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

CRISTIANO LAZZARI

Automatic Layout Generation of Static CMOS Circuits Targeting Delay and Power Reduction

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

Prof. Dr. Ricardo Augusto da Luz Reis
Advisor

Prof. Dr. José Luís Almada Güntzel
Coadvisor

Porto Alegre, December 2003

CIP – CATALOGING-IN-PUBLICATION

Lazzari, Cristiano

Automatic Layout Generation of Static CMOS Circuits Targeting Delay and Power Reduction / Cristiano Lazzari. – Porto Alegre: PPGC da UFRGS, 2003.

113 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Advisor: Ricardo Augusto da Luz Reis; Coadvisor: José Luís Almada Güntzel.

1. Automatic layout generation. 2. Static CMOS logic gates. 3. Power and timing optimization. I. Reis, Ricardo Augusto da Luz. II. Güntzel, José Luís Almada. III. Título.

The most dangerous moment comes with victory.
NAPOLEON BONAPARTE

AGRADECIMENTOS

Gostaria de agradecer primeiramente a meus pais Eloi e Lurdes pelo apoio dado durante minha vida e, principalmente, a educação que sempre me ofereceram, servindo de exemplo para minhas atitudes.

Agradeço a minha amada esposa Patrícia pelo apoio que me tem dado nos momentos difíceis. Ela que sempre me motivou a batalhar por meus objetivos e soube me incentivar nos momentos mais difíceis.

Sou grato aos meus orientadores Ricardo Reis e José Guntzel. Sem seus conselhos, suporte e encorajamento, este trabalho não teria sido possível. Agradeço pela confiança que depositaram em em mim e pela amizade que surgiu a partir deste trabalho.

Agradeço a meus professores Altamiro Amadeu Suzim, Luigi Carro, Marcelo Johann, Marcelo Lubaszewski e Sergio Bampi por me mostrar o valor da pesquisa e me auxiliar no meio acadêmico.

Gostaria de agradecer as pessoas que trabalharam diretamente comigo, Fabrício Bastian, Cristiano Domingues, Cristiano dos Santos, Daniel Ferrão, Gustavo Wilke, Renato Hentschke, Cláudio Menezes, Reginaldo Tavares e aos colegas do grupo de microeletrônica, especialmente a Alessandro Girardi, Fernando Paixão Cortes, Alexandre Morais Amory, Érika Fernandes Cota, Lucas Brusamarello, Fernanda Lima, Felipe Marques, Lisane Brisolara e Gustavo Neuberger.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

ATPG	Automatic Test Pattern Generation
CAD	Computer-Aided Design
CDL	Circuit Description Language
C_L	Load Capacitance
CMOS	Complementary Metal-Oxide-Silicon
DAG	Direct Acyclic Graph
DFF	D Flip-Flop
DSP	Digital Signal Processing
EDA	Electronic Design Automation
FOTC	Full Over-the-cell
FSM	Finite State Machine
GME	Microelectronics Group
LVS	Layout versus Schematic
IC	Integrated Circuit
PI	Primary Input
PO	Primary Output
RMS	Root Mean Square
RTL	Register-Transfer Language
SCCG	Static CMOS Complex Gates
TROPIC	Transparent Reconfigurable Optimized Parameterizable Integrated Circuit
VHDL	Very high speed integrated circuits High Description Level
VLSI	Very Large Scale Integration

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The evolution of integrated circuits technologies demands the development of new CAD tools. The traditional development of digital circuits at physical level is based in library of cells. These libraries of cells offer certain predictability of the electrical behavior of the design due to the previous characterization of the cells. Besides, different versions of each cell are required in such a way that delay and power consumption characteristics are taken into account, increasing the number of cells in a library.

The automatic full custom layout generation is an alternative each time more important to cell based generation approaches. This strategy implements transistors and connections according patterns defined by algorithms. So, it is possible to implement any logic function avoiding the limitations of the library of cells. Tools of analysis and estimate must offer the predictability in automatic full custom layouts. These tools must be able to work with layout estimates and to generate information related to delay, power consumption and area occupation.

This work includes the research of new methods of physical synthesis and the implementation of an automatic layout generation in which the cells are generated at the moment of the layout synthesis. The research investigates different strategies of elements disposition (transistors, contacts and connections) in a layout and their effects in the area occupation and circuit delay. The presented layout strategy applies delay optimization by the integration with a gate sizing technique. This is performed in such a way the folding method allows individual discrete sizing to transistors.

The main characteristics of the proposed strategy are: power supply lines between rows, over the layout routing (channel routing is not used), circuit routing performed before layout generation and layout generation targeting delay reduction by the application of the sizing technique. The possibility to implement any logic function, without restrictions imposed by a library of cells, allows the circuit synthesis with optimization in the number of the transistors. This reduction in the number of transistors decreases the delay and power consumption, mainly the static power consumption in submicrometer circuits.

Comparisons between the proposed strategy and other well-known methods are presented in such a way the proposed method is validated.

Keywords: Automatic layout generation, static CMOS logic gates, power and timing optimization.

Geração Automática de Leiautes de Circuitos CMOS Estáticos Visando Diminuição de Atraso e Consumo

RESUMO

A crescente evolução das tecnologias de fabricação de circuitos integrados demanda o desenvolvimento de novas ferramentas de CAD. O desenvolvimento tradicional de circuitos digitais a nível físico baseia-se em bibliotecas de células. Estas bibliotecas de células oferecem certa previsibilidade do comportamento elétrico do projeto devido à caracterização prévia das células. Além disto,diferentes versões para cada célula são requeridas de forma que características como atraso e consumo sejam atendidos, aumentando o número de células necessárias em uma bilioteca.

A geração automática de leiautes é uma alternativa cada vez mais importante para a geracão baseada em células. Este método implementa transistores e conexões de acordo com padrões que são definidos em algoritmos sem as limitações impostas pelo uso de uma biblioteca de células. A previsibilidade em leiautes gerado automaticamente é oferecida por ferramentas de análise e estimativa. Estas ferramentas devem ser aptas a trabalhar com estimativas do leiaute e gerar informações relativas a atraso, potência e área.

Este trabalho inclui a pesquisa de novos métodos de síntese física e a implementação de um gerador automático de leiautes cujas células são geradas no momento da síntese do leiaute. A pesquisa investiga diferentes estratégias de disposição dos componentes (transistores, contatos e conexões) em um leiaute e seus efeitos na ocupação de área e no atraso e de um circuito. A estratégia de leiaute utilizada aplica técnicas de otimização de atraso pela integração com uma técnicas de dimensionamento de transistores. Isto é feito de forma que o método de folding permita diferentes dimensionamentos para os transistores.

As principais características da estratégia proposta neste trabalho são: linhas de alimentação entre bandas, roteamento sobre o leiaute (não são utilizados canais de roteamento) e geração de leiautes visando a redução do atraso do circuito pela aplicação da técnica de dimensionamento ao leiaute e redução do comprimento médio das conexões. O fato de permitir a implementação de qualquer combinação de equações lógicas, sem as restrições impostas pelo uso de uma biblioteca de células, permite a síntese de circuitos com uma otimização do número de transistores utilizados. Isto contribui para a diminuição de atrasos e do consumo, especialmente do consumo estático em circuitos submicrônicos.

Comparações entre a estratégia proposta e outros métodos conhecidos são apresentadas de forma a validar a proposta apresentada.

Palavras-chave: Geração Automática de Leiautes, Portas Lógicas Estáticas CMOS, Otimização em Atraso e Consumo.

1 INTRODUCTION

CAD (Computer-Aided Design) tools are used since of the beginning of the microelectronics. They have great importance because of the need for obtaining products that lies on time-to-market. As new technologies are available each time more frequently, it is fundamental the use of CAD tools to design circuits.

Another point of influence in the use of CAD tools is the complexity of VLSI systems. Small commercial circuits may contain hundred or thousand of transistors, becoming hard the manual development.

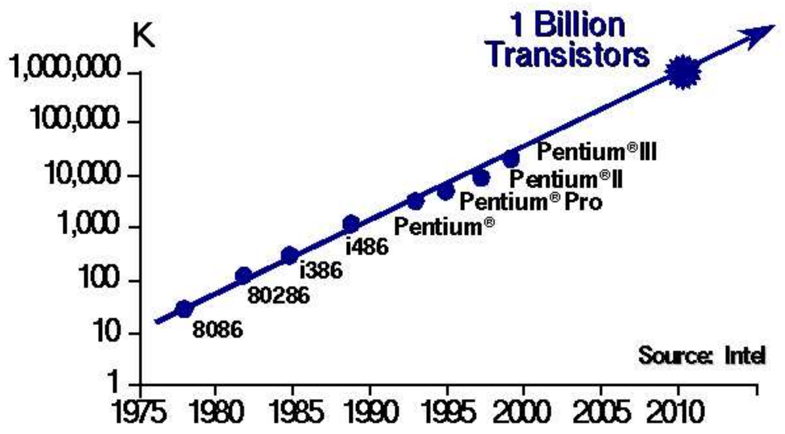


Figure 1.1: Increase of the Number of Transistors in Microprocessors

Figure 1.1 shows the increase of the number of transistors in a microprocessor family. It is possible to verify that the increase of the number of transistors grows linearly and it is estimated to manufacture processors with one billion of transistors by 2010.

In state-of-the-art and next generation VLSI circuits, geometries get smaller, clock frequencies keeps increasing and on-chip interconnect gains increased importance (CONG; SARRAFZADEH, 2000). In addition, problems in physical design are getting more complex and EDA (Electronic Design Automation) tools are essential to solve current design problems (SARRAFZADEH et al., 2001).

From $0.13\mu m$ and smaller, ICs are more susceptible to breakdown during fabrication (antenna effect) or to wear-out over time (electro-migration). Dealing with these issues requires careful planning (OTTEN; CAMPOSANO; GROENEVELD, 2002). This emphasizes the need for EDA tools able to automatically implement and validate integrated circuits.

Traditional physical level design relies on standard cell libraries. Standard cell methodology can be used for ASIC design because the layout process is partially automated. In addition, cell libraries offer some predictability to design because cells are characterized before synthesis.

However, standard cell libraries have limited number of cells. This imposes some restrictions in the layout synthesis. Also, different versions of each cell are required in order to drive different strengths, thus, increasing the number of elements in libraries to some hundreds of cells.

An alternative to the standard cell based layout generation is the automatic custom layout generation approach. In this approach it is not used cells from a library. Instead, it generates each element (transistors, contacts and connections) according to a layout pattern that is intrinsically programmed within its algorithms. In addition, automatic custom generation can be flexible to create optimized layouts to each location where they are inserted. Thus, it is a library free approach.

The automatic layout generation appeared with the Weinberger matrix (WEINBERGER, 1967). Other works on automatic cell generation were those of Lopez and Law (LOPEZ; LAW, 1980) and Uehara and Cleemput (UEHARA; CLEEMPUT, 1981). The formers introduce a layout style known as gate matrix, while the latters present the so-called linear matrix layout style. Both works were originally developed having in mind the automatic generation of cell libraries. Current layout generators are mainly based on the linear matrix style and they are able to generate modules with thousands of transistors.

In automatic custom layout generation, analysis and estimation tools must offer the predictability. An analysis tool can work well with layout estimates and generate accurate information about the layout.

In (GÜNTZEL, 2000) it is presented important concepts about timing analysis in VLSI circuits. Guntzel reports that timing verification targets at determining whether the timing constraints imposed to the design may be satisfied or not. More strictly, timing verification is concerned with estimating the critical delay of circuits and the maximal operating frequency, in case of clocked circuits.

The accuracy of timing verification is completely dependent on the accuracy of the used circuit models. By circuit models it is meant not only the physical delay model used to quantify the delay of each component, but also the models for computing circuit component delay and the circuit delay itself.

Timing analysis associated to layout generation can improve accurate timing optimization characteristics to the circuit design. Besides, timing analysis can be used to improve layout optimization by gate sizing and buffer insertion techniques.

Gate sizing is a technique to optimize each individual gate of the circuit, given certain characteristics. Characteristics refer to widths of the transistors, relations between gates and their implication in the timing.

Buffer insertion consists on the insertion of buffers in the circuit in order to distribute the capacitance of the original gate over the buffers reducing the delay of the path.

Santos et al. present in (SANTOS et al., 2003) a gate sizing tool able to perform timing optimization. The technique uses a timing analysis tool in order to estimate long paths and apply gate sizing in the critical path.

The number of transistors in a circuit is directly related to the power consumption.

Thus, logic optimization targeting reduction on the number of transistors associated to a layout generation able to implement a large set of logic functions can reduce the total power consumption of a circuit.

Layout optimization is very important in the development of integrated circuits (IC). No chip is designed without analyzing the environment where it is used. According the project application, characteristics as area, speed and power dissipation are studied to generate best results.

In this work it is presented a research about physical synthesis and the implementation of an automatic custom layout generator. The research is related to layout strategies and its effects on area occupation and on the delay of the circuit. As result it is presented a new strategy to automatic custom layout generation in which small area occupation and delay reduction are aimed.

The layout generation strategy applies techniques to timing optimization by the integration with a gate sizing technique. This is performed with the application of a folding technique that resize the transistors.

A tool called Parrot Punch was developed in order to validate the proposed strategy and it is presented in this work.

1.1 Organization of This Work

The work is organized as follows. Chapter 2 presents some basic concepts about layout generation available in the literature and the time line of techniques and tools developed at the UFRGS microelectronics group. Besides, it is shown an overview on the design flow, starting from the high level design until the physical level.

Chapter 3 presents different techniques often used in layout optimization, its characteristics and its effects in circuits. Works related to layout optimization are reported and the method proposed in (SANTOS et al., 2003) is highlighted.

Main aspects on layout synthesis are presented in chapter 4. It is shown important aspects about layout generation, effects on layout optimization and the power supply distribution. The placement and routing problem is also reported. To finish this chapter, some experiences on automatic layout generation is emphasized as consequence to the proposed layout generation strategy.

Chapter 5 presents a new strategy to the automatic generation problem. It is proposed a new method to layout generation without the use of library cells and the used algorithms are reported.

Experimental results are reported in chapter 6. Results obtained from the comparison between the strategy described in chapter 5 and other well known methods to layout generation are shown and layout optimization results are discussed.

Finally, some concluding remarks and suggestions to future works are offered.

2 LAYOUT GENERATION OF INTEGRATED CIRCUITS

2.1 Introduction

The complexity of the systems increase and also the need for design automation on more abstract levels where functionality and tradeoffs are easier to understand (GAJSKI et al., 1992). Thus, techniques on different abstraction levels can be used in order to generate accurate integrated circuits given the circuit specification.

When ASIC is aimed, the design of circuits may starts in higher abstraction level than the transistor level. In section 2.2 it is given some basic concepts about abstraction levels and characteristics that implies on the layout generation.

Researches on automatic layout generation have been done since the 60's. Nowadays, academic and commercial tool are able to generate layouts with thousands of transistors. In section 2.3 it is shown some layout techniques found in the literature. Section 2.4 presents a brief description of the physical synthesis at the UFRGS microelectronics group.

2.2 A High Level to Layout Level Flow Overview

Figure 2.1 shows a flow in which a system design is separated in tree levels of abstraction. When the goal is develop an ASIC, the design may starts in any level in function of circuit complexity. In each design level, tools are used to optimize and to validate automatically the design according to the designer requirements.

