annealing algorithm.

```
input: System Graph (system), Initial temperature (t), Stop point (e)
output: Optimized System Graph

1.  temperature ← t
2.  stop ← e

3.  clone ← system
4.  bcost ← cost(clone)

5.  while ( temperature *= value ) >  stop
6.      perturb(clone)
7.      ncost  ← cost(clone)
8.      if ( ( ncost <  bcost ) || ( P( ncost, bcost, temperature ) > random( ) ) ) then
9.          bcost ← ncost
10.         system ← clone
11.     else
12.         clone ← system
13.     end
14. end
```

5.6 POST-PROCESSING PROCEDURES

After running the simulating annealing algorithm, we hope to get a graph with the minimum global function cost as a result of the procedure. So, to make possible to use those data into different commercial tools, basically tree files are created.

5.6.1 Behavioral Netlist

Describe the behavior of the system pre-mapping as a structure decomposed in 2-input NAND/NOR gates and inverters. All inputs, outputs and special input and outputs are expressed as ports in a default verilog netlist description. The goal of writing this description is to obtain a truly behavior description of the system to be used in formal verification checks.

Table 5: Behavioral circuit description.

```
1.  module b01_generic (LINE1, LINE2, Q_reg_0, Q_reg_1, ( … ), D_reg_0, D_reg_1);
2.      input wire LINE1, LINE2, Q_reg_0, Q_reg_1, ( … );
3.      output   ( … ), D_reg_0, D_reg_1, ( … );
4.       ( … )
5.      wire i_14, i_13, i_15, ( … );
6.       ( … )
7.      assign OUTP_REG = Q_reg_4;
8.       ( … )
9.      nand g19 (i_14, Q_reg_3, U38);
10.     not    g20 (i_13, i_14);
```

```
11.     nand g21 (i_15, i_13, Q_reg_2);
12.     not   g22 (D_reg_0, i_15);
13.     nand g23 (i_19, U65, U66);
14.     not   g24 (i_18, i_19);
15.      ( … )
16.  endmodule
```

5.6.2 Structural Netlist

Describe the structure of the system post-mapping according to networks created on logical mapping process. All inputs, outputs, special input and outputs are expressed as ports in a default verilog netlist description. The goal of writing this description is to obtain a truly structure description of the system to use as input in commercial place and route tools.

Table 6: Structural circuit description.

```
1.   module b01_synthesized (LINE1, LINE2, Q_reg_0, Q_reg_1, ( … ), D_reg_0, D_reg_1);
2.      input wire LINE1, LINE2, Q_reg_0, Q_reg_1, ( … );
3.      output   ( … ), D_reg_0, D_reg_1, ( … );
4.       ( … )
5.      wire i_14, i_13, i_15, ( … );
6.       ( … )
7.      assign OUTP_REG = Q_reg_4;
8.       ( … )
9.      G9 inst2 ( .W0(Q_reg_3), .O1(i_13) );
10.     G9 inst3 ( .W0(Q_reg_2), .O1(i_19) );
11.     G0 inst4 ( .W0(U43), .W1(U67), .W2(i_68), .O1(D_reg_3) );
12.     G1 inst5 ( .W0(Q_reg_3), .W1(U54), .W2(U47), .W3(i_23), .W4(i_52), .O1(i_25) );
13.     G7 inst6 ( .W0(i_55), .W1(i_25), .W2(i_45), .O1(D_reg_2) );
14.     G8 inst7 ( .W0(LINE1), .W1(LINE2), .O1(i_30) );
15.      ( … )
16.  endmodule
```

5.6.3 Networks of Transistor Netlist

Describe the structure of the networks of transistors created in the mapping process. The description is exported in spectre format and the goal is to know the structure of the cells that form the system to use as input in a commercial characterization tool, and in an automated layout generator, to create the layout of the design-specific cells to be on a place and route tool.

Table 7: Network of transistor used in structural description.

```
1.   ************************************************
2.   simulator lang=spectre
3.   ************************************************
```

```
4.  * Implements Equation: O1 = !((W0*W1)*W2)
5.  * Transistor Count: 6
6.  ************************************************
7.  subckt G0 ( W0 W1 W2 O1 VDD GND )
8.      MP0 ( O1 W0 VDD VDD  p L=0.35u W=1*1*1.6u
9.      ( … )
10.     MN2 ( O1 W2 O4 GND ) n L=0.35u W=1*3*1.0u
11. ends G0
12. ************************************************
13. * Implements Equation: O1 = !((W1*W3)+((W0*W2)*W4))
14. * Transistor Count: 10
15. ************************************************
16. subckt G1 ( W1 W3 W0 W2 W4 O1 VDD GND )
17.     MP0 ( O6 W0 VDD VDD ) p L=0.35u W=1*2*1.6u
18.     ( … )
19.     MN4 ( O1 W4 O8 GND ) n L=0.35u W=1*3*1.0u
20. ends G1
21. ( … )
```

6 EXPERIMENTS AND RESULTS

In order to verify the effectiveness of the mapping method described in preview sections, a set of experiments were performed using several testbenchs from literature. The same testbenchs circuits were logically mapped using two approaches: commercial cell based logical synthesis tool and network of transistor mapping method based on sized transistors, presented above. As figure of merit, power and timing was compared for both methods, using devices models from STMicroelectronics 65nm technology process.

To ensure a fair comparison between both approaches we follow the flow illustrated in Figure 55. For each experiment the same circuit was used for both methods. Furthermore, the cell library used as input to the commercial mapping tool was recharacterized according to the simulation parameter and transistor models used in the network of transistor characterization.

