UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

CAIO GRACO PRATES ALEGRETTI

Analytical Logical Effort Formulation for Local Sizing

Thesis presented in partial fulfillment of the requirements for the degree of Doctor in Microelectronics.

Prof. Dr. André Inácio Reis
Advisor

Prof. Dr. Renato Perez Ribas
Co-advisor

Porto Alegre, June 2013.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Alegretti, Caio Graco Prates

Analytical Logical Effort Formulation for Local Sizing [manuscrito] / Caio Graco Prates Alegretti. – 2013.

84 f.:il.

Advisor: André Inácio Reis; Co-advisor: Renato Perez Ribas.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR – RS, 2013.

1. Subcircuit sizing. 2. Active area minimization. 3. Design constraints. 4. Logical Effort. I. Reis, André Inácio. II. Ribas, Renato Perez. III. Título.

ACKNOWLEDGMENTS

There are so many people I am thankful to…

First of all, I would like to express my most sincere gratitude to my advisors, professors André Inácio Reis and Renato Perez Ribas. Their advices, availability, guidance, and patience out of this world were vital for the conclusion of this work. In them, I could find a combination of technical expertise and soft/human skills that is very rare to come upon in the Engineering environment.

Working at LogiCS Labs was a very fruitful experience. I had the opportunity to share the companionship of brilliant, highly motivated, and funny colleagues. It was never ever a boring place. In particular, Vinicius Dal Bem helped a lot in the implementation of the thesis. Here goes a non-exhaustive, lexicographic list of the people I am indebted with: Alessandro Goulart, Digeorgia Natalie Silva, Felipe Marques, Felipe Marranghello, Jody Matos, Leomar Soares Jr., Mayler Martins, Nivea Schuch, Osvaldo Martinello Jr., Paulo Butzen, Rodrigo Mancuso, and Vinicius Callegaro. Either by technical discussions or random conversation, they have all contributed to my work.

Professor Paulo Fernandes, my master's advisor, introduced me to the academic research. I will always be thankful for that, even though my professional trajectory drifted me away from the brave new world of stochastic automata networks.

It is widely known that money makes the world go round. In this sense, I would like to thank the Brazilian funding agency CAPES and the Danish company Nangate A\S for the financial support, as well as my current employer, IFRS. Although there was no institutional stimulus for pursuing a doctorate degree, my colleagues at IFRS were always willing to help.

None of this would have been possible without the support and unconditional love from my family. My parents José Dante and Maria Elizabeth, as well as my brothers Tito Lívio and Francisco José, have taught me the value of reading, studying, and honest, hardworking. My wife Cristiane gave me Alexandre, by very, very far, the greatest happiness in my life. I will always be thankful for her. Just seeing Alexandre smiling, laughing, and playing makes life worth living.

Last, but not least, a perhaps unusual acknowledgment in a doctorate thesis. I would like to register my gratitude to Deolinda, my first teacher in a poor, rotten, public elementary school in a poor fisherman village. With her I learned how to read, to write, and to do basic calculations. Basic education is of foremost importance for the economic, technological, and social development of any nation. Regretfully, this is hardly ever remembered.

Formulação Analítica Baseada em *Logical Effort* para Dimensionamento Local

RESUMO

A indústria de microeletrônica tem recorrido cada vez mais à metodologia de projeto baseado em células para fazer frente à crescente complexidade dos projetos de circuitos integrados digitais, uma vez que circuitos baseados em células são projetados mais rápida e economicamente que circuitos *full-custom*. Entretanto, apesar do progresso ocorrido na área de *Electronic Design Automation*, circuitos digitais baseados em células apresentam desempenho inferior ao de circuitos *full-custom*. Assim, torna-se interessante encontrar maneiras de se fazer com que circuitos baseados em células tenham desempenho próximo ao de circuitos *full-custom*, sem que isso implique elevação significativa nos custos do projeto. Com tal objetivo em vista, esta tese apresenta contribuições para um fluxo automático de otimização local para circuitos digitais baseados em células. Por otimização local se entende a otimização do circuito em pequenas janelas de contexto, onde são feitas otimizações considerando o contexto global. Deste modo, a otimização local pode incluir a detecção e isolamento de regiões críticas do circuito e a geração de redes lógicas e de redes de transistores de diferentes topologias que são dimensionadas de acordo com as restrições de projeto em questão. Como as otimizações locais atuam em um contexto reduzido, várias soluções podem ser obtidas considerando as restrições locais, entre as quais se escolhe a mais adequada para substituir o subcircuito (região crítica) original. A contribuição específica desta tese é o desenvolvimento de um método de dimensionamento de subcircuitos capaz de obter soluções com área ativa mínima, respeitando a capacitância máxima de entrada, a carga a ser acionada, e a restrição de atraso imposta. O método é baseado em uma formulação de *logical effort*, e a principal contribuição é calcular analiticamente a derivada da área para obter área mínima, ao invés de fazer a derivada do atraso para obter o atraso mínimo, como é feito na formulação tradicional do *logical effort*. Simulações elétricas mostram que o modelo proposto é muito preciso para uma abordagem de primeira ordem, uma vez que apresenta erros médios de 1,48% para dissipação de potência, 2,28% para atraso de propagação e 6,5% para os tamanhos dos transistores.

Palavras-Chave: Dimensionamento de Subcircuitos, Minimização de Área Ativa, Restrições de Projeto, *Logical Effort*.

Analytical Logical Effort Formulation for Local Sizing

ABSTRACT

Microelectronics industry has been relying more and more upon cell-based design methodology to face the growing complexity in the design of digital integrated circuits, since cell-based integrated circuits are designed in a faster and cheaper way than full-custom circuits. Nevertheless, in spite of the advancements in the field of Electronic Design Automation, cell-based digital integrated circuits show inferior performance when compared with full-custom circuits. Therefore, it is desirable to find ways to bring the performance of cell-based circuits closer to that of full-custom circuits without compromising the design costs of the former circuits. Bearing this goal in mind, this thesis presents contributions towards an automatic flow of local optimization for cell-based digital circuits. By local optimization, it is meant circuit optimization within small context windows, in which optimizations are done taking into account the global context. This way, local optimization may include the detection and isolation of critical regions of the circuit and the generation of logic and transistor networks; these networks are sized according to the existing design constraints. Since local optimizations act in a reduced context, several solutions may be obtained considering local constraints, out of which the fittest solution is chosen to replace the original subcircuit (critical region). The specific contribution of this thesis is the development of a subcircuit sizing method capable of obtaining minimum active area solutions, taking into account the maximum input capacitance, the output load to be driven, and the imposed delay constraint. The method is based on the logical effort formulation, and the main contribution is to compute the area derivative to obtain minimum area, instead of making the delay derivative to obtain minimum delay, as it is done in the traditional logical effort formulation. Electrical simulations show that the proposed method is very precise for a first order approach, as it presents average errors of 1.48% in power dissipation, 2.28% in propagation delay, and 6.5% in transistor sizes.

Keywords: Subcircuit Sizing, Active Area Minimization, Design Constraints, Logical Effort.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AOI	AND–OR–Inverter
ASIC	Application Specific Integrated Circuit
AT	Arrival Time
CAD	Computer Aided Design
CMOS	Complementary Metal Oxide Semiconductor
CP	Convex Programming
CSP	Complementary Series Parallel
DP	Dynamic Programming
DSP	Digital Signal Processor
EDA	Electronic Design Automation
GGP	Generalized Geometric Programming
GND	Ground
GP	Geometric Programming
HDL	Hardware Description Language
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IP	Integer Programming
LE	Logical Effort
LP	Linear Programming
LSH	Locality Sensitive Hashing
LUT	Lookup Table
m	meter
MFI	Maximum FanIn
MINLP	Mixed Integer Nonlinear Program
MLE	Modified Logical Effort

MOS	Metal Oxide Semiconductor
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
n	nano (10^{-9})
NLDM	Nonlinear Delay Model
NLP	Nonlinear Programming
NMOS	*n*-type MOS transistor
NOCL	NANGATE 45nm Open Cell Library
OAI	OR–AND–Inverter
PDN	Pull-Down Network
PMOS	*p*-type MOS transistor
PUN	Pull-Up Network
RC	Resistor–Capacitor
RT	Required Time
RTL	Register Transfer Level
SCMS	Series-Connected MOSFET Structure
SDC	Synopsys Design Constraints
SOC	System on a Chip
STA	Static Timing Analysis
tdhl	delay time high to low
tdlh	delay time low to high
TILOS	TImed LOgic Synthesizer
UFRGS	*Universidade Federal do Rio Grande do Sul* (Federal University of Rio Grande do Sul)
V_{DD}	Supply voltage
V_t	threshold voltage of a transistor
V_{th}	Logic threshold voltage of a logic gate

TABLE OF CONTENTS

1 INTRODUCTION

Electronic devices become more and more present on people's daily lives, which makes out of the market of semiconductor devices an important parcel of world economy. A fair chance of good profits has attracted a great number of companies for such market, and so the electronics industry is induced to deliver products every time with higher quality standards and shorter time to market. Therefore, any way to reduce design time of integrated circuits is very appealing for this industry.

Adding to this fact the current stage of miniaturization and integration in microelectronics, which makes the design of an integrated circuit steadily more complex, it is set up scenery for a design methodology capable of dealing with these boundaries efficiently. According to Weste (2006), there are seven design methodologies currently in use:

- Structured design techniques;
- Microprocessor/DSP (Digital Signal Processor);
- Programmable Logic;
- Gate array and Sea of gates design;
- Cell-based design;
- Full-custom design;
- Platform-based design/System on a Chip (SOC).

These design methodologies take advantage of CAD (Computer Aided Design) tools, in a way to allow them to cope with the current scale of miniaturization and integration of semiconductor devices. This is a way to reduce considerably the design time (SHERWANI, 1999).

This research work is related to cell-based design. Cell-based design delivers faster, smaller and less power consuming integrated circuits than those integrated circuits produced by programmable logic or gate array. Nevertheless, cell-based design shows a higher design cost (WESTE, 2006). Compared with full-custom design, cell-based design demands much less man-hours and, therefore, it has a lower cost. Nevertheless, cell-based design generates integrated circuits that may be up to three to seven times more power consuming (CHINNERY, 2005) or three to eight times slower (CHINNERY, 2002) than equivalent integrated circuits designed via full-custom. Even though cell-based design delivers integrated circuits with inferior performance (as compared with full-custom integrated circuits), the market share of this methodology has been steadily increasing.

Full-custom design inherently generates either faster or less power consuming integrated circuits than those generated via cell-based design. This happens because, in the former methodology, a human designer can manually optimize transistor sizing,

placement and routing of devices, even the layout of the circuit. Cell-based design, in its turn, relies on a limited set of options to accomplish the circuit sizing, among other factors that restrain its capacity of generating either faster or less power consuming integrated circuits (HU, 2007).

Cell-based design must deal with discrete sizing of logic gates, since a standard cell library contains a finite number of different sizes for each cell. The problem of choosing, out of a limited number of options, the size of a cell to correspond to a node of a circuit under design is an NP-complete problem (LI, 1993; REZVANI, 2003). Besides, it is likely that the ideally-sized cell to map a given node does not belong to the current standard-cell library.

As a consequence of the factors just mentioned, human intervention in cell-based design became a common practice (ROY, 2005), especially in the nodes that represent a performance bottleneck. This thesis aims specifically at developing a subcircuit sizing method oriented towards on-the-fly cell generation, within the context of local optimization. This sizing method obtains minimum active area solutions, taking into account the maximum input capacitance, the output load to be driven, and the delay constraint. The method is based on the logical effort formulation, and the main contribution is to compute the area derivative to obtain minimum area, instead of making the delay derivative to obtain minimum delay, as it is done in the traditional logical effort formulation.

This thesis is organized as follows. Chapter 2 tackles the fundamental concepts for this research work. In chapter 3, some reference works on digital circuit sizing are shown, as well as concepts intrinsically related to this research. Next, in chapter 4, the subcircuit sizing method is developed. Chapter 5 is devoted to the experimental results obtained with the new sizing method. Finally, chapter 6 brings the conclusion.

2 BASIC CONCEPTS

In this chapter, the basic concepts needed for a better understanding of this thesis will be reviewed. This research work refers to cell-based design methodology, within the broader context of digital integrated circuits. The design of an integrated circuit, from specification to final implementation, is an intricate process that shall be divided into several stages. These stages involve well established concepts and methodologies. Therefore, initially general concepts about digital integrated circuit design will be shown. Based upon these general concepts, cell-based design flow will be tackled, since this methodology came as an answer to the growing complexity of integrated circuit design.

Next section deals with design constraints. When conceiving a digital circuit, there are specifications and design constraints that must be observed in every proposed modification to the circuit. Within design constraints, there are also cost functions that may be associated with a circuit, which are useful for optimization purposes.

After that, the concepts of delay and timing will be seen, since they have foremost importance in digital circuit design. Every digital circuit introduces a delay between its input and output signals. For a digital circuit design to reach its performance goals, these delays must be handled appropriately.

In the same fashion, the concepts regarding sizing of semiconductor devices deserve a section of its own. It is useless to have a very fast circuit that does not respect either area or power consumption limitations, hence the need for adequate sizing.

The final section is devoted to the mathematical tools related to the resolution of the proposed problem. There are different ways to model the digital circuit sizing problem, and for every model there may exist one or more mathematical techniques for its resolution. Grosso modo, these techniques may be divided into mathematical programming and analytical resolution of minimization problems, as reviewed in the end of this chapter.

2.1 General Concepts about Digital Integrated Circuit Design

In this section, concepts referring to integrated circuit design are reviewed. Some of these concepts are consensually defined in the literature, meanwhile others have different definitions according to different authors, and there are even concepts that have not been formally defined in the consulted literature. Therefore, in this section, concepts alluding to integrated circuit design are either reviewed or defined, in a way to conceive a consistent set of definitions to be referred to throughout this work.

The first concept to be reviewed is that of *logic networks*. Logic gates may be organized into *logic networks*, which are sets of logic gates that implement a non-elementary logic function. A *transistor network* is an array of transistors capable of implementing a logic function, in which these transistors are not necessarily organized into logic gates. For a given logic function, different logic networks and transistor networks may be implemented. Each of these networks may belong to different logic styles (RABAEY, 2003) — also known as circuit families (WESTE, 2006) —, as shown in Figure 2.1. Logic networks and transistor networks differ from each other by logic style and topology. Just to mention two examples, the logic style CSP CMOS — *Complementary Series Parallel* CMOS —is composed of logic gates, meanwhile the *pass transistor* logic style is composed of structures that are not organized as basic logic gates. Therefore, this logic style is based on transistor networks.

This research work deals only with *combinational digital circuits*, i.e., circuits whose outputs are functions exclusively of current inputs. Therefore, these are memoryless circuits.

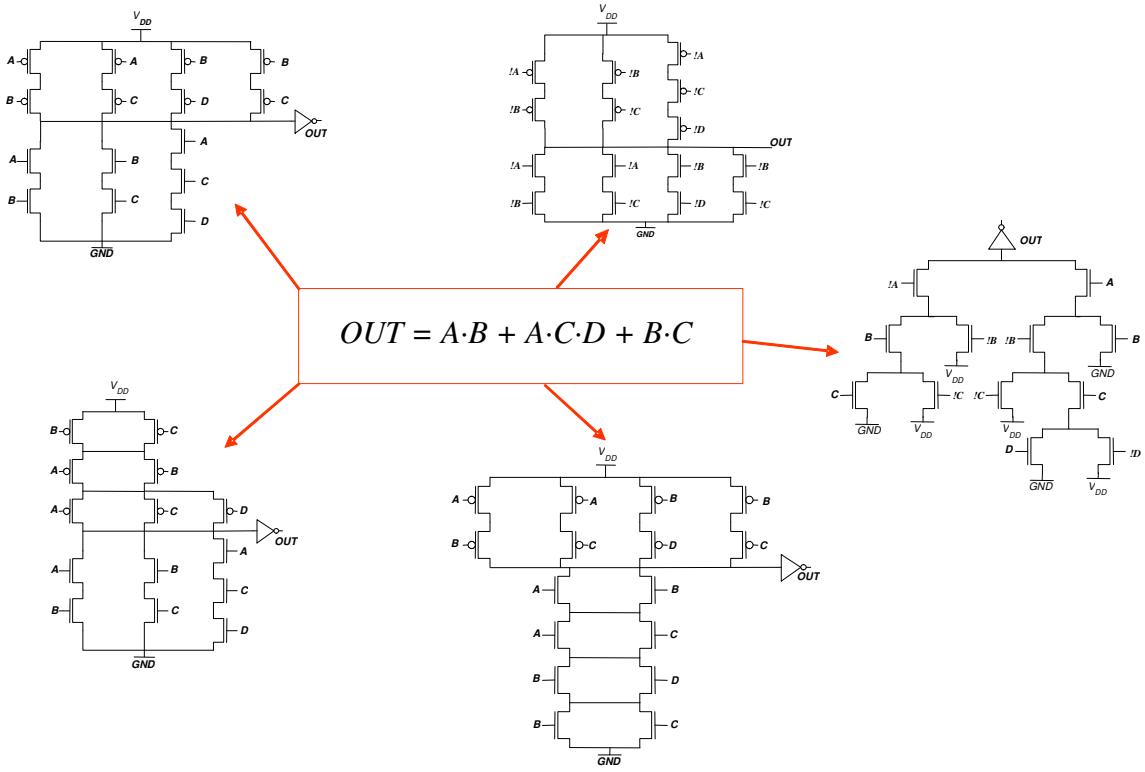
A concept closely related to logic gate is that of *cell*, although there is a difference between these two concepts: Logic gate refers to the logic function under implementation, its Boolean equation and truth table. In turn, cell alludes to physical aspects of implementation, such as layout, etc. That is, the concept of logic gate is related to logic functionality, meanwhile the concept of cell is related to electrical aspects and physical implementation. For instance, an inverter logic gate may correspond either to a small-sized inverter cell or to an inverter cell capable of driving a large capacitance in its output. In short, according to Gajski-Kuhn's Y diagram (WESTE, 2006), cell belongs to physical domain, and logic gate belongs to structural domain. A cell provides a Boolean logic function (e.g., inverter, AND, NAND, OR, NOR, XOR, XNOR, AOI, OAI, adder, multiplexer) or storage function (latch, flip-flop). Therefore, it can be stated that the cell is the basic unit of organization of a digital circuit designed under cell based design.

Cells are conceived by experienced designers, who spend a long time optimizing each cell and taking full advantage of layout properties. Besides, each cell must be verified and characterized individually, which is a time demanding activity. However, once concluded the design, the cell can be reused several times, without need for redesign.

Logic threshold and drive strength are two of the characteristics that define the functioning of a cell. *Logic threshold* (V_{th}), also named input threshold or switching threshold, is the voltage that, when applied to the input of a cell, causes the output voltage V_{out} to be equal to the input voltage V_{in} (RABAEY, 2003; WESTE, 2005). *Drive strength* (or *driving strength*) is the ability of the cell in charging or discharging a given capacitance in its output. This drive strength is directly related to the size of the transistors in the cell. Logic threshold may be used in the calculation of a cell propagation delay. Drive strength is very important in the sizing of digital integrated circuits, as shown throughout this work.

Usually, cells belong to a *standard cell library* (or cell library). This library is a set of cells with compatible layout template, in which the cells provide different logic functions and storage functions. These cells are used to map a given logic network, according to the cell-based design methodology. Table 2.1 shows an example of a standard cell library.

Figure 2.1 – Different logic and transistor networks for the same logic function



Source: Logics (2013).

Designers find the cell-based design methodology appealing because the power distribution and compatibility among neighbor cells is greatly improved by using a compatible layout template for all cells. This compatible layout template forces the cells to have compatibility with neighbor cells by using power lines (V_{DD} and *GND*) at compatible positions, so that placement and routing can be done automatically. Cells are organized in rows, and these rows may be separated one from another by routing channels.

Usually, a standard cell library contains cells with different sizes that implement the same logic function. X1 stands for the minimum sized cell, X2 represents the cell twice as big as X1 cell, and so forth. Larger cells have greater drive strength and therefore are faster, but at the cost of greater power consumption. On the contrary, smaller cells have less drive strength and therefore are slower, but they consume less power.

The concepts of template and scale factor are needed for a better understanding of the subject of cell size. The *template* of a cell is the minimum sized version of this cell, which defines the cell topology and the ratio between the sizes of its transistors. Each and every cell obtained from this template has the same topology and the same transistor size ratios. An example of a NOR2 cell template is shown in Figure 2.2a. It is worthy to emphasize that the current concept of template is related to neither layout aspects nor physical implementation aspects. Therefore, the current concept of template shall not be confused with the notion of layout template mentioned in the definition of standard cell library. This latter sort of template refers solely to physical implementation

aspects. Every standard cell in a library has an exclusive layout template, but several cells may have the same template.

Table 2.1 – Example of a standard cell library

Gate type	Variations	Options
Inverter, buffer, tristate buffer		1X, 2X, 4X, 8X, 16X, 32X minimum size inverter
NAND, AND	2-8 inputs	High, normal, low power
NOR, OR	2-8 inputs	High, normal, low power
XOR, XNOR		High, normal, low power
AOI, OAI		High, normal, low power
Multiplexer	Inverting, non inverting	High, normal, low power
Schmitt trigger		High, normal, low power
Adder, half adder		High, normal, low power
Latch		High, normal, low power
Flip-flop	D, with and without synch/asynch set and reset, scan	High, normal, low power
I/O pad	Input, output, tristate, bidirectional, boundary scan, slew rate limited, crystal oscillator	Various drive levels (1-16 mA) and logic levels

Source: Weste (2006, p. 426).

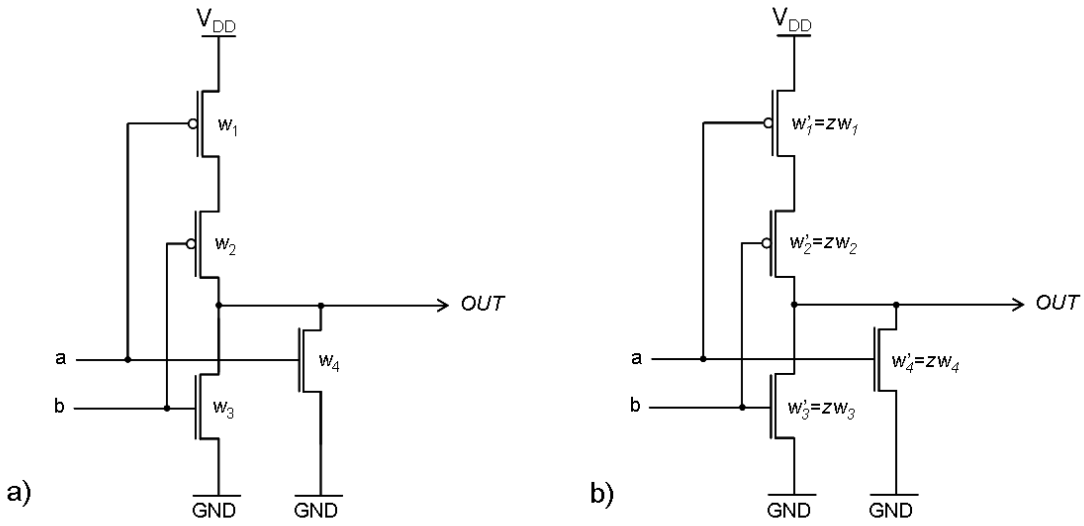
The *scale factor* (BOYD, 2005; HEDLUND, 1987) is the number greater than or equal to 1 that multiplies the widths of the transistors in the template in order to obtain the corresponding cell with the expected drive strength. Figure 2.2 shows two instances of a NOR2 cell: a minimum sized cell (i.e., *template* — Figure 2.2a) and another cell with size z. In this case, the cell depicted in Figure 2.2b has a scale factor equal to z. These two cells have different sizes, but the transistor sizes in each cell keep the same ratio between each other, i.e., $w_1 : w_2 : w_3 : w_4$, where w_i, $i \in \{1, 2, 3, 4\}$, is the width of every transistor in the template.

Although the scale factor may in principle assume any value greater than or equal to 1, usually the standard cells in a library have scale factors given by natural numbers. In this work, this set of standard cells in a library, which have the same template and that differ from each other just by their scale factors, is named *supercell*[1]. As examples of supercells, one can mention: a) the supercell constituted by the six inverter cells in the library depicted in Table 2.1; b) the supercell given by the two NOR2 cells in Figure 2.2

[1] This definition of supercell is different from the one given by Zhou (2007).

and c) the three supercells depicted in Figure 2.3, each of which is composed by three cells.

Figure 2.2 – Scale factor of a NOR2 cell



Source: Logics (2013).

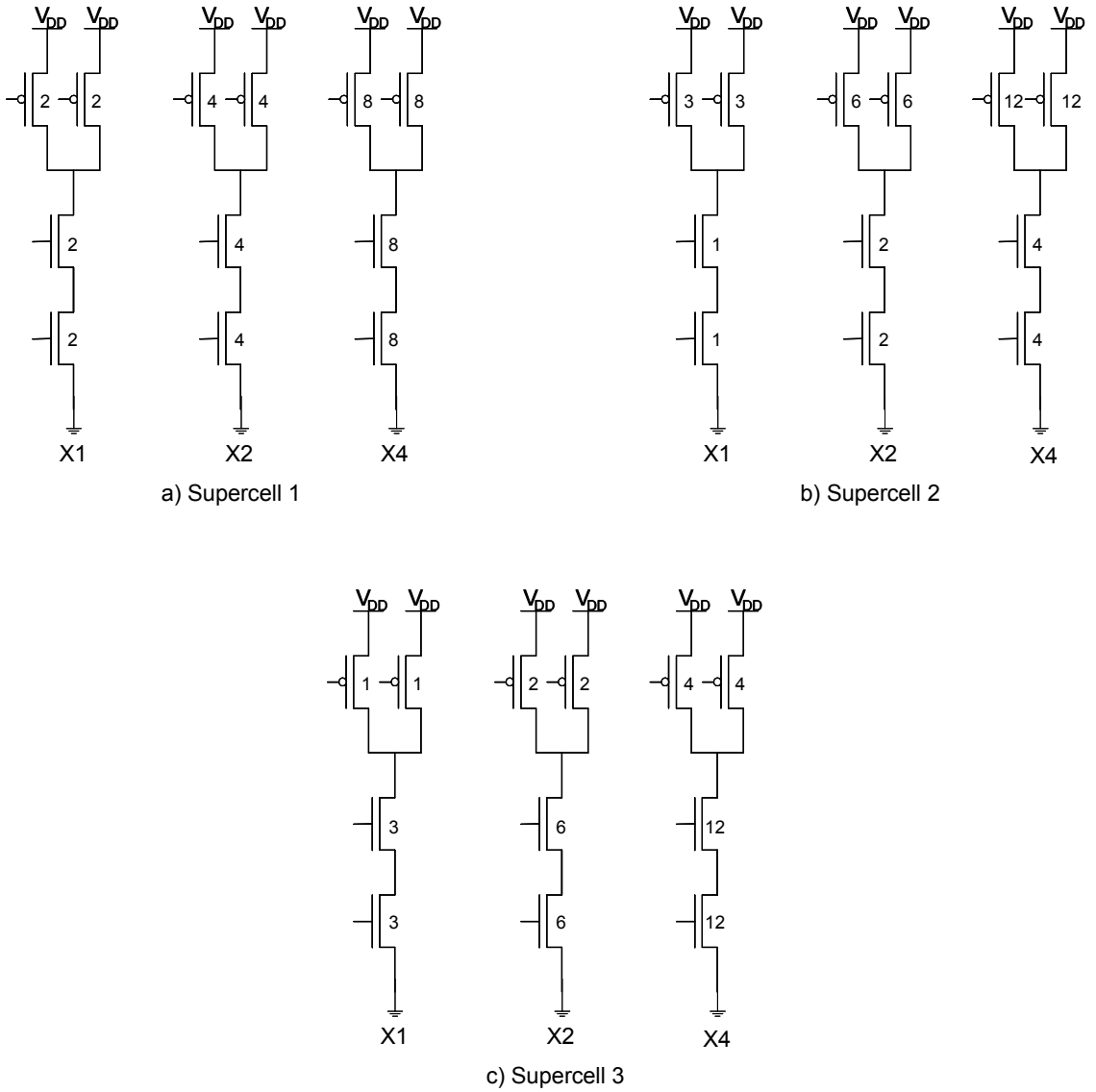
In Figure 2.3, the numbers in each transistor represent the width of the transistor channel (i.e., its size), parameterized with respect to a reference value (e.g., the channel width of the smallest NMOS transistor that can be implemented with the technology). Although the three supercells implement the same logic function, they have different templates. This can be observed in the ratio between PMOS and NMOS transistor sizes in each supercell: in supercell 1, this proportion is equal to 1:1; in supercell 2, it is worth 3:1; and in supercell 3 it is equal to 1:3. This example shows that there may exist standard cells (with the same logic function) that show differences among themselves other than their sizes (drive strengths). In other words, there are standard cell libraries with only one supercell for every logic function implemented, and there are also standard cell libraries with more than one supercell for a given logic function.