Following is reported some characteristics of the related flow and it is given a brief description about *high level design*, *logic level design* and *physical level design*. As the focus of this work is the automatic custom layout generation, the discussion is detailed on the physical level design.

Nowadays, design flows may starts from a higher abstraction level than the high level design. System C and System Verilog are examples of description languages in system level. They have been used as a solution to describe and validate systems.

High Level Design

System designers think in terms of states and actions triggered by external and internal events, and in terms of computations and communications (CAMPOSANO; WOLF, 1991). This emphasizes the use of high-level descriptions on the development of integrated circuits.

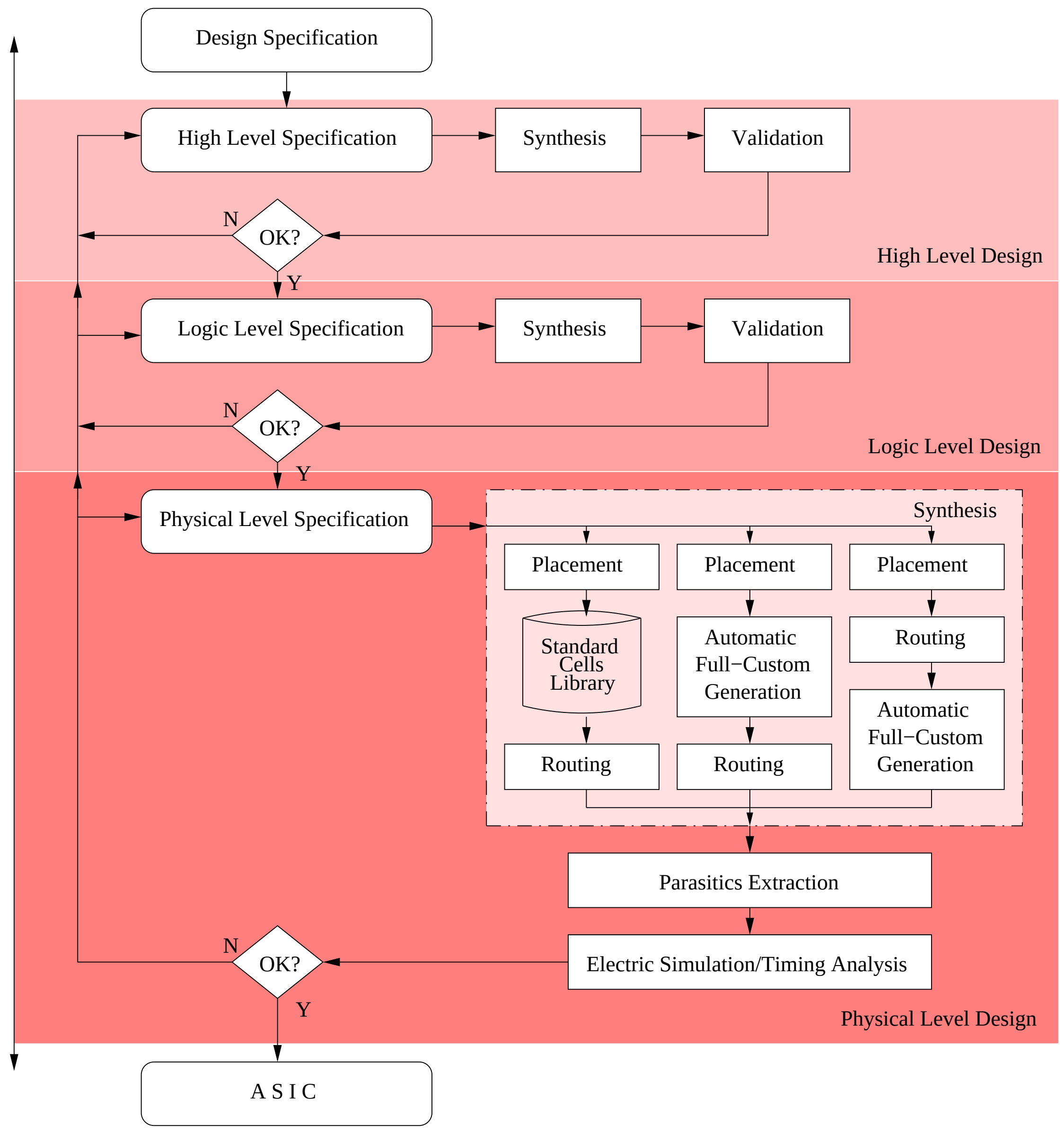


Figure 2.1: Different Levels in a Project Design

High-level synthesis takes a specification of the behavior of a digital system with a set of constraints and goals on the resulting hardware to be satisfied, and finds a structure that realizes the given behavior while satisfying the given goals and constraints (KU; MICHELI, 1992).

The behavior is described as an algorithm in which the implementation is separated in *data-path* and *control* parts. The data-path contains functional units, register and their interconnection. The control is responsible to activate the behavior of the data-path.

A high level coding can be developed in behavior level or RTL (Register-Transfer Language). In a behavior description, the coding is based on the behavior of each component used in the HDL code while a RTL description presents structural elements close to the logic of the hardware. Figures 2.2 and 2.3 illustrates VHDL behavior description and VHDL RTL description, respectively.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ripple_carry is
    generic( N : natural := 4 ;
    port( a :in std_logic_vector(N-1 downto 0);
          b :in std_logic_vector(N-1 downto 0);
    s :out std_logic_vector(N-1 downto 0);
    cin :in std_logic;
    cout :out std_logic );
end entity;

architecture estrut of ripple_carry is
   signal carry : std_logic_vector(N downto 0);
begin
   carry(0) <= cin;
   cout <= carry(N);

   add:for i in 0 to N-1 generate
   begin
      s(i) <= carry(i) xor a(i) xor b(i);
      carry(i+1) <= (carry(i) and a(i)) or
                    (carry(i) and b(i)) or (a(i) and b(i));
   end generate;
end architecture;
```

Figure 2.2: VHDL Behavior Description

The most known and used high level description languages are VHDL and Verilog. Usually VHDL is more frequently found in academic projects and Verilog is widely used in the industry.

```
entity fulladder is
  port (V1,
        V2,
        Carry_in : IN  bit;
        Carry_out,
        S : OUT bit );
end entity;