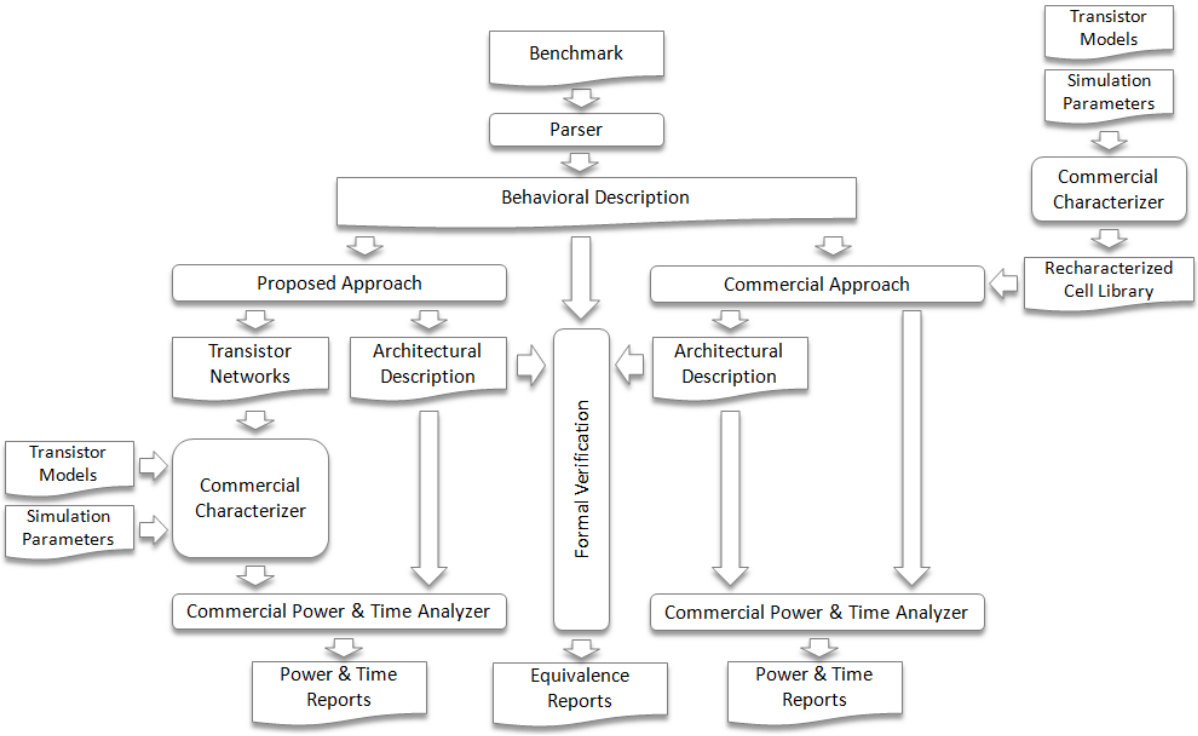

Figure 55: Flow follow to run the experiments.

The circuits used for this experiment are a set of benchmarks from [CORNO, 2000] and summarized on Table 8.

6.1 BENCHMARK DESCRIPTION

Table 8: Set of circuits under analyzes.

NAME	GATE LEVEL					FUNCTIONALITY
	PI	PO	FF	L0	L1	
b01	2	2	5	0	0	FSM that compares serial flows
b02	1	1	4	0	0	FSM that recognizes BCD numbers
b03	4	4	30	0	0	Resource arbiter
b04	8	11	66	8	4	Compute min and max
b06	2	6	9	0	0	Interrupt handler
b07	1	8	49	0	0	Count points on a straight line
b08	9	4	21	0	0	Find inclusions in sequences of numbers
b09	1	1	28	0	0	Serial to serial converter
b10	11	6	17	0	0	Voting system
b11	7	6	31	6	5	Scramble string with variable cipher
b12	5	6	121	0	0	1-player game (guess a sequence)
b13	10	10	53	2	1	Interface to meteorology sensors

PI: Number of primary inputs
PO: Number of primary outputs
FF: Number of *flip-flops*
L0/1: Number of *logic-zero*/*logic-one*

6.2 RESULTS ANALYSIS

To perform analysis it was used a commercial power and timing engine for both approaches. Several reports for dynamic power, static power (due leakage) and worst delay were obtained related to the mapping approach and technology process. Once power analysis requires a stimulus to be calculated, power was evaluated according to a constant switching probability and toggle rate on the input ports of the circuit and propagated thought the system according to the switching probability of the logical function.

The follow values were used for each analyzed circuit, which are:

- **Switching probability:** 50%;
- **Toggle Rate:** 0.02 toggles per nanosecond.

According some related works, these values are overestimated, i.e. higher than those obtained on a real system operation. Nonetheless, it is a good starting point for our analysis.

6.2.1 Technology process - STMicroelectronics 65 nm

As described above, logical mapping process for STMicroelectronics 65 nm fellow the flow illustrated on Figure 55. For all benchmarks from Table 8, logical mapping method was prepared in the two approaches, commercial and the proposed one. After mapping process, the resultant structural circuit was imported on a commercial power and timing engine in order to measure power and delay.

6.2.1.1 Power Analysis

For power analysis, two are the most important components and evaluated on the analysis: dynamic and static power (related to leakage). For both mapping approaches, we measured and plotted power results, as illustrated on Figures 56 and 57, respectively, dynamic and leakage power.