In earlier times, there were a few sizes for every template in a standard cell library. Nowadays, the cell size may vary from X1 to X16. As represented in Figure 2.4, there is a huge difference between an X1 cell and an X16 cell. This difference is reflected in the delay, area, and power consumption of each cell.

With respect to the amount of standard cells, there are libraries ranging from just tens of standard cells (ROSA JUNIOR, 2008) up to thousands of standard cells (SHERWANI, 1999; TRIHY, 2008). Regarding the amount of standard cells in a library, there is a tradeoff that must be respected: the bigger the number of logic functions and the number of different options for every logic function implemented as standard cells, the easier the obtention of a cell with the desired logic function and a close to ideal size for a given node of the circuit under design. However, as more

standard cells are added to the library, the runtime of the cell chooser algorithm increases polynomially and there is also the additional work of designing, verifying, and characterizing extra standard cells. In the literature, there are authors who plead libraries with a few standard cells — e.g., Ricci (2007), Seo (2008) —, and there are also those who defend libraries with many standard cells — e.g., Berkelaar (1988), Correia (2004), Gavrilov (1997), Keutzer (1987), Marques (2007), Scott (1994), Sechen (1996) —, with no consensus whatsoever about this matter.

Figure 2.3 – Examples of supercells for NAND2 logic function

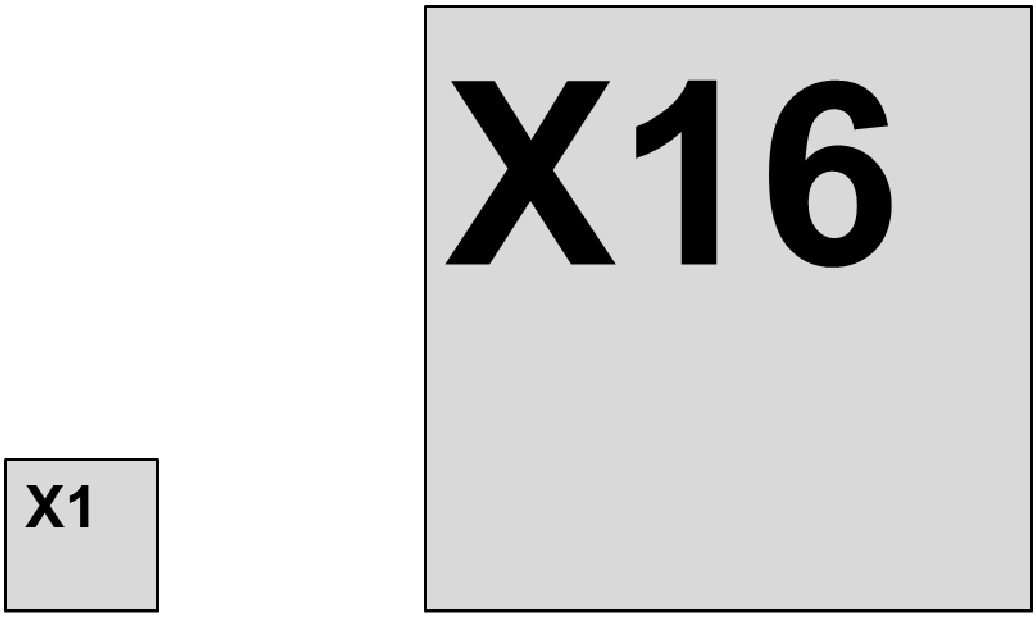


Source: Logics (2013).

All cells in a standard cell library have been previously verified and characterized. Therefore, the designer can take its correct functioning for granted. However, the designer is limited to using in his (her) project only the cells made available by the standard cell library. This is indeed a limiting factor in the cell-based design methodology, which is tackled in this research work.

Technology mapping is the process of expressing a logic network in terms of cells chosen from a standard cell library. Usually, technology mapping aims at the optimized usage of standard cells in order to implement a circuit that obeys given timing constraints, with a minimum area. In its most generic fashion, technology mapping is free to choose both the logic functionality and the size of the standard cells, out of the available sizes in the standard cell library (HU, 2003; KARANDIKAR, 2004). However, in some particular cases, technology mapping just determines the logic functionality of the cells (BERKELAAR, 1988; CORREIA, 2004; MARQUES, 2007) and, in some other cases, technology mapping can choose just the size of the cells (COUDERT, 1997, 2002; HU, 2009).

Figure 2.4 – Current difference in cell sizes



Source: Logics (2013).

The concepts of standard cell, standard cell library, and technology mapping are used in the explanation of cell-based design methodology. Prior to that, however, it is necessary to differentiate the concepts of full-custom circuits, ASIC circuits, and cell-based circuits, since there are in the literature conflicting definitions about this terminology.

There is a strict relationship between the design methodology and the integrated circuit thus produced. By definition, a *full custom circuit* is the one produced via full custom design methodology (also named custom design methodology). A full custom circuit is also named custom circuit. A characteristic of full custom design methodology is to give the designer a wide freedom of action. The designer may conceive each device without interfering in the design of the other devices. If necessary, the designer may act directly into the integrated circuit layout, by manually designing, positioning, and routing the devices. The full custom methodology design is suited for the design of

high performance integrated circuits, such as microprocessors and DSP processors. However, such a high performance is obtained because designers work for a long time in the project, which raises the costs.

The definition of ASIC circuits is somewhat different from the other categories. According to Chinnery (2002), strictly speaking, the expression "*Application Specific Integrated Circuit*" as well as the acronym ASIC refer to an integrated circuit designed for a particular application. As examples of ASIC circuits, one can cite: an IC for a speaking doll; an IC for military gear; an IC to interface the memory and the microprocessor of a workstation (SMITH, 1997).

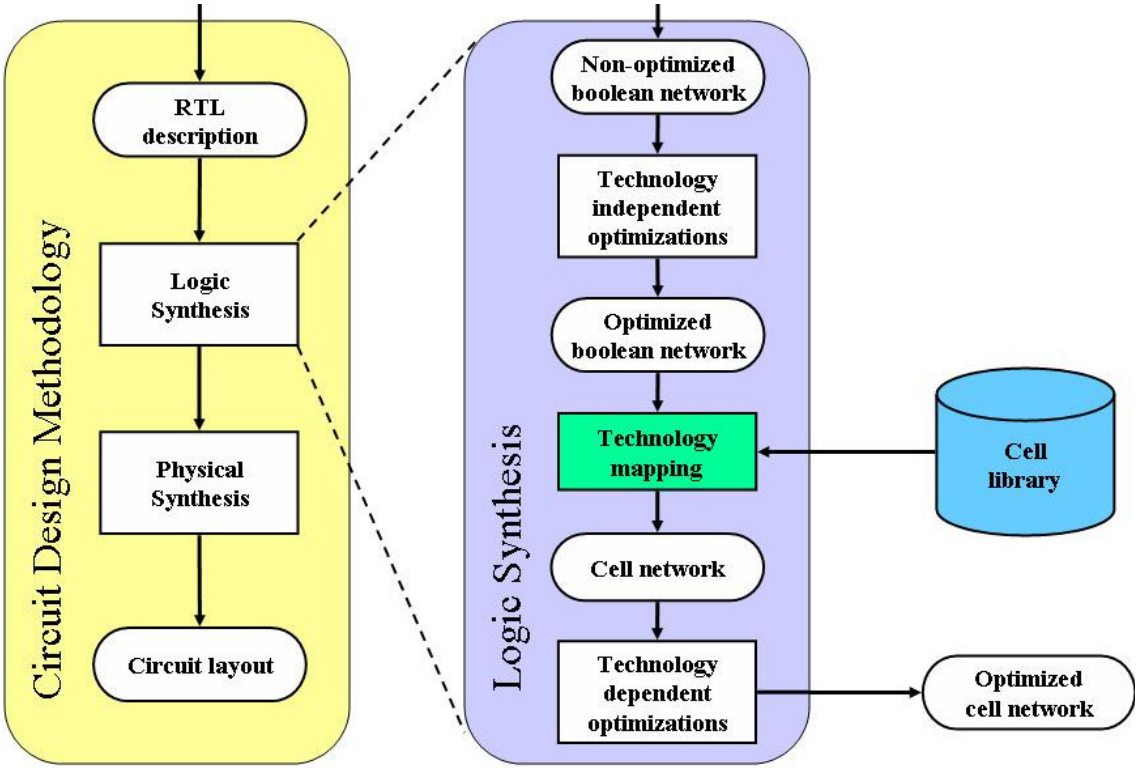
There is no consensus in the literature about the utilization and real meaning of the concepts "ASIC circuit", "full custom circuit", and "cell-based circuit". Although the notion of ASIC circuit refers just to the purpose with which a given integrated circuit was conceived – and not to the design methodology used (cell-based design, full custom design etc.) –, most of the times the acronym ASIC refers to an integrated circuit designed via cell-based design methodology (CHINNERY, 2002). This confusion is due to the fact that, according to Sapatnekar (1993-a), cell-based design methodology is the most widely used methodology in the design of ASIC circuits. However, just like there are ASIC circuits conceived via cell-based design methodology, there are also ASIC circuits conceived under full custom methodology. This way, for the sake of coherence, in this research work only cell-based and full custom are referred to henceforth.

That said, the next topic is the definition of *cell based design methodology*. In this methodology, the circuit under design is mapped to a standard cell library, and the placement and routing are done automatically. Figure 2.5 depicts the cell-based design flow. By *design flow*, it is meant a set of procedures that allow the designer to start from the specification of an integrated circuit and come to the error-free, final implementation of this circuit (WESTE, 2005).

The design starts with its behavioral description in HDL (Hardware Description Language). From this description, logic synthesis generates the circuit netlist, as follows. The RTL (Register Transfer Level) description is interpreted as a Boolean network not necessarily optimized. This network goes through technology-independent optimizations, which use properties of logic functions to generate an optimized Boolean network. Next, technology mapping is done. As result of such mapping, the circuit is now described as a cell network, which goes through technology-dependent optimizations, in order to generate the final, optimized cell network. At this moment, logic synthesis happens: based on the optimized cell network, and on the processes of floorplanning, placement and routing, all the information needed to completely specify the circuit layout is obtained (RABAEY, 2003).

The main objective of cell based design is to reduce implementation costs, by reusing a standard cell library. In this methodology, the cells need to be designed, verified, and characterized only once for a given technology. This is a great advantage of this methodology, because the cells may be reused, thus reducing design cost (RABAEY, 2003). Besides, once the standard cell library is ready to be used, the design time of a new circuit is considerably reduced (as compared with full-custom integrated circuits). The design time is shortened even further, because placement and routing is done automatically in this methodology.

Figure 2.5 – Cell based design flow



Source: Schneider (2007).

2.2 Design Constraints

Regardless of the design methodology in use, there are design constraints to be taken into account. In terms of engineering design, a *design constraint* refers to a characteristic that the design must possess. According to Design (2008), design constraints are declarations that define the design goals in terms of measurable characteristics of the circuit, such as timing, area, and capacitance. Typically, design constraints state that a circuit must have a delay less than or equal to a given maximum delay, a maximum input pin capacitance (MFI — *Maximum FanIn*), and a given load to be driven by output pin (C_{out}).

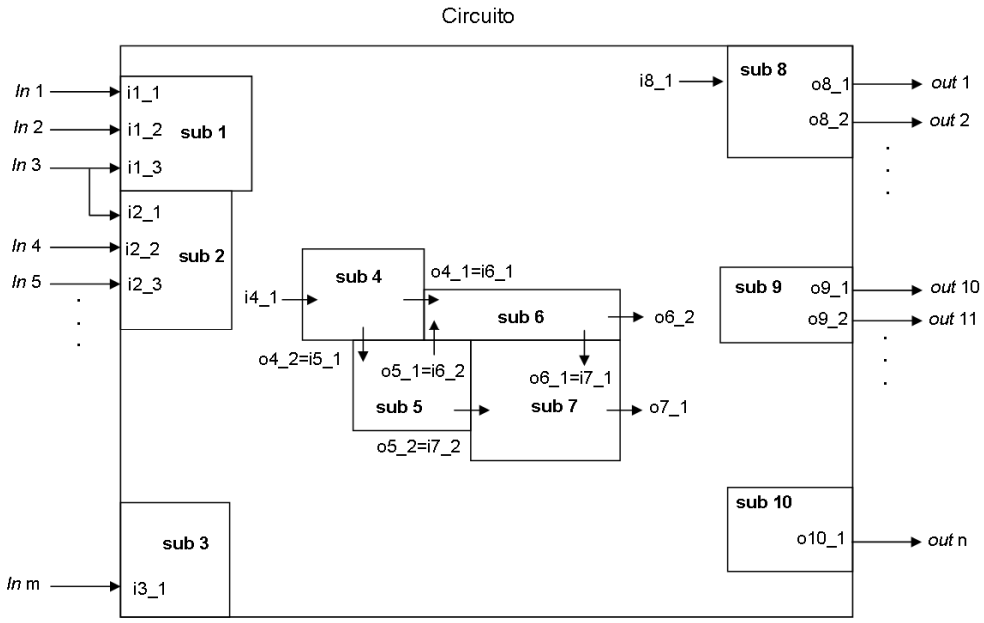
In this research work, a design constraint is regarded as an objective or characteristic to be attained by the circuit. For instance, if a given circuit path has a delay constraint d_1, but the circuit is optimized in such a way that this path comes to have a delay $d_2 < d_1$, then the delay constraint is updated to d_2. The same applies to the other design constraints.

Design constraints may also contain directives for the optimization process. These directions point to the costs to be reduced in this process. Usually, some constraints (e.g., maximum input pin capacitance and maximum delay) must be respected, meanwhile some costs (such as area and power consumption) shall be minimized.

When analyzing or designing a circuit, attention can be focused on a small piece of the global circuit, named subcircuit. Just like the global circuit has global design constraints, each subcircuit has its own local design constraints. In the same fashion as there is a relationship between the subcircuit and the global circuit it belongs to, there is

also a relationship between the corresponding global and local design constraints, as depicted in Figure 2.6. The global circuit has m primary inputs (in_1, in_2, ... , in_m) and n primary outputs (out_1, out_2, ... , out_n). Suppose this circuit is divided into subcircuits, out of which ten are emphasized in Figure 2.6. Each subcircuit sub_j has inputs i_{j_k} and outputs o_{j_l}, where $j \in \{1, 2, 3, ... ,T\}$, $k, l \in \{1, 2, 3, ..., U\}$, T is the total number of subcircuits and U is the maximum number of inputs or outputs of the subcircuits.

Figure 2.6 – Local and global design constraints



Source: Logics (2013).

Figure 2.6 shows that there is a strict relationship between the maximum input pin capacitance of the global circuit and the maximum input pin capacitances of the subcircuits in the input of the global circuit. This relationship is expressed in the following set of equations:

$$MFI(in_1) = MFI(i_{1\_1}) \tag{2-1}$$
$$MFI(in_2) = MFI(i_{1\_2})$$
$$MFI(in_3) = MFI(i_{1\_3}) + MFI(i_{2\_1})$$
$$MFI(in_4) = MFI(i_{2\_2})$$
$$\vdots$$
$$MFI(in_m) = MFI(i_{3\_1})$$

The maximum capacitance of the input pin in_3 of the global circuit is given by the sum of the maximum capacitances of two inputs of two subcircuits — $MFI(i_{1\_3})$ and $MFI(i_{2\_1})$ — due to the bifurcation in in_3 that drives these two inputs. The set of

equations (2-1) shows that an alteration in the maximum input capacitance of the global circuit implies the same alteration in the maximum input capacitance of the subcircuit in the input of the global circuit and vice versa. When there is a bifurcation in the input signal, a modification in the corresponding maximum input capacitance of the global circuit may be spread over the maximum input capacitances of one or more subcircuits.

With respect to the load to be driven by the output pins, there is always a biunivocal correspondence between the capacitances of the output pins of the global circuit and the capacitances of the output pins of the subcircuits in the output of the global circuit. Let C_x be the capacitance of a generic pin x:

$$\begin{aligned} C_{out1} &= C_{o8\_1} \\ C_{out2} &= C_{o8\_2} \\ &\vdots \\ C_{out10} &= C_{o9\_1} \\ C_{out11} &= C_{o9\_2} \\ &\vdots \\ C_{out\,n} &= C_{o10\_1} \end{aligned} \tag{2-2}$$

The set of equations (2-2) shows that an alteration in the capacitance to be driven by the output pin of the global circuit implies the same alteration in the capacitance to be driven by the output pin of the subcircuit in the output of the global circuit and vice versa.

In its turn, the global design constraint on maximum delay in every path (connecting an input to an output of the global circuit) is equivalent to the sum of the local design constraints on maximum delays of the subcircuits that constitute this global path. An alteration in the global design constraint on maximum delay implies alterations in one or more (or even all) local design constraints on maximum delay. In the same fashion, an alteration in a local design constraint on maximum delay implies the same alteration in the global design constraint on maximum delay. Mathematically speaking, let P_x be an arbitrary path between an input and an output of the global circuit; let $d(P_x)$ be the global design constraint on maximum delay of such path. The relationship between the global and local design constraints on maximum delay is given by:

$$d(P_x) = \sum_{i \in P_x} d_i \tag{2-3}$$

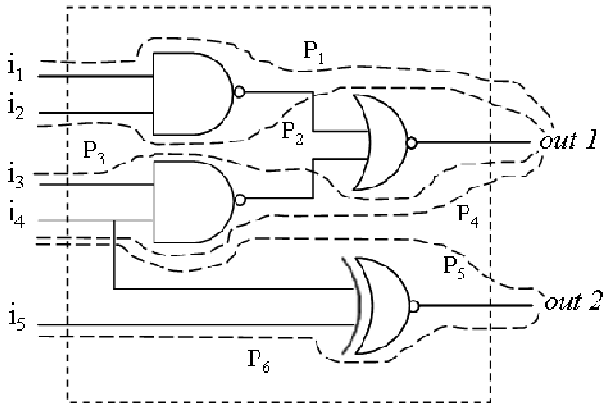
where d_i denotes the local design constraint on maximum delay of the path within subcircuit i that belongs to global path P_x. A subcircuit may have several inputs and outputs (as seen in Figure 2.7) and therefore several local design constraints on maximum delay, but for equation (2-3) only the path within subcircuit i that belongs to global path P_x matters.

Now let us suppose that the global path P_x has been optimized, its maximum delay has changed from $d_1(P_x)$ to $d_2(P_x)$, where $d_2(P_x) = d_1(P_x) - \delta$ and $\delta > 0$. The outcome of this optimization is a variation $\delta_i > 0$ on the local design constraint on maximum delay of the subcircuit i that belongs to global path P_x, so that

$$\sum_{i \in P_x} \delta_i = \delta \tag{2-4}$$

That is, the global path optimization may involve the optimization of one or more subcircuits that belong to such path. In the case of a subcircuit with more than one output, only the path within the subcircuit that belongs to the global path needs to be optimized.

Figure 2.7 – Subcircuit with several paths (P_1, P_2, ..., P_6) between input and output



Source: Logics (2013).

With respect to power consumption, an alteration in the power consumption of a subcircuit implies the same alteration in the power consumption of the global circuit. As for the area, an alteration in the area of a subcircuit is also reflected in the area of the global circuit, as long as layout aspects of the global circuit do not dim or even nullify this area variation. When the area or power consumption of the global circuit is minimized, this minimization is spread over the subcircuits in a similar way to equation (2-4).

2.2.1 Example of design constraint specification format

The SDC format (Synopsys Design Constraints) is a popular design constraint specification format. Several companies use this format in their EDA (Electronic Design Automation) tools as a means of communicating design intentions (USING).

In SDC format, design constraints are divided in two categories: design rule constraints and optimization constraints (also known as user specified constraints). Design rule constraints are due to the cell-based design methodology. The synthesis tool uses a standard cell library, and for every standard cell in the library there is a set of design rules that must be rigorously obeyed by this synthesis tool. Examples of design rule constraints are:

- Maximum transition time of the cell input signal, also referred to as slew;
- Maximum capacitance: maximum capacitive load that a cell output is capable of driving;

- Cell degradation: some standard cell libraries supply cell degradation tables, which indicate the maximum capacitance that a cell is capable of driving as a function of the transition time of its input signals (SYNOPSYS-b).

Optimization constraints are designer-made specifications that define design intentions such as timing, area, and power consumption. These constraints act as a guide for the synthesis tool to do its task. Examples of optimization constraints are:

- Input and output delays, which restrict the delays of the external paths in the borders of the circuit under design. The input delay specifies the path delay between an external input signal and the first register in the circuit, meanwhile the output delay specifies the delay between the last register in the circuit and its outputs;
- Minimum and maximum path delays, which specify, for every path between an input and an output, its timing constraint;
- Total circuit area (SYNOPSYS-b).

The synthesis tool tries to respect both design rule constraints and optimization constraints, but the former constraints have precedence over the latter ones. That is, some optimization constraints may be sacrificed so that all design rule constraints are met.

Figure 2.8 shows parts of a SDC file for the design of a ripple carry adder (DESIGN COMPILER). In this example, the clock signal has a period of 4 ns, pulse ratio 50/50, uncertainty 0.1 ns, latency 0.2 ns, and transition time 0.1 ns. The commands (attributes) `set_dont_touch_network`, `set_dont_touch`, and `set_ideal_network` are directives for the design. The command `set_driving_cell` specifies the cells that drive the adder. The command `set_load` specifies the load that must be driven by the adder. The commands `set_input_delay` and `set_output_delay` specify the time the input signal takes to arrive at the adder input and the time it takes to pass through the external block until the end of the logic path. The difference between these two delays is the amount of time available for the adder internal logic. Finally, the last design constraint sets the maximum area allowed for the design.

Figure 2.8 – Example of part of SDC file

```
create_clock -name "clk" -period 4 -waveform {0 2} {clk}
set_clock_uncertainty 0.1 clk
set_clock_latency 0.2 clk
set_clock_transition 0.1 clk
set_dont_touch_network clk
set_dont_touch rst_n
set_ideal_network rst_n
set_driving_cell -library umcl18g212t3_tc_180V_25C -lib_cell
HDDFFPB1 -pin Q [get_ports a]
set_driving_cell -library umcl18g212t3_tc_180V_25C -lib_cell
HDINVD1 -pin Z [get_ports b]
set_load [load_of umcl18g212t3_tc_180V_25C/HDDFFPB1/D]
[get_ports s]
set_input_delay 0.67 [get_ports b] -clock clk
set_output_delay 0.5 [get_ports s] -clock clk
set_max_area 1000
```

Source: DESIGN COMPILER.

2.3 Delay

In this section, the concepts related to signal propagation delay in digital integrated circuits are seen. Regardless of the logic path implemented as a digital integrated circuit, there is always a delay between the input and output signals, due to the electrical charge propagation through resistive and capacitive components of the circuit. In its current stage of miniaturization, microelectronics industry delivers integrated circuits with transistors whose channel length is equal to 22 nm (e.g., processors Intel Core i5 and i7). Regarded as semiconductor switching devices, these transistors change state very quickly, thanks to their tiny dimensions. Therefore, these devices can deliver a very high maximum operation frequency. However, for an integrated circuit to reach a good effective operation frequency, signal propagation delay in this circuit must be kept under control. The concepts related to delay and timing are seen in this section. Three delay models are also reviewed, namely, Elmore delay model, the gain-based logical effort delay model and the nonlinear delay model (NLDM). Elmore delay is a classic model, still used in recent sizing algorithms — e.g., Hu (2007) —, meanwhile the logical effort delay model was adopted in this research work. The NLDM is mentioned as an example of an empirical model, for the sake of comparison with the former models, which are theoretical delay models.

2.3.1 Fundamental concepts

The first concept worth mentioning is *transition time*, which encompasses the concepts of *fall time* (t_f) and *rise time* (t_r). In this research work, fall time corresponds to the time a waveform takes to fall from 90% to 10% of its steady value. In its turn, rise time corresponds to the time a waveform takes to rise from 10% to 90% of its steady value.

The second concept is *propagation delay* (t_d) — or *delay* (SAPATNEKAR, 2004), for short —, which is defined in this research work as the maximum time interval which starts when the input signal (whose transition causes a transition in the output) reaches $V_{DD}/2$ to the time the output signal reaches the same voltage $V_{DD}/2$. V_{DD} is the supply voltage, corresponding to logic level 1.

Since the propagation delay uses to be different for rise and fall transitions of the output signal of the cell, there are two distinct propagation delays: tdlh — *delay time low to high* and tdhl — *delay time high to low*, respectively.

The *arrival time* (*AT*) denotes the time a signal takes to propagate from the primary inputs until a given node. In its turn, the *required time* (*RT*) represents the time the signal must arrive at a given node, so that the local delay design constraint is satisfied. The *slack S* is defined as:

$$S = RT - AT \tag{2-5}$$

The *critical path* of a circuit is the logic path between a primary input and a primary output that shows the largest delay.

Timing analysis consists in computing the delay in a path of a digital circuit, in order to verify if the timing constraints imposed by the remaining of the circuit are satisfied (NOWE, 2003). *Timing closure* represents the timing analysis of an entire circuit (RABAEY, 2003). When a circuit path does not comply with the timing constraints, it is said that a *timing violation* occurs.

Theoretical delay model denotes the closed form equationing that gives the propagation delay of a logic gate, based on the electrical description of this gate. For instance, there is the Elmore delay model (ELMORE, 1948) and the logical effort delay model (SUTHERLAND, 1999). Besides these theoretical (physical) delay models, there are also empirical delay models, based upon experimental data. Initially, the theoretical delay models are reviewed.

2.3.2 Elmore delay model

The Elmore delay model is a classic method for estimating the delay in circuits, when the circuit is excited by a step function. It is a not very precise model, in which the delay is computed on RC networks, and transistors are modeled as resistances. The propagation delay (t_d) for the response of the circuit to this step function may be approximated by the first momentum of the response of the circuit to the impulse function:

$$t_d = \int_0^\infty te'(t)dt \tag{2-6}$$

where: t – time.
$e(t)$ – response of the circuit to the step function.
$e'(t)$ – derivative of the response of the circuit to the step function (i.e., response of the circuit to the impulse function).