architecture str of fulladder is
   signal t1, t2 : bit;
begin
   t1 <= V1 xor V2;
   t2 <= V1 and V2;
   S  <= Carry_in xor t1;
   Carry_out <= t2 or ( Carry_in and t1 );
end architecture str;
```

Figure 2.3: VHDL RTL Description

Logic Level Design

The benefits of automating the logic design process are lost if the result does not meet area, speed and power constraints, while optimizing the design tradeoffs as well as an expert designer could (HACHTEL; SOMENZI, 1996). This highlights the importance of the quality of the tools used in synthesis not only in the logic level but also in whole design flow.

Hachtel and Somenzi also report accepted optimization criteria for logic synthesis. Logic tools must be able to workout minimize on:

1. Area occupied by logic gates and interconnects;
2. The delay of the critical path through the logic;
3. The degree of testability of the circuit (fault coverage);
4. Power consumed by the logic gates.

Algorithms for logic optimization must be able to satisfy these quantities based on designer requirements. The area of logic gates and their interconnect is directly related to the area occupied by the circuit while the delay of the circuit is given by its critical path. The minimization of power consumption is very important nowadays because of the increasing use of mobile technologies.

Manufacturing for testing is another important aspect in integrated circuits design that determines when a fabricated chip works as expected.

Physical Level Design

At the physical level, the layout generation of ICs is done based on technology process and design specifications. The result of the physical design is a layout and a set of estimates (i.e. circuit delay, area and power) to guarantee that the circuit can be manufactured in accordance to its specifications.

Usually, this level can start from a transistor schematic or a description containing a transistors and connections lists(i.e. SPICE netlist). In both cases it is presented full information needed for generating the IC. In addition, other specifications circuit delay and power consumption demanded by the designer can be constrained to the physical design.

Synthesis based on a transistor netlist can be separated in two branches. First, the standard cell approach is important because it is the most used in academic and industry environment. The other method is related to the automatic custom generation. The custom generation can be separated into other two parts which is explained below.

Traditional physical level design relies on standard cells libraries. These cell libraries can be generated automatically or manually based on some constraints defined by the designer.

However, standard cell libraries have a limited number of cells that imposes restrictions to layout synthesis because it is not possible to implement any logic function but only the ones that are available in the library. Also, different versions of each cell are required in order to drive different strengths increasing the number of cells in libraries to some hundreds.

Automatic full-custom layout of circuits is an alternative to the standard cell approach. This method consists on generating a block or a complete circuit in a fully automatic way without using any cell library. This means that the circuit is generated based only on a transistor netlist and the cells are generated on the fly.

It is clear that the standard cell approach is faster to generate solutions because of the possibility to search a cell in a library and to place it into the layout. This search, targeting a specified logic gate, can find a set of cells with the same logic function in which they are distinguished in delay, area and power optimization. The synthesis tool must be able to choose a cell with the best as possible solution given the demand for a logic function.

Automatic full-custom layout generation differs from the standard cell approach by the methodology flow where it is generated layout on demand. Thus, all transistors are placed into the layout and their connections are performed on-the-fly.

In order to produce automatic custom synthesis able to generate layouts that relies with the circuit specifications, other tools for area, delay and power estimation must be associated to the synthesis flow.

As reported above, automatic custom generation can be separated into two synthesis flows. The difference lies on the integration between automatic layout generation and the routing step. The two methods are similar but different results can be obtained from each other.

The first physical synthesis flow is the traditional one in which the placement of the logic gates are done based on area estimates. After, the synthesis of the transistors is performed and the layout is generated. As last step, the routing is done to implement gates connections.

Tavares (TAVARES, 2003) proposes another method for automatic layout generation. This method is very interesting due to the possibility to perform the routing before the layout generation. The placement is done exactly as the traditional method but the circuit layout is generated only when the routing is completed.

This idea is more interesting when the routing algorithm has, not fixed points in a net, but a range of possible positions in each logic gate. Thus, the routing can be easily performed and the result given by the routing algorithm is better than the method with fixed point to connections.

After synthesis, tools for parasitics extraction and simulation are used to validate the generated layout. Related to parasitics are capacitances and resistances present in the layout. These parasitics are responsible by the delay and power consumption of a circuit.

Cadence Diva^{TM}, Mentor ICExtract^{TM} and LASCA are examples of parasitics extraction tools. Diva and ICExtract are commercial tool widely used in the industry and LASCA is an academic tool presented in (FERREIRA; MORAES; REIS, 2000). These tools are able to extract capacitances and resistances of transistors and connections.

There are two main ways to validate the layout after synthesis and extraction. First, an electric simulation based in an electric model may be used to generate a waveform in the PO (primary outputs) based on sources applied to the PIs (primary inputs). Thus, it is possible to calculate the delay of the circuit based on the waveform differences.

A timing analysis tool is a faster way to estimate the delay of a circuit. In this type of delay estimation, the analysis tool extracts the critical path of the circuit and it is able to present the delay and the patterns that applied to the PIs are responsible by this delay.

2.3 Layout Generation Strategies

The *Weinberger matrix* (WEINBERGER, 1967) is the first method to synthesize boolean functions. In this method, a circuit containing only NOR logic are implemented in a NMOS matrix. In figure 2.4 it is possible to verify that this approach is very simple.

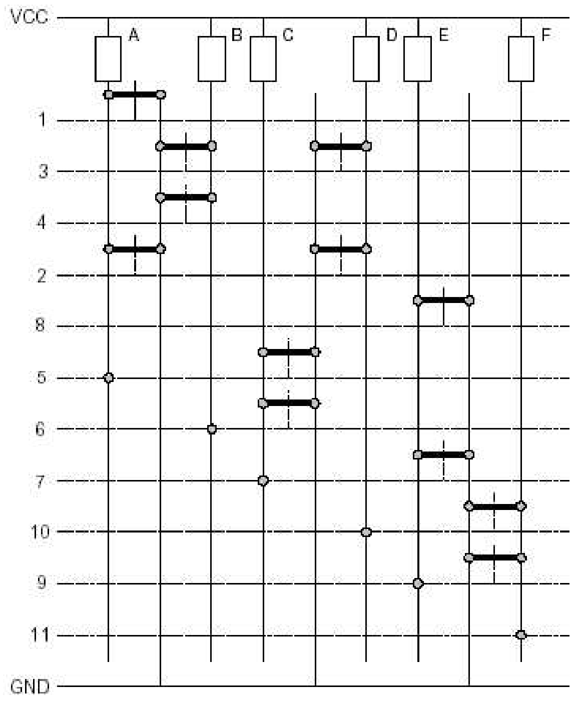


Figure 2.4: Weinberger matrix

Note in this example (figure 2.4) that the lines does not use all the space. The algorithm *left edge* is presented in (HASHIMOTO; STEVENS, 1971) in which the area of the circuit is reduced with a better utilization of the lines. Figure 2.5 shows the *left edge* algorithm applied to the example in figure 2.4.

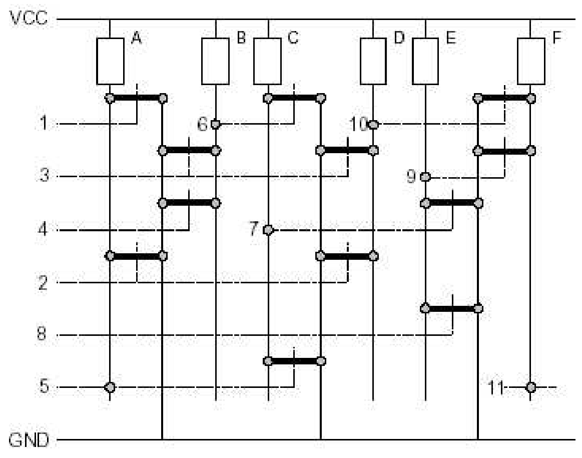


Figure 2.5: Left edge algorithm applied to an Weinberger matrix

The *gate array* consists on using a set of prediffused cells in a matrix. Metal layers inserted in the layout define the logic function. In order to support different circumstances, the cells present transistors with width greater than the necessary. Thus, power consumption and timing performance problems can be found in this approach.

In the *standard cell* approach, cells stored in a library are used to generate the layout. Each cell is defined by a logic function and presents specific characteristics related to power consumption and timing.

The layout generation based on the standard cell approach is basically divided into two steps. First, the *placement* is performed in which the position of each cell is found in the layout surface. Once the placement is done, the routing of the cells can be performed. The main goal of the placement step is to reduce the congestion in order to guarantee the complete routing of the circuit.

The *gate matrix* (LOPEZ; LAW, 1980) is introduced as an approach to VLSI systems. In this method, complex gates are placed in vertical polysilicon lines with spaces used to connect the gate of transistors. Usually, the number of inputs of a gate defines the number of polysilicon columns. Figure 2.6 illustrates the gate matrix approach.

In (UEHARA; CLEEMPUT, 1981) it is proposed a technique called *linear matrix*. In this layout style, only 2 diffusion strips, one PMOS and one NMOS, are placed horizontally and the polysilicon lines are placed vertically over the diffusions in order to generate the transistors. The ordering of the transistors in the diffusion strips is also defined in (UEHARA; CLEEMPUT, 1981) by the search of the *Euler Path* algorithm.

The main characteristics of the linear matrix style are:

- Used in static CMOS layouts;
- Implement series/parallel transistors;
- Dual P/N planes;

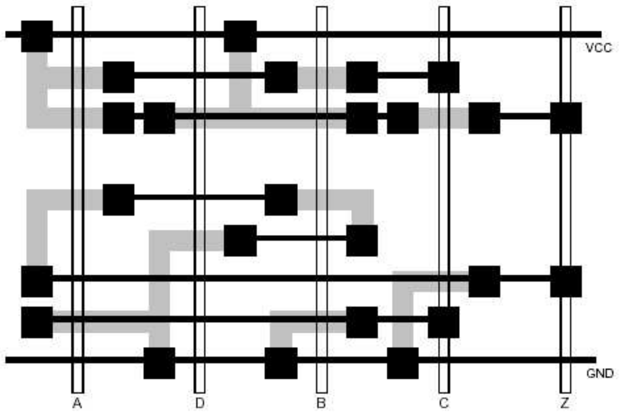


Figure 2.6: An example of gate matrix

- 2 diffusion strips to implement transistors;
- Transistors with the same gate signal are aligned.

Many algorithms are presented to increase the efficiency of this layout style. New algorithms for ordering transistors in the linear matrix are presented in (MAZIASZ; HAYES, 1987; WIMMER; PINTER; FELDMAN, 1987; WANG, 1989, 1990). These algorithms show techniques to improve better results with the relative position of the transistor inside the cells, decreasing the occupied area.

Algorithms for reduction of the rows height is presented in (ONG; LI; LO, 1989; MAZIASZ; HAYES, 1991; NAKAGAKI; YAMADA; FUKUNAGA, 1992). These algorithms try to optimize the area occupied by connections in the channel routing in order to reduce the area of the cells.

In (WANG, 1993) it is presented an alternative to the previous algorithms in which the power supply lines are placed between P and N diffusion strips. In this method, the routing is performed outside the row and connections between P and N nodes are implemented in metal 2.

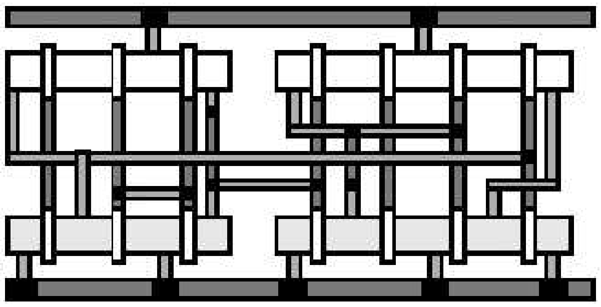
Kaneko and Jialin present in (KANEKO; JIALIN, 1997) a concurrent cell generation and mapping for CMOS circuits. This approach includes the following problems:

- Generation of the transistor topology for a cell, which is done concurrently with cell assignment;
- Determination of each transistor size;
- Generation of physical layout of each cell.

In (GUPTA; HAYES, 2000), it is presented a technique for the automatic generation of layouts of CMOS cells in the two-dimensional (2D) style. The technique, CLIP (Cell Layout via Integer Programming) is based on integer-linear programming and solves both width and height minimization problems for 2D cell.

Width minimization is formulated in form that combines factors influencing the 2D cell width in a common problem space: transistor placement, diffusion sharing and vertical inter-row connections. This space is searched in a systematic manner by the branch-and-bound algorithms used in ILP solvers. For height minimization, cell height is modeled based on the horizontal wire density.

The CLIP run time for width minimization is in seconds for circuits with 30 or more transistors. For both height and width optimization, the CLIP is practical for circuits with up to 20 transistors. To extend the algorithms to larger circuits, hierarchical methods are necessary.


(a) One-dimension layout

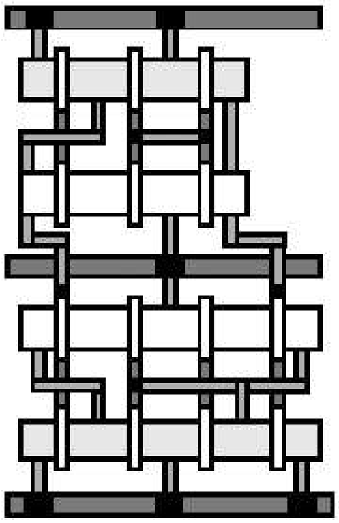

(b) Two-dimension layout

Figure 2.7: the CLIP layout style

Figure 2.7 shows an one-dimension layout in figure 2.7(a) and the same two-dimension layout in figure 2.7(b). It is important to note that the three routing horizontal tracks in the two-dimension layout are distributed in the two-dimensional layout.

A transistor level placement tool is presented in (DASH et al., 2000). The used technique divides large circuits into groups of strongly connected transistors. In the proposed algorithm, series connected transistors having the same substrate potential can be placed together to form chains of transistors called stacks and adjacent transistors in a stack share a common source/drain diffusion area.

Each stack represents a locally optimal placement of transistors. The width of each stack can also be controlled using transistors folding. The entire circuit can be broken up into stacks. In this work, the simulated annealing algorithm is used to place these stacks.

A datapath tile placement and routing at transistor level is presented in (SERDAR; SECHEN, 2001). In this work, there are two signal flows in the layouts. One is the data flow, which runs vertically along the power rails. The other is the control flow, which goes horizontally. Since a tile is replicated across an entire row, it is sufficient to optimize the area of a single tile at a time.

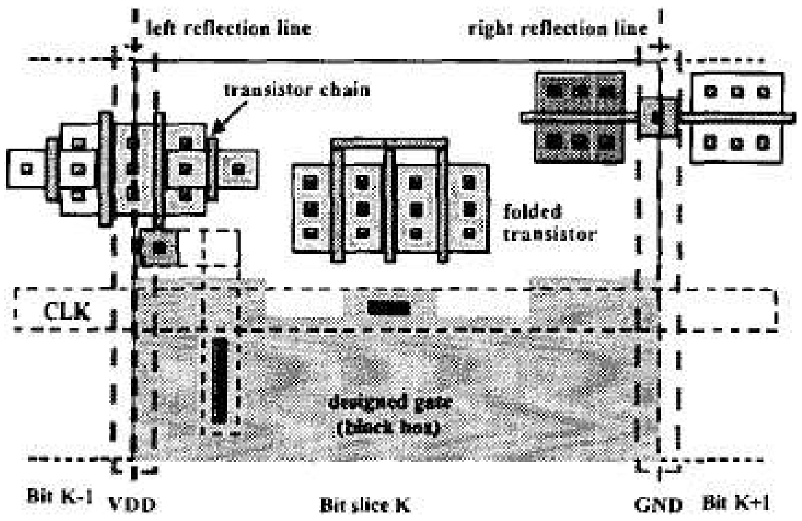


Figure 2.8: Possible placeable devices within a datapath tile