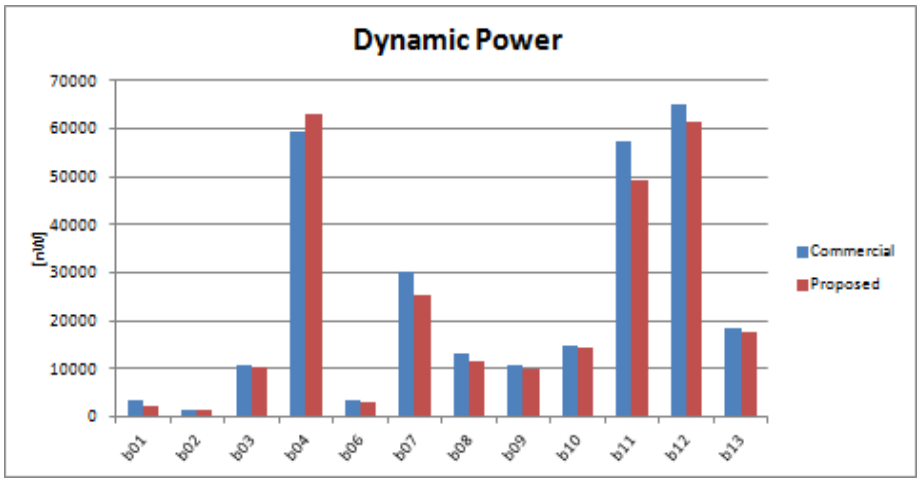

Figure 56: Dynamic power results for commercial and proposed mapping tool.

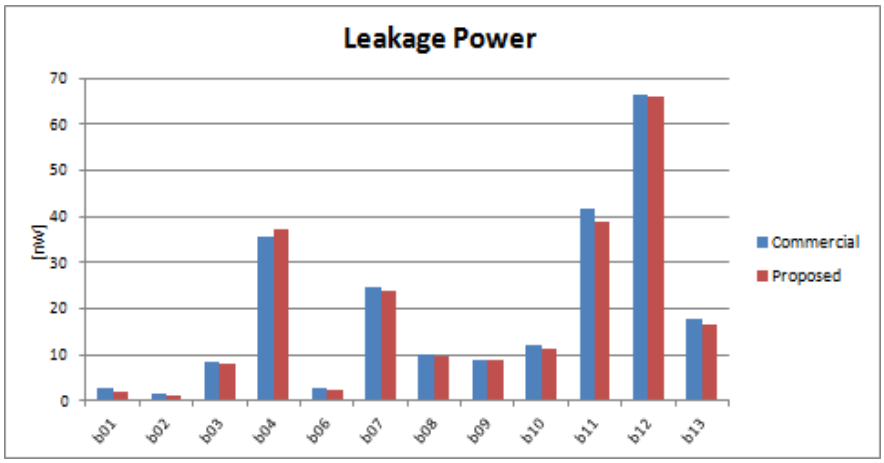

Figure 57: Static power (due leakage), results for commercial and proposed mapping tool.

According to the analysis, Figure 58 summarizes the difference between power consumption measured on both approaches. As results, power consumption was optimized for the biggest part of testbenchs on the proposed work, presenting an average reduction of 6.35% in dynamic power and 8.26% in leakage power.

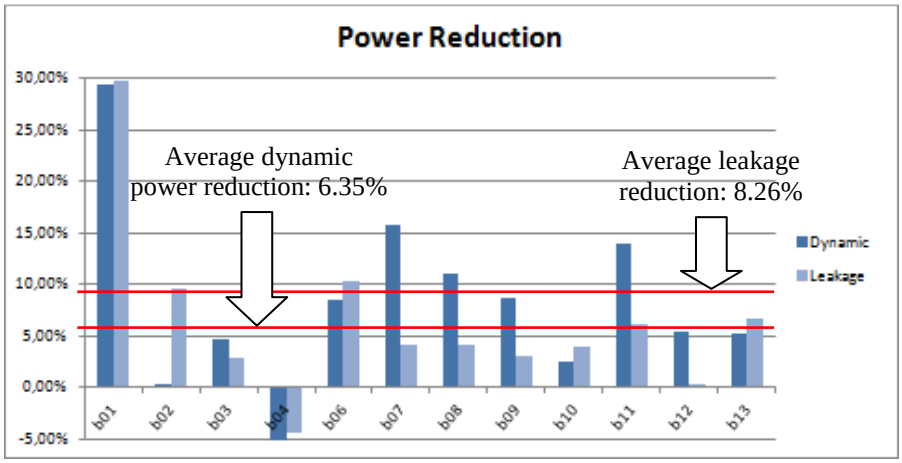

Figure 58: Dynamic and leakage power reduction between commercial and proposed work.

6.2.1.2 Delay Analysis

For delay analysis, we considered the worst delay path of the evaluated circuit. Figure 59 illustrate the worst delays on each testbenchs when mapped using commercial mapping tool and proposed one.

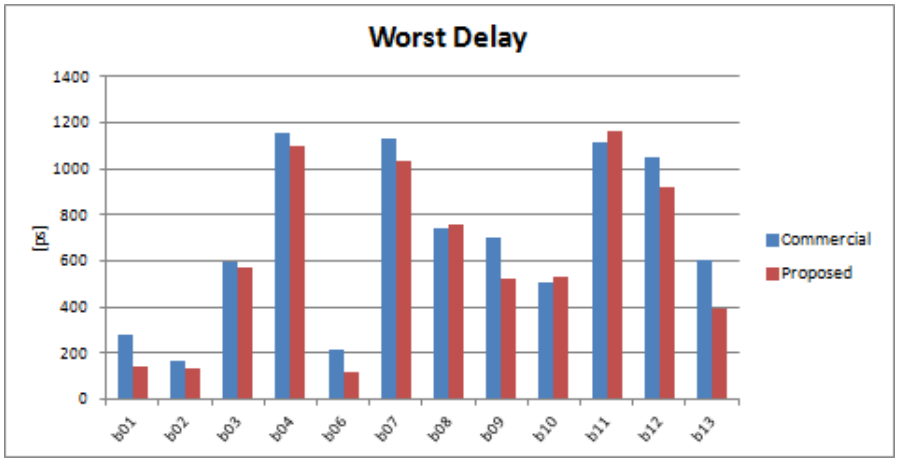

Figure 59: Delay results for commercial and proposed work.