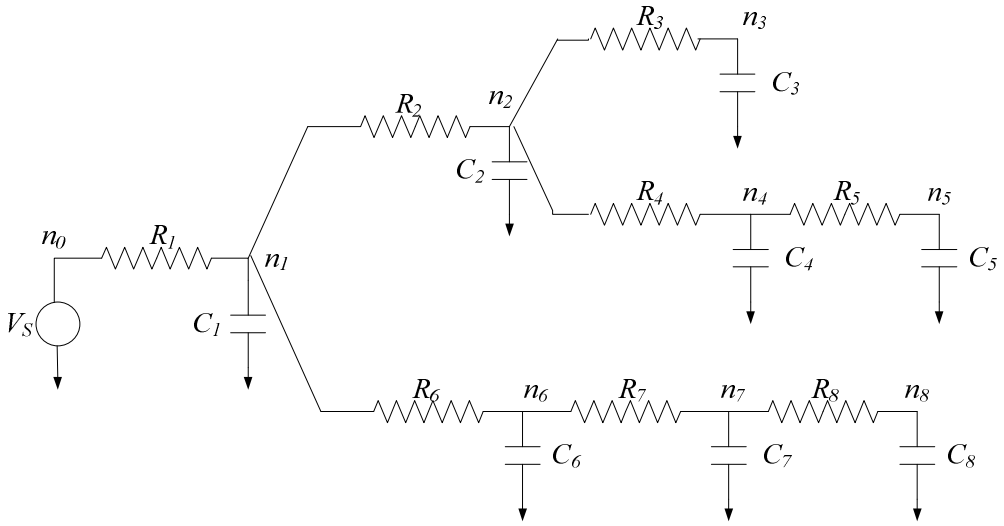
This is the formal definition of the Elmore delay. Obtaining the response of a circuit to the impulse function, as defined, may be an arduous task. Nevertheless, when the objective is to evaluate the delay in an *RC* tree, the Elmore delay becomes friendlier. According to Rubinstein (1983) and Sapatnekar (2004), an *RC* tree – depicted in

Figure 2.9 – is a connection of resistors structured as a tree, with two restrictions: a) a capacitance connected to any node in the tree must be grounded; and b) there is no resistor grounded. An *RC* tree thus defined shows two fundamental characteristics: there are no resistive loops and, if a node other than the ground is the input, then there is only one path from this input node to any other non-ground node in the tree. Besides, if two *RC* trees with the same ground are connected by a non-ground node, then they form a new *RC* tree.

The Elmore delay in node n_i of an *RC* tree may be obtained as follows. Let P_i the path between input node n_0 and n_i, which may be any tree node other than ground and input node. Analogously, let P_j be the path between n_0 and a node n_j. Let $P_{ij} = P_i \cap P_j$ be the part of the path between n_0 and n_i that is common with the path between n_0 and n_j. For convenience of notation, P_{ij} also denotes the set of resistances that belong to the path P_{ij}. The Elmore delay between the input node n_0 and the node n_i of the RC tree is given by:

$$T_{D_i} = \sum_{j=0}^{n} C_j \sum_{k \in P_{ij}} R_k \tag{2-7}$$

Figure 2.9 – Example of *RC* tree



Source: Sapatnekar (2004).

For example, in

Figure 2.9:

$$T_D(n_4) = R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2)C_3 + (R_1 + R_2 + R_4)C_4 + (R_1 + R_2 + R_4)C_5 + R_1C_6 + R_1C_7 + R_1C_8 \quad (2\text{-}8)$$

$$T_D(n_8) = R_1C_1 + R_1C_2 + R_1C_3 + R_1C_4 + R_1C_5 + (R_1 + R_6)C_6 + (R_1 + R_6 + R_7)C_7 + (R_1 + R_6 + R_7 + R_8)C_8 \quad (2\text{-}9)$$

2.3.3 Logical effort delay model

The logical effort (LE) delay model is a gain-based model, which emphasizes the linearity between delay and gain (i.e., the ratio between the input and output capacitances) in a given logic gate. This model has been used in several papers – e.g., Boyd (2005), Hu (2003), Joshi (2008), Karandikar (2004; 2005; 2008), Rezvani (2003), and Zeydel (2006).

The model expresses the delay of a logic gate as a function of four parameters, which are defined and explained along the deduction of the model:

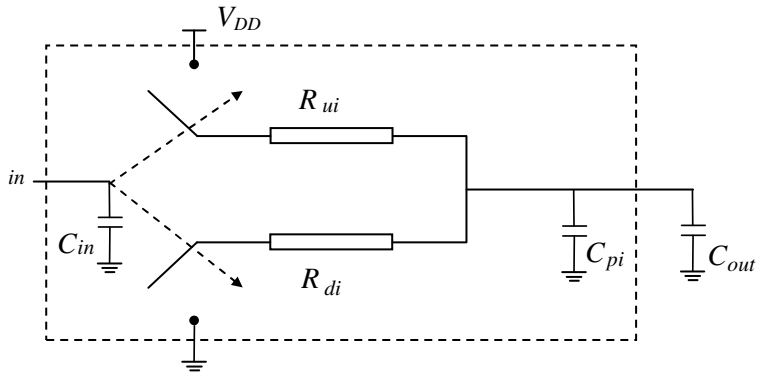
- τ: characteristic delay of the given technology;
- g: logical effort of the gate;
- h: electrical effort of the gate;
- p: parasitic (intrinsic) delay of the logic gate.

What matters now is that these four parameters allow the designer to obtain the delay of any logic gate in a quick, straightforward way. Besides, the logical effort differentiates the parasitic delay p (due to the intrinsic parasitic capacitances of the logic gate) from the delay due to the load driven by the logic gate (electrical effort h) and from the delay due to the topological characteristics of the logic gate (logical effort g).

This way, it becomes easy to realize the contribution of each of these factors to the total delay of the logic gate. Next, the logical effort is deducted, which better explains the four parameters previously mentioned. Examples of obtaining the parameters logical effort (g) and parasitic delay (p) for basic logic gates are also given.

According to Sutherland (1999), a single-input logic gate is modeled according to Figure 2.10. C_{in} represents the capacitance of each input of the logic gate. C_{in} is given by the sum of the gate capacitances of the transistors whose gates are connected to such input. C_{pi} denotes the intrinsic parasitic capacitance of the logic gate. C_{pi} is given basically by the sum of the drain capacitances of the transistors whose drains are connected to the output of the logic gate. The capacitance to be driven by the logic gate is named C_{out}.

Figure 2.10 – Logical effort delay model of a logic gate



Source: Sutherland (1999).

When the pull-up key is closed, the pull-up network offers a resistance R_{ui} to the current between V_{DD} and the logic gate output. When it is the turn for the pull-down key to be closed, the pull-down network offers a resistance R_{di} to the current between *GND* and the logic gate output (The concepts of pull-up and pull-down network are explained in section 3.2). To keep the model simple, it is assumed that $R_{ui} = R_{di} = R_i$, so that the logic gate has equal t_r and t_f.

In order to assure that R_{ui} is indeed equal to R_{di}, it suffices to use the convenient γ parameter for the technology, where γ is the ratio between the channel width of the PMOS transistor (w_p) and the channel width of the NMOS transistor (w_n) in a minimum sized inverter:

$$\gamma = w_p / w_n \tag{2-10}$$

The logical effort model uses the parameters C_{in}, C_{pi}, C_{out}, and R_i, along with the concepts of template and scale factor of a logic gate seen in section 2.1, to determine the delay of a generic logic gate, with arbitrary size and topology. In order to obtain the transistor sizes of a generic logic gate, the size of each transistor in the corresponding template is multiplied by a convenient scale factor x. The template of the logic gate in Figure 2.10 has input capacitance C_t, intrinsic parasitic capacitance C_{pt} and pull-up/pull-

down resistance R_t. The logic gate parameters and the corresponding template parameters obey the following relationship:

$$C_{in} = xC_t \tag{2-11}$$

$$C_{pi} = xC_{pt} \tag{2-12}$$

$$R_i = x^{-1}R_t \tag{2-13}$$

According to Sutherland (1999), the absolute delay (d_{abs}) of the logic gate depicted in Figure 2.10 is given by:

$$d_{abs} = kR_i(C_{out} + C_{pi}) \tag{2-14}$$

where k is a dimensionless constant, that is specific to the used technology. Let:

$$\tau = kR_{inv}C_{inv} \tag{2-15}$$

$$g = \frac{R_tC_t}{R_{inv}C_{inv}} \tag{2-16}$$

$$h = C_{out} \Big/ C_{in} \tag{2-17}$$

$$p = \frac{R_tC_{pt}}{R_{inv}C_{inv}} \tag{2-18}$$

where:

- R_{inv} is the *pull-up/pull-down* resistance of the minimum-sized inverter (i.e., inverter template);
- C_{inv} is the input capacitance of the minimum-sized inverter;
- τ is the characteristic delay of the used technology;
- g is the logical effort of the logic gate;
- h is the electrical effort of the logic gate;
- p is the parasitic delay of the logic gate.

Rewriting equation (2-14) as a function of (2-11), (2-12), (2-13), (2-15), (2-16), (2-17), and (2-18), we have:

$$d_{abs} = \tau(gh + p) \tag{2-19}$$

The logical effort g depends solely on the topology of the logic gate; it does not depend on the size of this logic gate. The logical effort expresses how much the logic gate is inferior to the minimum sized inverter of the same technology in delivering output current. By definition, the logical effort of an inverter is equal to 1. The electrical effort gives the ratio between the fanout and the fanin of the logic gate. Therefore, the electrical effort depends on the size of the logic gate and also on the load that it must drive.

The parasitic delay p is so called because it represents the delay due to the intrinsic parasitic capacitance C_{pt} of the logic gate. In the hypothetical situation of the logic gate driving no load whatsoever ($h = 0$), the delay is equal to p. The parasitic delay p denotes how much the parasitic delay is greater in a generic logic gate than the parasitic delay of the minimum inverter in the given technology. There is no relationship between the size

of the logic gate and its parasitic delay, since this delay depends only on the topology of the logic gate.

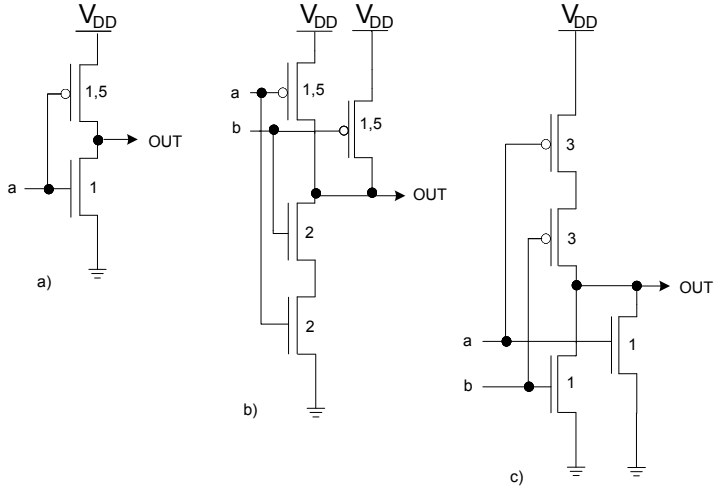
From equation (2-19), one can see that the characteristic delay τ may be understood in two different ways: 1) τ is the delay of an inverter with no intrinsic parasitic capacitance (p = 0) driving an identical inverter; or 2) τ is the delay of a loadless inverter whose transistors have gate capacitances equal to the drain capacitances. The absolute delay d_{abs} may be parameterized as a function of τ:

$$d_{abs} = d\tau \tag{2-20}$$

This parameterization is useful for comparing the dimensionless relative delay d of circuits developed in different technologies.

According to Sutherland (1999), there are two rules to determine the logical effort g and the parasitic delay p of a logic gate. The first rule refers to the parameter γ, which is obtained based upon the inverter; γ gives the relative sizes of PMOS and NMOS transistors, and it is valid for all logic gates in a given technology. The second rule refers to series transistors, connected between the output of the logic gate and either V_{DD} or *GND*. If there are n series transistors, then each one must be n times bigger than a single transistor, in the same situation. Figure 2.11 exemplifies this procedure for three basic logic gates.

Figure 2.11 – Examples of basic logic gates modeled according to logical effort



Source: Logics (2013).

For the technology used in Figure 2.11, parameter γ is equal to 1.5. Therefore, the inverter (Figure 2.11a) has input capacitance C_{in_INV} equal to 2.5. The logical effort g of the inverter is equal to 1. Parasitic delay p is also equal to 1, as long as the transistors have equal gate and drain capacitances.

In the NAND2 logic gate (Figure 2.11b), the PMOS transistors have the same size as the PMOS transistor in the inverter. However, the two NMOS transistors are in series and, therefore, they must have twice the size of the NMOS transistor in the inverter.

Thus, input capacitance C_{in_NAND2} is equal to 3.5. The logical effort g is equal to 1.4 ($g = C_{in\_NAND2}/C_{in\_INV}$) and the parasitic delay p is equal to 2 (there are two PMOS transistors and one NMOS transistor connected to the output of the logic gate).

In the NOR2 logic gate (Figure 2.11c), the PMOS transistors are in series and have twice the size of the PMOS transistor in the inverter. The input capacitance C_{in_NOR2} is equal to 4, the logical effort g is equal to 1.6 and the parasitic delay is equal to 2.

The logical effort delay model is very simple and easy to use. It is especially well suited for sizing for minimum delay. Nevertheless, its simplicity causes inaccuracy. This model assumes that the gate and drain capacitances of a transistor are equal, but in fact the gate capacitance uses to be greater than drain capacitance. This issue may be solved via model calibration (SUTHERLAND, 1999). Besides, the sizing of series transistors does not take into account the velocity saturation of carriers. This phenomenon is more evident in NMOS transistors, since electrons have a greater mobility than holes. Moreover, series transistors are less prone to showing this phenomenon than single transistors. Therefore, series transistors, especially NMOS transistors, must have a smaller size than that predicted by logical effort. An adequate calibration solves this problem.

Furthermore, the logical effort model computes the delay of a logic gate regardless of the transition time of the input signal. This behavior does not correspond to the real world, since a logic gate shows different delays for different transition times of the input signal. Besides, the logical effort model makes no distinction whatsoever between the transition time in the output of a logic gate and the propagation delay of this logic gate. These problems were solved by Lasbouygues (2006). Other papers also extended the logical effort delay model — e.g., Kabbani (2005) and Keane (2006).

2.3.4 Nonlinear Delay Model – NLDM

The nonlinear delay model (NLDM) is mentioned here as an example of an empirical delay model, in order to show the differences with respect to theoretical (Elmore and logical effort) delay models. This research work is based upon the logical effort delay model, since both mathematical programming techniques and analytical solution of the minimization problem are well suited for solving a problem modeled this way.

The NLDM model, introduced by Synopsys and embodied by the Liberty format (SYNOPSYS-a), consists in a discrete lookup table (LUT) filled by pre-characterization, which gives the delay and the output transition time of a generic cell as a function of both the transition time of the input signal and the output load of this cell. For values of transition time of the input signal and of output load that do not belong in the table, an interpolation is done. A lookup table is empirically constructed for every standard cell in the library.

By construction, the NLDM model is exact in the points of transition time of the input signal and output load in which the table was built. It is an accurate model, but its complexity turns out the simulation of the logic gates to be very slow. The NLDM model demands a high computational effort to simulate the standard cells in order to build the lookup tables, especially during the characterization of these cells (TRIHY, 2008).

There is no consensus in the literature as to which delay model (theoretical or empirical) is the most appropriate one. Even though there is a survey showing that lookup-table based models are more accurate than theoretical models (MARTINEZ apud COUDERT, 1996-b), the interpolation done in the former models may generate inaccurate delay values.

2.3.5 Delay models comparison

Theoretical and empirical delay models have different characteristics and objectives. These differences make each category of delay model more suitable for a specific stage of the industrial design flow, as it is explained next.

Timing analysis is performed twice in an industrial design flow. Initially, an early timing estimation is done. This early timing estimation must be fast enough so that it can be used in the initial sizing of complete circuits, even though it is not sufficiently accurate for timing closure. Theoretical delay models such as Elmore and logical effort delay models are well suited for such early timing estimation.

In a later stage of the industrial design flow, the back-end timing estimation is accomplished. It is too slow for the initial sizing of complete circuits, but its accuracy allows it to be used for timing closure. Empirical delay models are appropriate models for this back-end timing estimation. In fact, the NLDM model is currently the industry standard delay model for timing closure.

2.4 Sizing

Circuit sizing is accomplished in order to determine the size of the components of the circuit, aiming at optimizing a given objective function, and always respecting the design constraints. Within the scope of this research work, the goal is to minimize the active area of the circuit, respecting the maximum delay, maximum input pin capacitance, as well as the load to be driven in each output pin. Depending on the case, the delay constraint may be reduced, and a new sizing is done so that all design constraints are fully respected.

Sizing is of foremost importance in the design flow of digital integrated circuits, since it guarantees that the circuit is conceived within the specifications and required design constraints. More than that, an adequate sizing is capable of delivering an optimized version of the circuit, with more demanding design constraints. In this section, the concept of sizing is introduced. Examples and more details about sizing algorithms and methods are seen in chapter 3.

Transistor sizing is the first type of sizing. In it, each transistor in a logic gate (or in a circuit as a whole) can have its size modified individually, without necessarily provoking modifications in the size of the remaining transistors in the logic gate (or in the circuit). Transistor sizing uses to be employed for sizing standard cells in a library. Based on this sizing, a table of transistor sizes is generated. Then, this table is used for obtaining the template of each standard cell in the library. Transistor sizing was used in Kung (1999), Roy (2007), and Shah (2006).

Once the transistor sizing is done, the next step is cell sizing. This latter sizing aims at determining the size of a cell that is used to map a circuit node. This means to find the scale factor that will be applied to each transistor in the cell template, in order to

obtain the cell with the desired size. In cell sizing, the sizes of all transistors in the cell are multiplied by the found scale factor. For the sake of compatibility with the terminology adopted in the literature, the expressions ‘cell sizing’ and ‘gate sizing’ are used indistinctively in this research work.

Sizing may also be classified with respect to its scope, which may be either global or local. In global sizing, the circuit as a whole is sized. This can be done via either transistor sizing or cell sizing. Transistor sizing offers considerable freedom to the designer, since the size of each transistor can be adjusted to the design needs. This is the sizing used in full custom design. However, technological progress fosters the miniaturization and integration in the microelectronics industry, leading to an increase in the number of transistors in a circuit. The complexity of sizing transistors individually grows the same way. Consequently, cell sizing becomes more and more relevant. This kind of sizing handles circuits of high complexity, but this is accomplished at the expense of a lesser sizing granularity: it is impossible to change the size of a single transistor in one cell without changing the size of all the other transistors in the same proportion.

In Joshi (2008), it is proposed a global sizing method that is capable of optimizing a circuit with more than one million logic gates, always finding the global optimum for the circuit. Nevertheless, this huge scale is achieved at the expense of using an inaccurate delay model, which might compromise the optimality of the solution. In order to achieve a high accuracy, local sizing must be used. In local sizing, only a small part (subcircuit) of the circuit is sized. Since there are fewer components to be sized, a more accurate delay model may be used. Global and local sizing may be used in a complementary way. Initially, the entire circuit is globally sized, generating a first version of the circuit. Then, this version of the circuit is optimized, one subcircuit at a time, via local sizing.

Local sizing is advantageous when it is known beforehand that the circuit has just a few specific bottlenecks, in which design constraints are not met. In this case, there is no need to resize globally the entire circuit, it suffices to attack the problematic parts. Local sizing is part of the wider concept of local optimization, which is better explained in the coming chapter.

A third criterion to classify the sizing techniques refers to the way the transistor dimensions can vary. In continuous scale sizing – or just continuous sizing –, the transistor size is free to assume any calculated value. In discrete scale sizing – or discrete sizing, for short –, the transistor sizes cannot assume an arbitrary value, but just one out of a finite set of allowed values.

Discrete sizing is used in cell-based design. In this case, the designer (or CAD tool) is aware, e.g., that the standard cell library contains a given cell with drive strength 1X, 2X, and 4X. Therefore, when sizing a circuit, the designer (CAD tool) is aware that, in a given node, this cell may have drive strength 2X or 4X, but never an intermediate value. Based on this knowledge, the sizing of this node generates a value close enough either to 2X or 4X.

Continuous sizing may also be used in cell-based design, but there is a handicap. The designer (CAD tool) sizes the circuit supposing that the standard cells can assume any drive strength. Later on, when mapping the circuit to the standard-cell library, the designer rounds up the calculated value to a drive strength value available in the library. Let us make a comparison with the example in the previous paragraph. Let us suppose

that a second designer, when conceiving the same circuit, finds a value of 2.82X for the drive strength in the given node. This value must be rounded for either 2X or 4X. No matter which rounding is done, there is a considerable inaccuracy in the sizing of such node, because the calculated (ideal) value is geometrically equidistant from the two values available in the standard cell library.

2.5 Mathematical Programming

By mathematical programming, it is meant the resolution techniques for optimization problems that can be modeled as a set of equations and inequations. Mathematical programming has a wide area of application, including sizing of digital integrated circuits.The origin of mathematical programming dates back to the first half of the last century, with the pioneer work on linear programming (LP) by George Dantzig and Leonid Kantorovich (VANDERBEI, 2008). Since then, with the development of new techniques and depending on the kind of problem to be solved, the equations and inequations of the mathematical program may have particular characteristics, such as: linear functions, nonlinear, convex, integer variables etc. In its most generic form, a mathematical program may be defined as follows (BOYD, 2004):

$$\text{minimize } f_0(x) \tag{2-21}$$

$$\text{subject to } f_i(x) \le b_i, \quad i \in \{1,\ldots,m\}$$

where: $x = (x_1,\ldots,x_n)$ – vectorial optimization variable;

$f_0 : \Re^n \to \Re$ – objective function;

$f_i : \Re^n \to \Re$, $i \in \{1,\ldots,m\}$ – constraint functions;

$b_1,\ldots,b_m$ – constants that limit the constraint functions.

The optimal vector x^* – solution vector to the problem stated in equation (2-21) – is the vector for which $f_0(x^*)$ has the minimum value, among all vectors that satisfy the *m* constraint functions.

Linear programming was the first kind of mathematical programming to be developed. Diverse problems can be modeled as linear programs, and there are reliable solution methods for linear programming (e.g., simplex method and interior-point methods) (VANDERBEI, 2008). This made out of linear programming a very popular technique, most of all in Economics and Business Administration. However, in Science, Technology, and Engineering, nonlinear problems are very common, which lessens the use of linear programming. For instance, digital circuit sizing is a nonlinear problem, because the delay of a cell is an affine function of the inverse of the scale factor of the cell. A standard linear program may be stated as:

$$\text{minimize } c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{2-22}$$

$$\text{subject to } a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \le b_i, \quad i \in \{1,\ldots,m\}$$

$$x_j \ge 0, \quad j \in \{1,\ldots,n\}$$

where: $x = (x_1, \ldots, x_n)$ – real vectorial optimization variable;

$a_{ij}, c_j \in \Re$.

Convex programming (CP) is a generalization of linear programming, which became popular more recently. Convex programming is capable of filling part of the gap left behind by linear programming, mainly in Science, Technology, and Engineering. A convex program is described as in equation (2-21), in which the objective function f_0 and the constraint functions f_i are convex functions of the vectorial optimization variable. A convex function is a continuous function whose value at the midpoint of every interval in its domain does not exceed the arithmetic mean of its values at the ends of the interval. More generally, in formal terms:

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y), \quad \forall x, y \in \Re^n \text{ and } \forall t \in [0,1] \qquad (2\text{-}23)$$

It can be said that geometric programming is a specific type of convex programming since, via variable change and rewrite of the objective function and the constraint functions, a generic geometric program becomes a convex program. Besides, in some sense geometric programming is a generalization of linear programming, even though some linear programs cannot be modeled as geometric programs. Geometric programming is more appropriate than linear programming for modeling problems in Science, Technology, and Engineering, and the programs can be solved efficiently.

According to Boyd (2004), a geometric program is defined as:

$$\begin{aligned} &\text{minimize} \quad f_0(x) \qquad (2\text{-}24)\\ &\text{subject to} \quad f_i(x) \leq 1, \quad i \in \{1, \ldots, m\}\\ &\qquad\qquad h_i(x) = 1, \; i \in \{1, \ldots, p\} \end{aligned}$$

where: $x = (x_1, \ldots, x_n)$ – vectorial optimization variable;

$f_0 : \Re_+^{*n} \rightarrow \Re$ – posynomial objective functions;

$f_i : \Re_+^{*n} \rightarrow \Re, \; i \in \{1, \ldots, m\}$ – posynomial constraint functions;

$h_i : \Re_+^{*n} \rightarrow \Re, \; i \in \{1, \ldots, p\}$ – monomial constraint functions.

Among all kinds of mathematical programming, nonlinear programming (NLP) is the most generic type; i.e., it has the greatest capacity of describing a real world problem as equations and inequations. However, nonlinear programs have the most difficult resolution. In fact, only some particular types can be solved (LUENBERGER, 2008). In mathematical terms, a nonlinear program is a program as specified in (2-21), in which the objective function f_0 or one or more constraint functions f_i are nonlinear (BERTSEKAS, 1999).

Integer programming (IP) is a particular case of linear programming as stated in equation (2-22), in which the scalar components of the vectorial optimization variable are required to take on integer values only. Unlike the previous categories of mathematical programming, integer programming is solved by combinatorial optimization techniques, which usually demand more computational resources than mathematical programs based on real (continuous) variables (GOLDBARG, 2005).

2.6 Analytical Resolution of the Minimization Problem

Mathematical programming is widely used for solving optimization problems, inclusive digital circuit sizing — e.g., Berkelaar (1990), Chen (1996), Joshi (2008), Kasamsetty (2000), Mahalingam (2005), Menezes (1995), Nguyen (2003), Pattanaik (2003), Roy (2007), Sapatnekar (1995), and Singh (2008). This is due to the fact that mathematical programming has a wide area of application, as well as it can deal with problems with a large number of variables easily. However, mathematical programming is not the only possible approach for optimization problems. Mathematical analysis, a technique precedent to mathematical programming, offers an efficient way for sizing digital circuits.

By mathematical analysis, it is meant the study — via infinitesimal calculus — of extreme points (local and global maxima and minima) of real functions of real variables. These functions shall be piecewise continuous and piecewise differentiable at least up to the second order. There are many practical situations in which mathematical programming may be used and mathematical analysis may not. Nevertheless, when it may be used, mathematical analysis tends to give an answer faster and using less computational resources than mathematical programming would. There are two conditions that must be satisfied, so that mathematical programming may be used for sizing digital circuits:

- The delay model must express analytically the active area of the digital circuit as a function of the scale factors of every logic gate in the circuit;
- This function must have first and second order derivatives.

As seen in section 2.3.3, the logical effort delay model provides an analytical expression for the active area of the circuit. In chapter 4, it is shown that this expression may be manipulated via mathematical analysis, resulting in one univariate equation. The solution of this univariate equation gives the value of the scale factor of the last logic gate in the logic path. By replacing the value of the scale factor just obtained in other equations of the model, it is calculated the value of the penultimate scale factor in the logic path, and so forth.

2.6.1 Minimization of multivariable functions

Let $f(x_1, x_2, \ldots, x_n)$ be a real function of n real variables with continuous second order partial derivatives. The local minima of f are the points $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$ in which we have:

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \cdots = \frac{\partial f}{\partial x_n} = 0 \tag{2-25}$$

and $\Delta_k > 0, \ \forall \ k \ | 1 \le k \le n$

where: $\Delta_k = \det\left(\dfrac{\partial^2 f}{\partial x_i \partial x_j} \right) \ \forall \ i, j \ | \ 1 \le i, j \le k$

That is, the local minima are the critical points $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$ in which the determinants of all principal submatrices of the Hessian matrix $\mathbf{H}(f)$ are strictly positive (ROBINSON).