The placement process is divides into two steps: a global placement step using simulated annealing, and a detailed placement step based on extensive modifications using the O-tree algorithm. The proposed algorithm places the tiles mirrored. Thus, devices can be placed such that geometry sharing is possible between adjacent tiles in the datapath array. In figure 2.8 the transistor chain shares the diffusion contact in the left side and the single transistor shares the poly/metal1 contact in the right side.

As obtained results, it is presented four small circuits (10 to 70 transistors) generated by the proposed algorithm. It is important to note that these small circuits are used in big ones in order to generate real datapath circuits.

2.4 Layout Generation at UFRGS Microelectronics Group

Many tools have been developed at the UFRGS microelectronics group (GME) for researching and developing new techniques related to the physical synthesis. Figure 2.9 shows the time line of tools developed at GME.

In (REIS, 1983) it is presented a research about control parts of microprocessors designed in the 70's. A new set of concepts is presented in (REIS, 1987) called TRANCA (TRANsparent-Cell Approach) methodology. The main goal of this methodology was

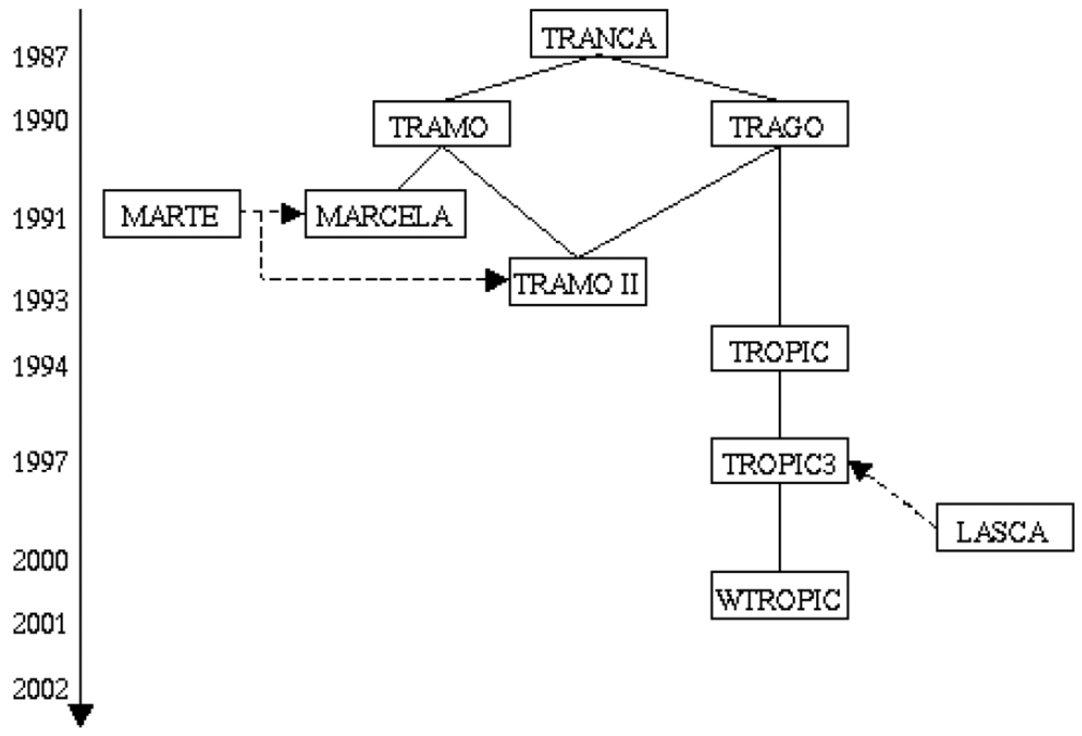


Figure 2.9: Time line of physical synthesis at GME

to use the knowledge acquired in the research of commercial layouts and to apply the techniques in automatic tools.

TRAMO (TRAnca MOdule) generator (LUBASZEWSKI, 1990) was the first developed system based on the TRANCA methodology. In this work, layouts were generated based on a library of cells.

In TRAGO (TRanca Automatic GeneratOr) (MORAES, 1990), synthesis was performed in such a way, layouts were automatically generated without cells library. The system used a description similar to a netlist spice to generate the layout.

Gate-array and sea-of-gates are examples of circuits generated by Marcela (GÜNTZEL; RIBAS; REIS, 1991; GÜNTZEL, 1993; GÜNTZEL et al., 1995) approach. In this approach, over-the-cell routing and track allocation is used to avoid routing congestion.

MARLA (GÜNTZEL et al., 1993) and MARTE (JOHANN; REIS, 1993) are tools to generate layouts based on the MARCELA approach. MARLA was responsible to generate the layout under a matrix and, MARTE was used to route the cells. The main characteristics of the circuits were the full over-the-cell (FOTC) routing.

The main goal of TRAMOII (REIS; REIS, 1991; REIS, 1993) was the layout generation of ICs in which cells are transparent to metal 1 and metal 2 connections. In TRAMO, only one metal layers was used.

In (MORAES, 1994) it is presented a tool called TROPIC (Transparent Reconfigurable Optimized Parametrizable Integrated Circuit generator) based on the concept of layout generation based on a transistors netlist. In this work, layouts can be generated on-the-fly based on design rules. Layouts generated by TROPIC were limited to technologies with 2 metal layers.

TROPIC3 is presented in (MORAES; REIS; LIMA, 1997). In this work, 3 metal layers were used in the routing. This modification decreased up to 50% in the area between rows.

A tool for electric extraction of circuits generated by TROPIC was presented in (FERREIRA; MORAES; REIS, 2000). It is possible to extract resistances, coupling capacitances and intrinsic capacitances by layouts of ICs.

WTROPIC (FRAGOSO, 2001) is presented as a tool to layout generation to be used via Internet. Besides, some optimizations are presented (in comparison of TROPIC3) in which the transistor density increase up to 4%.

2.5 Conclusion

The complexity of the systems demand higher abstraction levels than a transistor description. Thus, the design of circuits can be developed in a high level description and optimized in the high, logic and physical level in order to meet the designer specifications.

At the physical level, three main options are available to designers according the synthesis strategy. The classic standard cell approach can be used in such a way elements stored in a library are placed and routed.

The layout can be generated fully automatic without using any library of cells in which the placement, the layout generation and the routing are performed in this sequence. As last option to layout generation, the full custom automatic layout generation can be done in which the layout generation is performed only when the routing is performed. This strategy offers to the layout generation some freedom concerning the routing process.

Tools have been used to automate the layout generation process since the 60's. Many strategies are proposed in the literature starting from the Weinberger matrix until the standard cell generation. At UFRGS microelectronics group, researches on full custom automatic layout generation have been developed in order to generate layouts similar to done manually.

3 OPTIMIZATION TECHNIQUES USED IN ASSOCIATION WITH LAYOUT GENERATION

3.1 Introduction

The association with analysis techniques improves the efficiency of the layout generation. Logic optimization, timing analysis and power consumption estimate are examples of techniques essential for generating layouts able to be manufactured.

Logic optimization and its effects are reported in section 3.2. It is shown some techniques for logic optimization targeting reduction of power consumption and the importance of the technology to the layout generation.

In section 3.3 it is reported important aspects about critical paths in combinational and sequential circuits, path sensitization criteria and clock distribution techniques. Besides, clock tree synthesis and the concept of clock skew is also reported.

Techniques for timing optimization are presented in section 3.4. Cost models for area occupation and timing analysis are reported and techniques for gate sizing and buffer insertion are shown. In section 3.5 it is reported details about a gate sizing tool developed with the goal of resize circuits generated by automatic full custom generators.

3.2 Logic Optimization and Power Consumption Reduction

With the increased popularity of portable devices, battery size and lifetime are becoming important factors in the design time. At the same time, the amount of data to be processed is increasing at a rapid pace (IMAN; PEDRAN, 1996). These considerations have resulted in a growing need for minimizing power consumption in digital systems.

There are three main components of power consumption in digital CMOS VLSI circuit (SHAMS; BAYOUMI, 2000).

1. *Switching Power:* consumed in charging and discharging of the circuit capacitances during transistor switching;

2. *Short-Circuit Power:* consumed due to short-circuit current flowing from the power supply to ground during transistor switching;

3. *Static Power:* consume due to static and leakage currents flowing while the circuit is in a stable state.

In non-deep-submicrometer technologies, dynamic power consumption is the dominant source of power consumption. Moreover, the dominant source of power dissipation is due to capacitive currents because of the charging and discharging of the capacitances. Many works (ALIDINA et al., 1994; PRADHAN et al., 1996; IMAN; PEDRAN, 1996; WANG; VRUDHULA, 1996) related to power estimation and optimization use the following equation to approximate the average power dissipated by a gate:

$$P_{avg} = \frac{1}{2} \times v_{DD}^2 \times C_{load} \times f \times E(transitions) \tag{3.1}$$

where v_{DD} is the supply voltage, C_{load} is the load capacitance, f if the clock frequency and $E(transitions)$ is the number of gate output transitions per global clock cycle.

However, high leakage current in deep-submicrometer regimes is becoming a significant contributor to power consumption of CMOS circuits when threshold voltage, channel length and gate oxide thickness are reduced (ROY; MUKHOPADHYAY; MEIMAND, 2003). Thus, the dynamic (switching) power can be estimated by the equation 3.1, but when the transistors are not switching, the power consumption is approximated by

$$P_{LEAK} = I_{LEAK} \times V_{DD}. \tag{3.2}$$

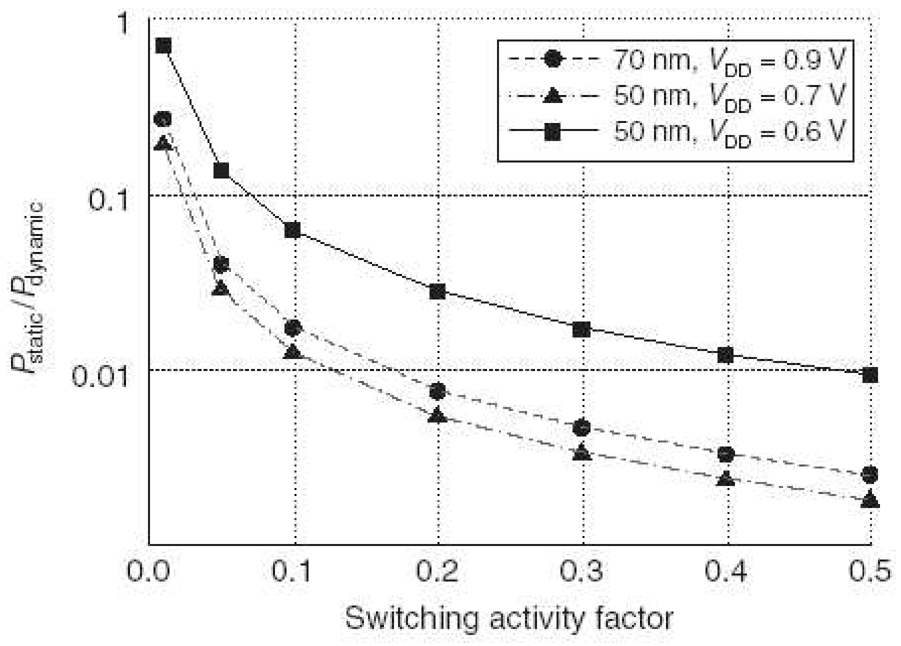


Figure 3.1: Ratio of static to dynamic power consumption

Figure 3.1 is presented in (SYLVESTER; KAUL, 2001). The figure illustrates the relative importance of static and dynamic power for an inverter driving a fan-out of four gates identical to the inverter. It is shown that for logic with switching activities on the order to 0.01 to 0.1, static power can approach and exceed 10% of dynamic power.

In (NGUYEN et al., 2003) it is emphasize that at a 90nm technology, leakage power may make up 42% of total power. The primary reason for this increase in leakage power is the reduction of threshold voltage of devices, and an exponential increase in leakage current that it causes.

Based on equations 3.1 and 3.2, it is possible to conclude that, at a circuit level, low power application can be obtained from two ways. Reducing the switching activity it is possible to reduce the dynamic power and modifying the logic of the circuit and, consequently, reducing the number of transistors to decrease the static power.

Many authors show that including additional logic in the circuit can reduce the switching activity. Logic optimization techniques are presented in (ALIDINA et al., 1994; MONTEIRO et al., 1995) which precomputation is used to selectively disable the inputs of a sequential logic circuit, thereby reducing switching activity and power dissipation.

In (WANG; VRUDHULA, 1996), it is presented a technique to reduce the switching activity based on local logic transformations. These transformations consist on adding redundant connections or gates to reduce the power consumption. The obtained results show an average decrease in the switching activity of 13% and the increase in delay up to 29%.

These techniques increase circuit area and can adversely impact circuit performance. It is clear that the additional logic inserted in the circuit reduces the switching activity but the static power is not reduced. On the contrary, the consumed power is increased by the additional logic.

Iman and Pedran present in (IMAN; PEDRAN, 1996) a framework for specifying and maintaining power relevant circuit information of low power circuits. This methodology uses simplification techniques to optimize the logic and uses a technology mapping to make area-power trade-off during logic optimization.

In all techniques targeting low power by logic optimization presented in this section, any of them explore the possibility to reduce the number of transistors by the utilization of logic function of complex gates. The reason is related to the library cell mapping in which a limited number of defined logic functions available to the logic synthesis.

Reis presents in (REIS et al., 1997) a method for mapping a set of boolean equations into a set of static CMOS complex gates (SCCG) under a constraint in the number of serial transistors. In this work, a tool called TABA is used to generate a virtual library of complex gates. The tool is able to optimize a circuit to a set of complex gates in which the number of transistors is reduced.

In a free library mapping as used in a full automatic layout generator it is possible to generate an optimized set of cells with reduced number of transistor in comparison of a standard cell mapping.

A given limitation in the number of serial transistors induces the number of complex functions available in the virtual library. In a logic mapping where the number of serial transistors is constrained to 4, the number of possible logic functions is 396 (DETJENS et al., 1987).

The library free mapping is very important because of the reduction of the number of transistors in a circuit in comparison with a library with restricted number of logic functions. Thus, the dynamic and static power consumption is reduced due to the utilization of a bigger set of complex gates.

3.3 Timing Verification

3.3.1 Critical Delay of Combinational Blocks

Many logic paths in combinational blocks do not require any conscious effort when it comes to speed. However, usually there are a number of paths that require attention to timing details (WESTE; ESRAGHIAN, 1993). These paths are called the *critical paths*.

There are many paths to a signal that cross a circuit according the gates and the connections among them. The critical path is the slowest path to a signal cross a circuit, starting at the terminal inputs (PIs) and arriving at one of the outputs (POs).

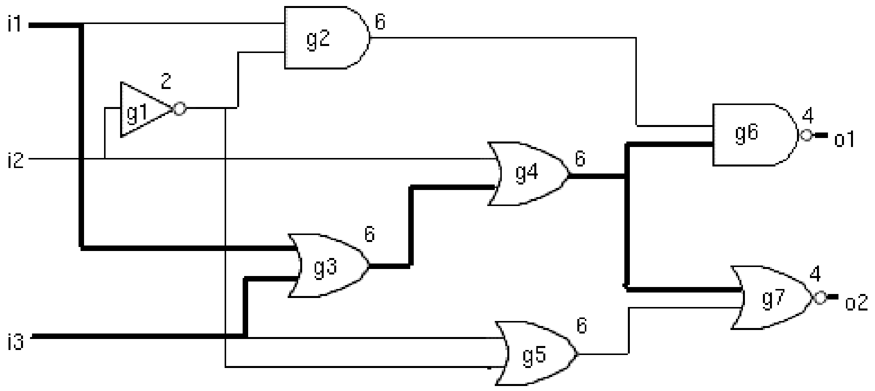


Figure 3.2: A simple critical path example

In figure 3.2 it is shown a simple example of the critical path. Consider the value at the output of each gate the delay of the gate. The critical path may starts in the inputs i_1 or i_3 and finishes in any output. The gates in the critical path may be g_3, g_4 and g_6 or g_3, g_4 and g_7.

In a circuit, critical paths require attention to timing details. These may be recognized by experience or timing simulation, but most designer use a timing analyzer, which is a design tool that automatically finds the slowest paths in a logic design (WESTE; ESRAGHIAN, 1993).

The critical path can be affected at four main levels:

- The architectural level
- The RTL/logic gate level
- The circuit level
- The layout level