Comparing results from each mapping approach we could create the graph presented on Figure 60. Results present that worst delay was also optimized on proposed mapping process. In average 15.96% of worst delay was optimized.

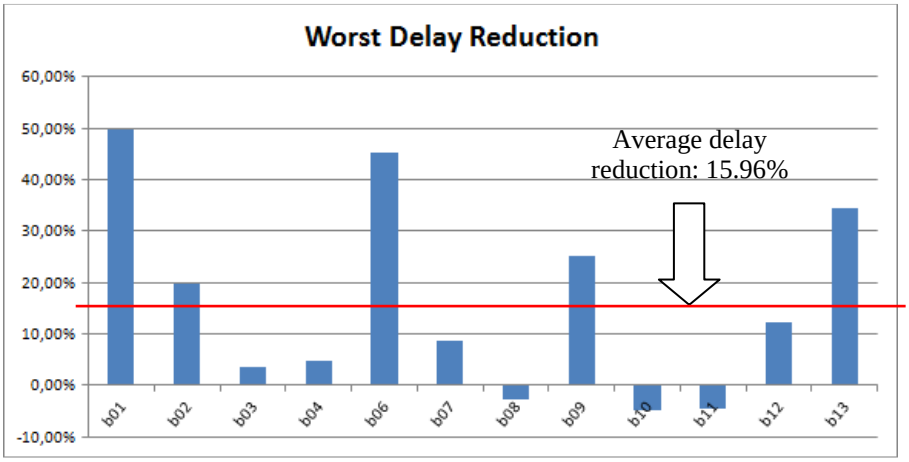

Figure 60: Delay reduction between commercial and proposed work.

6.2.1.3 Transistor Analysis

For transistors analysis, we considered the sum of all transistors presented on the evaluated circuit. Figure 61 illustrate the number of transistor on each testbenchs when mapped using commercial mapping tool and proposed one.

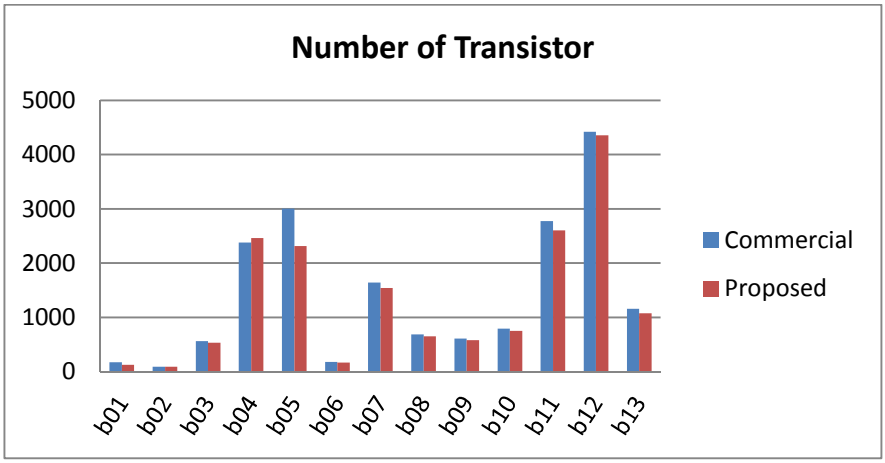

Figure 61: Transistor count results between commercial and proposed work.

Comparing results from each approach we could create the graph presented on Figure 62. As expected, proposed mapping method optimized the number of transistors of the circuit. In average 7.3% of transistor count was optimized.

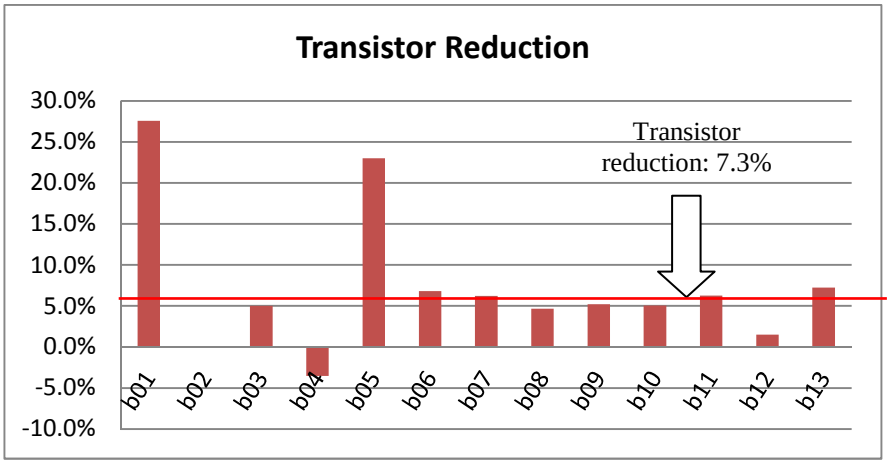

Figure 62: Transistor reduction between commercial and proposed work.

6.3 LOGICAL EQUIVALENCE CHECK

A logical equivalence check was performed to verify the logical equivalence between different circuit descriptions. Technological mapping process modifies the circuit at every moment while running, thus, failures may occur resulting in logical mismatching. Using a logical equivalence check tool, we can confirm the correctness of the mapped circuit.