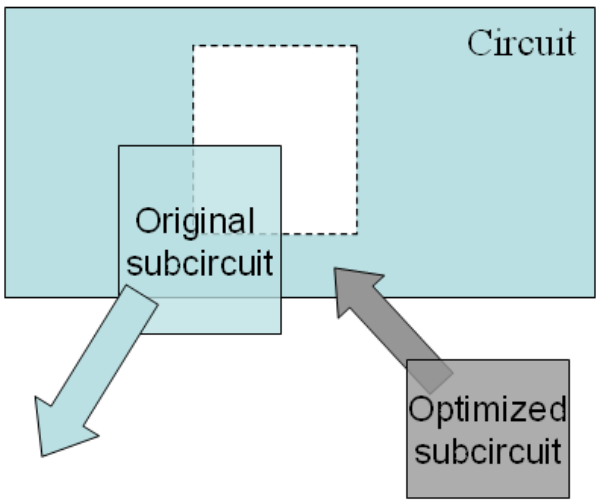
3 BIBLIOGRAPHICAL REVIEW

In this chapter, concepts more intrinsically related to this thesis are reviewed, as well as some reference works about digital circuit sizing. Initially, the concept of local optimization is reviewed, along with some examples. Next, the kind of logic network this research work refers to is seen. After that, modifications and extensions to the logical effort delay model are analyzed, in order to evaluate the feasibility of using such modifications and extensions within the scope of this research work. This thesis focuses on the sizing of logic networks for local optimization. Therefore, the most relevant methods for sizing digital circuits are also analyzed in this chapter.

3.1 Local Optimization

Local optimization consists in successively replacing specific subcircuits (each subcircuit with an approximate size of a few two-input logic gates) of a digital circuit by more efficient subcircuits whose logical functionalities are indistinguishable from the logical functionalities of the original subcircuits (GOPALAKRISHNAN, 1999). Each subcircuit is optimized according to the environment imposed by the remainder of the circuit. This environment is characterized by the local design constraints. Figure 3.1 depicts such idea.

Figure 3.1 – Local optimization



Source: Logics (2013).

In Figure 3.1, both the original and the optimized subcircuits implement the same logic function. However, the optimized subcircuit shows at least one local design constraint that is optimized with respect to the corresponding local design constraint of the original subcircuit, at the same time assuring that the remaining local design constraints do not deteriorate. Only when these conditions are fulfilled, the original subcircuit is replaced by the optimized one.

Unlike methods that optimize an entire circuit, with hundreds of thousands or even millions of logic gates, local optimization focuses on a small piece of the circuit and, consequently, it has more freedom for actuating. Indeed, local optimization can handle the logic functions implemented by the subcircuit. This is done by changing the logic networks or transistor networks by other ones with the same logic functionality. Besides choosing the most appropriate logic network or transistor network, local optimization can also perform the local sizing of these networks. To do that, local optimization may rely upon delay models more accurate than those used in global sizing.

In the literature, there can be found works on local optimization that focus exclusively on the exchange of logic networks and transistor networks (WERBER, 2007; YOSHIDA, 2006), as well as papers that deal solely with the sizing of networks previously defined (MAURINE, 2002; PANDA, 1998).

3.2 Transistor Networks

As seen in section 2.1, a transistor network is an array of transistors capable of implementing a logic function, in which these transistors are not necessarily organized into logic gates. Specifically within this research work, only MOS transistor networks are studied.

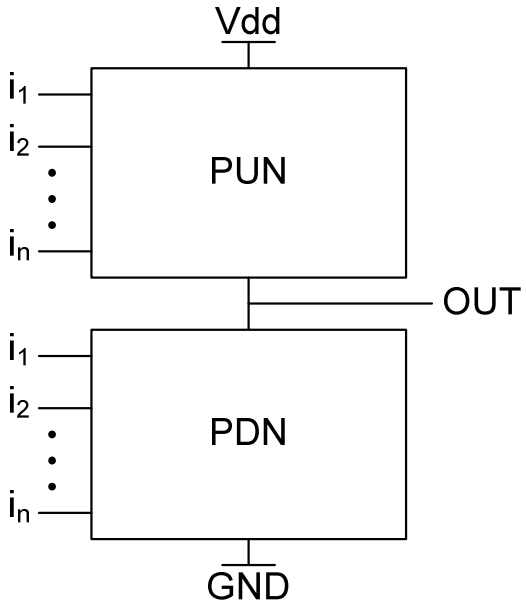
A specific logic function may be implemented by diverse logic and transistor networks, as seen in Figure 2.1. Each network has its own tradeoff between active area and delay. This is particularly true after the network has been sized according to the design constraints of a particular application context. As examples of transistor network generating methods, one can mention Kagaris (2007), Poli (2003), Reis (2009), Rosa Junior (2006), and Schneider (2005).

Transistor networks may be classified in different logic styles (RABAEY, 2003), also named circuit families (WESTE, 2006). There are numerous circuit families, each of them with its own characteristics and specific properties. Out of these logic styles, this research work focuses on static CSP CMOS networks. In static networks, the output is always connected to either V_{DD} or *GND* through a low resistance path. CSP CMOS networks represent the most widely used logic style in the microelectronics industry, due to its low noise sensitivity, good performance, low power consumption, and no static power dissipation (RABAEY, 2003).

Figure 3.2 shows that a CSP CMOS logic gate is made up of a pull-up network (PUN) and a pull-down network (PDN). Each of the n inputs of the logic gates feeds both networks. Basically, the role of the PUN is to establish a connection – i.e., a low resistance path – between the output *OUT* and supply voltage V_{DD} whenever the output should have a high value (true – '1'), according to the values of the n input signals. Analogously, the function of the PDN is to make a connection between the output *OUT* and the ground *GND* whenever the output should have a low value (false – '0'),

according to the values of the *n* input signals. The pull-up and pull-down networks are conceived in a mutually exclusive way, in order to assure that, in steady state, one and only one of the networks is conducting, at any given time. In other words, the output is always connected to either V_{DD} or *GND*, it is never in a high impedance state or connected to V_{DD} and *GND* simultaneously (RABAEY, 2003).

Figure 3.2 – CSP CMOS logic gate seen as a combination of a pull-up and a pull-down network



Source: Rabaey (2003).

The pull-up network is made of PMOS transistors, while the pull-down network is made of NMOS transistors. This is that way because a PMOS transistor generates strong '1' and weak '0', meanwhile an NMOS transistor generates strong '0' and weak '1'. Hence, each kind of transistor is used in its most efficient way (RABAEY, 2003).

In pull-up and pull-down networks, the transistors are associated in series-parallel. Depending on the association, different logic functions may be implemented. Two series NMOS transistors implement the AND function over the transistor gate signals: only when both transistor gate signals are equal to '1', the composition conducts. Analogously, two parallel NMOS transistors implement the OR function over the transistor gate signals: it suffices that one of the transistor gate signals be equal to '1' for the composition to conduct. In the same fashion, two series PMOS transistors implement the NOR function over the transistor gate signals: only when both transistor gate signals are equal to '0', the composition conducts. Analogously, two parallel PMOS transistors implement the NAND function over the transistor gate signals: it suffices that one of the transistor gate signals be equal to '0' for the composition to conduct.

In static CMOS networks, the transistors work as keys controlled by the input signal in their gates. Every input signal is applied to the gate of a PMOS transistor and also to the gate of an NMOS transistor. Therefore, an *n*-input logic gate is made of 2*n* transistors. This is due to the mutually exclusive way in which the PUN and PDN are

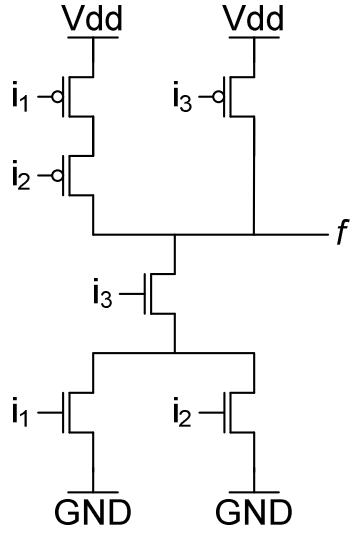
conceived. These are dual networks, i.e., for every series connection of PMOS transistors in the PUN, there is a parallel connection of NMOS transistors in the PDN and vice versa. Since PMOS (NMOS) transistors only conduct with a '0' ('1') signal in its gate, only one of the two transistors driven by an input signal is allowed to conduct at a given moment. This fact, along with the duality principle, assures that one and only one of the networks is conducting at a given time, except for the transition time.

The duality principle may be observed in Figure 3.3. For convenience of notation, the transistors are identified by the input signals in their gates. In the pull-down plane, the NMOS transistors i_1 and i_2 are in parallel. Therefore, in the pull-up plane, the PMOS transistors i_1 and i_2 are in series. The NMOS transistor i_3, in the pull-down plane, is in series with the subnetwork composed of the NMOS transistors i_1 and i_2. Henceforth, in the pull-up plane, the PMOS transistor i_3 is in parallel with the subnetwork composed of the PMOS transistors i_1 and i_2.

Since the pull-down network is made of NMOS transistors that conduct with a signal '1' in their gates, CSP CMOS logic gates implement negated logic. That is, logic functions INV, NAND, NOR etc. may be implemented by only one CSP CMOS logic gate. Logic functions AND and OR need an additional inverter to be implemented.

In order to build a CSP CMOS logic gate that implements a logic function *f*, it suffices to create a pull-down network that implements function !*f*, and the pull-up network is implemented as the dual of the PDN. In other words, the PDN is obtained from the off-set of logic function *f*. By off-set it is meant the set of values of the variables of logic function *f* for which the function is equal to '0'. This is the standard procedure for the generation of CSP CMOS logic gates.

Figure 3.3 – Series-parallel duality in CSP CMOS logic gate



$f = !((i_1 + i_2)\, i_3)$

Source: Logics (2013).

3.3 Delay Models

In the previous section, the logic style of the subcircuits to be optimized in this research work was described. For the optimization to be successful, the delay model must be at the same time both accurate and capable of generating a system of equations and inequations that can be solved by mathematical techniques. The logical effort delay model analyzed in subsection 2.3.3 is suited for such use, but it is somewhat inaccurate. In this section, delay models that have evolved from the original logical effort delay model are analyzed. These models are likely to be used in future works based upon this thesis.

3.3.1 Extension of the logical effort model

According to Lasbouygues (2006), the logical effort delay model makes no distinction between propagation delay and transition time in order to compute the delay of a logic gate. The logical effort delay model simply calculates the transition time of the output signal of the logic gate and takes this as the propagation delay of the logic gate. Besides, this delay is computed without taking into account: a) if the input signal is rising or falling; b) transition time of the input signal; and c) coupling capacitance. As a result, the logical effort produces inaccurate results, which lead to non-optimal sizings, especially when rigorous timing design constraints are at stake.

In order to solve such problems, Lasbouygues (2006) proposes an extension to the logical effort delay model that is capable of considering coupling capacitance effects, as well as the transition time of the input signal, in the computation of the delay of the logic gate. This is done via modeling the charge and discharge current during the switching of a basic logic gate.

This extension to the logical effort delay model has two different formulations, depending on the transition time of the input signal. This fact, as well as the very equationing of the extension, must be analyzed to verify the compatibility of this extension to the logical effort delay model with mathematical techniques.

3.3.2 Modified logical effort model

Another approach to compensate for the deficiencies of the logical effort delay model is found in the paper of Kabbani (2005). This work introduces the Modified Logical Effort (MLE), a delay model for CSP CMOS logic gates that takes into account the behavior of series-connected MOSFET structures (SCMS), the input transition time, internodal charges, mobility degradation, and velocity saturation. Since MLE evaluates internodal charges, it is capable of dealing with input signals switching in different series transistors. This is advantageous over the original logical effort delay model, which considers solely the switching of transistors directly connected to the output of the logic gate. The properties of the MLE model are reflected into its mathematical formulation, which might compromise the feasibility of using it as the delay model of a sizing method.

3.3.3 Logical effort model considering transition time

In Wang (2009), it is proposed another improvement to the original logical effort delay model. Based in one additional experimental parameter, this work manages to account for the transition time of the input signal in a more straightforward way than the approaches seen in the two previous subsections, while keeping the same accuracy

level. Nevertheless, the delay model of Wang (2009) – as well as the models of Lasbouygues (2006) and Kabbani (2005) – is not a posynomial model. This fact imposes a severe restriction with respect to the mathematical programming techniques that may be used with it.

3.4 Sizing Algorithms

As seen in the previous chapter, sizing is of foremost importance in the design flow of digital integrated circuits. There are diverse approaches for sizing, as described in papers about the subject.

The works of Chu (2001), Kasamsetty (2000), Matson (1986), Otten (2000), Pattanaik (2003), Sapatnekar (1993-b; 1995), and Tennakoon (2002) are examples of delay models compatible with geometric programming that are applied in the sizing of transistors and logic gates. In Berkelaar (1990), a convex delay model is used for sizing logic gates, and the problem is solved via linear programming. In Tennakoon (2005), a piecewise convex delay model is presented. It is more accurate than a conventional convex delay model, but the sizing problem must be solved via lagrangian relaxation, since the delay model as a whole is no longer convex. In Shah (2005), dual-V_t is added to an originally convex delay model. Since V_t can assume only two different values, the problem becomes a mixed integer nonlinear program (MINLP), which is solved heuristically.

The objective of this section is to analyze six algorithms that solve the sizing problem via distinct approaches. Based on this analysis, the sizing technique of this research work is proposed. The first two algorithms rely on convex optimization and are suitable for continuous sizing of digital integrated circuits. The third algorithm is popular among designers and may be regarded as the predecessor of the sizing method proposed in this research work. The algorithms of subsections 3.4.4 e 3.4.5 perform discrete sizing and are adequate for cell-based design. At last, the fifth method to be analyzed introduces the concept of flex cells for optimizing digital integrated circuits.

3.4.1 TILOS: An initial approach in geometric programming

Fishburn (1985) presents the software TILOS (TImed LOgic Synthesizer), which is based on a transistor and interconnection sizing algorithm. TILOS uses convex programming techniques, and the delay in each logic gate is modeled according to Elmore (1948). The gate, drain, and source capacitances of a transistor are supposed to be directly proportional to the size of this transistor, meanwhile the drain–source resistance is supposed to be inversely proportional to the size of the transistor. Therefore, the delay in the circuit is expressed as a posynomial function – hence, convex – of the sizes of the transistors.

The software TILOS works in an interactive way. Initially, all the transistors in the circuit have minimum size. A static timing analysis (STA) is performed, thus detecting the path with the greatest timing violation. This path is traversed until the transistor with the greatest timing sensitivity with respect to its sizing is found. The size of this transistor is increased, therefore diminishing the delay in the path. This process goes on, until there are no more timing violations left.

TILOS was one of the first proposals based in geometric programming for sizing digital circuits. TILOS was capable of sizing circuits with 26,000 transistors, which was a large circuit at that time. However, the optimality of the solution of the convex program is not granted (BERKELAAR, 1990), due to the greedy way with which the highest sensitivity transistor is enlarged. This is done without taking into account the circuit as a whole (CHINNERY, 2005). Besides, the modeling of the delay of a logic gate with a distributed *RC* network is not capable of handling input signals with high transition time (FISHBURN, 1985). These signals have an equationing that is not necessarily convex. It is worthy to remember that TILOS performs just transistor sizing, it does not modify the circuit topology.

3.4.2 A modern approach in geometric programming

In Boyd (2005), there is a survey about the use of geometric programming (GP) and generalized geometric programming (GGP) for sizing semiconductor devices, logic gates, and interconnections. The proposed method encompasses the transition time of the input signals, as well as robust project, multimode project, statistical project, and projects in which the threshold voltage V_t and V_{DD} are variables to be determined. The method can optimize the area or power for a given timing constraint, as well as it can optimize the delay for a given area or power constraint.

This method consists in applying GP for the global sizing of digital circuits. It performs cell sizing, based on a convex delay model (Elmore, logical effort, or another one) for the logic gates. A scale factor is associated to each logic gate, and the corresponding delay is expressed as a convex function of this scale factor. The method performs continuous sizing: the scale factor may, at principle, assume any real value greater than or equal to 1. When the method of Boyd (2005) is used to size a circuit to be mapped to a standard-cell library, the rounding problem discussed in section 2.4 arises.

Geometric programming may be used for modeling the delay in a logic path, as long as the delay model is convex. The delay in a digital circuit is given by the maximum delay of all paths between the inputs and outputs of the circuit. This maximum delay cannot be modeled by geometric programming, but rather by generalized geometric programming (GGP)

This method, based on GP, has the characteristics inherent to this kind of mathematical programming, namely:

- if there is a local minimum for the sizing objective function, then this minimum is found by the method and it is guaranteed to be the global minimum;
- if this minimum does not exist, this is quickly detected by the method.

The method of Boyd (2005) can perform the global sizing of large circuits, as long as the convex delay model is simple.

This method preserves the pre-defined topology of the circuit. The only thing to change is the drive strength of each logic gate of the circuit. Therefore, the method of Boyd (2005) is unable to obtain an optimization that demands changes in the topology.

3.4.3 The logical effort sizing method

The logical effort sizing method (SUTHERLAND, 1999) is very popular, due to its simplicity and didactic aspects. The groundwork for this sizing method is the homonymous gain-based delay model, as seen in subsection 2.3.3. Based on this delay

model, the logical effort sizing method may be derived as follows. It is assumed that the total delay of a logic path is given by the sum of the delays of every logic gate in such path. Taking the derivative of this total delay with respect to the electrical effort *h*, one can see that minimum delay is obtained when the product *gh* is the same for each logic gate in the path. This result is valid regardless of the number of logic gates in the path. Therefore, the logic path is sized for minimum delay, regardless of area or power considerations. This is the major handicap of this sizing method, but the proposed sizing method offers a solution for such problem.

3.4.4 GS: A well-succeeded discrete sizing algorithm

In Coudert (1996-a), it is introduced the GS algorithm for sizing of logic gates. It is a general purpose optimizer, that can optimize the area or power for given timing constraints, as well as it can optimize the delay for given area or power constraints. The GS algorithm performs discrete sizing and it was developed aiming at the cell-based design methodology.

This algorithm has a combinatorial optimization approach for the sizing of a circuit mapped to a standard cell library, as described next. Let a generic logic path of this circuit, and let $Cell_i$ be the cell in the *i*-th stage of this path. The GS algorithm determines locally which is the standard cell with best timing for the node ($Cell_i$') and replaces $Cell_i$ for $Cell_i$'. Since $Cell_i$ and $Cell_i$' do not necessarily have the same input capacitances, propagation delays, and drive strengths, it is necessary to verify the impact of this change in the global delay of this logic path.

In order to compute this change in the delay, the algorithm takes advantage of the following fact: the impact of changing a cell decays nearly geometrically, as one moves forward or backward in the logic path, taking as reference the replaced cell. Changing from cell $Cell_i$ to cell $Cell_i$', the cell $Cell_{i-1}$ now sees as output capacitance the (new) input capacitance of $Cell_i$'. Therefore, the propagation delay of $Cell_{i-1}$ may be altered, and this must be taken into account to verify the feasibility of replacing $Cell_i$ by $Cell_i$'. The propagation delay of $Cell_{i-2}$ does not change, because $Cell_{i-1}$ remains unchanged.

Now moving forward in the logic path, one can see that, if $Cell_i$' and $Cell_i$ have different drive strengths, then the transition time of the input signal in $Cell_{i+1}$ will change accordingly. As mentioned in subsection 2.3.3, the propagation delay of a cell depends on the transition time of its input signals. Therefore, the replacement of $Cell_i$ by $Cell_i$' may cause a change in the propagation delay of $Cell_{i+1}$. Besides, according to Lasbouygues (2006), the transition time of the output signal of a logic gate depends on the transition time of the input signal of this logic gate. Therefore, a modification in the transition time of the input signal in $Cell_{i+1}$ will carry out a change in the transition time of its output signal, which is the input signal of $Cell_{i+2}$. This, in turn, affects the propagation delay of $Cell_{i+2}$. This effect, in principle, propagates until the end of the logic path under analysis. However, the algorithm of Coudert (1996-a) takes into account the roughly geometric decay of this impact and it supposes that, from cell $Cell_{i+3}$ (inclusive) on, this effect is so mitigated that it can be discarded. Hence, only the cells $Cell_{i+1}$ and $Cell_{i+2}$ are verified with respect to the change of $Cell_i$ by $Cell_i$'.

It is worthy to remember that a cell with high drive strength may overload the previous cell in the logic path, due to its likely high input capacitance. Therefore, the standard cell with best timing for the logic path is not necessarily the fastest one.

The GS algorithm proceeds this way, replacing the cell in a given node by a candidate cell to better timing. GS verifies the new global delay in the logic path and so forth, until coming to the cell that gives the smallest delay for this logic path. The algorithm is able to size fairly large circuits, despite its iterative approach.

The algorithm performs a nonlinear, non-convex, plurimodal (i.e., several minima), design constraint bounded optimization. Therefore, GS can handle complex delay models (or another cost function). In Coudert (1996-a), a nonlinear delay model (NLDM) was used. The algorithm uses a heuristics based on perturbation propagation to avoid gradient recomputing and a global perturbation technique to avoid local minima. Thence, GS algorithm is better than greedy methods for sizing.

Nevertheless, despite all the advantages, the GS algorithm is based on a heuristics, which may not find the optimal solution. In Hu (2007), it is shown an algorithm with better results than Coudert (1996-a). The GS algorithm was presented here as an example of an efficient and popular method for performing the global sizing of a cell-based-designed digital integrated circuit, via combinatorial optimization and discrete sizing.

3.4.5 NEW: A recent discrete sizing algorithm

In Hu (2007), it is introduced the algorithm NEW for discrete sizing, used in cell-based design. This algorithm aims at the same objectives as those of Coudert (1996-a), but it works in a considerably different manner. NEW performs an approach similar to dynamic programming (DP) and guided by a continuous solution, as described next.

The algorithm performs a breadth-first traversal on the circuit graph, processing one node at a time. Instead of verifying every possible cell implementation for a node, it just verifies a small number of implementations. These verified implementations are close to the optimized continuous solution. By doing so, the search space is drastically reduced, but practically without compromising the quality of the solutions found. The algorithm also concentrates the efforts in the nodes of greater criticality, i.e., nodes with less slack. Since NEW searches for solutions in a more ordered fashion than Coudert (1996-a) algorithm, the former is able of obtaining solutions more efficiently.

In order to preserve the diversity and representativeness of the intermediate solutions, as well as to eliminate the solutions of inferior quality, the NEW algorithm uses the LSH — Locality Sensitive Hashing (GIONIS, 1999) — technique.

It is important to emphasize that the NEW algorithm does not perform continuous sizing of a circuit. The NEW algorithm is just guided by a continuous solution, meanwhile it executes the discrete sizing. This is rather different from algorithms that simply perform continuous sizing for cell-based design and, when the sizing is concluded, just round up the final continuous results to the cells of closest size. By doing so, these algorithms incur in the rounding problem explained in section 2.4. Once the NEW algorithm executes a discrete sizing since the beginning, there is no such inconvenience. The bigger the gap between the sizes of the standard cells in the library, the bigger the rounding error introduced by the continuous sizing and, therefore, the better the advantage of using the NEW algorithm.

For the same timing design constraint, the NEW algorithm designs a circuit with less area than that of the circuit conceived by Coudert (1996-a) algorithm, at the expense of a longer execution time (HU, 2009).

3.4.6 The flex-cell approach for local optimizations

In Roy (2005), it is introduced the concept of flex cells for optimizing digital integrated circuits. This approach arose as a means of improving the performance of cell-based digital integrated circuits. According to Chinnery (2005), cell-based digital IC's show inferior performance when compared with full-custom digital IC's with the same functionality. There are several reasons why full-custom circuits overcome cell-based design IC's (CHINNERY, 2002), one of which is the limited number of available standard cells in a predefined library. The designer – or EDA tool – counts on a limited number of cell options to map the circuit, which narrows his/her/its freedom of action. Once it is highly unlikely that the cells in a library are optimized for a given digital circuit, it is fairly probable that the cell-based design circuit has a non-optimal performance.

Consequently, human intervention in the cell-based design – or any other automatic design flow – of integrated circuits became commonplace. This is particularly true in the cell-based design of high performance digital circuits, in which the design constraints are so demanding that sometimes they cannot be achieved via mere automatic execution of CAD tools.

Tactical cell insertion is a manner to proceed with this human intervention. These cells are created manually and are also manually inserted in the manually identified nodes in the circuit, in substitution to the existing standard cells (ROY, 2005). Usually, the need for manual intervention only happens on isolated nodes in the circuit, in which the design constraints are not met. When this is the case, there is no need to redesign the entire circuit; it suffices to tackle the problems in the specific nodes where they occur. This is the essence of local optimization seen in section 3.1 and is also the approach of flex cells.

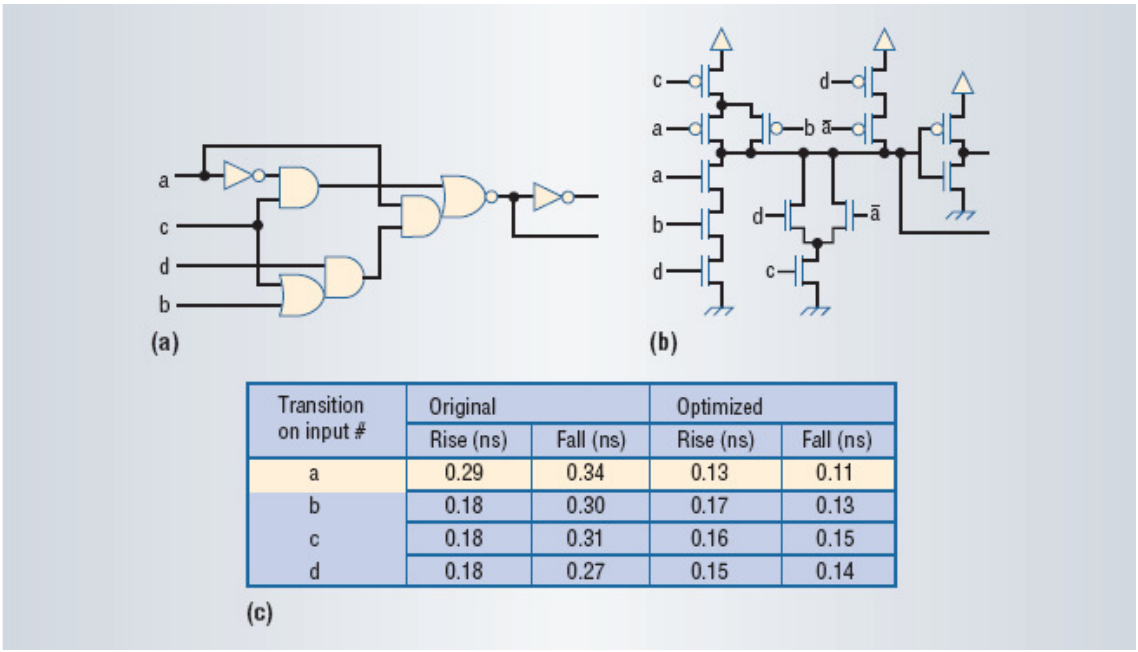
In a very succinct way, the flex cell approach proposed by Roy (2005) consists in the automatic generation of tactic cells. Flex cells are generated specifically to the design at issue and their characteristics are determined according to the local design constraints of the nodes in which they will be inserted. The flex cell approach happens in a post-processing stage, when technology mapping is over. The final result is an optimized circuit, made up of standard cells from a standard cell library and of flex cells. These flex cells are generated in a way to be layout compatible with the standard cells.