In a high level implementation (architecture level), the knowledge of the algorithms that implement the function and chip parameter is required. Functions refer to how many gate delays fit in a clock cycle, how fast operation occurs or how fast memories access.

Timing optimization can be done in the RTL/logical level. In this level, pipelining, gate types (inverter, nand, etc), fan-in and fan-out are taken into account.

In the circuit level, sizing transistors or other technique to optimize the circuit can improve the critical speed path.

The speed of a circuit can be affected by rearranging the physical layout. At the layout level, placement and routing algorithms can modify the critical path in order to organize the layout in such a way that critical nets are routed with the minimum interconnect length between them.

Furthermore, only sensitizable paths contribute to the delay of a circuit. Thus, false paths must be excluded in optimizing the delay of the circuit. A sensitizable path is a path that can be activated by at least one input vector (LIN; HWANG, 1994).

3.3.1.1 Path Sensitization Criteria

Most of sensitization criteria are defined by topological parameters. Güntzel classifies the sentization criteria either as delay-dependent or delay-independent (GÜNTZEL, 2000). In delay-independent criteria, only logic values are considered while in a delay-dependent, the time such signals become stables are also considered.

In addition, Güntzel presents four sensitization criteria in the context of the floating mode. Figure 3.3 shows the reported criteria.

Static Sensitization

Static sensitization is a delay-independent criterion. It was one of the first sets of sensitization conditions used in timing analysis. When pairs of vectors assume the delay (i.e. transition mode), it corresponds as the concept of sensitization in the propagation of error signals in stuck fault test generation. Static sensitization is defined by:

Definition 3.1 *A path P is said to **statically sensitizable** if and only if there is at least one input vector w such as for each v_i, $1 \leq i \leq n$, each side-input of v_i settles to $nc(v_i)$ under w.*

Figure 3.3(a) shows the condition for static sensitization.

Static Cosensitization

Static cosensitization is also a delay-independent criteria similar to but less restrictive than static sensitization. Static cosensitization is defined as follow.

Definition 3.2 *An input vector w is said to **statically cosensitizable** to 1(0) path P in a circuit C if and only if the value of v_{n+1} is 1(0), and for each v_i, $1 \leq i \leq n$, if v_i is a controlled value, then edge e_{i-1} presents a controlling value.*

Figure 3.3(b) shows conditions for static sensitization.

Viability Analysis

In viability analysis, a set of delay-dependent conditions is used to test if a path can be considered as responsible for the delay of a circuit. A viable path is defined by:

Definition 3.3 *A path P is said **viable** if and only if there is at least one input vector w such that for each gate v_i, $1 \leq i \leq n$, and for each side-input e of v_i, if $st(e, w) < st(e_{i-1}, w)$, then $sv(e, w)$ must be equal $nc(v_i)$.*

Figure 3.3(c) shows the condition for path viability.

Exact Floating-mode Sensitization

Exact sensitization condition may declare a path as responsible for a delay of a circuit under the floating mode. Exact floating-mode sensitization is a delay-dependent sensitizable criteria defined by:

Definition 3.4 *A path P is said **exactly sensitizable** if and only if there is at least one input vector w such that for each gate v_i, $1 \leq i \leq n$, and one of the following conditions are satisfied*

1. *if $sv(e_{i-1}) = c(v_i)$, then for each side-input e of v_i, if $sv(e) = c(v_i)$ then $st(e) \geq st(e_{i-1})$*

2. *if $sv(e_{i-1}) = nc(v_i)$, then for each side-input e of v_i, $sv(e) = nc(v_i)$ then $st(e) \leq st(e_{i-1})$*

Figure 3.3(d) shows conditions for exact sensitization.

3.3.1.2 A Timing Verification Tool

In (WILKE et al., 2002), it is presented a timing verification tool called $TicTac$ based on floating vector simulation and path tracing able to identify the true critical delay of combinational blocks. It is also able to identify the long false paths that a combinational block may contain. Although being limited by the number of inputs of the combinational block to be analyzed, it also furnishes some valuable information on the circuit under analysis that may help the designer to take some important decisions and also to understand the impact of the chosen technology mapping parameters on the timing behavior of the circuit.

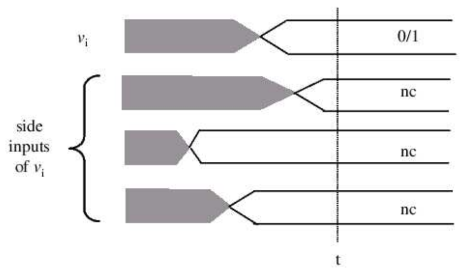
The $TicTac :: KPaths$ is a timing verification tool able to provide timing information of small combinational blocks by path tracing and floating vector simulation. Figure 3.4 shows a simple example of the graph used as data structure in the tool.

The critical delay of a circuit is computed by input floating vector simulation through the timed implication process. The path trace tool traces the k longest paths in which one expects to find the path responsible for the critical delay of the circuit. The critical path can be identified by comparing the critical delay with the delay of the k paths in the list. The main drawback of this process is the huge simulation time, which restricts its practical use to only circuits with around twenty inputs.

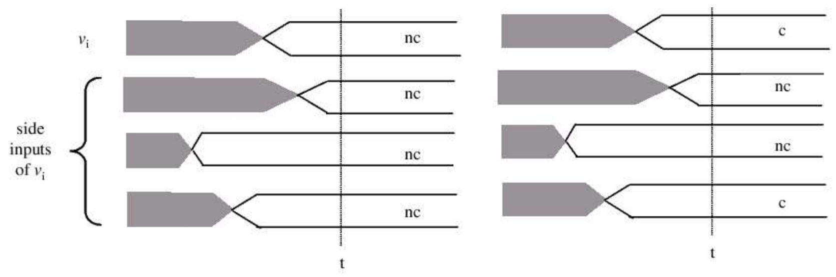
3.3.2 Clock Distribution

Clock signal is the most important signal in sequential circuits. Synchronous systems use a clock structure in order to distribute the clock signal to whole IC. Clock transitions provide a time reference to the sequential elements (state machines, registers, latches, etc).

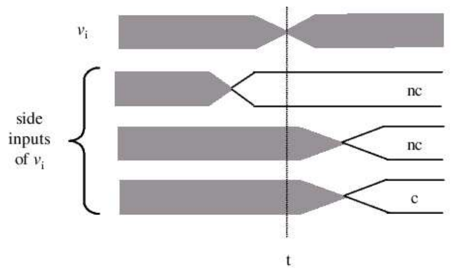
The fan-out of these sequential elements acts as a large capacitive load on the clock line. The clock wire distributed over the chip increases the final capacitance. In an Alpha



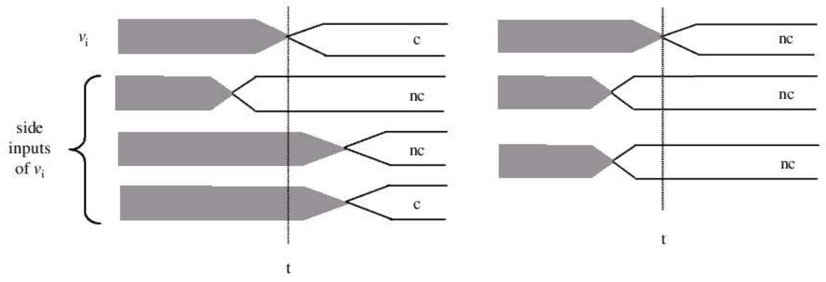
(a) Condition for static sensitization



(b) Conditions for static cosentization



(c) Condition for path viability



(d) Conditions for exact fbating-mode sensitization

Figure 3.3: Path Sensitization Criteria

microprocessor, the total clock equals 40% of the total effective switching capacitance on the chip (RABAEY, 1996).

The challenge in the clock distributions is how to distribute the clock signal to all sequential elements in the system with a optimal clock skew. Concerning clock distribution, two main techniques are used:

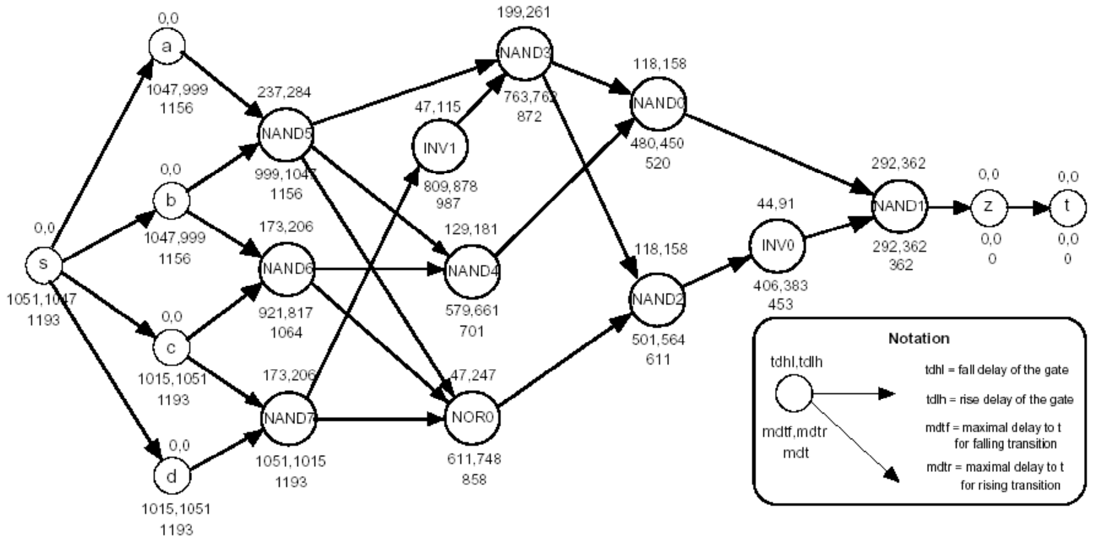


Figure 3.4: A TIC TAC example

- *a single large buffer*: used to drive a global clock.

- *a distributed clock tree structure*: uses a tree of clock buffers to feed all the modules of the system.

The first approach is used in designs that have a large number of modules (i.e. a microprocessor or a digital signal processor). The distributed clock tree approach is preferred for DSP structures. In this approach, the delay of each datapath must be carefully simulated and matched or the distribution of the clock signal can be damaged (WESTE; ESRAGHIAN, 1993). These techniques are shown in the figure 3.5.

3.3.2.1 Clock Tree Synthesis

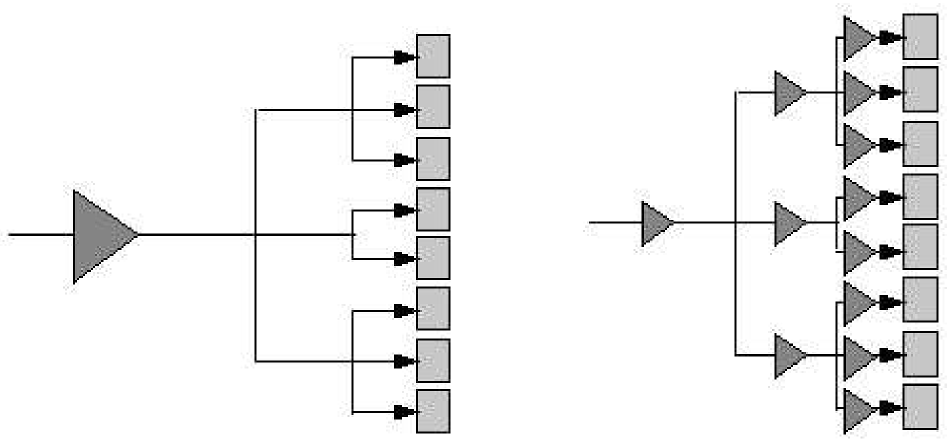
In traditional project flows, clock tree generation is a separated task. Usually, the clock implementation is built once the location of sequential elements and gated clocks are know, i.e. after placement (COUDERT, 2002).

By taking into account this problem, clock tree synthesis must be part of the refinement process. It can contribute significantly to congestion and power dissipation. Moreover, a study in the clock structure can be used to optimize the timing in the circuit.

In order to increase the performance of a circuit, simultaneous placement and buffer insertion in the clock tree may reduce the congestion and routing problems. Since the placement is flexible enough, it allows the control of the skew for timing optimization.

3.3.2.2 Clock Skew

Clock skew occurs when the clock signals arrive at storage elements at different times. The clock skew can be defined as a derivation of clock-input timings with respect to a central clock (RAVINDRAN, 2003). Furthermore, in a typical framework, all the clock-inputs to registers are at zero-skew and the best achievable cycle period is equal to the maximum delay across a combinational block.



(a) Single large buffer (b) Distributed clock tree approach

Figure 3.5: Techniques to clock distribution

A technique used to reduce the effects of the clock skew in the circuits is the *clock skew scheduling*. The objective of clock skew scheduling is to decrease the overall cycle time by balancing combinational path delays between registers.

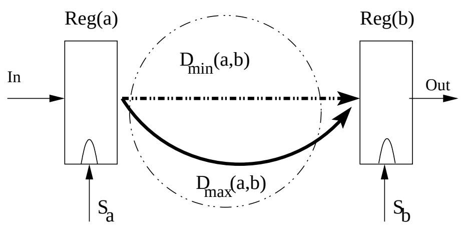


Figure 3.6: Typical synchronous circuit

In (FISHBURN, 1990), the clock skew optimization problem is formulated as setup and hold time constraints on a synchronous circuit. A typical synchronous circuit is shown in the figure 3.6. Reg_a and Reg_b are two registers, one before and another after a combinational block. $D_{min}(a, b)$ and $D_{max}(a, b)$ are, respectively, the minimum delay and the maximum delay between registers Reg_a and Reg_b. S_a and S_b are individuals skews. The circuit works using clock signals with period equals T.

The *setup* or *zero-clocking* constraint, is given by

$$S_a + D_{max}(a, b) + SETUP_b \leq S_b + T. \tag{3.3}$$

This means that the signal reaches Reg_b earlier than the next clock edge. The *hold* or *double-clocking* constraint specifies that the signal at Reg_b is safely clocked an not overwritten before the clock edge. The hold constraint is given by

$$S_a + D_{min}(a, b) \geq S_b + HOLD_b. \tag{3.4}$$

When zero-skew clock is considered, S_a and S_b are equal to zero and the minimum possible clock period T is equal to the maximum value.

In (RAVINDRAN, 2003), it is proposed an algorithm to minimize the cycle period of a synchronous circuit by optimizing clock skews. The algorithm uses the bound imposed by the critical cycle of the circuit and setup (or hold) time constraints on individual registers to compute the clock period.

A methodology for determining an optimal set of clock path delays for designing high performance VLSI-based clock distribution networks is presented in (NEVES; FRIEDMAN, 1996). The minimum clock is reduced using intentional clock skew by calculating a permissible clock skew range for each local datapath while incorporating process dependent values of the clock signal paths.

In this methodology, graph-based algorithms are used for determining the minimum clock period and for selecting a range of clock skews for each local data path in the circuit.

This work have demonstrated examples of clock distribution networks with worst case variations of up to 30% without causing circuit failure while increasing the system-wide maximum clock frequency by up to 20% over zero skew-based systems.

3.4 Timing Optimization

Layout optimization consists on choosing the best implementation for each gate in the circuit. Following are some cost models to estimate area and delay of a gate. These techniques are essential to apply any optimization technique.

3.4.1 Cost models

3.4.1.1 Area modeling

Since the area of each gate and routing area are known, the total area of a circuit can be accurately estimated. When the standard cell approach is used, to estimate the occupied area of the circuit is very realistic, but if the layout is generated without the use of a library of cells, the area estimate is not so trivial.

For the purposes of this work, the area is modeled by assuming that gates are generated using the row-based style and all strips of diffusion have the same width. Thus, the height of a row h_i is defined by

$$h_i = w_{n,i} + w_{p,i} + k \tag{3.5}$$

where k is the spacing between *p*-diffusion and *n*-diffusion, the $w_{n,i}$ is the width of NMOS transistors and the $w_{p,i}$ is the width of PMOS transistors. The total row height is the height of any gate in the row.

The length of a gate is variable in the gate sizing process. Assuming discrete gate sizes, the row length can be modified during the optimization and the gate length needs to be calculated again.

Another consideration is that in the works done by the TRANCA group at GME (see section 2.4), the gates are generated automatically on-the-fly without use of any cell library. So, the design technology rules must be considered and the length of the row L_i is given by

$$L_i = \left(t \times \left(l + 2 \times \left(spc + \frac{c}{2}\right)\right)\right) + \sum_{g=1}^{\max G_{p,n}} \left(gap_g + 2 \times \left(edc + \frac{c}{2}\right)\right) \tag{3.6}$$

where t is the number of transistors in *p*- or *n*-diffusion, l, spc and c are the minimal values given by the technology representing the transistor length, the spacing between the transistor and the fixed value to the contact, respectively. The $G_{p,n}$ is the number of the gaps in the *p*- or *n*-diffusion and the gap_g is the minimum size of a gap. The variable edc is the value to the enclosure of the diffusion to the contact.

Given this information, the total area occupied by the chip is estimated by

$$A = \max_{\forall i} L_i \times \left(\sum_{j=1}^{N} h_j + \sum_{j=1}^{N-1} K_{j,j+1} \right) \tag{3.7}$$

where the total area is estimated by the maximum length to the rows L_i multiplied by the sum of the rows height h_n and the sum of the channel spacing between rows $K_{j,j+1}$.