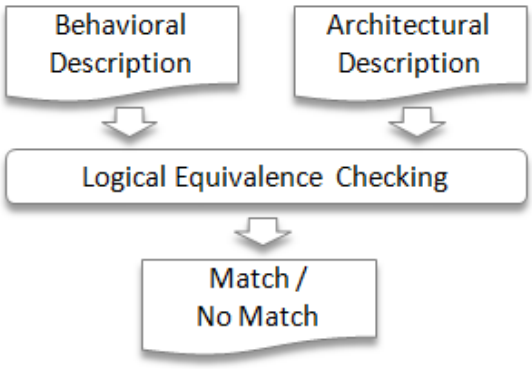

Figure 63: Flow used on logical equivalence check.

For this experiment, all the circuits were verified through equivalence check, comparing the behavioral description (pre-mapping) with the structural one (pos-mapping). According this verification, all circuits were mapped keeping the logical equivalence.

7 CONCLUSIONS

In newest technologies process, power became a critical issue faced for the process scaling. Problems involving heat removal and cooling are worsening stimulated for the power dissipated on those systems. Along last years, many low-power techniques were developed trying to solve this issue; however, this process could still be improved. Traditional libraries cells are mature and reliable, but limited with respect to fine-tuning optimizations. Even an extensive cell library has the disadvantage of being discrete, which means mapping tools cannot efficiently implement some logical functions, creating bottlenecks for process optimization.

In this work, it was presented the use of library free methodology as an alternative for low-power designs. Throughout this work, we studied the influence of the number of transistors and the importance of transistors sizing in power dissipation. In general, fewer transistors are desirable to reduce power dissipation, however, long chains of transistors required to implement specific logical functions leads to slow transition time, and hence more power dissipation due short-circuit current.

In order to prevent this issue, we modeled a mapping function to control the number of transistors, at the same time preventing long resistive chains of transistors. We applied this function as cost function on simulating annealing algorithm to solve the optimization problem. The use of this method provided an efficient tuned set of networks of transistors for the experimented IC designs, where efficient represents metrics like power consumption and delay, measured against standard cell implementation.

By allowing the networks of transistors to be freely manipulated, and controlling the global transistor gate sizing, we could avoid the creation of blocks with “unlimited” transistors; at the same time allows the creation of unrestricted inputs blocks. Such flexibility does not come for free. A couple of issues ranging from logical mapping, to minimizing the number of networks of transistors used in a design, to efficient characterization, to control issues like physical design information, need to be addressed.

From experiments presented in chapter 6, we can observe a significant reduction in power consumption, both static and dynamic power, and delay, when compared the proposed work against commercial standard cells based mapping tool. As results, power was optimized for the biggest part of benchmarks, presenting an average reduction of 6.35% in

dynamic power and 8.26% in leakage power on STMicroeletronics 65 nm, illustrated on Figure 58. Analogous result was presented for delay analysis, in average 15.6% of worst delay was optimized, illustrated on Figures 60. As expected, power and timing results fellow the optimization on the number of transistor, illustrated on Figure 62.

Evaluating these results, we could justify the use of networks of transistors, and the presented mapping method on designing low-power circuits, where there is great potential for reducing the number of components and should comply with restricted power constraints.

REFERENCES

1. BEECE, D. K. et al. Transistor sizing of custom high-performance digital circuits with parametric yield considerations. In: DESIGN AUTOMATION CONFERENCE, 47., 2010. **Proceedings…** Anaheim, CA, USA: ACM/IEEE, 2010. p. 781-786.

2. BENINI, L. et al. Designing low-power circuits: practical recipes. **Circuits and Systems Magazine**, IEEE, vol.1, no.1, pp.6-25, First Quarter 2001 doi: 10.1109/7384.928306.

3. BERKELAAR, M.R.C.M. et al. Technology mapping for standard-cell generators. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1988. **Proceedings…** Santa Clara, CA, USA: IEEE, 1988. p. 470-473.

4. BORAH, M. et al. Transistor sizing for low power CMOS circuits. **Computer-Aided Design of Integrated Circuits and Systems**, IEEE Transactions on, vol.15, no.6, pp.665,671, Jun 1996, doi: 10.1109/43.503935.

5. BRATTACHARYA, D.; BOPPANA, V. Design optimization with automated flex-cell creation. In: **Closing the gap between ASIC & custom.** [S.l.] : Kluwer, 2002. p. 241-266.

6. CHANDRAKASAN, A.P. et al. Minimizing power consumption in digital CMOS circuits. **Proceedings of the IEEE**, vol.83, no.4, pp.498-523, Apr 1995 doi: 10.1109/5.371964.

7. CHEN, W. (Ed.). **The VLSI Handbook**. CRC Press, Inc., Boca Raton, FL, USA, 2000, ISBN:0-8493-8593-8.

8. CHINNERY, D.G. at al. Closing the Gap between ASIC and Custom: An ASIC Perspective. In: DESIGN AUTOMATION CONFERENCE, 37., 2000. **Proceedings…** Los Angeles, California, USA: ACM/IEEE, 2000. p. 637-642.

9. CORNO, F. et al. RT-level ITC'99 benchmarks and first ATPG results**. Design & Test of Computers**, IEEE, vol.17, no.3, pp.44-53, Jul/Sep 2000, doi: 10.1109/54.867894.

10. CORREIA, V. et al. Advanced technology mapping for standard-cell generators. In: INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 17., 2004. **Proceedings…** Porto de Galinhas, Pernambuco, Brazil: IEEE, 2004. p. 254- 259.