The starting point of the flex cell approach is the STA analysis of the circuit to be optimized. As a result of such analysis, the regions of the circuit that are prone to go through local optimization are detected. Each region is composed of a cluster of one or more standard cells. The series of operations for detecting and delimiting each cluster to be optimized is named clustering process (ROY, 2005).

The clusters and their local design constraints are the starting point for the mapping process, which generates candidate flex cells to replace such clusters. These flex cells may be generated by a new sizing of predefined standard cells or by the generation of new transistor networks (with appropriate sizing) with the same logic functionalities of the clusters to be replaced (ROY, 2005). In Figure 3.4, there is an example of flex cell generation.

The clustering process detects the standard cell cluster depicted in Figure 3.4a, from which the mapping process generates the flex cell shown in Figure 3.4b. The table in Figure 3.4 shows the gain in timing, especially for input '*a*', which corresponds to the critical path of the original cluster. It is worthy to emphasize that the mapping process just described is a stage of the flex cell approach, it should not be confused with the technology mapping, which is part of every cell-based design.

Figure 3.4 – Example of flex cell generation



Transition on input #	Original		Optimized	
	Rise (ns)	Fall (ns)	Rise (ns)	Fall (ns)
a	0.29	0.34	0.13	0.11
b	0.18	0.30	0.17	0.13
c	0.18	0.31	0.16	0.15
d	0.18	0.27	0.15	0.14

(c)

Source: Roy (2005).

The flex cell approach, as well as the sizing algorithms seen in subsections 3.4.4 and 3.4.5, perform the sizing of cell-based design digital integrated circuits. The method of Boyd (2005) seen in subsection 3.4.2 may also be used in cell-based design methodology, as long as the rounding problem seen in section 2.4 is taken into account. However, there is an important difference between the flex cell approach and the three other sizing methods just mentioned: meanwhile the latter use just standard cells already defined in the library to map the circuit, the former approach generates on the fly the flex cells. These newly created cells need to be characterized, which happens in the very mapping process. Since each flex cell is used in a specific context – in terms of local design constraints and input signal combinations –, its characterization is much simpler than that of a conventional standard cell (ROY, 2005).

The flex cell approach is an example of local optimization in which both the original topology and sizing of the circuit may be modified.

4 ANALYTICAL METHOD FOR MINIMIZING THE ACTIVE AREA OF DIGITAL SUBCIRCUITS UNDER DELAY CONSTRAINT

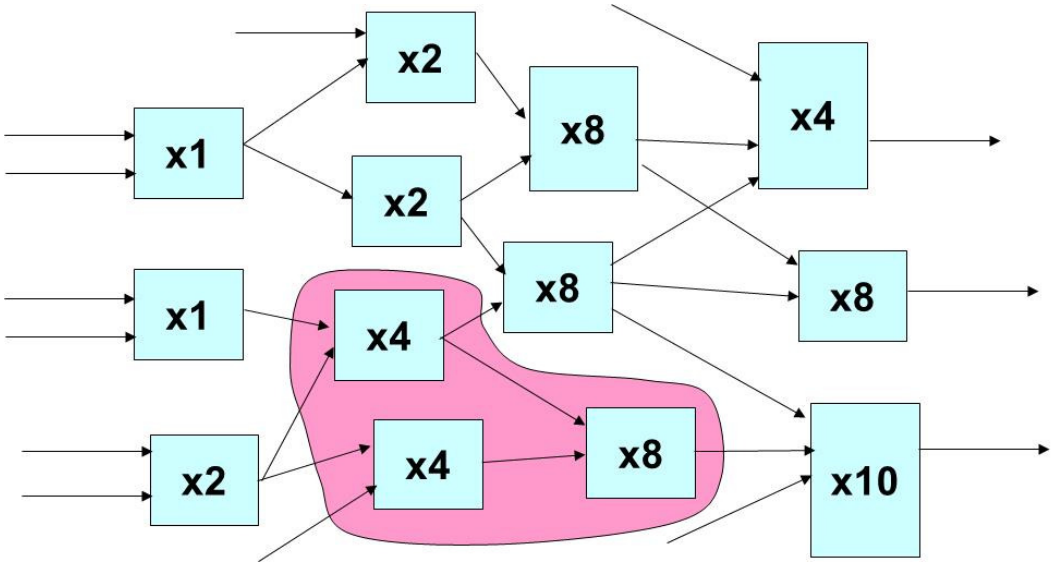
In this chapter, it is developed the specific contribution of this thesis, namely, a method for sizing subcircuits that shall respect the design constraints. As seen in section 2.6, mathematical analysis offers a straightforward way to size digital subcircuits. The proposed sizing method relies on this mathematical technique and on the logical effort delay model to come to a sort of generalization of the logical effort sizing method. Meanwhile the logical effort sizing method only delivers circuits with minimum achievable delay, the proposed method aims at achieving minimum active area for a specific delay D. Therefore, the expression for the total active area of the subcircuit is derived with respect to the size of each logic gate. This size can be represented either as the input capacitance of the logic gate or as the scale factor (BOYD, 2005) of such gate, since the scale factor is the ratio between the input capacitance of the logic gate and the input capacitance of the corresponding seed size in NANGATE 45nm Open Cell Library (NOCL). When the delay constraint D is made equal to the minimum achievable delay of the circuit, the proposed method delivers exactly the same results as the logical effort sizing method.

This chapter is organized as follows. Initially, it is explained how the proposed sizing method is positioned in the design flow collectively developed in the laboratory in which this thesis was elaborated. Next, the specific contribution of this thesis is emphasized. After that, the sizing method is derived. To do so, the digital subcircuits to be sized are classified according to three orthogonal criteria, since a specific set of equations must be derived for each category. These criteria are as follows: a) fixed or variable input capacitance; b) number of stages in the subcircuit; c) fanout free or branched subcircuits. Initially, all subcircuits are supposed to be fanout free. These subcircuits are then analyzed for fixed input capacitance (section 4.3), considering two-stage and three-stage subcircuits (sections 4.3.1 and 4.3.2, respectively). Then, in section 4.4, the same approach is applied for subcircuits with variable input capacitance. Next, in section 4.5, the implications of subcircuits with branching are taken into account. Finally, section 4.6 brings a discussion about how the proposed analytical method may be used in the evaluation of power delay product.

4.1 Sizing Method Contextualization

As mentioned in the introduction, the proposed sizing method is tailored for local optimization. The need for such sizing method arose in the context of the research being conducted in the laboratory, as it will be explained in this section. Figure 4.1 shows a digital circuit to be optimized. This circuit has already been previously sized, and each one of the thirteen boxes represents a standard cell, with the corresponding scale factor in it. Despite the previous sizing, there is room for further improvement in this circuit, which may be achieved via local optimization.

Figure 4.1 – Laboratory approach for local optimization

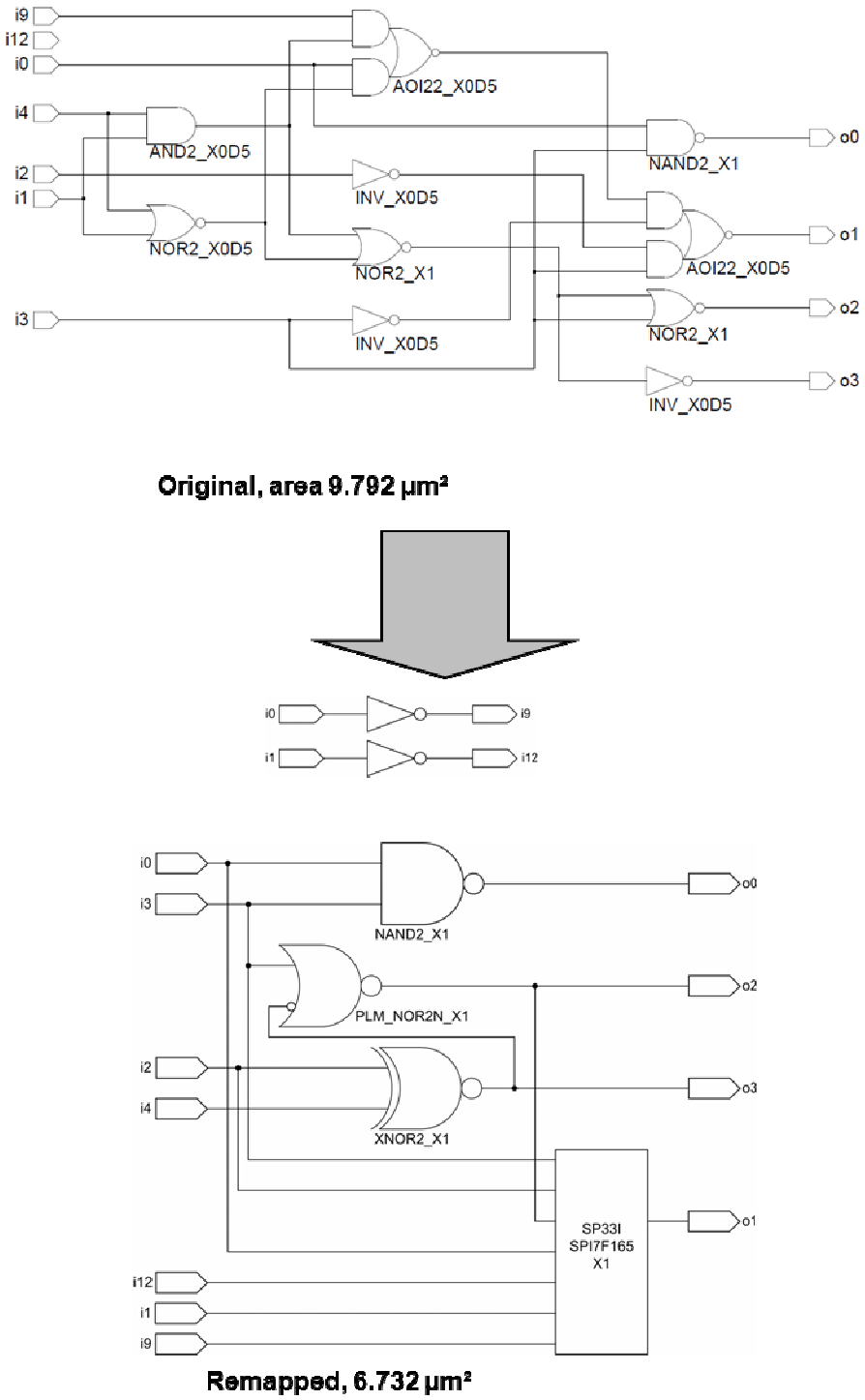


Source: Logics (2013).

Initially, a subcircuit is selected, e.g., the three shaded standard cells in Figure 4.1. In the laboratory approach, the selection algorithm is the KL–Cuts algorithm described by Martinello (2010), which finds a subcircuit with $K = 4$ inputs and $L = 2$ outputs. This algorithm was chosen since it preserves the logic relationship among circuit inputs, which is important for using highly Boolean methods of optimization such as functional composition (REIS, 2009; MARTINS, 2012) or read-polarity-once Boolean functions (CALLEGARO, 2013). These Boolean optimization methods can rewrite the logic functions in several different ways, which eventually leads to a minimal area solution, as depicted in Figure 4.2. It is worthy to mention that the subcircuit under remapping in Figure 4.2 is distinct from the shaded subcircuit in Figure 4.1, it is a different example.

The remapped subcircuit in Figure 4.2 has the same logical functionalities as the original subcircuit, but its outputs must be buffered before being reinserted again in the original circuit. This is exactly the point in which the proposed sizing method fits into the laboratory approach for local optimization: the contribution of this thesis is an analytical solution for the sizing algorithm needed at the outputs.

Figure 4.2 – Example of local remapping



Source: Logics (2013).

4.2 Contribution of this Thesis

The original contribution of this thesis is the development of a subcircuit sizing method appropriate for obtaining minimum active area solutions, taking into

consideration the maximum input capacitance, the output load to be driven, and the imposed delay constraint.

The method is based on the logical effort formulation, but it was derived for obtaining minimum area (under design constraints) rather than minimum delay. The original logical effort sizing method solves the delay equation analytically and concludes that, in order to achieve minimum delay, all the stages in the logic path must bear the same effort. However, this is costly in terms of power and area.

The proposed method, instead, computes the subcircuit area derivative, and thus finds an analytical solution for minimum area. For modern sizing, minimizing area under delay constraints is much more important than minimizing delay.

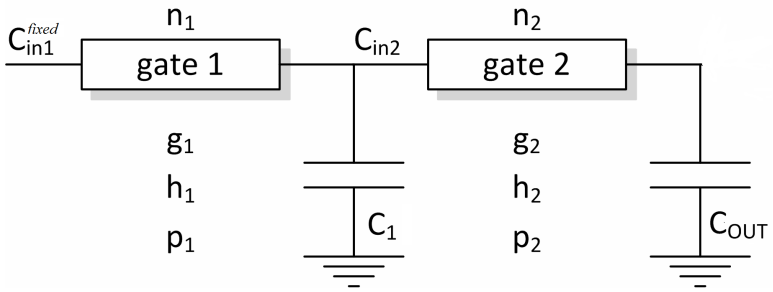
4.3 Fixed Input Capacitance

The proposed method was developed so that the input capacitance of the subcircuit under design may have either a fixed or a variable value. Each case has its own set of equations, and they are shown in different sections. These two approaches can be combined, allowing us to size a subcircuit with a variable, yet limited, input capacitance. In terms of subcircuit input capacitance, this is the ultimate goal of a subcircuit sizing method.

4.3.1 Two-stage subcircuits

In this subsection, the proposed method is deducted for a 2-stage fanout free subcircuit, with fixed topology, fixed extra parasitic capacitances, and fixed subcircuit input capacitance, as depicted in Figure 4.3:

Figure 4.3 – Model of a 2-stage subcircuit with fixed input capacitance



Source: Logics (2013).

In the subcircuit shown, parameters g_i, h_i, and p_i (i = 1, 2) come from the logical effort delay model. n_i is the ratio between the total input capacitance of gate i and the capacitance of the input pin of gate i that belongs to the logic path under analysis. For a symmetric gate (SUTHERLAND, 1999), the numerical value of n_i is equal to the number of input pins of gate i. C_{ini} is the capacitance of the input pin of gate i that belongs to the logic path. Since the subcircuit has a fixed input capacitance, it is labeled C_{in1}^{fixed}. C_1 is a fixed extra parasitic capacitance. C_{out} is the output capacitance, which might encompass another fixed extra parasitic capacitance in the output of gate 2.

This is a straightforward case, since there are only two stages, and the input capacitance of the first one has already been defined. Therefore, there is just one variable left, namely, C_{in2}, which can be found algebraically. In this particular case, there is no need for mathematical analysis. According to the logical effort delay model, the delay equation may be expressed as:

$$D = g_1\left(\frac{C_{in2} + C_1}{C_{in1}^{fixed}}\right) + g_2\frac{C_{out}}{C_{in2}} + P \tag{4.1}$$

where $P = p_1 + p_2$. Solving this equation for the variable C_{in2}, we have:

$$DC_{in1}^{fixed}C_{in2} = g_1(C_{in2} + C_1)C_{in2} + g_2C_{out}C_{in1}^{fixed} + PC_{in1}^{fixed}C_{in2}$$

$$g_1C_{in2}^2 + (g_1C_1 - DC_{in1}^{fixed})C_{in2} + g_2C_{out}C_{in1}^{fixed} + PC_{in1}^{fixed}C_{in2} = 0$$

$$g_1C_{in2}^2 + (g_1C_1 + (P-D)C_{in1}^{fixed})C_{in2} + g_2C_{out}C_{in1}^{fixed} = 0$$

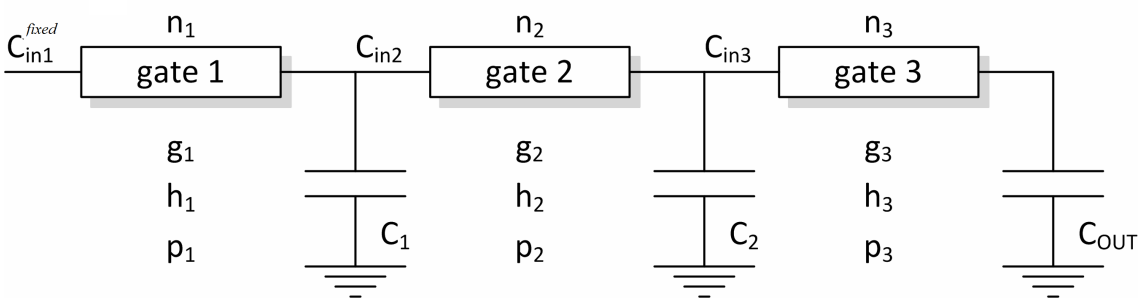
$$\therefore C_{in2} = \frac{-g_1C_1 + (D-P)C_{in1}^{fixed} \pm \left((g_1C_1 + (P-D)C_{in1}^{fixed})^2 - 4g_1g_2C_{out}C_{in1}^{fixed}\right)^{1/2}}{2g_1} \tag{4.2}$$

Equation (4.2) can give either positive or negative results for C_{in2}. Negative values are physically meaningless and, in the case of two distinct positive values, one shall choose the smallest one, since the objective is to attain the minimum active area.

4.3.2 Three-stage subcircuits

In this subsection, the proposed method is deducted for a 3-stage fanout free subcircuit, with fixed topology, fixed extra parasitic capacitances, and fixed subcircuit input capacitance, as shown in Figure 4.4:

Figure 4.4 – Model of a 3-stage subcircuit with fixed input capacitance



Source: Logics (2013).

Just like the previous subsection, in the subcircuit shown, parameters g_i, h_i, and p_i (i = 1, 2, 3) come from the logical effort delay model. n_i is the ratio between the total input capacitance of gate i and the capacitance of the input pin of gate i that belongs to the logic path under analysis. For a symmetric gate, the numerical value of n_i is equal to the number of input pins of gate i. C_{ini} is the capacitance of the input pin of gate i that belongs to the logic path. C_1 and C_2 are fixed extra parasitic capacitances. C_{out} is the

output capacitance, which might encompass another fixed extra parasitic capacitance in the output of gate 3.

Following the logical effort delay model, the total relative delay (D) for the subcircuit depicted in Figure 4.4 is specified by a delay constraint and is given by:

$$D = (g_1 h_1 + g_2 h_2 + g_3 h_3 + p_1 + p_2 + p_3) \tag{4.3}$$

Since $h_1 = (C_{in2} + C_1)/C_{in1}$, $h_2 = (C_{in3} + C_2)/C_{in2}$, $h_3 = C_{out}/C_{in3}$, $p_1 + p_2 + p_3 = P$, and $C_{in1} = C_{in1}^{fixed}$, equation (4.3) may be rewritten as

$$D = g_1 \frac{(C_{in2} + C_1)}{C_{in1}^{fixed}} + g_2 \frac{(C_{in3} + C_2)}{C_{in2}} + g_3 \frac{C_{out}}{C_{in3}} + P \tag{4.4}$$

Eliminating the denominators, we have:

$$\begin{aligned} g_1 C_{in3} C_{in2}^2 + (g_1 C_1 C_{in3} + C_{in1}^{fixed} g_3 C_{out} + C_{in1}^{fixed} P C_{in3} - C_{in1}^{fixed} D C_{in3}) C_{in2} \\ + C_{in1}^{fixed} g_2 C_{in3} (C_{in3} + C_2) = 0 \end{aligned} \tag{4.5}$$

Equation (4.5) shows the relationship between the variables C_{in2} and C_{in3} so that all design constraints for the subcircuit are fulfilled. Since the input capacitance of the subcircuit is fixed (C_{in1}^{fixed}), C_{in2} and C_{in3} are the only variables in this problem. Moreover, equation (4.5) may be regarded as a univariate polynomial equation of the second degree on C_{in2}. Therefore, C_{in2} may be expressed as a function of C_{in3}, since all the other terms are constant parameters.

$$C_{in2} = \frac{-\beta \pm \sqrt{\beta^2 - 4 g_1 C_{in3} \gamma}}{2 g_1 C_{in3}} \tag{4.6}$$

where:

$$\beta = g_1 C_1 C_{in3} + C_{in1}^{fixed} g_3 C_{out} - C_{in1}^{fixed} (D - P) C_{in3} \tag{4.7}$$

$$\gamma = C_{in1}^{fixed} g_2 C_{in3} (C_{in3} + C_2) \tag{4.8}$$

The new variables β and γ were introduced just for the sake of better visualization of the equations. According to Boyd (2005), Kasamsetty (2000), Tennakoon (2008), Joshi (2008), Lefebvre (1997), and Otten (2000), the active area of the subcircuit in Figure 4.4 may be considered as

$$A(C_{in2}, C_{in3}) = n_1 C_{in1}^{fixed} + n_2 C_{in2} + n_3 C_{in3}. \tag{4.9}$$

I.e., the active area of a logic gate is monotonically related to its total input capacitance. Replacing the expression obtained in equation (4.6) for C_{in2} into (4.9), $A(C_{in2}, C_{in3})$ becomes a univariate equation on C_{in3}:

$$A(C_{in3}) = n_1 C_{in1}^{fixed} + n_2 \frac{(-\beta \pm \sqrt{\beta^2 - 4 g_1 C_{in3} \gamma})}{2 g_1 C_{in3}} + n_3 C_{in3} \tag{4.10}$$

Now, we find the derivatives of β, γ, and ν (new variable). This was done so in order to find the final expression for the derivative of $A(C_{in3})$. Otherwise, the equations would be too long to fit in the page.

$$\beta' = \frac{d\beta(C_{in3})}{dC_{in3}} = g_1 C_1 - C_{in1}^{fixed} (D - P) \tag{4.11}$$

$$\gamma' = \frac{d\gamma(C_{in3})}{dC_{in3}} = 2C_{in1}^{fixed} g_2 C_{in3} + C_{in1}^{fixed} g_2 C_2 \tag{4.12}$$

$$\nu = \beta^2 - 4g_1 C_{in3}\gamma \tag{4.13}$$

$$\nu = \left(g_1 C_1 C_{in3} + C_{in1}^{fixed} g_3 C_{out} - C_{in1}^{fixed}(D-P)C_{in3}\right)^2 - 4g_1 g_2 C_{in1}^{fixed} C_{in3}^2 (C_{in3} + C_2) \tag{4.14}$$

$$\nu' = \frac{d\nu(C_{in3})}{dC_{in3}} = 2\beta\beta' - 4(g_1\gamma + g_1 C_{in3}\gamma') \tag{4.15}$$

$$\begin{aligned}\nu' = {} & 2g_1^2 C_1^2 C_{in3} + 2C_{in1}^{fixed} g_1 g_3 C_1 C_{out} \\ & - 4C_{in1}^{fixed} g_1 C_1 (D-P) C_{in3} \\ & - 2C_{in1}^{fixed\,2} g_3 C_{out} (D-P) \\ & + 2C_{in1}^{fixed\,2} (D-P)^2 C_{in3} \\ & - 4(3C_{in1}^{fixed} g_1 g_2 C_{in3}^2 + 2C_{in1}^{fixed} g_1 g_2 C_2 C_{in3})\end{aligned} \tag{4.16}$$

Taking the derivative of $A(C_{in3})$ in equation (4.10) and introducing the expressions in (4.11)-(4.16), we have:

$$\frac{dA(C_{in3})}{dC_{in3}} = \frac{n_2}{2g_1 C_{in3}}\left(-\beta' \pm \frac{\nu'}{2\nu^{1/2}}\right) + \frac{n_2}{2g_1 C_{in3}^2}\left(\beta \mp \nu^{1/2}\right) + n_3 \tag{4.17}$$

At this point, it is straightforward to obtain the minimum active area; it suffices to find the zeroes of equation (4.17):

$$n_2\left[\left(-\beta' \pm \frac{\nu'}{2\nu^{1/2}}\right) g_1 C_{in3} + (\beta \mp \nu^{1/2}) g_1\right] + 2n_3 (g_1 C_{in3})^2 = 0 \tag{4.18}$$

Equation (4.18) can be solved numerically, giving C_{in3} for minimum active area. Solving equation (4.6) with C_{in3} just obtained, we have the corresponding value of C_{in2}. Eventually, more than one pair of values may be attained, but only one of them corresponds to minimum active area. Besides, due to the mathematical formulation of the problem, either negative or nonreal values for C_{in2} and C_{in3} may be obtained from equations (4.6) and (4.18). However, discarding these inconsistent results, there is always one and only one pair of values for minimum active area.

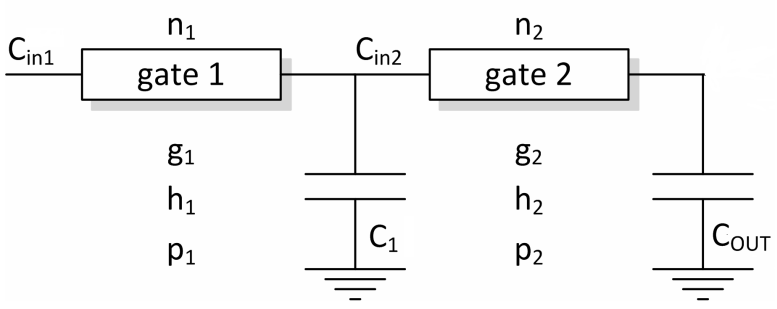
4.4 Variable Input Capacitance

The proposed sizing method is capable of coping with both fixed input capacitance and variable input capacitance. In the next two subsections, it is derived the set of equations for variable input capacitance.

4.4.1 Two-stage subcircuits

In this subsection, the proposed method is deducted for a 2-stage fanout free subcircuit, with fixed topology, fixed extra parasitic capacitances, and variable subcircuit input capacitance, as depicted in Figure 4.5:

Figure 4.5 – Model of a 2-stage subcircuit with variable input capacitance



Source: Logics (2013).

Just like subsection 4.3.1, in the current subcircuit, parameters g_i, h_i, and p_i (i = 1, 2) come from the logical effort delay model. n_i is the ratio between the total input capacitance of gate i and the capacitance of the input pin of gate i that belongs to the logic path under analysis. For a symmetric gate, the numerical value of n_i is equal to the number of input pins of gate i. C_{ini} is the capacitance of the input pin of gate i that belongs to the logic path. C_1 is a fixed extra parasitic capacitance. C_{out} is the output capacitance, which might include another fixed extra parasitic capacitance in the output of gate 2.