Figure 3.7 shows the variables used to estimate the area occupation. These variables come from the technology rules and they are presented in equations 3.5, 3.6 and 3.7.

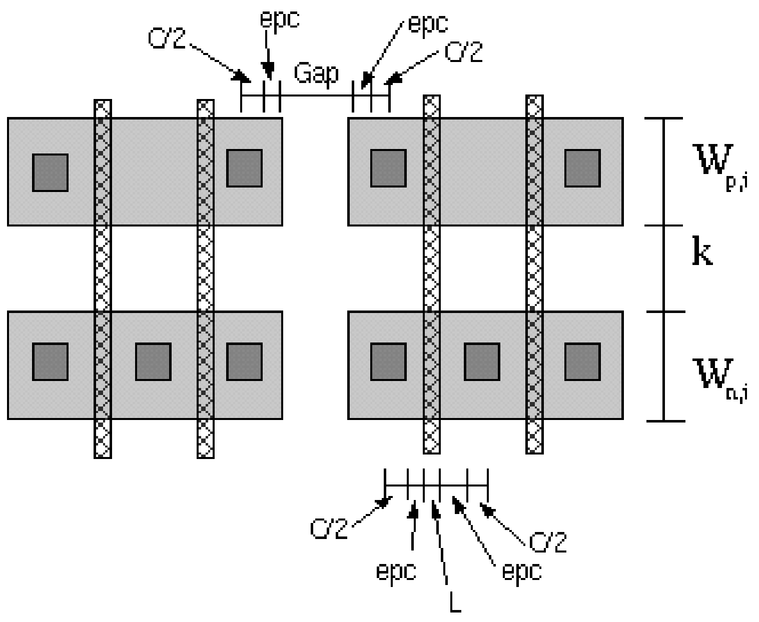


Figure 3.7: Technology rules used to estimate the area.

Table 3.1 presents a preliminary estimate of area occupation to some circuits. The results show the difficulty to estimate the size of a circuit generated automatically. Despite the estimate area in the C1908 benchmark is very close to the real value, the average error is about 20%.

3.4.1.2 *Delay modelings*

The delay propagation of the circuit is very important in the layout optimization. To arise to realistic delay propagation estimation, gate and wire delays are assumed. Moreover, the delay propagation can assume different models. These models are reported in (JACOBS, 2001) and they are shown below.

Table 3.1: Error amount in the estimated area.

Name	Gates	Error
craig	110	17.47%
7seg	230	8.30%
t481	234	18.00%
Lal	958	33.24%
Alu2	1632	22.24%
C1908	2460	3.56%
C6288	10850	24.33%
Alu4	13478	32.88%
average		19.22%

- *Zero delay model*: In the zero delay model, zero gate and wire delay are assumed in the circuit. This means that glitches are not considered and no delay differences exist.

- *General delay model*: The general model consists on delay propagation in the circuit and glitches are considered in the delay propagation.

- *Unit delay model*: In the unit model, only discrete timing events are taken into account. Thus, the propagation delay is defined in multiples of predefined units.

- *Inertial delay model*: Inertial delay model is an extension of the general model. In this model, the filtering effects displayed by CMOS-gates are included. Filtering effects make the signal shorter than the propagation delay of the gate. These effects are used to come to a more accurate estimation of the delay propagation estimation.

These models are used in commercial and academic tools. Some more detailed and more accurate than other, they are necessary in the development of timing-driven algorithms and tools. In (UEBEL, 1995), some definitions are discussed in order to classify the timing verification techniques.

1. *Electric Simulation* : The simulator uses non-linear equations to model characteristics as voltage-current (I-V) and voltage-capacitance (C-V) in the active devices.

2. *Timing Simulation* : The timing simulator uses a simplified transistor model. Usually, a linear or non-linear resistance is used in the transistor model.

3. *Timing Verification* : It is a timing simulator associated to an automatic input vector generator. Sometimes, a more simplified model is used aiming at reduce the processing time.

4. *Searching Path Algoritms* : Algorithms to search critical paths are used to find sensitizable paths in the circuit. They generate input vectors and calculate the delay of the paths.

5. *Timing Analyzer* : Timing analyzer are tools used to analyze the propagation delay to signals in the circuit. They can be electrical simulators, timing simulators or timing verification tools.

6. *Gate Level Simulation* : This kind of simulation models transistors as ideal gates. The gate level simulator assumes the logic signals and it cannot be used in circuits with analog characteristics.

Propagation delay

The delay of a CMOS gate is limited by the time taken to charge and discharge the output load capacitance C_L. An input transition results in an output transition that either charges C_L towards V_{DD} or discharges C_L towards V_{SS}.

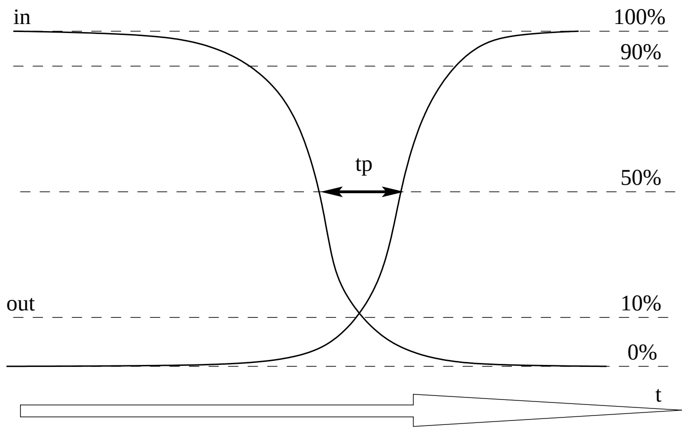


Figure 3.8: Propagation delays of an inverter

Figure 3.8 shows the propagation delay of an inverter. The logic level is high when the voltage level is more than 50% of V_{DD} and the logic level is low when the voltage level is less than 50% of V_{DD}.

The propagation delay in CMOS gates is measured from the time of the input signal in the 50% of V_{DD} to the time of the output signal in the 50% of V_{DD}. In other words, the propagation delay is the time taken for a logic transition to pass from input to output. Rising delay is measured by the time to a signal change from 10% of V_{DD} to 90% of V_{DD} and falling delay is the time between 90% of V_{DD} to 10% of V_{DD}.

An analytic model

In (WESTE; ESRAGHIAN, 1993), it is presented an analytic model that describes switching characteristics of a CMOS gate. The error is 7% to 10% in comparison to spice results (level 1). In spite of that, more detailed analysis or simulation is usually required to yield models that accurately predict the performances of today's processes.

The average delay of a CMOS gate for the analytic model is approximated by the output rise and fall time, and it is given by

$$t_{av} = \frac{t_{df} + t_{dr}}{2} \tag{3.8}$$

where t_{df} and t_{dr} are formulated by

$$t_{df} = A_N \frac{C_L}{\beta_n}, \tag{3.9a}$$

$$t_{dr} = A_P \frac{C_L}{\beta_p}. \tag{3.9b}$$

The A_N and A_p are process constants for a specific supply voltage and has been derived as

$$A_N = \frac{1}{V_{DD}(1-n)}\Big[\frac{2n}{1-n} + ln\Big(\frac{2(1-n)-V_O}{V_O}\Big)\Big], \tag{3.10}$$

where

$$n = \frac{V_{tn}}{V_{DD}}$$

and

$$A_P = \frac{1}{V_{DD}(1+p)}\Big[\frac{-2p}{1+p} + ln\Big(\frac{2(1+p)-V_O}{V_O}\Big)\Big], \tag{3.11}$$

where

$$p = \frac{V_{tp}}{V_{DD}},$$

$$V_O = \frac{V_{out}}{V_{DD}}$$

In the equations 3.9a and 3.9b are used the MOS transistor gain factor β_n and β_p. The β is dependent on the process parameters and the device geometry, and is given by

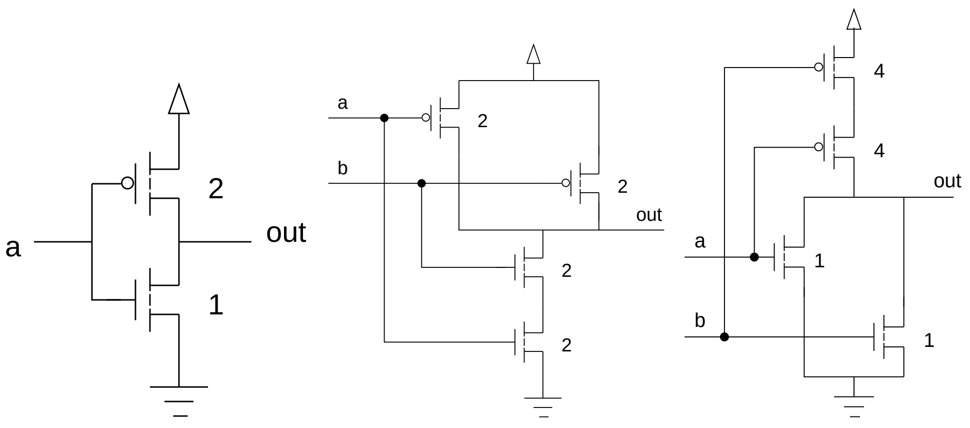
$$\beta = \frac{\mu\varepsilon}{t_{ox}}\Big(\frac{W}{L}\Big) \tag{3.12}$$

where μ is the effective surface mobility of the carriers in the channel, ε is the permittivity of the gate insulator and t_{ox} is the thickness of the gate insulator. W is the width of the gate and the L is the length of the gate channel.

Logical effort

Sutherland introduces a method called *Logical Effort* in (SUTHERLAND; SPROULL; HARRIS, 1999). The model describes delays caused by the capacitive load C_L that the logic gate drives and by the topology of the logic gate. When the load increases, the delay also increases, but the delay also depends on the logic function of the gate. In this method, the logical effort g normalizes the output capability of a gate to match a minimum sized inverter for static CMOS. Thus, the value to g of an inverter is equal to 1.

Logical effort g of other gates represents how much more input capacitance a gate must present to produce the same output current as the inverter. Figure 3.9 shows the logical effort g to an inverter (figure 3.9(a)), a NAND2 (figure 3.9(b)) and a NOR2 (figure 3.9(c)). Table 3.2 presents the logical effort to common gates assuming P/N size ratio equals 2.



(a) An inverter (g=1) (b) A NAND (g=4/3) (c) A NOR (g=5/3)

Figure 3.9: Logical effort g to INV, NAND and NOR CMOS gates

Table 3.2: Logical effort of common gates

Gate Type	Number of inputs					
	1	2	3	4	5	n
Inverter	1					
NAND		4/3	5/3	6/3	7/3	(n+2)/3
NOR		5/3	7/3	9/3	11/3	(2n +1)/3
MUX		2	2	2	2	2
XOR		4	12	32		

The delay model is normalized to dimensionless units to isolate effects of fabrication process. Thus, the delay of a gate is given by

$$d_{abs} = d * \tau \tag{3.13}$$

where τ is the delay of a minimum inverter driving another minimum inverter with no parasitics. Furthermore, the delay of a gate d is composed by two components:

- a fixed part called *parasitic delay* p (no load delay). This parasitic delay is caused by the internal capacitance of the gate. It is constant and independent of the transistor size because when the transistor size is increased, the capacitances of the gate/source/drain are also increased keeping the delay constant.

- a part proportional at the load on the output called the *effort delay* or *stage effort* f.

The total delay is measured in units of τ, and is composed of the sum of these delays.

$$d = f + p \tag{3.14}$$

The effort delay f can be expressed as a product of two terms:

$$f = g + h \tag{3.15}$$

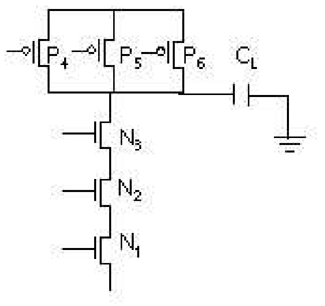
where g is the logical effort and h is electrical effort which captures properties of load and the effects of transistor sizes on loading. The electrical effort h is given by

$$h = C_{out}/C_{in} \tag{3.16}$$

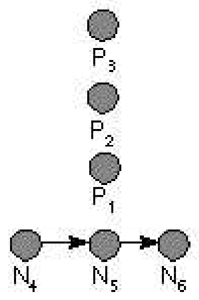
where C_{out} is the capacitance that loads the output and C_{in} is the capacitance presented at the input of the gate. For a gate, the electrical effort f is reduced when the transistor becomes wider since C_{in} increases.

Direct acyclic graph model

In (SUNDARARAJAN; SAPATNEKAR; PARHI, 2002) it is proposed an approach which each CMOS gate is converted into an equivalent directed acyclic graph (DAG) model. In the DAG structure, each vertex is a transistor with an associated delay attribute. This delay attribute is given as the worst case discharging (or charging) path delay through the transistor.



(a) A 3-input static CMOS NAND gate



(b) DAG corresponding to a NAND gate

$$Delay(N_3) = \frac{A}{x_3}\left(B^p_{x4} + B^p_{x5} + B^p_{x6} + C_L + E\right)$$

(c) Delay for the vertex N_3

Figure 3.10: Direct Acyclic Graph (DAG) for CMOS gates

In figure 3.10 is shown a DAG structure. In figure 3.10(a) a static CMOS NAND gate is shown and its DAG structure is shown in 3.10(b). In 3.10(c), a delay formulation for the vertex N_3.

Considering the Elmore delay, the delay of a 3-input NAND CMOS gate is given by

$$\begin{aligned} Delay^{pulldown} = \; & \left(\frac{A}{x_1}\right)(B_{x_1} + C_{x_2}) + \left(\frac{A}{x_1} + \frac{A}{x_2}\right) \\ & .(B_{x_2} + C_{x_3} + D)\left(\frac{A}{x_1} + \frac{A}{x_2} + \frac{A}{x_3}\right) \\ & .(B_{x_3} + B^p_{x_4} + B^p_{x_5} + B^p x_6 + C_L + E) \end{aligned} \tag{3.17}$$

where x_i is the size of the transistor N_i (P_i) in the pulldown (pullup) network. A, B and C are the constant resistance, drain and source capacitance. D and E are related to the wire capacitance. B^p is the drain capacitance of a unit sized PMOS transistor and C_L is the load capacitance.

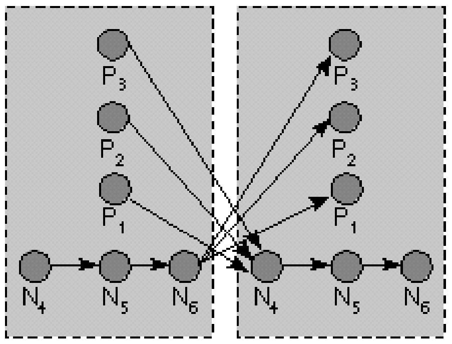


Figure 3.11: The DAG corresponding to a circuit of two 3-input static CMOS NAND gates in series

The entire circuit consisting on static CMOS gates can be represented as an equivalent DAG. In figure 3.11 is illustrated the construction of a circuit DAG of two 3-input CMOS NAND gates.

Delay estimation of Complex MOS gates

Complex gates are used frequently in the layout generation in order to reduce the number of transistors of the circuits. The mapping to complex gates is usually done at the logic level.

In (KONG; HUSSAIN; OVERHAUSER, 1997), a two step technique is proposed to estimate the performance of complex gates. A complex gate is first mapped to an equivalent NAND gate and them the NAND gate is mapped to an inverter macro model.

The algorithm for generalized reduction techniques for complex gates is defined by

1. Estimate the conducting path from the output node to supply source;
2. Find the maximum number of series transistors and the corresponding NAND gate;
3. Find the scaling factor;
4. Estimate the internal node capacitances;
5. Find v_a (figure 3.12);

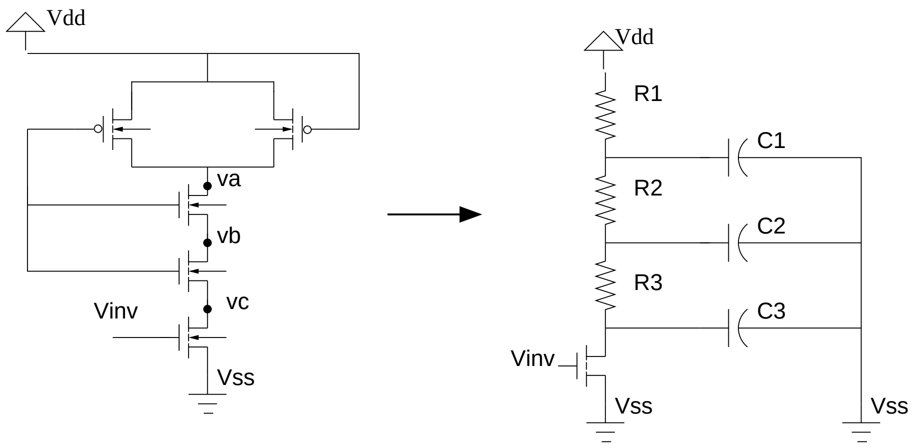


Figure 3.12: Estimating τ

6. Estimate initial β_{eq} and λ_{eq} using the scaling factor and v_a
7. Find the effective resistances of transistors at $v_{in} = v_{inv}$. Estimate the delay τ using internal node capacitances and effective resistances;
8. Find initial C_L;
9. Apply the same simulation scheme for the corresponding NAND gate.

The scaling factor is determined from the concept of equivalence of the conducting transistor, τ and v_a are found in dc analysis. β_{eq} of a complex gate is modeled as a function of the scaling factor:

$$\beta_{eq}(AOI) = \frac{N}{N_{scale}} \beta_{eq}(NAND). \tag{3.18}$$

and λ_{eq} is given by

$$\lambda_{eq} = \frac{V_{DD} - v_a}{V_{DD}} \lambda, \tag{3.19a}$$

and

$$\lambda_{eq}(AOI) = \frac{N}{N_{scale}} \lambda_{eq}(NAND). \tag{3.19b}$$

where v_a depends on the gate type.

Table lookup approach

The most accurate delay estimation is a simulation using differential equation (*e.g.*, with SPICE). However it is too much CPU expensive (COUDERT, 1997). In (COUDERT, 1996), is reported that a table lookup approach is more accurate than multi-coefficient (linear, polynomial or posynomial) approximation. The table lookup based non-linear interpolation delay model is about 3% of SPICE, while 2-or 3-coefficient linear delay models present values as much as 20% from SPICE.

Unfortunately, this table lookup is a technique used in the industry and characteristics about its implementation is not reported. So, it is impossible to use this technique in this work.

3.4.2 Gate Sizing

Gate sizing is a technique to optimize each individual gate of the circuit, given certain characteristics. Characteristics refer to widths of the transistors, relations between gates and their implication in the timing. In addition, the optimization process may be directed to minimal area, minimal power, minimal delay (maximal performance) of the circuit or any combination among them.

Area, power and delay are conflicting objectives and the search for an acceptable solution is the key point in any sizing technique. Sizing requires a careful balancing of these conflicting characteristics, and an optimal solution will require the coordination of the correct sizes of the gates along *all* critical paths (COUDERT, 2002).