11. CRASTES, M. et al. A technology mapping method based on perfect and semi-perfect matching. In: DESIGN AUTOMATION CONFERENCE, 28., 1991. **Proceedings…** New York, NY, USA: ACM/IEEE. p. 93-98.

12. DE-SHIUAN, C. et al. Timing driven power gating. In: DESIGN AUTOMATION CONFERENCE, 43., 2006. **Proceedings…** San Francisco, CA, USA: ACM/IEEE, 2006. p. 121-124.

13. DETJENS, E. et al. Technology Mapping in MIS. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1987. **Proceedings...** Los Alamitos, CA, USA: IEEE, 1987. p.116-119.

14. ELMORE, W. C. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. **Journal of Applied Physics**, vol.19, no.1, pp.55,63, Jan 1948. doi: 10.1063/1.1697872.

15. FARADAY, Technology Corporation. Available at: http://www.faraday-tech.com. Access in: 7th of March 2014.

16. FISHBURN J. et al. TILOS: a posynomial programming approach to transistor sizing. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1985. **Proceedings...** Santa Clara, CA, USA: IEEE, 1985. p. 326–328.

17. FISHER, C. et al. Optimization of standard cell libraries for low power, high speed, or minimal area designs. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1996. **Proceedings...** San Diego, CA, USA: IEEE, 1996. p. 493-496.

18. GONZALEZ, R. et al. Supply and threshold voltage scaling for low power CMOS. **Solid-State Circuits, IEEE Journal of**, vol.32, no.8, pp.1210-1216, Aug 1997. Doi: 10.1109/4.604077.

19. GREGORY, D. et al. Socrates: a system for automatically synthesizing and optimizing combinatorial logic. In: A. R. Newton. **Papers on thwenty-five years of electronic design automation**. New York: ACM, 1996. p. 1210-1216. 25 years of DAC.

20. HENDERSON, D.; JACOBSON, S.H.; JOHNSON, A. W. The theory and practice of simulated annealing. In: A. R. Glover; G. Kochenberger. **Handbook of metaheuristics.** Boston: Kluwer, 2003. p. 287-319.

21. HER, T. W. et al. Cell area minimization by transistor folding. In: DESIGN AUTOMATION CONFERENCE, EURO-DAC, 1993. **Proceedings…** Hamburg: IEEE, 1993. p. 172-177.

22. HILL, D. Sc2: A Hybrid Automatic Layout System. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1985. **Proceedings…** Santa Clara, CA, USA: IEEE, 1985. p. 172-174.

23. JEONG T. T. et al. Design Trade-Offs and Power Reduction Techniques for High Performance. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ITS APPLICATIONS, 2006. **Proceedings…** Glasgow, UK: IEEE, 2006. p. 531-536.

24. JIANG Y. et al. Technology mapping for high-performance static CMOS and pass transistor logic designs. In: VERY LARGE SCALE INTEGRATION SYSTEMS, 2001. **Proceedings…** Piscataway, NJ, USA: IEEE, 2001. p. 577 – 589.

25. KAGARIS, D. et al. Transistor-Level Synthesis for Low-Power Applications. In: QUALITY ELECTRONIC DESIGN, 8., 2007. **Proceedings…** San Jose, CA, USA: IEEE, 2007. p. 607 – 612.

26. KAPUR, P. et al. Power optimization of future transistors and a resulting global comparison standard. In: INTERNATIONAL ELECTRON DEVICES MEETING, 2004. **Proceedings…** Piscataway, NJ, USA: IEEE, 2004. p. 415- 418.

27. KEATING, M; et al,; 2007. **Low Power Methodology Manual:** For System-On-Chip Design. Springer Publishing Company, Incorporated.

28. KEUTZER, K. Dagon: technology binding and local optimization by DAG matching. In: A. R. Newton. **Papers on thwenty-five years of electronic design automation**. New York: ACM, 1996. p. 617-624. 25 years of DAC.

29. KUKIMOTO, Y. et al. Delay-optimal technology mapping by DAG covering. In: DESIGN AUTOMATION CONFERENCE, 1998. **Proceedings…** San Francisco, CA, USA: IEEE, 1998. p. 348-351.

30. LI, P. et al. A simulated annealing based technology mapping method for sequential circuits. In: INTERNATIONAL CONFERENCE ON FUTURE INFORMATION NETWORKS, 1., 2009. **Proceedings…** Beijing: IEEE, 2009. p. 303-307.

31. LIU, W., **Techniques for Leakage Power Reduction in Nanoscale Circuits**: A Survey, Department of Informatics and Mathematical Modeling, Technical University of Denmark, IMM Technical Report 2007

32. MARQUES, F.; **Technology mapping for virtual libraries based on cells with minimal transistor stacks**. Phd Thesis in Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação. 2008.

33. MARQUES, F.S. et al. DAG Based Library-Free Technology Mapping. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, 17., 2007. **Proceedings…** Streesa-Lago Maggiore, IT: ACM, 2007.

34. MICHELI, G. D. **Synthesis and Optimization of Digital Circuits**. McGraw-Hill Science/Engineering/Math; 1 edition (January 1, 1994), ISBN-10: 0070163332.

35. MISHCHENKO A.; et al., **Technology mapping with Boolean matching, supergates and choices**. ERL Technical Report, EECS Dept., UC Berkeley, March 2005.

36. MOORE, G. E. Cramming more components onto integrated circuits. **Electronics Magazine**, Vol. 38, No. 8. (19 April 1965), pp. 114-117, doi:10.1109/JPROC.1998.658762.

37. PEDRAN, M. Design technologies for low power VLSI. **In Encyclopedia of Computer Science and Technology**, Vol. 36, Marcel Dekker, Inc., 1997, pp. 73-96.