Unlike section 4.3.1, now C_{in1} has a variable value. Therefore, there are two variables in this problem, namely, C_{in1} and C_{in2}. According to the logical effort delay model, the delay equation may be expressed as:

$$D = (g_1 h_1 + g_2 h_2 + p_1 + p_2) \tag{4.19}$$

Since $h_1 = (C_{in2} + C_1)/C_{in1}$, $h_2 = C_{out}/C_{in2}$, and $p_1 + p_2 = P$, equation (4.19) may be rewritten as

$$D = g_1 \frac{(C_{in2} + C_1)}{C_{in1}} + g_2 \frac{C_{out}}{C_{in2}} + P \tag{4.20}$$

Rearranging the terms in the last equation, one can express C_{in1} as a function of C_{in2}:

$$C_{in1} = \frac{g_1 (C_{in2} + C_1)}{D - P - g_2 \dfrac{C_{out}}{C_{in2}}} \tag{4.21}$$

According to Boyd (2005), Kasamsetty (2000), Tennakoon (2008), Joshi (2008), Lefebvre (1997), and Otten (2000), the active area of the subcircuit in Figure 4.5 may be regarded as

$$A(C_{in1}, C_{in2}) = n_1 C_{in1} + n_2 C_{in2}\,. \tag{4.22}$$

Replacing the expression obtained in equation (4.21) for C_{in1} into equation (4.22), $A(C_{in1}, C_{in2})$ becomes a univariate equation on C_{in2}:

$$A(C_{in2}) = \frac{n_1 g_1 (C_{in2} + C_1)}{D - P - g_2 \dfrac{C_{out}}{C_{in2}}} + n_2 C_{in2} \tag{4.23}$$

or

$$A(C_{in2}) = \frac{n_1 g_1 (C_{in2} + C_1) C_{in2}}{(D-P)C_{in2} - g_2 C_{out}} + n_2 C_{in2} \tag{4.24}$$

as long as $C_{in2} \neq \frac{g_2 C_{out}}{D-P}$. From the definition of the logical effort delay model, it can be seen that $D > P$. Now that we have $A(C_{in2})$, the values of C_{in2} that correspond to the minimum active area are given by the solutions of the equation $\frac{dA(C_{in2})}{dC_{in2}} = 0$. For the sake of an easier algebraic manipulation, let the functions:

$$u(C_{in2}) = n_1 g_1 (C_{in2} + C_1) C_{in2} \tag{4.25}$$

$$v(C_{in2}) = (D-P)C_{in2} - g_2 C_{out} \tag{4.26}$$

Therefore, $A(C_{in2})$ may be written as:

$$A(C_{in2}) = \frac{u(C_{in2})}{v(C_{in2})} + n_2 C_{in2} \tag{4.27}$$

Hence,

$$A'(C_{in2}) = \frac{dA(C_{in2})}{dC_{in2}} = \frac{u'(C_{in2})v(C_{in2}) - u(C_{in2})v'(C_{in2})}{v^2(C_{in2})} + n_2 \tag{4.28}$$

Note that:

$$u(C_{in2}) = n_1 g_1 (C_{in2} + C_1) C_{in2} = n_1 g_1 C_{in2}^2 + n_1 g_1 C_1 C_{in2}$$

$$\Rightarrow u'(C_{in2}) = \frac{du(C_{in2})}{dC_{in2}} = 2n_1 g_1 C_{in2} + n_1 g_1 C_1 \tag{4.29}$$

and

$$v(C_{in2}) = (D-P)C_{in2} - g_2 C_{out} \Rightarrow v'(C_{in2}) = \frac{dv(C_{in2})}{dC_{in2}} = D - P \tag{4.30}$$

Therefore,

$$u'v - uv' = (2n_1 g_1 C_{in2} + n_1 g_1 C_1)((D-P)C_{in2} - g_2 C_{out}) - n_1 g_1 (C_{in2} + C_1) C_{in2} (D-P) =$$

$$= 2n_1 g_1 (D-P) C_{in2}^2 - 2n_1 g_1 g_2 C_{out} C_{in2} + n_1 g_1 C_1 (D-P) C_{in2} - n_1 g_1 g_2 C_1 C_{out}$$
$$- n_1 g_1 (D-P)(C_{in2}^2 + C_1 C_{in2})$$

$$= n_1 g_1 (D-P) C_{in2}^2 - 2n_1 g_1 g_2 C_{out} C_{in2} - n_1 g_1 g_2 C_1 C_{out} \tag{4.31}$$

Replacing equation (4.31) into (4.28) and setting it to zero, we have:

$$\frac{dA(C_{in2})}{dC_{in2}} = \frac{n_1 g_1 (D-P) C_{in2}^2 - 2n_1 g_1 g_2 C_{out} C_{in2} - n_1 g_1 g_2 C_1 C_{out}}{((D-P)C_{in2} - g_2 C_{out})^2} + n_2 = 0$$

Eliminating the denominator, we have:

$$n_1 g_1 (D-P) C_{in2}^2 - 2n_1 g_1 g_2 C_{out} C_{in2} - n_1 g_1 g_2 C_1 C_{out}$$
$$+ n_2 ((D-P)^2 C_{in2}^2 - 2g_2 C_{out} (D-P) C_{in2} + g_2^2 C_{out}^2) = 0$$

After algebraic manipulation, we have:

$$(n_1 g_1 (D-P) + n_2 (D-P)^2) C_{in2}^2 - 2(n_1 g_1 g_2 C_{out} + n_2 g_2 C_{out} (D-P)) C_{in2} - n_1 g_1 g_2 C_1 C_{out} + n_2 g_2^2 C_{out}^2 = 0 \quad (4.32)$$

Written this way, equation (4.32) is a univariate polynomial equation of the second degree on C_{in2}, whose solutions are given by:

$$C_{in2} = \frac{\kappa \pm \sqrt{\kappa^2 - 4\eta\lambda}}{2\eta} \quad (4.33)$$

where:

$$\eta = n_1 g_1 (D-P) + n_2 (D-P)^2 \quad (4.34)$$

$$\kappa = 2(n_1 g_1 g_2 C_{out} + n_2 g_2 C_{out} (D-P)) \quad (4.35)$$

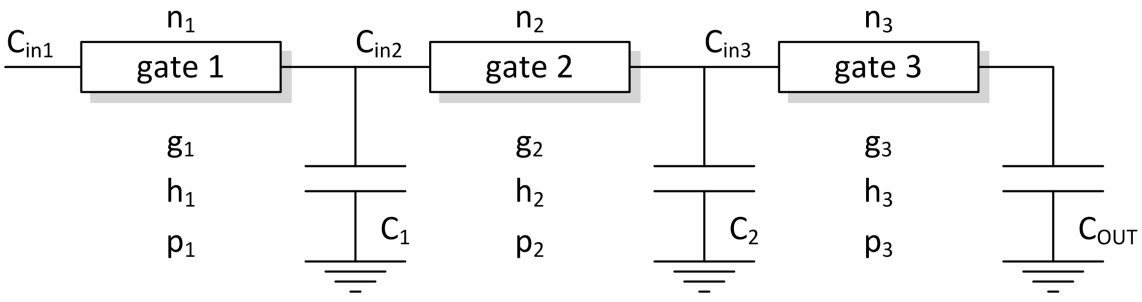
$$\lambda = n_2 g_2^2 C_{out}^2 - n_1 g_1 g_2 C_1 C_{out} \quad (4.36)$$

The new variables η, κ, and λ were introduced for the sake of better visualization of equation (4.33). The pair of equations (4.21) and (4.33) defines the sizing of the subcircuit. Since these equations have been algebraically solved, the numerical values are obtained by straightforward calculation. Eventually, two pairs of values for C_{in1} and C_{in2} may be obtained, but only one of them corresponds to the minimum active area.

4.4.2 Three-stage subcircuits

In this subsection, the proposed method is deducted for a 3-stage fanout free subcircuit, with fixed topology, fixed extra parasitic capacitances, and variable subcircuit input capacitance, as depicted in Figure 4.6:

Figure 4.6 – Model of a 3-stage subcircuit with variable input capacitance



Source: Logics (2013).

Just like subsection 4.3.2, in the subcircuit shown, parameters g_i, h_i, and p_i (i = 1, 2, 3) come from the logical effort delay model. n_i is the ratio between the total input capacitance of gate i and the capacitance of the input pin of gate i that belongs to the logic path under analysis. For a symmetric gate, the numerical value of n_i is equal to the number of input pins of gate i. C_{ini} is the capacitance of the input pin of gate i that belongs to the logic path. C_1 and C_2 are fixed extra parasitic capacitances. C_{out} is the

output capacitance, which might encompass another fixed extra parasitic capacitance in the output of gate 3.

Unlike subsection 4.3.2, now C_{in1} has a variable value. Therefore, there are three variables in this problem, namely, C_{in1}, C_{in2}, and C_{in3}. According to the logical effort delay model, the delay equation for each stage may be expressed as:

$$d_i = g_i h_i + p_i, \ i = 1,2,3 \tag{4.37}$$

$$\therefore D = \sum_{i=1}^{3} d_i = (g_1 h_1 + g_2 h_2 + g_3 h_3 + p_1 + p_2 + p_3) \tag{4.38}$$

Since $h_1 = (C_{in2} + C_1)/C_{in1}$, $h_2 = (C_{in3} + C_2)/C_{in2}$, $h_3 = C_{out}/C_{in3}$, and $p_1 + p_2 + p_3 = P$, equation (4.38) may be rewritten as

$$D = \frac{g_1}{C_{in1}}(C_{in2} + C_1) + \frac{g_2}{C_{in2}}(C_{in3} + C_2) + \frac{g_3 C_{out}}{C_{in3}} + P \tag{4.39}$$

Isolating C_{in1} in the last equation:

$$C_{in1} = \frac{g_1(C_{in2} + C_1)}{D - P - \frac{g_2}{C_{in2}}(C_{in3} + C_2) - g_3 \frac{C_{out}}{C_{in3}}} \tag{4.40}$$

As stated before, according to Boyd (2005), Kasamsetty (2000), Tennakoon (2008), Joshi (2008), Lefebvre (1997), and Otten (2000), the active area of the subcircuit in Figure 4.6 is given by:

$$A(C_{in1}, C_{in2}, C_{in3}) = n_1 C_{in1} + n_2 C_{in2} + n_3 C_{in3}. \tag{4.41}$$

Replacing the expression obtained in equation (4.40) for C_{in1} into equation (4.41), $A(C_{in1}, C_{in2}, C_{in3})$ becomes a bivariate equation:

$$A(C_{in2}, C_{in3}) = \frac{n_1 g_1(C_{in2} + C_1)}{D - P - \frac{g_2}{C_{in2}}(C_{in3} + C_2) - g_3 \frac{C_{out}}{C_{in3}}} + n_2 C_{in2} + n_3 C_{in3}$$

Therefore, after some algebraic manipulation:

$$A(C_{in2}, C_{in3}) = \frac{n_1 g_1(C_{in2} + C_1) C_{in2} C_{in3}}{(D - P) C_{in2} C_{in3} - g_2 C_{in3}(C_{in3} + C_2) - g_3 C_{out} C_{in2}} + n_2 C_{in2} + n_3 C_{in3}$$

And hence:

$$A(C_{in2}, C_{in3}) = \frac{n_1 g_1 C_{in2}^2 C_{in3} + n_1 g_1 C_1 C_{in2} C_{in3}}{(D - P) C_{in2} C_{in3} - g_2 C_{in3}^2 - g_2 C_2 C_{in3} - g_3 C_{out} C_{in2}} + n_2 C_{in2} + n_3 C_{in3} \tag{4.42}$$

According to subsection 2.6.1, the minimum active area is attained when the partial derivatives of $A(C_{in2}, C_{in3})$ are zero. That is,

$$\frac{\partial A(C_{in2}, C_{in3})}{\partial C_{in2}} = 0 \tag{4.43}$$

and

$$\frac{\partial A(C_{in2}, C_{in3})}{\partial C_{in3}} = 0\,. \tag{4.44}$$

For the sake of an easier algebraic manipulation, let the functions:

$$u(C_{in2}) = n_1 g_1 C_{in2}^2 C_{in3} + n_1 g_1 C_1 C_{in2} C_{in3} \tag{4.45}$$

$$v(C_{in2}) = (D-P) C_{in2} C_{in3} - g_2 C_{in3}^2 - g_2 C_2 C_{in3} - g_3 C_{out} C_{in2} \tag{4.46}$$

It is easy to notice that $u(C_{in2})$ corresponds to the numerator of the ratio in equation (4.42), meanwhile $v(C_{in2})$ corresponds to the denominator in this fraction. In a truly rigorous notation, the functions $u(C_{in2})$ and $v(C_{in2})$ should be written as $u(C_{in2}, C_{in3})$ and $v(C_{in2}, C_{in3})$ respectively, since they are functions of both C_{in2} and C_{in3}. However, when dealing with partial derivatives with respect to one specific variable, the other variables may be considered as constant values, which justifies the current notation. Later on, these same functions are regarded as $u(C_{in3})$ and $v(C_{in3})$. Replacing equations (4.45) and (4.46) into (4.42), the partial derivative of $A(C_{in2}, C_{in3})$ with respect to C_{in2} may be expressed as:

$$A(C_{in2}, C_{in3}) = \frac{u(C_{in2})}{v(C_{in2})} + n_2 C_{in2} + n_3 C_{in3} \tag{4.47}$$

Therefore,

$$\frac{\partial A(C_{in2}, C_{in3})}{\partial C_{in2}} = \frac{u'(C_{in2}) v(C_{in2}) - u(C_{in2}) v'(C_{in2})}{v^2(C_{in2})} + n_2 \tag{4.48}$$

Note that

$$u'(C_{in2}) = \frac{\partial u(C_{in2})}{\partial C_{in2}} = 2 n_1 g_1 C_{in2} C_{in3} + n_1 g_1 C_1 C_{in3} \tag{4.49}$$

and

$$v'(C_{in2}) = \frac{\partial v(C_{in2})}{\partial C_{in2}} = (D-P) C_{in3} - g_3 C_{out} \tag{4.50}$$

Therefore,

$$\begin{aligned} u'(C_{in2}) v(C_{in2}) = {} & (2 n_1 g_1 C_{in2} C_{in3} + n_1 g_1 C_1 C_{in3}) \cdot \\ & \cdot [(D-P) C_{in2} C_{in3} - g_2 C_{in3}^2 - g_2 C_2 C_{in3} - g_3 C_{out} C_{in2}] \end{aligned}$$

Rearranging the terms:

$$\begin{aligned} u'(C_{in2}) v(C_{in2}) = {} & (2 n_1 g_1 C_{in2} C_{in3} + n_1 g_1 C_1 C_{in3}) \cdot \\ & \cdot \{[(D-P) C_{in3} - g_3 C_{out}] C_{in2} - g_2 C_{in3} (C_{in3} + C_2)\} \end{aligned}$$

Expanding the terms:

$$u'(C_{in2})v(C_{in2}) = 2n_1g_1C_{in3}[(D-P)C_{in3} - g_3C_{out}]C_{in2}^2 - 2n_1g_1g_2C_{in3}^2(C_{in3}+C_2)C_{in2} + n_1g_1C_1C_{in3}[(D-P)C_{in3} - g_3C_{out}]C_{in2} - n_1g_1g_2C_1C_{in3}^2(C_{in3}+C_2)$$

Expressing as a polynomial equation of the second degree on C_{in2}:

$$u'(C_{in2})v(C_{in2}) = 2n_1g_1C_{in3}[(D-P)C_{in3} - g_3C_{out}]C_{in2}^2 + \{n_1g_1C_1C_{in3}[(D-P)C_{in3} - g_3C_{out}] - 2n_1g_1g_2C_{in3}^2(C_{in3}+C_2)\}C_{in2} - n_1g_1g_2C_1C_{in3}^2(C_{in3}+C_2) \quad (4.51)$$

In the same fashion,

$$u(C_{in2})v'(C_{in2}) = (n_1g_1C_{in2}^2C_{in3} + n_1g_1C_1C_{in2}C_{in3})[(D-P)C_{in3} - g_3C_{out}]$$

Expanding the terms:

$$u(C_{in2})v'(C_{in2}) = n_1g_1(D-P)C_{in3}^2C_{in2}^2 - n_1g_1g_3C_{out}C_{in3}C_{in2}^2 + n_1g_1C_1(D-P)C_{in3}^2C_{in2} - n_1g_1g_3C_1C_{out}C_{in3}C_{in2}$$

Expressing as a polynomial equation of the second degree on C_{in2}:

$$u(C_{in2})v'(C_{in2}) = [n_1g_1(D-P)C_{in3}^2 - n_1g_1g_3C_{out}C_{in3}]C_{in2}^2 + [n_1g_1C_1(D-P)C_{in3}^2 - n_1g_1g_3C_1C_{out}C_{in3}]C_{in2} \quad (4.52)$$

Merging equations (4.51) and (4.52), we have:

$$u'(C_{in2})v(C_{in2}) - u(C_{in2})v'(C_{in2}) = [n_1g_1(D-P)C_{in3}^2 - n_1g_1g_3C_{out}C_{in3}]C_{in2}^2 - 2n_1g_1g_2C_{in3}^2(C_{in3}+C_2)C_{in2} - n_1g_1g_2C_1C_{in3}^2(C_{in3}+C_2) \quad (4.53)$$

Putting the result just obtained into equation (4.48), we have:

$$\frac{\partial A(C_{in2},C_{in3})}{\partial C_{in2}} = \frac{[n_1g_1(D-P)C_{in3}^2 - n_1g_1g_3C_{out}C_{in3}]C_{in2}^2}{\{[(D-P)C_{in3} - g_3C_{out}]C_{in2} - g_2C_{in3}(C_{in3}+C_2)\}^2} - \frac{2n_1g_1g_2C_{in3}^2(C_{in3}+C_2)C_{in2}}{\{[(D-P)C_{in3} - g_3C_{out}]C_{in2} - g_2C_{in3}(C_{in3}+C_2)\}^2} - \frac{n_1g_1g_2C_1C_{in3}^2(C_{in3}+C_2)}{\{[(D-P)C_{in3} - g_3C_{out}]C_{in2} - g_2C_{in3}(C_{in3}+C_2)\}^2} + \frac{n_2\{[(D-P)C_{in3} - g_3C_{out}]C_{in2} - g_2C_{in3}(C_{in3}+C_2)\}^2}{\{[(D-P)C_{in3} - g_3C_{out}]C_{in2} - g_2C_{in3}(C_{in3}+C_2)\}^2} \quad (4.54)$$

In order to find the values of C_{in2} and C_{in3} that satisfy equation (4.43), two conditions must be satisfied: a) the denominator in the right-hand side of equation (4.54) must be different from zero; and b) the corresponding numerator must be equal to zero. The first condition can be checked. The second condition leads to the following equation, after conveniently reordering the constants and variables:

$$\{n_1 g_1 (D-P) C_{in3}^2 - n_1 g_1 g_3 C_{out} C_{in3} + n_2 [(D-P) C_{in3} - g_3 C_{out}]^2\} C_{in2}^2$$
$$- \{2 n_1 g_1 g_2 C_{in3}^2 (C_{in3} + C_2) + 2 n_2 g_2 C_{in3} (C_{in3} + C_2)[(D-P) C_{in3} - g_3 C_{out}]\} C_{in2}$$
$$+ n_2 g_2^2 C_{in3}^2 (C_{in3} + C_2)^2 - n_1 g_1 g_2 C_1 C_{in3}^2 (C_{in3} + C_2) = 0 \tag{4.55}$$

Expressed in such a suitable way, equation (4.55) may be regarded as if it were a univariate polynomial equation of the second degree on C_{in2}. Doing so, the variable C_{in2} can be isolated from C_{in3} and its values are given by:

$$a C_{in2}^2 - b C_{in2} + c = 0 \Rightarrow C_{in2} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4.56}$$

where:

$$a = n_1 g_1 (D-P) C_{in3}^2 - n_1 g_1 g_3 C_{out} C_{in3} + n_2 [(D-P) C_{in3} - g_3 C_{out}]^2 \tag{4.57}$$

$$b = 2 n_1 g_1 g_2 C_{in3}^2 (C_{in3} + C_2) + 2 n_2 g_2 C_{in3} (C_{in3} + C_2)[(D-P) C_{in3} - g_3 C_{out}] \tag{4.58}$$

$$c = n_2 g_2^2 C_{in3}^2 (C_{in3} + C_2)^2 - n_1 g_1 g_2 C_1 C_{in3}^2 (C_{in3} + C_2) \tag{4.59}$$

The three new variables *a*, *b*, and *c* have been introduced for the sake of an easier algebraic manipulation. At this point, out of the three variables in the problem, C_{in1} has already been expressed as a function of C_{in2} and C_{in3} in equation (4.40), and C_{in2} has been expressed as a function of C_{in3} in equation (4.56). Therefore, C_{in3} is the only variable that remains to be determined. This can be done as follows. Equation (4.45) may now be expressed as:

$$u(C_{in3}) = n_1 g_1 C_{in2} (C_{in2} + C_1) C_{in3} \tag{4.60}$$

Idem for equation (4.46):

$$v(C_{in3}) = -g_2 C_{in3}^2 + [(D-P) C_{in2} - g_2 C_2] C_{in3} - g_3 C_{out} C_{in2} \tag{4.61}$$

Replacing equations (4.60) and (4.61) into (4.42), the partial derivative of $A(C_{in2}, C_{in3})$ with respect to C_{in3} may be expressed as:

$$A(C_{in2}, C_{in3}) = \frac{u(C_{in3})}{v(C_{in3})} + n_2 C_{in2} + n_3 C_{in3} \tag{4.62}$$

Therefore,

$$\frac{\partial A(C_{in2}, C_{in3})}{\partial C_{in3}} = \frac{u'(C_{in3}) v(C_{in3}) - u(C_{in3}) v'(C_{in3})}{v^2(C_{in3})} + n_3 \tag{4.63}$$

Since

$$u'(C_{in3}) = \frac{\partial u(C_{in3})}{\partial C_{in3}} = n_1 g_1 C_{in2} (C_{in2} + C_1) \tag{4.64}$$

and

$$v'(C_{in3}) = \frac{\partial v(C_{in3})}{\partial C_{in3}} = -2g_2C_{in3} + (D-P)C_{in2} - g_2C_2 \tag{4.65}$$

we have:

$$\begin{aligned} u'(C_{in3})v(C_{in3}) = &-n_1g_1g_2C_{in2}(C_{in2}+C_1)C_{in3}^2 \\ &+ n_1g_1C_{in2}(C_{in2}+C_1)[(D-P)C_{in2} - g_2C_2]C_{in3} \\ &- n_1g_1g_3(C_{in2}+C_1)C_{out}C_{in2}^2 \end{aligned} \tag{4.66}$$

and

$$\begin{aligned} u(C_{in3})v'(C_{in3}) = &-2n_1g_1g_2C_{in2}(C_{in2}+C_1)C_{in3}^2 \\ &+ n_1g_1[(D-P)C_{in2} - g_2C_2]C_{in2}(C_{in2}+C_1)C_{in3} \end{aligned} \tag{4.67}$$

Therefore, merging equations (4.66) and (4.67):

$$\begin{aligned} u'(C_{in3})v(C_{in3}) - u(C_{in3})v'(C_{in3}) = &\, n_1g_1g_2C_{in2}(C_{in2}+C_1)C_{in3}^2 \\ &- n_1g_1g_3(C_{in2}+C_1)C_{out}C_{in2}^2 \end{aligned} \tag{4.68}$$

Consequently, inserting equation (4.68) into (4.63), we have:

$$\begin{aligned} \frac{\partial A(C_{in2},C_{in3})}{\partial C_{in3}} = &\frac{n_1g_1g_2C_{in2}(C_{in2}+C_1)C_{in3}^2 - n_1g_1g_3(C_{in2}+C_1)C_{out}C_{in2}^2}{[-g_2C_{in3}^2 + [(D-P)C_{in2} - g_2C_2]C_{in3} - g_3C_{out}C_{in2}]^2} \\ &+ \frac{n_3[-g_2C_{in3}^2 + [(D-P)C_{in2} - g_2C_2]C_{in3} - g_3C_{out}C_{in2}]^2}{[-g_2C_{in3}^2 + [(D-P)C_{in2} - g_2C_2]C_{in3} - g_3C_{out}C_{in2}]^2} \end{aligned} \tag{4.69}$$

In order to find the values of C_{in2} and C_{in3} that satisfy equation (4.44), two conditions shall be obeyed: a) the denominator in the right-hand side of equation (4.69) must be non-null; and b) the corresponding numerator must be null. The first condition can be verified. The second condition takes us to the coming equation, after conveniently reordering the constants and variables:

$$\begin{aligned} &n_3g_2^2C_{in3}^4 + 2n_3[g_2^2C_2 - g_2(D-P)C_{in2}]C_{in3}^3 \\ &+[n_3(D-P)^2C_{in2}^2 - 2n_3g_2C_2(D-P)C_{in2} + n_3g_2^2C_2^2 \\ &+ 2n_3g_2g_3C_{out}C_{in2} + n_1g_1g_2C_{in2}(C_{in2}+C_1)]C_{in3}^2 \\ &+ 2n_3[g_2g_3C_2C_{out}C_{in2} - g_3C_{out}(D-P)C_{in2}^2]C_{in3} \\ &+ n_3g_3^2C_{out}^2C_{in2}^2 - n_1g_1g_3(C_{in2}+C_1)C_{out}C_{in2}^2 = 0 \end{aligned} \tag{4.70}$$

The triple of equations (4.40), (4.56), and (4.70) defines the sizing of the subcircuit. By replacing the expression for C_{in2} – equation (4.56) – into equation (4.70), we end up with a univariate equation for C_{in3}. This equation must be solved numerically, giving the value of C_{in3} corresponding to minimum active area. This value of C_{in3} allows to calculate C_{in2}, via equation (4.56). Finally, equation (4.40) gives the value of C_{in1} for

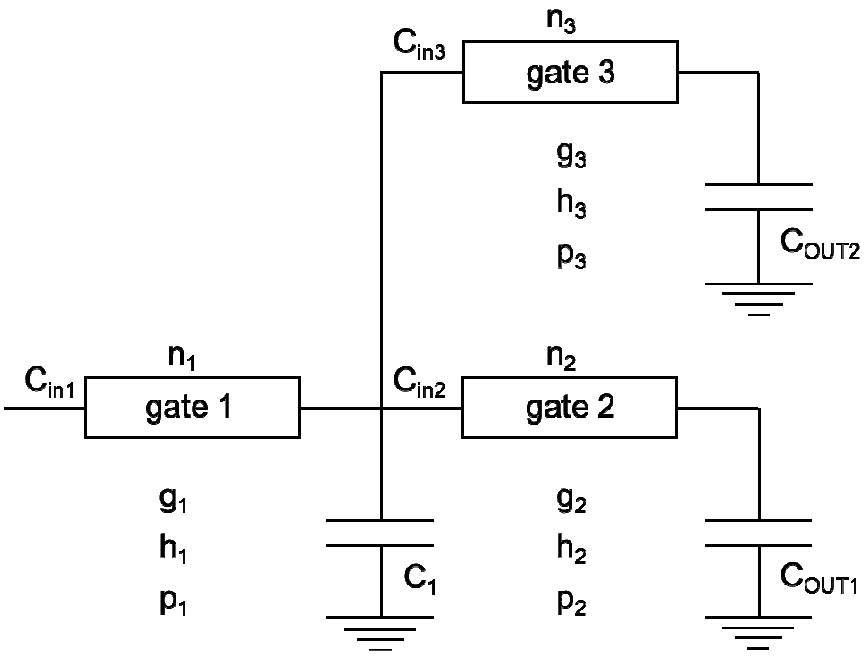
minimum active area. Eventually, more than one triple of values is attained, but only one of them corresponds to minimum active area. Besides, due to the mathematical formulation of the problem, either negative or complex values for C_{in2} and C_{in3} may be obtained from equations (4.56) and (4.70). However, after rejecting these inconsistent results, there is always one and only one triple of values for minimum active area.

4.5 Branching Subcircuits

In the previous sections, all the circuits analyzed were fanout free. However, the proposed sizing method is also suitable for branching subcircuits. In this section, the proposed method is deducted for a 2-stage branching subcircuit, with fixed topology, fixed extra parasitic capacitance, and variable subcircuit input capacitance, as depicted in Figure 4.7.

Just like subsection 4.3.2, in the subcircuit shown, parameters g_i, h_i, and p_i (i = 1, 2, 3) come from the logical effort delay model. n_i is the ratio between the total input capacitance of gate i and the capacitance of the input pin of gate i that belongs to the logic path under analysis. For a symmetric gate, the numerical value of n_i is equal to the number of input pins of gate i. C_{ini} is the capacitance of the input pin of gate i that belongs to the logic path. C_1 is a fixed extra parasitic capacitance. C_{out1} is the output capacitance seen by gate 2, which might encompass another fixed extra parasitic capacitance in the output of such gate. C_{out2} is the output capacitance seen by gate 3, which might encompass another fixed extra parasitic capacitance in the output of such gate. It is worthy to mention that now there are two delay constraints: D_1, corresponding to the logic path formed by gates 1 and 2, and D_2, corresponding to the logic path formed by gates 1 and 3.