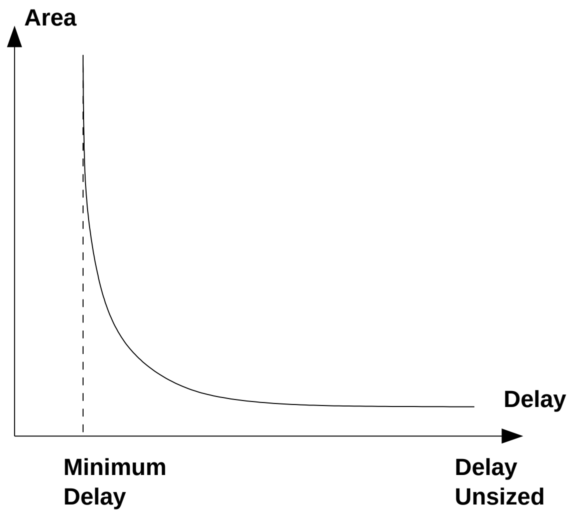


Figure 3.13: Area/delay curve for transistor sizing

Figure 3.13 shown a well-known curve that illustrates the relation between area and delay (JIANG et al., 1998a). It shows the nature of the area/delay tradeoff curve for transistor sizing. Usually, a small amount of sizing is adequate to reduce the delay. In any case, it is impossible to reduce the delay of a circuit indefinitely through sizing.

3.4.2.1 Discrete versus continuous sizing

Gate sizing methods can be classified in two main categories:

- discrete;
- continuous.

In discrete sizing methods, only a set of discrete sizes is allowed. On the other hand, in continuous sizing methods the gate sizes are continuous variables (CHEN; HSIEH; PEDRAM, 2000a), hence leading to a larger space of solutions.

A method to gate sizing for row-based layouts is presented in (MAHESHWARI; SAPATNEKAR, 1995). In this work, the transistors are sized continuously creating empty spaces between adjacent rows and the height of a row is determined by the height of the tallest gate in the row.

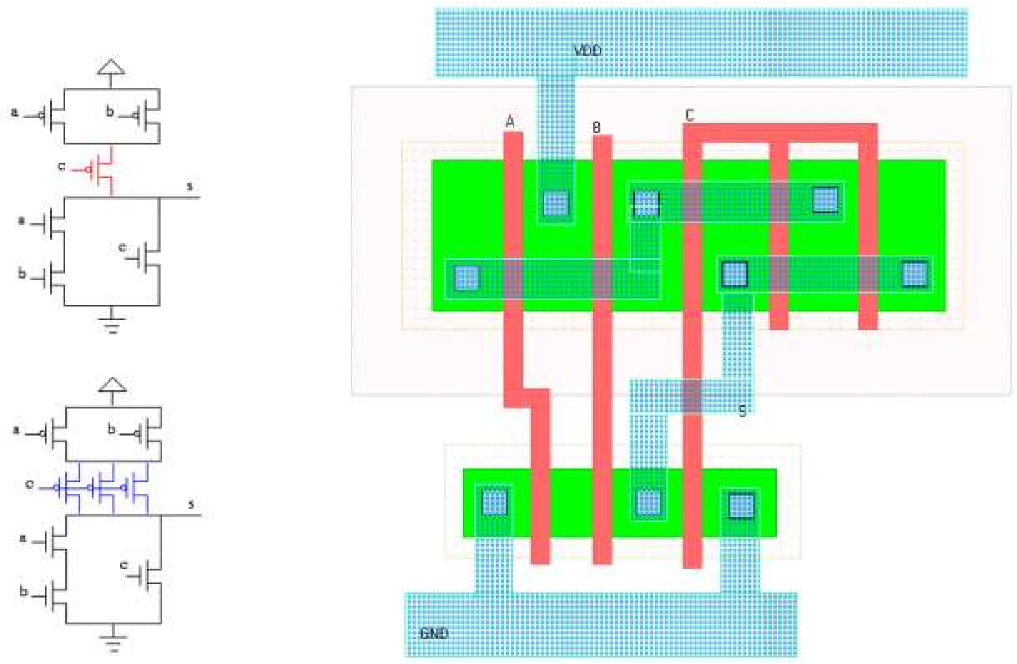


Figure 3.14: Example of transistor folding

In a row-based layout, different transistors width results in an area occupation problem. To avoid that different gate sizes increase the total height of the row, a possible solution is the transistor folding technique. Thus, all the cells in a row have the same height and the channel routing area between rows is uniform.

The objective of the transistor folding technique is to divide a single transistor in other transistors connected in parallel. Figure 3.14 shows an example of transistor folding application. In this example a transistor is folded in 3 parts, maintaining the diffusion width.

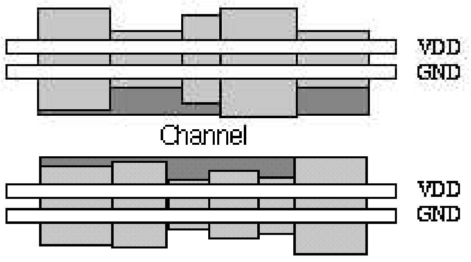
Despite the continuous sizing method presents a larger space of solutions, in row-based style, the spacing between rows can increase the area. Figure 3.15 shows an example of the transistor folding application. In figure 3.15(a), a row-based layout where the folding is not used. In figure 3.15(b), the folding technique is being used.

3.4.3 Buffer Insertion

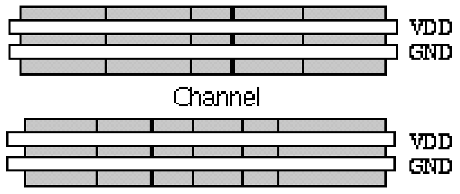
In the buffer insertion process, buffers are inserted in a tree structure in order to distribute the capacitance of the original gate over the buffers of the circuit. The main example of this technique is the insertion of buffers in clock wires, as shown in the section 3.3.2.

Coudert in (COUDERT, 2002) divides the buffering function in three:

- Long wires result in signal attenuation. Buffers are used to restore signal levels.



(a) Gate sizing without folding

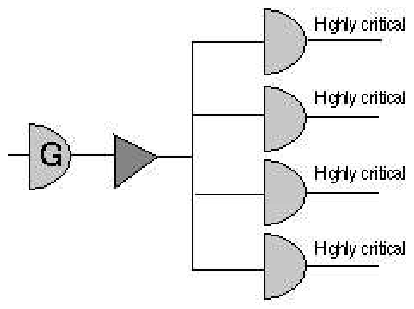


(b) Gate sizing with folding

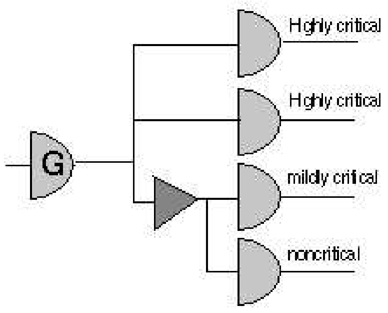
Figure 3.15: Row-based style gate sizing

- In order to increase the drive strength for a gate that is driving a large load, one buffer or a chain of buffers can be used.
- Buffers can be used to shield a critical path from a high-load off-critical path.

In the gate sizing technique, some gates may be sized excessively because of the large loads that they drive. The appropriate insertion of buffers in a circuit can be used to prevent excessive sizing (JIANG et al., 1998a). They relate also that the buffer insertion must be used only for highly critical paths. The gate sizing is more advantageous than buffer insertion for mildly or noncritical paths.



(a) Type A



(b) Type B

Figure 3.16: Buffer insertion strategies

Cases which a mildly critical path is being specified, using a small amount of sizing, good results may be aimed. When a buffer is inserted in the path, it implies that the area of the circuit increases corresponding to four transistors.

In (JIANG et al., 1998a,b), two situations where the buffering presents advantages are reported. It refers to type A and B buffer insertion scenarios. The buffer insertion types are shown in the figure 3.16.

- *Type A*: If a gate whose outputs are all highly critical drives a large capacitive fan-out, buffer insertion can help reducing the delay. By choosing an appropriate size of the buffer, the sum of the delays of the buffer and the gate may be smaller than with the unbuffered gate (figure 3.16(a)).

- *Type B*: If a gate has both highly critical, mildly critical and noncritical outputs, one may isolate the capacitance of the noncritical path by inserting a buffer (figure 3.16(b)).

Quantifying the criticality in a situation and to use them to determine locations for buffer insertion are the challenge.

3.4.3.1 Buffer insertion formulation

The buffer insertion process must be able to reduce the delay resulting in minimum increase of area. In (SATO et al., 1996), the optimization process is presented in four steps as follows.

1. Selects a wire on the worst critical path;

2. Selects a segment of the wire selected;

3. Selects the type of inserted buffer. A candidate buffer type is selected and the improvement of the delay is checked. If the buffer type causes fanout violation, it is removed from the candidates;

4. Search the best location of the segment to insert buffers.

A challenge in the construction of optimization algorithms is to find the best place to make the buffer insertion. By depending on the distance between the gates where the buffer is inserted, the location to the buffer influences on the delay of the segment of the circuit.

The location α of the inserted buffer is also reported in (SATO et al., 1996). In this work, it is presented two methods to calculate the location to the buffer insertion.

- *Method 1* : Buffers are inserted at constant intervals on the segment. For example, if the segment is partitioned in four, the locations α = 0,0.25,0.75,1 are checked one by one.

- *Method 2* : Buffer location α is chosen following the Type A and Type B situation. In this work, the type situations are called partition and isolation, respectively.

3.4.4 Cloning

In order to increase the drive strength driving the net, the cloning method may be used. Cloning is the process of duplicating identical gates and placing the gates in parallel. Theoretically, this is the same as duplicating the width of each transistor.

Jacobs cites in (JACOBS, 2001) some behavior differences between cloning and gate sizing:

- there is additional wiring between the original gate and its clone;
- clone gate and original gate may behave different because of physical margins of the fabrication process;
- clone gate may behave different from the original one because of the influences of the wires around the gates.

A note on implemented strategies is reported in (JIANG et al., 1998a) concerning the gate cloning technique. The gate cloning is treated as an alternative to the *Type B* buffer insertion. The idea is the use of cloned versions of the gate instead of create new buffers. The figure 3.17 illustrates the gate cloning and buffer insertion.

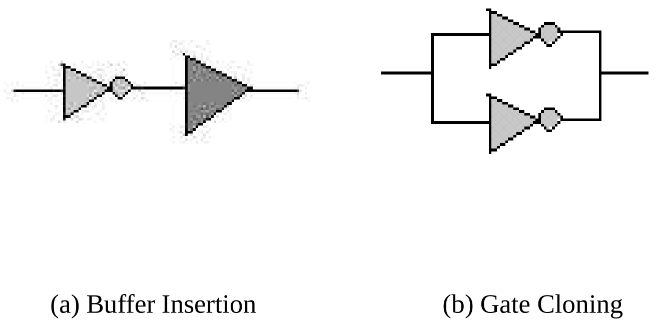

(a) Buffer Insertion (b) Gate Cloning

Figure 3.17: Gate cloning as alternative to buffer insertion

Gate cloning has two advantages to the buffer insertion:

1. if a buffer is inserted at the output of an inverter. It increases the transistor amounts to an expense of four transistors, while the gate cloning increases the circuit by only two transistors.
2. buffer insertion increases the number of levels in the circuit, causing unnecessary delay increases.

The results of their (JIANG et al., 1998a) implementations show that the situations related above do not occur sufficiently often to see the cloning advantages. On any of the ISCAS85 benchmark circuits, it is never obtained better results by using the gate cloning technique.

3.4.5 Related Works

Here are discussed some works and experiences concerning the layout optimization. Works are related to a specific kind of optimization, such as area or delay, and other for general purposes. Some of them are developed only to gate sizing, but other experiences treat the optimization using both techniques to more accurate results.

3.4.5.1 Gate Sizing and Path Sensitization

In the circuit model where output are latched and input vectors are successively applied at inputs, gate sizing used to reduce the delay of the critical path may not improve the performance. In (KONG; HUSSAIN; OVERHAUSER, 1998), a typical gate sizing example is given which algorithms are presented to optimize the performance using a gate selection method.

In this work, the path sensitization is used to detect the gate to be sized. The clock period is determined by the delay of both long and short paths where gates in the sensitizable long and short path can be selected for resizing.

In this method, most of the circuit (ISCAS'95) have their area increased in less than 10% of the examples, but with 30% gain in timing performance.

3.4.5.2 Gate Sizing and Placement

In (CHEN; HSIEH; PEDRAM, 2000b), an optimization technique is presented for improving delay optimization in integrated circuits. The technique consists on applying timing analysis to identify the set of k most critical paths in the circuit followed by resizing and placement along the critical path. The technique is repeated until no further reduction in circuit delay is possible.

A mathematical formulation for simultaneous sizing and placement subject to timing and position constraints is used to solve the problem. The algorithm has been applied to ISCAS benchmarks and the average improvement is about 15%.

3.4.5.3 A Convex Optimization Problem

The approaches to the transistor sizing problem suggests the fact that the problem can be mapped in a convex optimization problem; hence any local minimum is a global minimum. In (SAPATNEKAR; RAO; VAIDYA, 1991), it is presented a solution to the transistor sizing problem. This is done by modeling the delay and area as posynominal functions of transistor sizes.

Posynomials are products of the variables of the geometric programming formulation to any real number exponent. In this case, a simple variable transformation is used to transform posynomials to convex functions.

To finish, a convex programming algorithm is used to solve the resulting optimization problem.

3.4.5.4 A Statistical Delay Model

In (JACOBS; BERKELAR, 2000) is presented a work about gate sizing using a statistical delay model. The formulation used allows many different forms of objective functions. The gate sizing problem is formulated as a non-linear programming problem.

Gate sizing is usually based upon a static delay model, which assumes that a gate always have the same delay. Statistical delay analysis is basically a static delay analysis where each delay-inducing element has an associated delay probability function.

This expresses the fact that the delay in logic circuits is basically dynamic. It depends on the state and local temperature of the gate, or cross talk effects in wires.

3.4.5.5 Transistor Sizing Based on Iterative Relaxation

In (SUNDARARAJAN; SAPATNEKAR; PARHI, 2002), a fast technique to gate sizing is presented. This technique uses an iterative relaxation approach that involves a two-step optimization strategy:

1. A *D-phase* where transistor sizes are assumed fixed and transistor delays are regarded as variable parameters.

2. A *W-phase* where transistor/gate delays are assumed fixed and their sizes are regarded as variable parameters.

The primary features of the proposed approach are as follows.

- It is computationally fast
- It can be used for true transistor sizing as well as the relaxed problem of gate sizing.
- It uses the Elmore delay model, but more general delay models may be used

The objective function for the problem is the minimization of the circuit area. In the W-phase, this objective is addressed directly, and in the D-phase the objective is chosen to facilitate a move in the solution space in a direction that is know to lead to a reduction in the circuit area.

The transistor size optimization problem can be stated as

$$\begin{array}{ll} minimize & \sum_{all\ trans,i} x_i \\ subject\ to & Delay(Circuit) < T \\ & minsize \leq x_i \leq maxsize \end{array} \quad (3.20)$$

where x_i refers to the size of transistor i, T is timing requeriment specified as an input of the optimization, and $minsize$, $maxsize$ are minimum and maximum bounds of the sizes of transistors.

The technique for delay estimation is the DAG model as shown in the section 3.4.1.2. In this work, the Elmore delay model was implemented in a graph structure to whole circuit.

In addition to gate sizing, the wire sizing also can be implemented in the DAG model. This can be performed by modeling the problem adding vertices corresponding to each wire.

3.4.5.6 Combined Gate Sizing and Buffer Insertion

Timing constraints in modern VLSI circuits are becoming increasingly tighter and combined gate sizing and buffer insertion are widely used to meet these constraints (CHEN; HSIEH; PEDRAM, 2000a). Gate sizing reduces the circuit delay by adjusting the gate sizes and their drive strengths and input capacitances. On the other hand, buffer insertion (fan-out optimization) achieves circuit delay reduction by speeding up the timing critical signals.