38. PEDRAN, M., **Leakage Power Modeling and Minimization**, University of Southern California, Dept. of EE-Systems, Los Angeles, CA 90089, ICCAD 2004 Tutorial, www.ceng.usc.edu, 10/10/2007

39. PEDRAN, M. et al. Low-power RT-level synthesis techniques: a tutorial. **Computers and Digital Techniques**, IEE Proceedings - , vol.152, no.3, pp. 333- 343, 6 May 2005 doi: 10.1049/ip-cdt:20045111

40. POSSER, G. **Dimensionamento de Portas Lógicas Usando Programação Geométrica**; Master Thesis in Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação. 2011.

41. RABAEY, J. M. et al. **Digital Integrated Circuits:** A Design Perspective; Prentice-Hall, 2002.

42. REIS, A. et al. Library Free Technology Mapping. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, 1997. **Proceedings…** Gramado, RS, Brazil, 1997. p. 303-314.

43. REIS, R. Design automation of transistor networks, a new challenge. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2011. **Proceedings…** Rio de Janeiro, RJ, Brazil, 2011. p. 2485 – 2488.

44. REIS, R. Physical Design Automation at Transistor Level. In: DESIGN AND TECHNOLOGY OF INTEGRATED SYSTEMS IN NANOSCALE ERA, 3., 2008. **Proceedings…** Tozeur: IEEE, 2008. p. 241-245.

45. SANTOS, C. et al. A transistor sizing method applied to an automatic layout generation tool. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 16., 2003. **Proceedings…** São Paulo, SP, Brazil: IEEE, 2003. p. 303-307.

46. SAPATNEKAR, S. S. et al. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. **IEEE Transactions on Computer Aided Design of Integrated circuits and Systems**, [S.l.], v.12, n.11, p.1621–1634, 1993.

47. SHEU, B. J. et al. BSIM: Berkeley Short-Channel IGFET Model for MOS transistors. **IEEE journal of Solid State Circuits**. New York. V.SC-22, n.4, p. 558-566, Aug. 1987.

48. SYLVESTER, D. et al. Getting to the bottom of deep submicron. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1998. **Proceedings…** San Jose, CA, USA: IEEE/ACM, 1998. p. 203-211.

49. TIWARI, V. et al. Technology Mapping for Low Power. In: DESIGN AUTOMATION CONFERENCE, 30., 1993. **Proceedings…** Dallas, Texas, USA: IEEE, 1993. p. 74-79.

50. VEENDRICK, H. J. M. Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits, **IEEE Journal**. Solid-State Circuits, vol. SC-19, pp. 468-483, Aug. 1984.

51. WESTE, N. H. E.; HARRIS, D. **CMOS VLSI Design: a circuits and systems perspective**. Boston, USA: Addison-Wesley Publishing Company, 2005.

52. XFAB, Silicon Foundries. Available at: http://www.xfab.com. Access in: 7th of March 2014.

53. ZIESEMER, A. et al. Transistor Level Automatic Layout Generator for non-Complementary CMOS Cells. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, 2007. **Proceedings…** Atlanta, GA, USA: IEEE, 2007. p. 116-121.

Appendix A: Comparison between standard cells and networks of transistors implementation.