Figure 4.7 – Model of a 2-stage branching subcircuit with variable input capacitance



Source: Logics (2013).

There are three variables in this problem, namely, C_{in1}, C_{in2}, and C_{in3}. According to the logical effort delay model, the delay equation for each logic path may be expressed as:

$$D_1 = (g_1 h_1 + p_1 + g_2 h_2 + p_2) \tag{4.71}$$

$$D_2 = (g_1 h_1 + p_1 + g_3 h_3 + p_3) \tag{4.72}$$

Since $h_1 = (C_{in2} + C_{in3} + C_1)/C_{in1}$, $h_2 = C_{out1}/C_{in2}$, and $h_3 = C_{out2}/C_{in3}$, equations (4.71) and (4.72) may be rewritten respectively as

$$D_1 = \frac{g_1}{C_{in1}}(C_{in2} + C_{in3} + C_1) + p_1 + \frac{g_2}{C_{in2}} C_{out1} + p_2 \tag{4.73}$$

and

$$D_2 = \frac{g_1}{C_{in1}}(C_{in2} + C_{in3} + C_1) + p_1 + \frac{g_3}{C_{in3}} C_{out2} + p_3 \tag{4.74}$$

Isolating C_{in1} in equation (4.73):

$$C_{in1} = \frac{g_1 C_{in2}(C_{in2} + C_{in3} + C_1)}{(D_1 - p_1 - p_2)C_{in2} - g_2 C_{out1}} \tag{4.75}$$

Replacing the expression for C_{in1} – equation (4.75) – into equation (4.74):

$$D_2 = \frac{D_1 C_{in2} - g_2 C_{out1} - C_{in2}(p_1 + p_2)}{C_{in2}} + \frac{g_3}{C_{in3}} C_{out2} + p_1 + p_3 \tag{4.76}$$

Isolating C_{in2} in equation (4.76):

$$C_{in2} = \frac{g_2 C_{out1} C_{in3}}{C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2}} \tag{4.77}$$

Now replacing the expression for C_{in2} – equation (4.77) – into equation (4.75), we obtain C_{in1} as a function of solely C_{in3}:

$$\begin{aligned} C_{in1} = {} & \frac{g_1 g_2^2 C_{out1}^2 C_{in3}^2}{g_2 C_{out1}(C_{in3}(D_2 - p_1 - p_3) - g_3 C_{out2})(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2})} \\ & + \frac{g_1 g_2 C_{out1} C_{in3}(C_{in3} + C_1)(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2})}{g_2 C_{out1}(C_{in3}(D_2 - p_1 - p_3) - g_3 C_{out2})(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2})} \end{aligned} \tag{4.78}$$

In this case, the active area of the subcircuit in Figure 4.7 is given by:

$$A(C_{in1}, C_{in2}, C_{in3}) = n_1 C_{in1} + n_2 C_{in2} + n_3 C_{in3}. \tag{4.79}$$

Merging equations (4.77), (4.78), and (4.79):

$$A(C_{in3}) = \frac{n_1 g_1 g_2^2 C_{out1}^2 C_{in3}^2}{g_2 C_{out1}(C_{in3}(D_2 - p_1 - p_3) - g_3 C_{out2})(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2})} + \frac{n_1 g_1 g_2 C_{out1} C_{in3}(C_{in3} + C_1)(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2})}{g_2 C_{out1}(C_{in3}(D_2 - p_1 - p_3) - g_3 C_{out2})(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2})} + \frac{n_2 g_2 C_{out1} C_{in3}}{C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2}} + n_3 C_{in3} \tag{4.80}$$

That is, the active area is now a univariate function of C_{in3}. By taking the derivative of $A(C_{in3})$ and setting it to zero, the value of C_{in3} for minimum active area is attained. For the sake of an easier algebraic manipulation, let the functions:

$$u(C_{in3}) = n_1 g_1 g_2^2 C_{out1}^2 C_{in3}^2 \tag{4.81}$$

$$v(C_{in3}) = g_2 C_{out1}(C_{in3}(D_2 - p_1 - p_3) - g_3 C_{out2})(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2}) \tag{4.82}$$

$$w(C_{in3}) = n_1 g_1 g_2 C_{out1} C_{in3}(C_{in3} + C_1)(C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2}) \tag{4.83}$$

$$x(C_{in3}) = n_2 g_2 C_{out1} C_{in3} \tag{4.84}$$

$$y(C_{in3}) = C_{in3}(D_1 - D_2 + p_3 - p_2) + g_3 C_{out2} \tag{4.85}$$

It is easy to notice that $u(C_{in3})$ is the numerator of the first fraction in the right-hand side of equation (4.80), meanwhile $v(C_{in3})$ is the denominator of the two first fractions in the right-hand side of this same equation. In the same fashion, $w(C_{in3})$ is the numerator of the second fraction, and the numerator and denominator of the third fraction are given by $x(C_{in3})$ and $y(C_{in3})$, respectively. Therefore, $A(C_{in3})$ may be expressed as:

$$A(C_{in3}) = \frac{u(C_{in3})}{v(C_{in3})} + \frac{w(C_{in3})}{v(C_{in3})} + \frac{x(C_{in3})}{y(C_{in3})} + n_3 C_{in3} \tag{4.86}$$

Hence,

$$\frac{dA(C_{in3})}{dC_{in3}} = \frac{u'v - uv'}{v^2} + \frac{w'v - wv'}{v^2} + \frac{x'y - xy'}{y^2} + n_3 = 0 \tag{4.87}$$

where $\theta' = \frac{d\theta(C_{in3})}{dC_{in3}}, \theta \in \{u, v, w, x, y\}$.

Rearranging equation (4.87):

$$(u'v - uv')y^2 + (w'v - wv')y^2 + (x'y - xy')v^2 + n_3 v^2 y^2 = 0 \tag{4.88}$$

This is a sixth degree univariate (on C_{in3}) polynomial equation, whose solutions may be obtained numerically. Among these solutions, one corresponds to the minimum active area. Therefore, the triple of equations (4.77), (4.78), and (4.88) defines the sizing of the branching subcircuit.

4.6 Power Delay Product

In the design of power-efficient circuits, an important figure of merit is the power-delay ($P - D$) tradeoff. In its simplest form, it is expressed as the product PD. In order to achieve a thorough analysis in the $P - D$ space, this tradeoff may also be considered as $P^i D^j, i, j \geq 0$ (ALIOTO, 2011). The correct understanding of this tradeoff is essential for designing either high-speed or low-power circuits (DAO, 2006; MARKOVIC, 2004). One way of doing this analysis is based upon a power-efficient curve, which consists of the minimum points of each $P^i D^j$ curve, for several combinations of the exponents i and j. The minimum points of these curves use to be obtained via iterative optimization procedures (ALIOTO, 2011), which is a time-demanding task. In this section, it is shown that the proposed sizing method is suited for finding analytically the minimum points for power-delay product.

As an example, it is obtained analytically the minimum point of PD product for a 3-stage fanout free subcircuit, with fixed topology, fixed extra parasitic capacitances, and fixed subcircuit input capacitance, as shown in Figure 4.4. The delay and active area of this subcircuit are given respectively by equations (4.3) and (4.9). According to Weste (2006), the dynamic power P dissipated by the subcircuit is given by:

$$P = \alpha C V_{DD}^2 f \tag{4.89}$$

where f is the clock frequency, C is the total input capacitance (i.e., active area A), and α is an activity factor, which can be empirically determined (WESTE, 2006). Consequently, equation (4.89) may be rewritten as:

$$P = kA \tag{4.90}$$

where k is a constant. Hence, the PD product is given by:

$$PD = P \times D = k \times A(C_{in2}, C_{in3}) \times D(C_{in2}, C_{in3}) \tag{4.91}$$

Even though the same expression for the delay – equation (4.3) – is valid throughout this thesis, in this section there is a remarkable difference in its interpretation. Previously, the delay of the subcircuit was a design constraint, i.e., a fixed value. However, the objective now is to do an analysis in the PD space. Therefore, the delay is not only variable, but also a function of C_{in2} and C_{in3}. Merging equations (4.3) and (4.90) into (4.91), it follows that the PD product may be expressed as:

$$PD = k \times (n_1 C_{in1}^{fixed} + n_2 C_{in2} + n_3 C_{in3}) \times (g_1 h_1 + g_2 h_2 + g_3 h_3 + p_1 + p_2 + p_3) \tag{4.92}$$

Since $h_1 = (C_{in2} + C_1)/C_{in1}^{fixed}$, $h_2 = (C_{in3} + C_2)/C_{in2}$, $h_3 = C_{out}/C_{in3}$, and $p_1 + p_2 + p_3 = P$, equation (4.92) may be rewritten as

$$PD = k(n_1 C_{in1}^{fixed} + n_2 C_{in2} + n_3 C_{in3})(g_1 \frac{(C_{in2} + C_1)}{C_{in1}^{fixed}} + g_2 \frac{(C_{in3} + C_2)}{C_{in2}} + g_3 \frac{C_{out}}{C_{in3}} + P) \tag{4.93}$$

Expanding the terms in equation (4.93) and finding the two partial derivatives:

$$\frac{\partial PD(C_{in2},C_{in3})}{\partial C_{in2}} = k\left(n_1 g_1 - n_1 C_{in1}^{fixed} g_2 \frac{(C_{in3}+C_2)}{C_{in2}^2} + n_2 g_1 \frac{(2C_{in2}+C_1)}{C_{in1}^{fixed}} \right) + k\left(n_2 g_3 \frac{C_{out}}{C_{in3}} + n_2 P + n_3 g_1 \frac{C_{in3}}{C_{in1}^{fixed}} - n_3 g_2 C_{in3} \frac{(C_{in3}+C_2)}{C_{in2}^2} \right) \quad (4.94)$$

and

$$\frac{\partial PD(C_{in2},C_{in3})}{\partial C_{in3}} = k\left(n_1 g_2 \frac{C_{in1}^{fixed}}{C_{in2}} - n_1 g_3 C_{in1}^{fixed} \frac{C_{out}}{C_{in3}^2} + n_2 g_2 - n_2 g_3 C_{out} \frac{C_{in2}}{C_{in3}^2} \right) + k\left(n_3 g_1 \frac{(C_{in2}+C_1)}{C_{in1}^{fixed}} + n_3 g_2 \frac{(2C_{in3}+C_2)}{C_{in2}} + n_3 P \right) \quad (4.95)$$

Setting these two partial derivatives equal to zero:

$$\frac{\partial PD(C_{in2},C_{in3})}{\partial C_{in2}} = 0 \Rightarrow$$
$$n_1 g_1 C_{in1}^{fixed} C_{in2}^2 C_{in3} - n_1 C_{in1}^{fixed\,2} g_2 (C_{in3}+C_2) C_{in3} + n_2 g_1 (2C_{in2}+C_1) C_{in2}^2 C_{in3}$$
$$+ n_2 g_3 C_{out} C_{in1}^{fixed} C_{in2}^2 + n_2 P C_{in1}^{fixed} C_{in2}^2 C_{in3} + n_3 g_1 C_{in2}^2 C_{in3}^2 - n_3 g_2 C_{in1}^{fixed} C_{in3}^2 (C_{in3}+C_2) = 0 \quad (4.96)$$

and

$$\frac{\partial PD(C_{in2},C_{in3})}{\partial C_{in3}} = 0 \Rightarrow$$
$$n_1 g_2 C_{in1}^{fixed\,2} C_{in3}^2 - n_1 g_3 C_{in1}^{fixed\,2} C_{out} C_{in2} + n_2 g_2 C_{in1}^{fixed} C_{in2} C_{in3}^2 - n_2 g_3 C_{out} C_{in1}^{fixed} C_{in2}^2$$
$$+ n_3 g_1 (C_{in2}+C_1) C_{in2} C_{in3}^2 + n_3 g_2 C_{in1}^{fixed} (2C_{in3}+C_2) C_{in3}^2 + n_3 P C_{in1}^{fixed} C_{in2} C_{in3}^2 = 0 \quad (4.97)$$

Rearranging conveniently the terms in equation (4.97), it can be expressed as if it were a univariate polynomial equation of the second degree on C_{in2}:

$$\left(n_3 g_1 C_{in3}^2 - n_2 g_3 C_{out} C_{in1}^{fixed}\right) C_{in2}^2 + \left(n_2 g_2 C_{in1}^{fixed} C_{in3}^2 - n_1 g_3 C_{in1}^{fixed\,2} C_{out} + n_3 g_1 C_1 C_{in3}^2 + n_3 P C_{in1}^{fixed} C_{in3}^2\right) C_{in2}$$
$$+ n_1 g_2 C_{in1}^{fixed\,2} C_{in3}^2 + n_3 g_2 C_{in1}^{fixed} (2C_{in3}+C_2) C_{in3}^2 = 0 \quad (4.98)$$

Therefore, the variable C_{in2} can be expressed as a function of C_{in3}:

$$aC_{in2}^2 + bC_{in2} + c = 0 \Rightarrow C_{in2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (4.99)$$

where:

$$a = n_3 g_1 C_{in3}^2 - n_2 g_3 C_{out} C_{in1}^{fixed} \quad (4.100)$$

$$b = n_2 g_2 C_{in1}^{fixed} C_{in3}^2 - n_1 g_3 C_{in1}^{fixed\,2} C_{out} + n_3 g_1 C_1 C_{in3}^2 + n_3 P C_{in1}^{fixed} C_{in3}^2 \quad (4.101)$$

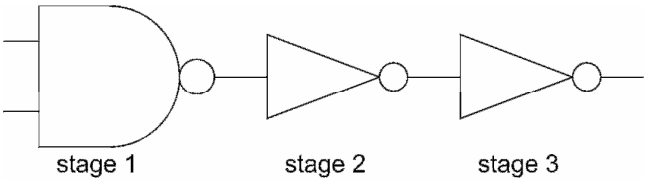
$$c = n_1 g_2 C_{in1}^{fixed\,2} C_{in3}^2 + n_3 g_2 C_{in1}^{fixed} (2C_{in3}+C_2) C_{in3}^2 \quad (4.102)$$

The variables *a*, *b*, and *c* have been introduced for the sake of an easier algebraic manipulation. Replacing equation (4.99) into (4.96), we have a univariate nonlinear equation on C_{in3}, which can be solved numerically. Replacing the value thus obtained for C_{in3} into equation (4.99), we have the corresponding value of C_{in2}. Eventually, either negative or nonreal values may be found, but there will always be one and only one pair of values for C_{in2} and C_{in3} that correspond to the minimum *PD* product.

5 EXPERIMENTAL RESULTS

In this section we investigate the validity of the method compared with results obtained with electrical simulations. The subcircuit used for validation is shown in Figure 5.1. Even though the proposed method is capable of coping with fixed extra parasitic capacitances, they are not taken into account in this validation. The comparison is based on the Nangate Open Cell Library (NOCL) library, and the specific values of logical effort parameters g and p for the logic gates are obtained by simulation according to the logical effort original method (SUTHERLAND, 1999). The formulation proposed herein is developed so that the input capacitance of the subcircuit under design may have either a fixed or a maximum value. Working with a variable – albeit limited – input capacitance would give an additional degree of freedom to the problem, making it easier to come to a global optimum. However, in this chapter, for the sake of comparison with the logical effort sizing method, the input capacitance of the subcircuit is made equal to the X1 NAND2 cell in NOCL.

Figure 5.1 – Subcircuit to be sized



Source: Logics (2013).

The proposed method is used to size the subcircuit of Figure 5.1 for 20 different configurations of output load and delay constraints, which are shown in Table 5.1. The four columns under the *Experiment Configuration* title specify the conditions under which each experiment was carried out. The column labeled *Case* presents the configuration identifier label, ranging from C1 to C20. The column labeled *Load* represents the output load used in the configuration. X*i* expresses a load whose value is *i* times the input capacitance of the X1 inverter in NOCL. The third column (*Const.*) represents the *delay constraint* of the experiment configuration, given in picoseconds (ps). The column entitled *LE ratio* explains how the delay constraint is obtained from the minimum achievable delay. These delay constraints are obtained as follows. First, we calculate the minimum possible delay for every load X*i*, namely, LE_i, given by the logical effort sizing method (SUTHERLAND, 1999). Then, the minimum achievable LE delay is augmented by a factor within the range 0.9^{-1} to 0.5^{-1}, thus introducing a slack in the minimum achievable delay constraint. This slack is increased as long as the logic gates of the corresponding optimized subcircuit are not sized to scale factors

smaller than 1. In this case, the scale factors would be automatically set to 1, and therefore any sizing comparison would be meaningless.

For each of the case studies (C01 to C20), the subcircuit is sized with the proposed method, and the corresponding results are shown in the three columns in Table 5.1 under the *Proposed method* title. The column labeled ΣW shows the sum of the scale factors of the two inverters (stages 2 and 3) in the subcircuit obtained with the method. The columns entitled *Delay* (in picoseconds) and *Pow.*(parameterized), respectively, show the corresponding delay and dynamic power obtained by HSPICE simulations for the subcircuit obtained with the method. *Pow.* is parameterized by the dynamic power consumption of a subcircuit with $\Sigma W = 2.9$ under load X1.

Table 5.1 – Sizing results compared with HSPICE reference and Kabbani (2010)

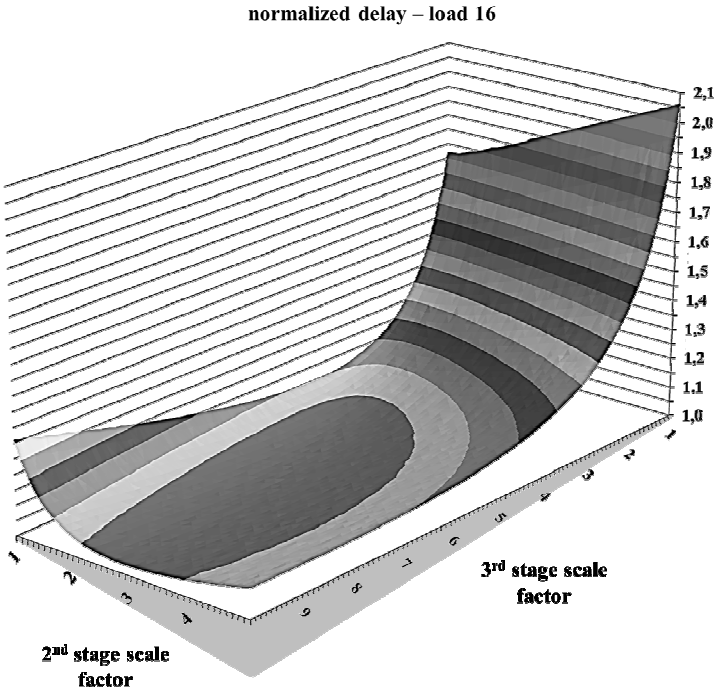
Experiment Configuration				Proposed method			Reference		Proposed method (%)			Kabbani [7] (%)		
Case	Load	Const.	LE Ratio	∑ W	Delay	Pow.	∑ W	Pow.	∑ W(%)	D(%)	Pw(%)	∑ W(%)	D(%)	Pw(%)
C01	X4	42.7	(LE_4/0.9)	2.02	43.603	1.21	2.2	1.26	-8.0	+2.00	-4.0	+70,90	-8,38	+26,90
C02	X16	56.2	(LE_{16}/0.9)	4.01	58.210	3.24	4.6	3.38	-12.8	+3.45	-4.1	+77,59	-7,21	+23,40
C03	X16	63.2	(LE_{16}/0.8)	3.00	64.703	3.03	3.2	3.07	-6.3	+2.29	-1.3	+155,29	-17,48	+35,86
C04	X16	72.3	(LE_{16}/0.7)	2.49	72.769	2.92	2.6	2.94	-4.3	+0.70	-0.68	+214,20	-27,87	+41,87
C05	X32	65.7	(LE_{32}/0.9)	6.14	68.041	5.83	7.1	6.05	-13.5	+3.50	-3.6	+72,40	-6,76	+19,01
C06	X32	73.9	(LE_{32}/0.8)	4.49	75.669	5.47	4.8	5.54	-6.4	+2.38	-1.3	+155,01	-17,10	+29,96
C07	X32	84.4	(LE_{32}/0.7)	3.51	84.816	5.26	3.6	5.28	-2.4	+0.47	-0.38	+240,01	-27,42	+36,36
C08	X32	98.5	(LE_{32}/0.6)	2.90	96.865	5.14	2.9	5.14	+0.1	-1.67	0.0	+322,08	-37,81	+40,08
C09	X40	69.2	(LE_{40}/0.9)	7.05	71.711	7.08	8.1	7.33	-12.9	+3.50	-3.4	+72,48	-6,59	+17,98
C10	X40	77.8	(LE_{40}/0.8)	5.18	79.677	6.69	5.5	6.75	-5.9	+2.29	-0.89	+154,02	-16,92	+28,12
C11	X40	89.0	(LE_{40}/0.7)	3.99	89.349	6.43	4.1	6.45	-2.7	+0.42	-0.31	+240,75	-27,37	+34,08
C12	X64	77.6	(LE_{64}/0.9)	9.44	80.381	10.8	10.9	11.1	-13.4	+3.49	-2.7	+69,85	-6,26	+15,77
C13	X64	87.3	(LE_{64}/0.8)	6.81	90.203	10.2	7.4	10.3	-8.0	+3.25	-0.97	+150,19	-16,68	+24,76
C14	X64	99.7	(LE_{64}/0.7)	5.35	99.945	9.90	5.4	9.90	-0.9	+0.20	0.0	+242,85	-27,04	+29,80
C15	X64	116	(LE_{64}/0.6)	4.21	113.78	9.65	4.1	9.63	+2.7	-2.27	+0.21	+351,56	-37,29	+33,44
C16	X100	86.8	(LE_{100}/0.9)	12.5	90.002	16.2	14.4	16.7	-13.3	+3.51	-3.0	+68,56	-5,78	+13,29
C17	X100	97.7	(LE_{100}/0.8)	9.26	99.562	15.5	9.8	15.6	-5.5	+1.87	-0.64	+147,68	-16,29	+21,28
C18	X100	112	(LE_{100}/0.7)	7.14	111.47	15.0	7.1	15.1	+0.6	-0.16	-0.67	+241,87	-26,98	+25,30
C19	X100	130	(LE_{100}/0.6)	5.54	126.98	14.7	5.3	14.6	+4.4	-2.58	+0.68	+357,98	-37,09	+29,59
C20	X100	156	(LE_{100}/0.5)	4.37	148.00	14.5	4.1	14.4	+6.6	-5.62	+0.69	+492,03	-47,58	+31,39

Source: Author.

Table 5.1 also presents two columns used as reference, listed under the title *Reference*. In order to generate the reference data, an exhaustive set of HSPICE simulations – level 6, using Predictive Technology Model 45 nm technology (PTM) – is performed for each output load (ranging from X4 to X100). The goal is to compare the results given by the sizing method with the minimum active area obtained by exhaustive electrical simulations. The results of the delay obtained by electrical simulations form a surface, as illustrated for the load X16 in Figure 5.2 for delay and in Figure 5.3 for

power. These data sets can be used to discover minimum active area respecting the delay constraint and minimum power respecting the delay constraint for cases C02 to C04, which have load X16. Similar surfaces are produced for loads X4 (case C01), X32 (cases C05 to C08), X40 (cases C09 to C11), X64 (cases C12 to C15) and X100 (cases C16 to C20). The columns under the title *Reference* labeled *ΣW* and *Pow.*, respectively, show the minimum possible ΣW and power respecting the design constraints, as obtained empirically from the dataset forming the surfaces.

Figure 5.2 – Delay vs. sizing of a subcircuit for minimum active area via exhaustive search (electrical simulation)



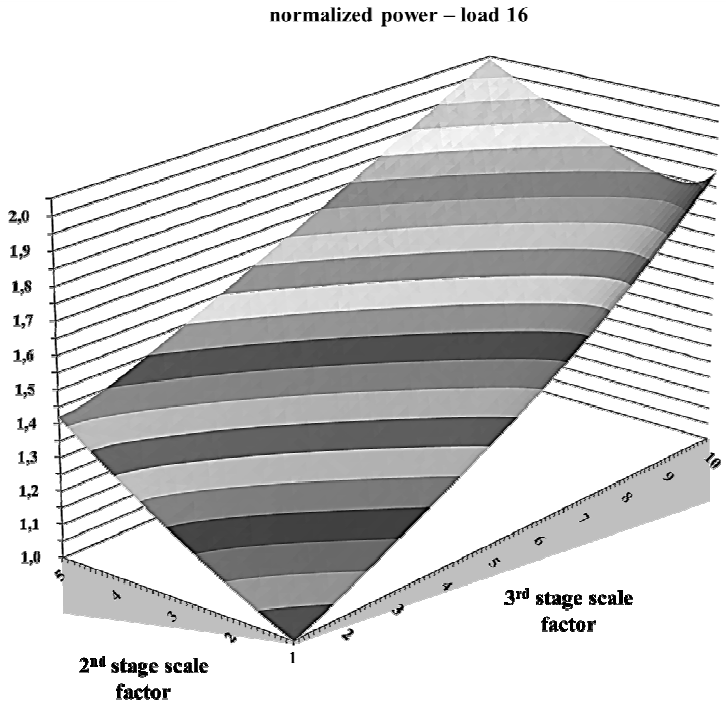
Source: Logics (2013).

Figure 5.2 plots the delay of the subcircuit for a range of scale factors for the two inverters, with an output load of X16. The delay is parameterized by LE_{16}. For any given delay constraint greater than the minimum possible delay, there are several possible pairs of scale factors. However, only one specific point corresponds to either minimum active area (sum of widths) or power consumption. Figure 5.3 plots the power consumption of the subcircuit for the same previous range of scale factors for the two inverters and load. Notice that, for better visualization, the horizontal scales in the plot are not the same as in Figure 5.2. The dynamic power consumption is parameterized by that of the minimum inverter. Comparing these plots, it is clear that the fastest circuit is not the most power consumer, as well as the slowest circuit is not the least power consumer. This summarizes the importance of appropriate sizing for optimizing a circuit. Figure 5.3 also shows a good correlation between the total input capacitances of the logic gates of a circuit and its power consumption. This can be inferred from the fact that the surface plotted is nearly a plane.

The comparison of the results from the method with HSPICE references is presented in three columns of Table 5.1, under the title *Proposed Method (%)*. The column labeled *ΣW(%)* gives the percentage difference between the sum of widths (i.e., scale factors of the two inverters) obtained by the proposed method and the minimum reference obtained from HSPICE simulation datasets. The subcircuit can be oversized by 6.6% in the worst case, but producing a better delay at a small power penalty. The column entitled *D(%)* gives the percentage difference between the delay for the subcircuit obtained by the proposed method and the delay constraint. Sometimes the delay is slightly larger than the delay constraint (by 3.5%), which is acceptable for a first fast computation. The column labeled *Pw(%)* gives the percentage difference between the dynamic power obtained by the proposed method and the minimum reference obtained from electrical simulation datasets. Notice that, in all cases but one (C18), the delay difference has opposite signs with respect to both power and sum of width differences, as expected. This exception is explained by the fact that the minimum value obtained from electrical simulation is not necessarily the minimum minimorum, since this value was obtained with a discrete simulation step (0.1 of scale factor). In case C18, the real minimum value was found by using a simulation step ten times smaller (0.01 of scale factor). This happens for reference ΣW = 7.14. In this situation, the delay difference has positive sign, while the power difference has negative sign.

Using 0.1 as simulation step, a total of 5,000 simulations have been performed. It would be unfeasible to use 0.01 as simulation step to fill in the table, since 500,000 simulations would be necessary.