Function	Standard Cell Implentation					Transistor Network Implementation					Reduction in:				
	TC	Average Transition Time (ns)	Average Delay (ns)	Average Dynamic Power (W)	Leakage (nW)	TC	Average Transition Time (ns)	Average Delay (ns)	Average Dynamic Power (W)	Leakage (nW)	TC	Average Transition Time	Average Delay	Average Dynamic Power	Leakage
(!((((A0 A1) A2)+B)+C))	14	0,0507449	0,052095	0,00946828	0,84074	10	0,0626034	0,0558041	0,0092566	0,461335	**29%**	**-23,37%**	**-7,12%**	**2,24%**	**45,13%**
(!((((A0+A1)+A2) B) C))	14	0,0447673	0,0487627	0,00934332	0,923039	10	0,0612669	0,0546037	0,009208	0,299208	**29%**	**-36,86%**	**-11,98%**	**1,45%**	**67,58%**
(!(((A0 A1)+B)+C))	12	0,0639686	0,062391	0,0116624	0,844204	8	0,0699389	0,0635836	0,0114404	0,461033	**33%**	**-9,33%**	**-1,91%**	**1,90%**	**45,39%**
(!(((A0 A1) A2)+B))	12	0,04661	0,0559449	0,0117293	0,793861	8	0,0662443	0,0619012	0,0113951	0,336299	**33%**	**-42,12%**	**-10,65%**	**2,85%**	**57,64%**
(!(((A0+A1) B) C))	12	0,0565432	0,058006	0,0114756	0,753128	8	0,0654068	0,0604499	0,0113714	0,295624	**33%**	**-15,68%**	**-4,21%**	**0,91%**	**60,75%**
(!(((A0+A1)+A2) B))	12	0,042687	0,0542693	0,0116304	1,08952	8	0,0688333	0,0633147	0,0113937	0,378106	**33%**	**-61,25%**	**-16,67%**	**2,04%**	**65,30%**
(!((((A0 A1) A2)+((B0 B1) B2))+C))	22	0,0354078	0,0382697	0,00685856	1,30596	14	0,0468142	0,0421056	0,0067455	0,463449	**36%**	**-32,21%**	**-10,02%**	**1,65%**	**64,51%**
(!((((A0+A1)+A2) ((B0+B1)+B2)) C))	22	0,031104	0,0360172	0,00668121	1,77218	14	0,0480732	0,0421473	0,0067398	0,372729	**36%**	**-54,56%**	**-17,02%**	**-0,88%**	**78,97%**
(!((A0 A1)+B))	10	0,0628452	0,0703786	0,0154133	0,797324	6	0,0754942	0,0732245	0,0150291	0,332709	**40%**	**-20,13%**	**-4,04%**	**2,49%**	**58,27%**
(!(((B0 B1) B2)+((A0 A1) A2)))	20	0,0299532	0,0384578	0,00788642	1,48103	12	0,0475783	0,0436959	0,0077488	0,339463	**40%**	**-58,84%**	**-13,62%**	**1,74%**	**77,08%**
(!((((A0 A1) A2)+(B0 B1))+C))	20	0,0414154	0,0437335	0,00796376	1,30943	12	0,0515724	0,0467641	0,0077911	0,462717	**40%**	**-24,52%**	**-6,93%**	**2,17%**	**64,66%**
(!(((((A0 A1) A2)+((C0 C1) C2))+((B0 B1) B2)))	30	0,0271366	0,0300969	0,00534917	2,11247	18	0,0378673	0,0337567	0,005351	0,464987	**40%**	**-39,54%**	**-12,16%**	**-0,03%**	**77,99%**
(!((A0+A1) B))	10	0,057666	0,0677164	0,015238	0,919603	6	0,0746302	0,0725196	0,0150082	0,260136	**40%**	**-29,42%**	**-7,09%**	**1,51%**	**71,71%**
(!(((B0+B1)+B2) ((A0+A1)+A2)))	20	0,0273231	0,0375555	0,00772374	1,79082	12	0,0514302	0,045484	0,0078069	0,527438	**40%**	**-88,23%**	**-21,11%**	**-1,08%**	**70,55%**
(!((((A0+A1)+A2) (B0+B1)) C))	20	0,036396	0,0409894	0,00774006	1,60226	12	0,0516118	0,0461793	0,0077675	0,313085	**40%**	**-41,81%**	**-12,66%**	**-0,35%**	**80,46%**
(!(((((A0+A1)+A2) ((C0+C1)+C2)) ((B0+B1)+B2)))	30	0,02379	0,0284858	0,00513289	2,67657	18	0,0398485	0,0341285	0,0053884	0,534928	**40%**	**-67,50%**	**-19,81%**	**-4,98%**	**80,01%**
(!((((A0 A1) A2)+(C0 C1))+((B0 B1) B2)))	28	0,0305517	0,0334997	0,00601141	1,95788	16	0,040601	0,0367744	0,0059608	0,464342	**43%**	**-32,89%**	**-9,78%**	**0,84%**	**76,28%**
(!((((A0+A1)+A2) (C0+C1)) ((B0+B1)+B2)))	28	0,0267713	0,031588	0,00575489	2,50664	16	0,042117	0,0368901	0,0059789	0,38374	**43%**	**-57,32%**	**-16,79%**	**-3,89%**	**84,69%**
(!((B0 B1)+((A0 A1) A2)))	18	0,0359638	0,0450791	0,00943264	1,4233	10	0,0527961	0,0497354	0,0092095	0,339181	**44%**	**-46,80%**	**-10,33%**	**2,37%**	**76,17%**
(!(((A0 A1)+(B0 B1))+C))	18	0,0498285	0,0507986	0,00946348	1,31289	10	0,0567901	0,0522967	0,0092455	0,461953	**44%**	**-13,97%**	**-2,95%**	**2,30%**	**64,81%**
(!((B0+B1) ((A0+A1)+A2)))	18	0,0327922	0,0438186	0,00921499	1,62089	10	0,0558428	0,0511293	0,009239	0,364639	**44%**	**-70,29%**	**-16,68%**	**-0,26%**	**77,50%**
(!(((A0+A1) (B0+B1)) C))	18	0,0438226	0,047376	0,00919062	1,43235	10	0,0548713	0,0505751	0,0092075	0,300273	**44%**	**-25,21%**	**-6,75%**	**-0,18%**	**79,04%**
(!((((A0 A1) A2)+(C0 C1))+(B0 B1)))	26	0,034921	0,0376106	0,00684394	1,91395	14	0,0441566	0,0402718	0,0067437	0,463554	**46%**	**-26,45%**	**-7,08%**	**1,46%**	**75,78%**
(!((((A0+A1)+A2) (C0+C1)) (B0+B1)))	26	0,0305961	0,0353507	0,00654546	2,33672	14	0,0447503	0,0399371	0,0067508	0,365499	**46%**	**-46,26%**	**-12,97%**	**-3,14%**	**84,36%**
(!((B0 B1)+(A0 A1)))	16	0,0449153	0,0540613	0,0116794	1,40186	8	0,0589766	0,0569525	0,0113862	0,334988	**50%**	**-31,31%**	**-5,35%**	**2,51%**	**76,10%**
(!(((A0 A1)+(C0 C1))+(B0 B1)))	24	0,0407102	0,0427283	0,00792336	1,9131	12	0,0480392	0,0443022	0,00778	0,462669	**50%**	**-18,00%**	**-3,68%**	**1,81%**	**75,82%**
(!((B0+B1) (A0+A1)))	16	0,0409583	0,052271	0,0114114	1,45097	8	0,060171	0,0573072	0,0114055	0,336495	**50%**	**-46,91%**	**-9,63%**	**0,05%**	**76,81%**
(!(((A0+A1) (C0+C1)) (B0+B1)))	24	0,0356702	0,0400197	0,00758254	2,16681	12	0,0472742	0,0432574	0,0077797	0,365041	**50%**	**-32,53%**	**-8,09%**	**-2,60%**	**83,15%**
										Average	**40,68%**	**-39,05%**	**-10,25%**	**0,53%**	**71,30%**

TC: Transistor Count