Figure 5.3 – Power consumption of the subcircuit under design



Source: Logics (2013).

The proposed method presents improvements over previous approaches. According to Kabbani (2010), the efforts (SUTHERLAND, 1999) of the logic gates in the subcircuit should have the same value for attaining minimum area. In mathematical terms, for the subcircuit under design, we would have:

$$g_1 \frac{C_{in2}}{C_{in1}} = g_2 \frac{C_{in3}}{C_{in2}} = g_3 \frac{C_{out}}{C_{in3}} \tag{5.1}$$

Nevertheless, electrical simulations show that, for the load X16 and delay constraint LE_{16}/0.9, the efforts (SUTHERLAND, 1999) for the first, second, and third stages are given by 1.60, 2.54, and 4.85, respectively. The proposed method finds 1.39, 2.56, and 5.55, respectively. This verification is conducted for several combinations of loads and delay constraints, and the discrepancy between HSPICE results and equation (5.1) – as stated by Kabbani (2010) – is always present. In turn, the results given by the proposed method, for all experiments fulfilled, are much closer to the optimal results obtained by electrical simulations. This happens because, unlike Kabbani (2010), the proposed method takes into account the following facts: a) the input capacitance of the subcircuit may have either fixed or variable – although limited – value; b) the number of stages in the subcircuit may differ from the ideal number predicted by the logical effort sizing method (SUTHERLAND, 1999); c) the cost function for the subcircuit active area in equation (4.8) encompasses each logic gate in its entireness, not just the capacitance of the input pin that belongs to the logic path under design.

6 CONCLUSION AND FUTURE WORK

This thesis presented a new method for sizing subcircuits based on the logical effort delay model. The method is able to find the minimum active area of a subcircuit analytically, thus dismissing the use of iterative methods such as mathematical programming or algorithmic approaches. An analytical solution – rather than an iterative one – has some advantages. The minimum active area is achieved by solving a one-variable equation, which tends to be faster than iterative methods. Since power consumption is closely related to active area, this method is also capable of minimizing power.

Besides, an analytical solution offers a better understanding of the problem, which may be the starting point for future works. As seen in section 4.6, the method may be generalized in order to optimize power delay product. More specifically, the minimum point for the PD curve was obtained. It is likely that the minimum points for $P^i D^j$, for small values of *i* and *j*, may also be obtained with the proposed method.

Another extension of the proposed method is related to branching circuits. In section 4.5, the method was developed for two-stage branching subcircuits. It is very likely that branching circuits may be treated to the same extension – in terms of the number of stages – that fanout free circuits may be treated by the proposed method.

The model accuracy has been validated with respect to electrical simulations, which showed that the proposed method was very precise for a first order approach, as it presented average errors of 1.48% in power dissipation, 2.28% in propagation delay, and 6.5% in transistor sizes.

The maximum delay error was 3.5%. This inaccuracy is inherent to the logical effort delay model. In order to minimize such error, the usage of more accurate versions of the logical effort delay model – such as Lasbouygues (2006), Wang (2009), and Masry (2011) – is under study. Such model version should consider the impact of the input signal slope on the delay of a logic gate. This way, the new sizing method would be able to cope with non-posynomial delay models. Such models cannot be solved by convex programming (TENNAKOON, 2008), and their solution by non-convex programming is not granted.

Just like many other cases in engineering, there is a tradeoff relationship between analytical solutions and numerical solutions. The former gives more elegant and less computing power consuming solutions, meanwhile the latter relies on iterative methods that may be computationally expensive. However, it is not always that analytical solutions for sizing may be applied with precise, non-trivial delay models. In this case,

numerical solutions may be more appropriate. This is the challenge to be faced when applying the proposed method with more accurate delay models.

Another future work is related to the generalization of the proposed sizing method for subcircuits with an arbitrary number of stages. Currently, the method is derived for a finite set of logic path lengths. Initially, it seems very likely that the method may be applied to subcircuits with up to five stages. It depends on the system of polynomial equations to be solved analytically, which may not always be possible. The next step would be to find the solution of the proposed method for an arbitrary number of stages. It is not granted that such problem has a closed-form solution. Therefore, for the sake of simplicity, the investigation would start by analyzing fanout free buffers. For every number of stages – up to a limit –, for every load – within a range –, and for every delay constraint – as long as the inverters are not sized to a scale factor smaller than 1 –, a buffer will be sized via exhaustive electrical simulations. The data set thus obtained might reveal a pattern in the sizing of buffers with minimum active area, which might lead to a new sizing formulation.

A somewhat peripheral future work is as follows. In Sutherland (1999), an analysis is conducted about the w_p/w_n ratio that gives the smallest average delay (rise and fall transitions) of a logic gate. Within the scope of the proposed method, an analogous analysis may be conducted to determine the w_p/w_n ratio that gives the logic gates with smallest active area.

To the best knowledge of the doctor's degree candidate, this is the first approach for analytical sizing under delay constraints based on the logical effort delay model. It differs considerably from the existing sizing methods, since its main contribution is to compute the area derivative to obtain minimum area, instead of making the delay derivative to obtain minimum delay, as it is done in the traditional logical effort formulation. At the same time, the proposed method shows a good precision – when compared to electrical simulations – and superior results to other sizing method (KABBANI, 2010) that has the same objectives. Equally important, the proposed method paves way for consistent future works.

REFERENCES

ALIOTO, M.; CONSOLI, E.; PALUMBO, G. Analysis and Comparison in the Energy-Delay-Area Domain of Nanometer CMOS Flip-Flops: Part I – Methodology and Design Strategies. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** [S.l.], v.19, n. 5, p. 725-736, May. 2011.

BERKELAAR, M.; JESS, J. Technology Mapping for Standard-Cell Generators. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1988 [S.l.:s.n], **Proceedings…** Santa Clara, CA: IEEE, 1988, p. 470-473.

BERKELAAR, M. R. C. M.; JESS, J. A. G. Gate Sizing in MOS Digital Circuits with Linear Programming. In: EUROPEAN DESIGN AUTOMATION CONFERENCE, EDAC, 1990, [S.l.], **Proceedings…** [S.l:s.n.], 1990, p. 217-221.

BERTSEKAS, D. P. **Nonlinear Programming**. 2nd ed. Belmont: Athena, 1999.

BOYD, S. P.; VANDENBERGHE, L. **Convex Optimization**. Cambridge: Cambridge, 2004.

BOYD, S. P.; KIM, S. J.; PATIL, D. D.; HOROWITZ, M. A. Digital Circuit Optimization via Geometric Programming. **Operations Research** [S.l.], v.53, n.6, p. 899-932, Nov.-Dec. 2005.

CALLEGARO, V. ; MARTINS, M.G.A. ; RIBAS, R.P. ; REIS, A.I. Read-polarity-once Boolean functions. In: 26th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2013. **Proceedings…** [S.l.]: IEEE, 2013, p. 1 – 6.

CHEN, G.; ONODERA, H.; TAMARU, K. Timing and Power Optimization by Gate Sizing Considering False Path. In: GREAT LAKES SYMPOSIUM ON VLSI, GLSVLSI, 1996. **Proceedings…** [S.l.]: IEEE, 1996, p. 154-159.

CHINNERY, D. G.; KEUTZER, K. **Closing the Gap between ASIC & Custom:** Tools and Techniques for High-Performance ASIC Design. Norwell: Kluwer, 2002.

CHINNERY, D. G.; KEUTZER, K. Closing the Power Gap between ASIC and Custom: An ASIC Perspective. In: DESIGN AUTOMATION CONFERENCE, DAC, 42., 2005, Anaheim, CA, USA. **Proceedings…** [S.l.]: ACM, 2005, p. 275-280.

CHU, C.; WONG, D. F. VLSI Circuit Performance Optimization by Geometric Programming. **Annals of Operations Research** [S.l.], v.105, p. 37-60, 2001.

CORREIA, V. P.; REIS, A. I. Advanced Technology Mapping for Standard-Cell Generators. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2004, Porto de Galinhas, PE. **Proceedings…** [S.l.]: SBC, 2004, p. 254-259.

COUDERT, O. Gate Sizing: a General Purpose Optimization Approach. In: EUROPEAN DESIGN AND TEST CONFERENCE, ED&TC, 1996-a, Paris, France. **Proceedings…** Los Alamitos: IEEE, 1996-a, p. 214-218.

COUDERT, O.; HADDAD, R.; MANNE, S. New Algorithms for Gate Sizing: A Comparative Study. In: DESIGN AUTOMATION CONFERENCE, DAC, 33., 1996-b, Las Vegas, NV, USA. **Proceedings…** [S.l.]: ACM, 1996-b, p. 734-739.

COUDERT, O. Gate Sizing for Constrained Delay/Power/Area Optimization. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** [S.l.], v.5, n.4, p. 465-472, Dec. 1997.

COUDERT, O. Timing and Design Closure in Physical Design Flows. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, ISQED, 2002 [S.l.] **Proceedings…** [S.l:s.n.] 2002, p. 511-516.

DAO, H.; ZEYDEL, B.; OKLOBDZIJA, V. Energy Optimization of Pipelined Digital Systems Using Circuit Sizing and Supply Scaling. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** [S.l.], v.14, n.2, p. 122-134, Feb. 2006.

DESIGN COMPILER Tutorial [S.l.:s.n]. Jul. 2010. Available at http://www.tkt.cs.tut.fi/tools/public/tutorials/synopsys/design_compiler/gsdc.html

DESIGN Compiler® Reference Manual: Constraints and Timing [S.l.:s.n]. Sept. 2008. Available at http://solvnet.synopsys.com

ELMORE, W. C. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. **Journal of Applied Physics** [S.l.], v. 19, p.55-63, Jan. 1948.

FISHBURN, J. P.; DUNLOP, A. E. TILOS: A Posynomial Programming Approach to Transistor Sizing. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1985 [S.l.:s.n], **Proceedings…** [S.l.] IEEE, 1985, p. 326-328.

GAVRILOV, S. et al. Library-less Synthesis for Static CMOS Combinational Logic Circuits. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1997 [S.l.:s.n], **Proceedings…** [S.l.] IEEE, 1997, p. 658-662.

GIONIS, A.; INDYK, P.; MOTWANI, R. Similarity Search in High Dimensions via Hashing. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 1999, Edinburgh, SCO, 25., **Proceedings…** [S.l.] ACM, 1999, p. 518-529.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear**. 2ª ed. Rio de Janeiro: Campus Elsevier, 2005.

GOPALAKRISHNAN, G.; KUDVA, P.; BRUNVAND, E. Peephole optimization of asynchronous macromodule networks. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** [S.l.], v.7, n.1, p.30-37, Mar. 1999.

HEDLUND, K. S. Aesop: A Tool for Automated Transistor Sizing. In: DESIGN AUTOMATION CONFERENCE, DAC, 24., 1987, Miami Beach, FL, USA. **Proceedings…** [S.l.]: ACM, 1987, p. 114-120.

HU, B. et al. Gain-Based Technology Mapping for Discrete-Size Cell Libraries. In: DESIGN AUTOMATION CONFERENCE, DAC, 40., 2003, Anaheim, CA, USA. **Proceedings…** [S.l.]: ACM, 2003, p. 574-579.

HU, S.; KETKAR, M.; HU, J. Gate Sizing for Cell Library-Based Designs. In: DESIGN AUTOMATION CONFERENCE, DAC, 44., 2007, San Diego, CA, USA. **Proceedings…** [S.l.]: ACM, 2007, p. 847-852.

HU, S.; KETKAR, M.; HU, J. Gate Sizing for Cell Library-Based Designs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 28, n. 6, p. 818-825, Jun. 2009.

JOSHI, S.; BOYD, S. An Efficient Method for Large-Scale Gate Sizing. **IEEE Transactions on Circuits and Systems** [S.l.], v. 55, n. 9, p. 2760-2773, Oct. 2008.

KABBANI, A.; AL-KHALILI, D.; AL-KHALILI, A. J. Delay Analysis of CMOS Gates Using Modified Logical Effort Model. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 24, n. 6, p. 937-947, Jun. 2005.

KABBANI, A. Logical effort based dynamic power estimation and optimization of static CMOS circuits. **Integration, the VLSI Journal** [S.l.], v.43, p.279-288, 2010.

KAGARIS, D.; HANIOTAKIS, T. A Methodology for Transistor-Efficient Supergate Design. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** [S.l.], v.15, n.4, p. 488-492, Apr. 2007.

KARANDIKAR, S. K.; SAPATNEKAR, S. S. Logical Effort Based Technology Mapping. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2004 [S.l.:s.n], **Proceedings…** [S.l.] IEEE, 2004, p. 419-422.

KARANDIKAR, S. K.; SAPATNEKAR, S. S. Fast Comparisons of Circuit Implementations. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** [S.l.], v.13, n.12, p. 1329-1339, Dec. 2005.

KARANDIKAR, S. K.; SAPATNEKAR, S. S. Technology Mapping Using Logical Effort for Solving the Load-Distribution Problem. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 27, n. 1, p. 45-58, Jan. 2008.

KASAMSETTY, K.; KETKAR, M.; SAPATNEKAR, S. S. A new class of convex functions for delay modeling and its application to the transistor sizing problem. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 19, n. 7, p. 779-788, Jul. 2000.

KEANE, J.; EOM, H.; KIM, T-H.; SAPATNEKAR, S.; KIM, C. Subthreshold Logical Effort: a Systematic Framework for Optimal Subthreshold Device Sizing. In: DESIGN AUTOMATION CONFERENCE, DAC, 43., 2006, San Francisco, CA, USA. **Proceedings…** [S.l.]: ACM, 2006, p. 425-428.

KEUTZER, K.; KOLWICZ, K.; LEGA, M. Impact of Library Size on the Quality of Automated Synthesis. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1987 [S.l.:s.n], **Proceedings…** [S.l.] IEEE, 1987, p. 120-123.

KUNG, D. S.; PURI, R. Optimal P/N Width Ratio Selection for Standard Cell Libraries. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1999 [S.l.:s.n], **Proceedings…** [S.l.] IEEE, 1999, p. 178-184.

LASBOUYGUES, B. et al. Logical Effort Model Extension to Propagation Delay Representation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 25, n. 9, p. 1677-1684, Sept. 2006.

LI, W. N. Strongly NP-hard Discrete Gate Sizing Problems. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 1993, Cambridge, MA, USA. **Proceedings…** [S.l.]: IEEE, 1993, p. 468-471.

LOGICS – logic circuit synthesis labs homepage, 2013. Available at: http://www.inf.ufrs.br/logics/

LUENBERGER, D. G.; YE, Y. **Linear and Nonlinear Programming.** 3rd ed. New York: Springer, 2008.

MAHALINGAM, V.; RANGANATHAN, N. A Nonlinear Programming Based Power Optimization Methodology for Gate Sizing and Voltage Selection. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, 2005. **Proceedings…** [S.l.]: IEEE, 2005.

MARKOVIC, D.; STOJANOVIC, V.; NIKOLIC, B.; HOROWITZ, M.; BRODERSEN, R. Methods for True Energy-Performance Optimization. **IEEE Journal on Solid-State Circuits** [S.l.] v. 39, n. 8, p. 1282-1293, Aug. 2004.

MARQUES, F. S.; ROSA JUNIOR, L. S.; RIBAS, R. P.; SAPATNEKAR, S. S.; REIS, A. I. DAG Based Library-Free Technology Mapping. In: GREAT LAKES SYMPOSIUM ON VLSI, GLSVLSI, 2007, Stresa. **Proceedings…** [S.l.]: ACM, 2007, p. 293-298.

MARTINELLO, O. ; MARQUES, F.S. ; RIBAS, R.P. ; REIS, A.I. KL-Cuts: A new approach for logic synthesis targeting multiple output blocks. In: Design, Automation & Test in Europe Conference & Exhibition, DATE, 2010. **Proceedings…** [S.l. :s.n] IEEE, 2010, p. 777-782.

MARTINS, M.G.A. ; RIBAS, R.P. ; REIS, A.I. Functional composition: A new paradigm for performing logic synthesis. In: 13th International Symposium on Quality Electronic Design, ISQED, 2012. **Proceedings…** [S.l.] IEEE, 2012, p. 236 – 242.

MASRY, H. El; KHALILI, D. Al. Cell stack length using an enhanced logical effort model for a library-free paradigm. In: 18th IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, 2011. **Proceedings…** [S.l.] IEEE, 2011, p.703-706.

MATSON, M. D.; GLASSER, L. A. Macromodeling and Optimization of Digital MOS VLSI Circuits. **IEEE Transactions on Computer-Aided Design** [S.l.], v. CAD-5, n. 4, p. 659-678, Oct. 1986.

MAURINE, P.; MICHEL, X.; AZEMARD, N.; AUVERGNE, D. Gate Speed Improvement at Minimal Power Dissipation. In: ASIA PACIFIC CONFERENCE ON CIRCUITS AND SYSTEMS, APCCAS, 2002 [S.l.:s.n], **Proceedings…** [S.l.] IEEE, 2002, p. 325-330.

MENEZES, N.; BALDICK, R.; PILEGGI, L. T. A Sequential Quadratic Programming Approach to Concurrent Gate and Wire Sizing. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1995, San Jose, CA USA, **Proceedings…** [S.l.] IEEE, 1995, p.144-151.

NANGATE 45nm Open Cell Library (NOCL) v1_3_v2010_12. Available at http://www.nangate.com

NGUYEN, D. et al. Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, ISLPED, 2003, Seoul, Korea. **Proceedings…** [S.l.]: ACM, 2003, p. 158-163.

NOWE, P. Timing (Analysis) is Everything. **Circuit Cellar** – The Magazine for Computer Applications [S.l.], n. 160, Nov. 2003.

OTTEN, R. H. J. M. Timing closure: the solution and its problems. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP-DAC, 2000, Yokohama, Japan, **Proceedings…** [S.l.] IEEE, 2000, p. 359-364.

PANDA, R. et al. Migration: A new technique to improve synthesized designs through incremental customization. In: DESIGN AUTOMATION CONFERENCE, DAC, 35., 1998, San Francisco, CA, USA. **Proceedings…** [S.l.]: ACM, 1998, p. 388-391.

PATTANAIK, M.; BANERJEE, S.; BAHINIPATI, B. K. GP Based Transistor Sizing for Optimal Design of Nanoscale CMOS Inverter. In: IEEE CONFERENCE ON NANOTECHNOLOGY, 2003, San Francisco, CA, USA. **Proceedings…** [S.l.] IEEE, 2003, p. 524-527.

POLI, R. E. B.; SCHNEIDER, F. R.; RIBAS, R. P.; REIS, A. I. Unified Theory to Build Cell-level Transistor Networks from BDDs. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2003, São Paulo, SP. **Proceedings…** [S.l.]: SBC, 2003, p. 199-204.

PTM 45 nm model. Available at: http://www.eas.asu.edu/~ptm

RABAEY, J. M.; CHANDRAKASAN, A.; NIKOLIĆ, B. **Digital Integrated Circuits** – a design perspective. 2nd ed. Upper Saddle River: Pearson, 2003.

REIS, A. I.; RASMUSSEN, A. B.; ROSA JUNIOR, L. S.; RIBAS, R. P. Fast Boolean Factoring with Multi-Objective Goals. In: INTERNATIONAL WORKSHOP IN LOGIC SYNTHESIS, IWLS, 2009, Berkeley, CA, USA, **Proceedings...** 2009.

REZVANI, P.; PEDRAM, M. A Fanout Optimization Algorithm Based on the Effort Delay Model. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 22, n. 12, p. 1671-1678, Dec. 2003.

RICCI, A.; DE MUNARI, I.; CIAMPOLINI, P. An Evolutionary Approach for Standard-Cell Library Reduction. In: GREAT LAKES SYMPOSIUM ON VLSI, GLSVLSI, 2007, Stresa. **Proceedings…** [S.l.]: ACM, 2007, p. 305-310.

ROBINSON, C. Second derivative test for a function with more variables. Available at http://www.math.northwestern.edu/~clark/232/handouts/max-2deriv.pdf

ROSA JUNIOR, L. S.; MARQUES, F. S.; CARDOSO, T. M. G.; RIBAS, R. P.; SAPATNEKAR, S. S.; REIS, A. I. Fast Disjoint Transistor Networks from BDDs. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2006, Ouro Preto, MG. **Proceedings…** [S.l.]: SBC, 2006, p. 137-142.

ROSA JUNIOR, L. S. **Automatic Generation and Evaluation of Transistor Networks in Different Logic Styles.** 2008. 147 f. Tese (Doutorado em Microeletrônica) – Programa de Pós-Graduação em Microeletrônica, UFRGS, Porto Alegre.

ROY, R.; BHATTACHARYA, D.; BOPPANA, V. Transistor-Level Optimization of Digital Designs with Flex Cells. **Computer** [S.l.], v. 38, n. 12, p. 53-61, Feb. 2005.

ROY, S.; CHEN, W.; CHEN, C. C. P.; HU, Y. H. Numerically Convex Form and Their Application in Gate Sizing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 26, n. 9, p. 1637-1647, Sept. 2007.

RUBINSTEIN, J.; PENFIELD JR., P.; HOROWITZ, M. A. Signal Delay in *RC* Tree Networks. **IEEE Transactions on Computer-Aided Design** [S.l.], v. 2, n. 3, p. 202-211, Jul. 1983.

SAPATNEKAR, S. S.; Kang, S. M. **Design Automation for Timing-Driven Layout Synthesis**. Norwell: Kluwer, 1993-a.

SAPATNEKAR, S. S.; RAO, V. B.; VAIDYA, P. M.; KANG, S. M. 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, Nov. 1993-b.

SAPATNEKAR, S. S.; CHUANG, W. Power vs. Delay in Gate Sizing: Conflicting Objectives? In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1995, San Jose, CA, USA, **Proceedings…** [S.l.] IEEE, 1995, p. 463-466.

SAPATNEKAR, S. **Timing**. Norwell: Kluwer, 2004.

SCHNEIDER, F. R.; RIBAS, R. P.; SAPATNEKAR, S. S.; REIS, A. I. Exact Lower Bound for the Number of Switches in Series to Implement a Combinational Logic Cell. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 2005, San Jose, CA, USA. **Proceedings…** [S.l.]: IEEE, 2005, p. 357-362.

SCHNEIDER, F. R. **Building transistor-level networks following the lower bound on the number of stacked switches.** 2007. 109 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pó-Graduação em Computação, UFRGS, Porto Alegre.

SCOTT, K.; KEUTZER, K. Improving Cell Libraries for Synthesis. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, CICC, 1994 [S.l.:s.n], **Proceedings…** [S.l.]: IEEE, 1994, p. 128-131.

SECHEN, C.; GUAN, B. Large Standard Cell Libraries and their Impact on Layout area and circuit performance. In: INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 1996 [S.l.:s.n], **Proceedings…** [S.l.]: IEEE, 1996, p. 378-383.

SEO, J.; MARKOV, I. L.; SYLVESTER, D.; BLÁAUW, D. On the Decreasing Significance of Large Standard Cells in Technology Mapping. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2008, San Jose, CA, **Proceedings…** [S.l.] IEEE, 2008, p. 116-121.

SHAH, S. et al. Discrete Vt Assignment and Gate Sizing Using a Self-Snapping Continuous Formulation. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2005, San Jose, CA, **Proceedings…** [S.l.] IEEE, 2005, p. 704-711.

SHAH, S.; GUPTA, P.; KAHNG, A. Standard Cell Library Optimization for Leakage Reduction. In: DESIGN AUTOMATION CONFERENCE, DAC, 43., 2006, San Francisco, CA, USA. **Proceedings…** [S.l.]: ACM, 2006, p. 983-986.

SHERWANI, N. **Algorithms for VLSI Physical Design Automation**. 3rd ed. Norwell: Kluwer, 1999.

SINGH, J.; LUO, Z. Q.; SAPATNEKAR, S. S. A Geometric Programming-Based Worst Case Gate Sizing Method Incorporating Spatial Correlation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 27, n. 2, p. 295-308, Feb. 2008.

SMITH, M. J. S. **Application-Specific Integrated Circuits**. Reading: Addison-Wesley, 1997.

SUTHERLAND, I.; SPROULL, B.; HARRIS, D. **Logical Effort:** Designing Fast CMOS Circuits. San Francisco: Morgan Kaufmann, 1999.

SYNOPSYS-a Open Source Liberty. [S.l.:s.n]. Available at http://www.opensourceliberty.org/

SYNOPSYS-b Design Compiler Tutorial. [S.l.:s.n]. Dec. 2009. Available at http:// www.tkt.cs.tut.fi/tools/public/tutorials/synopsys/design_compiler/gsdc.html

TENNAKOON, H.; SECHEN, C. Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Pre-Processing Step. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2002 [S.l.] **Proceedings…** [S.l.] IEEE, 2002, p. 395-402.

TENNAKOON, H.; SECHEN, C. Efficient and Accurate Gate Sizing with Piecewise Convex Delay Models. In: DESIGN AUTOMATION CONFERENCE, DAC, 42., 2005, Anaheim, CA, USA. **Proceedings…** [S.l.]: ACM, 2005, p. 807-812.

TENNAKOON, H.; SECHEN, C. Nonconvex Gate Delay Modeling and Delay Optimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 27, n. 9, p. 1583-1594, Sept. 2008.

TRIHY, R. Addressing Library Creation Challenges from Recent Liberty Extensions. In: DESIGN AUTOMATION CONFERENCE, DAC, 45., 2008, Anaheim, CA, USA. **Proceedings…** [S.l.]: ACM, 2008, p. 474-479.

USING the Synopsys Design Constraints Format. [S.l.:s.n]. Jun. 2009. Available at http://solvnet.synopsys.com

VANDERBEI, R. J. **Linear Programming** – Foundations and Extensions. 3rd ed. New York: Springer, 2008.

WAGNER, F. R.; REIS, A. I.; RIBAS, R. P. **Fundamentos de Circuitos Digitais.** Porto Alegre: Sagra Luzzatto, 2006.

WANG, C. C.; MARKOVIC, D. Delay Estimation and Sizing of CMOS Logic Using Logical Effort With Slope Correction. **IEEE Transactions on Circuits and Systems – II: Express Briefs** [S.l.], v. 56, n. 8, p. 634-638, Aug. 2009.

WERBER, J.; RAUTENBACH, D.; SZEGEDY, C. Timing Optimization by Restructuring Long Combinatorial Paths. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2007, San Jose, CA, USA. **Proceedings…** [S.l.] IEEE, 2007, p. 536-543.

WESTE, N. W.; HARRIS, D. **CMOS VLSI Design** – A Circuit and Systems Perspective. 3rd ed. Boston: Pearson, 2005.

WESTE, N. W.; HARRIS, D.; BANERJEE, A. **CMOS VLSI Design** – A Circuit and Systems Perspective. New Delhi: Pearson, 2006.

YOSHIDA, H.; IKEDA, M.; ASADA, K. A Structural Approach for Transistor Circuit Synthesis. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences** [S.l.], v. E89-A, n. 12, p. 3529-3537, Dec. 2006.

ZEYDEL, B. R.; OKLOBDZIJA, V. G. Methodology for Energy-Efficient Digital Circuit. In: INTERNATIONAL WORKSHOP ON POWER AND TIMING MODELING, OPTIMIZATION AND SIMULATION, PATMOS, 2006, Montpellier, France. **Proceedings…** [S.l.]: 2006.

ZHOU, L.; WAKAYAMA, C.; SHI, C.J.R. CASCADE: A Standard Supercell Design Methodology with Congestion-Driven Placement for Three-Dimensional Interconnect-Heavy Very Large Scale Integrated Circuits. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems** [S.l.], v. 26, n. 7, p. 1270-1282, Jul. 2007.