Many works report the development of algorithms to circuit optimization by using the gate sizing and buffer insertion. In (JIANG et al., 1998b) are presented strategies to insert buffers in circuits combined with gate sizing to achieve better power-delay and area-delay tradeoffs. A method for minimizing the delay of power constrained combinational logic networks is presented in (LOWE; GULAK, 1998) that can incorporate a mixture of unbuffered and buffered gates (or mixture of CMOS and BiCMOS gates).

Traditionally, gate sizing and buffer insertion are treated separately and at different stages of the design process. In (JIANG et al., 1998a), the buffer insertion and the transistor sizing is done into a single optimization. This technique allows the buffer insertion being used only is specified cases, and the gate sizing used with better results. In this work, the results are average 40% better compared with the sizing method.

Simultaneous gate sizing and buffer insertion are also presented in (CHEN; HSIEH; PEDRAM, 2000a). The problem is expressed in mathematical form and the delay model reflects the effect of the sizing and the insertion of buffers. The average delay improvement obtained is about 9%.

3.5 A Gate Sizing Method Applied to an Automatic Custom Layout Generation Tool

In case of a row-based layout generator that uses and approach derived from the linear matrix approach, transistor sizing may provoke less impact in the layout topology, mainly when a discrete sizing is adopted. In this case, each transistor that must be enlarged has a new width that is an integer multiple of the initial size of the transistors in the layout.

Most of sizing algorithms does not report the impact of the sized transistors in the layout of the circuits as shown in (SAPATNEKAR; RAO; VAIDYA, 1991; KIM; DU, 1998; SUNDARARAJAN; SAPATNEKAR; PARHI, 2002). In (SANTOS et al., 2003), it is proposed a method for performing delay optimization of combinational blocks that are generated by a row-based layout generator. Given a maximum delay imposed to the circuit, the method begins by identifying a set of critical paths that violate such delay. This is performed by a timing analysis tool that is able to guarantee that the longest sensitizable paths belong to this set (WILKE et al., 2002). Then, a gate belonging to these paths is selected to be sized. Transistors of the selected gate are sized in a discrete manner, by using the transistor folding technique. The area and delay of the sized circuit is estimated and, if necessary, a new gate is selected to be sized. This procedure is repeated until the delay constraint is satisfied or there is no gate to be resized.

Folding technique can be used in row-based layout optimization to reduce effects of the sizing in the layout. The folding technique consists on dividing the transistors in discrete sizes in order to maintain the height of the cells. In spite of the increase in the length of the row, the folding technique can eliminate spaces left in the channel between rows.

3.5.1 The Gate Sizing Method

The algorithm starts with the critical delay estimation of the circuit followed by searching of longest paths. Once longest paths are identified, gates in the critical path are selected for sizing. Only gates in the sensitizable critical path of the circuit are sized. Criteria for gate selection are:

- gates which appear with more frequency in longest paths are candidates for selection;
- gates with great loading factor LF.

The loading factor (CRÉMOUX; AZÉMARD; AUVERGNE, 1998) of the gate is proportional to the delay of the gate and it is given by:

$$LF = \frac{C_L}{C_{IN}}. \tag{3.21}$$

where C_L is the load capacitance constituted by the active, diffusion and routing capacitance and C_{IN} represents the input capacitance.

New gate delays are calculated and the process is repeated until area/delay constraints are satisfied or there are no gates to be sized.

3.5.2 Obtained Results

The sizing algorithm was tested in purely combinational circuits. Table 3.3 shows the obtained results, where T is the number of transistors and D_u is the delay of the gate with unsized transistors. The values in the table present the area increase and the topological and functional delay reduction. It is important to note that topological delay may be a pessimistic estimation for the critical delay because it does not take into account the logic behavior of the circuit.

Table 3.3: Experimental results

Bench	Critical Path Oriented Sizing		
	Area(%)	T. Delay(%)	F. Delay(%)
bw	1.34	12.26	12.26
T=894	4.70	20.35	20.35
D_u=879ps	13.42	30.61	30.61
	42.06	40.15	40.15
9sym	0.88	5.34	5.34
T=1140	4.21	10.69	10.69
D_u=1313ps	8.77	20.18	20.18
	27.37	30.17	30.17
alu2	2.58	10.00	8.61
T=1203	7.65	15.86	16.89
D_u=2629ps	11.47	20.01	11.47
	34.86	30.58	35.19
t481	0.18	5.13	5.13
T=6898	0.58	10.09	10.09
D_u=1675ps	1.55	15.04	15.04
	3.19	20.00	20.00

Results show that cases where the topological delay is not the critical delay of the circuit, the algorithm presents effective reduction of the critical delay.

Figure 3.18 shows the area increase in the circuit 9sym. The curve illustrates the area increase in circuits when delay is demanded. In this case, delay constraints upper 30% have little effect in the timing optimization, but the area increases significantly. It seems clear that timing optimization requirement cannot be guaranteed for any delay constraints. This also happen because of the limited value of the loading factor. The loading factor is always greater than one in order to avoid large transistors.

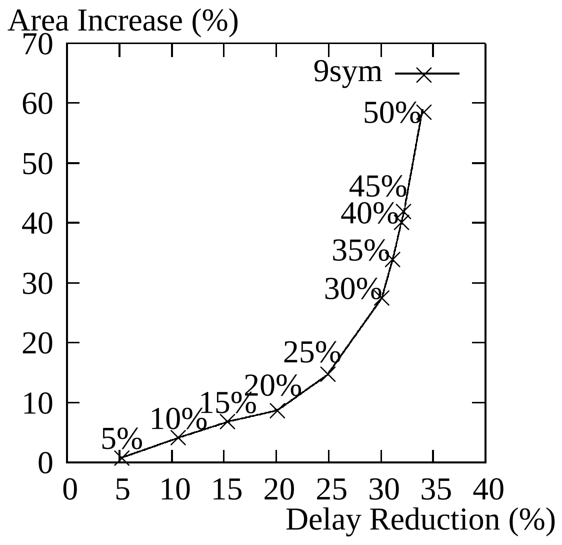


Figure 3.18: Area increase under timing constraints

3.6 Conclusion

Cost models are essential on the optimization of circuits. Efficient area and delay modeling result in more accurate techniques to layout optimization. Section 3.4.1 shows the difficulty to estimate the area occupation in automatic layout optimization, while some known delay models are presented.

Three techniques for optimizing layouts are reported:

- Gate sizing
- Buffer insertion
- Gate cloning

The optimization can be done through buffer insertion, cloning the gate or increasing the width of the transistors of the gate. The use of gate cloning techniques is limited to few works because of its disadvantages. Characteristics and formulation are discussed to both gate sizing and buffer insertion and shows that the use of these techniques is necessary in the layout development.

Related works concerning gate sizing and buffer insertion show that the best results are obtained by utilization of the both techniques in association.

In section 3.5, a row-based gate sizing method is reported. Custom layout constraints were applied to the sizing algorithm and it shows that it is able to optimize delay in the automatic custom layout generation.

4 MAIN ASPECTS ON LAYOUT SYNTHESIS

4.1 Introduction

In this chapter it is reported some aspects on the automatic layout generation. These aspects are very important due to the effort of the designers to generate circuits with the best possible compromise between area, delay and power according to project specifications.

In section 4.2 it is reported some basic aspects about full automatic layout generation. The effects of the power supply distribution and the diffusion area optimization are discussed. These characteristics are very important because of the influence on the occupied area and the circuit delay. Placement and routing characteristics are also reported in section 4.3 and 4.4.

Some experiences on automatic layout generation are presented in 4.5. These experiences are base to the development of a tool where different strategies were developed. Consequences of layouts developed as result of these experiences in the area occupation and circuit delay are reported.

4.2 Full Automatic Layout Generation

In (LEFEBVRE; MARPLE; SECHEN, 1997), it is highlighted that the realization of any custom circuit, whether it is a standard cell or a custom block, requires three primary implementation phases:

- Creation of a transistor circuit topology that provides a specific digital function. Topology is related to P and N type transistors and their connectivity.
- Sizing and ordering the transistors in the circuit topology. Sizing means altering the MOS transistor's width and ordering means swapping positions of MOS transistors connecting in series chains or swapping logical pins.
- Placing, routing and compacting the transistors into the layout.

4.2.1 Transistor Placement

Transistor placement is the problem of determining the best possible position of each transistor within the layout. In function of the placement of each transistor into the layout, diffusion areas may be reduced and connections between transistors (polysilicon or metal lines) may easily be performed.

Lefebvre reports in (LEFEBVRE; MARPLE; SECHEN, 1997) some operations that are related to transistor placement methods:

- Transistor pairing: $p-$ and $n-$transistors are paired if they have common gate and/or source/drain signal names.
- Group/chain formation: groups or chains formed of series connected transistors result more compact layouts.
- Transistor folding: large transistors may need to be decomposed into a set of smaller ones connected in parallel.

Transistor pairing may be performed by searching the *Euler path*. Once linear matrix topology is used, the search of the Euler path may reduce the connection complexity between transistors with similar gate signal.

Euler path search consists on finding a path, starting from a transistor source (drain) node and arriving to another node crossing only one time in each transistor of the logic gate. In CMOS circuits, the same path in PMOS and NMOS planes are searched in order to easily connect transistors with common signals with the polysilicon layer.

Transistor folding consists in increasing the width of a transistor discretely in order to maintain the height of the diffusion strips and consequently the height of a row. Techniques to gate sizing applied to row-based layouts are examples in which folding is widely used. Section 3.4.2.1 discusses the folding application in layouts.

4.2.2 Effects of Diffusion Area Optimization on the Delay of Circuits

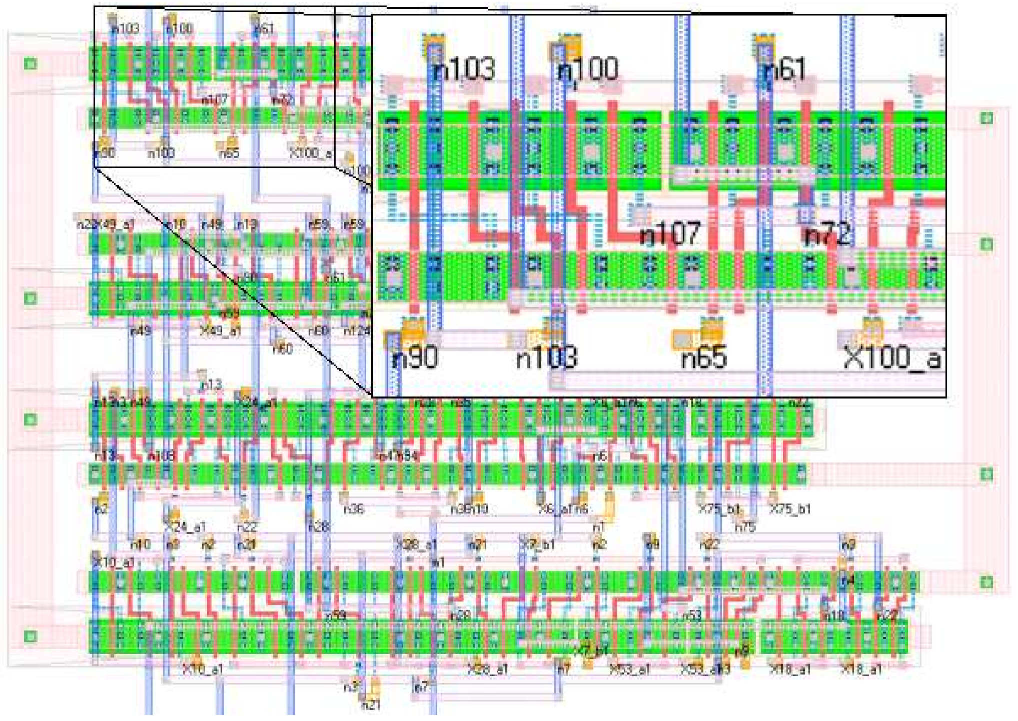
It is known that the size of the transistors drain and source are directly related to the delay of circuits. In addition, the switching speed of MOS gates are strongly dependent on the parasitic capacitances associated with the MOS device and interconnection capacitances (WESTE; ESRAGHIAN, 1993). The diffusion capacitance is one source of this parasitic capacitance and this value is related to the perimeter and the area of the diffusion.

Figure 4.1 shows two examples of layout implementation. In 4.1(a), the layout is generated in such a way drain and source areas are reduced while in 4.1(b) the preoccupation for reducing the diffusion areas are not taking into account. The same circuit is presented in both figures and the layouts present exactly the same placement and routing. Parasitics capacitances and resistances were extracted using the Cadence Diva^{TM} and some electric simulations were done using Cadence Spectre^{TM}.

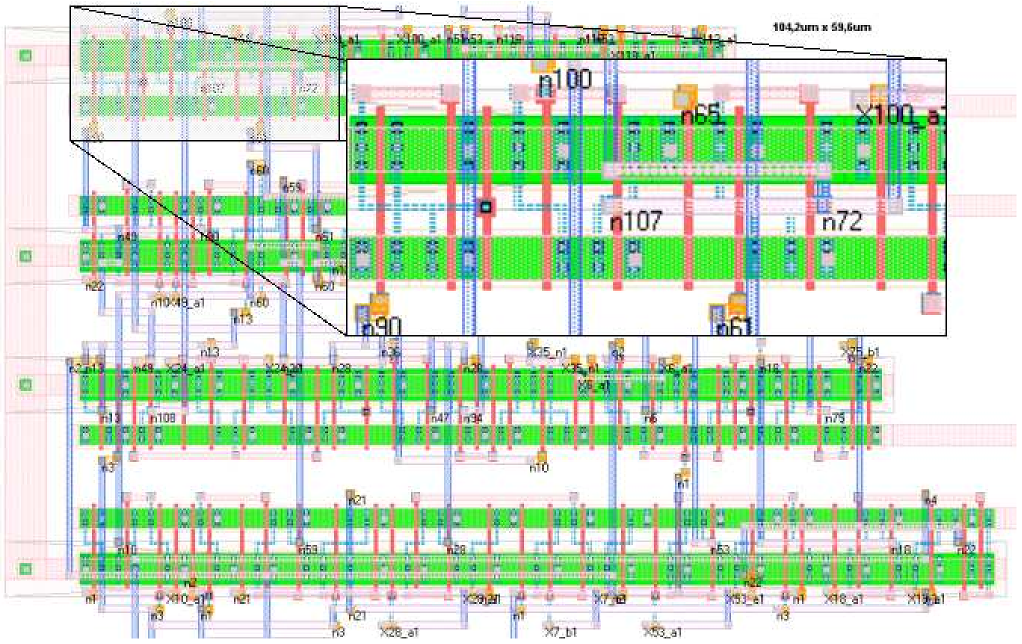
Results show a gain of around 20% related to the delay of the circuits. This emphasizes the need to develop layouts with smaller as possible source and drain areas.

4.2.3 Power Supply Lines Distribution

In CMOS gates supply lines are responsible to guarantee strong ’0’s and ’1’s in its output. The “strength” of a signal is measured by its ability to sink or source current (WESTE; ESRAGHIAN, 1993). It is assumed that the symbol ’0’ is related to the low voltage (normally set to zero volts) and the value appears in the output of the gate when there is a path between the GND and the output. On the contrary, the symbol ’1’ is related to the high voltage (set to 3.3 volts in a standard 0.35μm technology) and this value appears in the output of a cell when there is a path between the V_{DD} and the output.



(a) Drain and source areas reduced by using dog leg algorithms in polysilicon routing



(b) Short polysilicon connections but big transistors drain/sources areas

Figure 4.1: Effects of Diffusion Area Optimization

In row-based layouts, supply lines must be disposed in horizontal lines in order to distribute V_{DD} and GND signals to whole circuit. Circuits where the *linear matrix* style is used, the supply lines may be basically as following:

- Supply Lines between diffusion strips;
- Supply Lines over diffusion strips (over p and n transistors);
- Supply Lines between rows.

These three types of distributing the supply signals are explained below.

4.2.3.1 Supply lines between diffusion strips

Linear matrix-based layouts with supply lines between PMOS and NMOS diffusion have as main characteristic an easy placement of body ties in the layout. Due the needed of contacts to N type wells and the P substrate, these supply lines disposition style place the body ties directly connected with the V_{DD} and GND inside the row.

The TROPIC (Transparent Reconfigurable Optimized Parametrizable Integrated Circuit) (MORAES, 1994) tool is an example of automatic custom layout generator which supply lines are disposed between the PMOS and NMOS diffusion. Figure 4.2 shows an example of generated layout with this supply style.

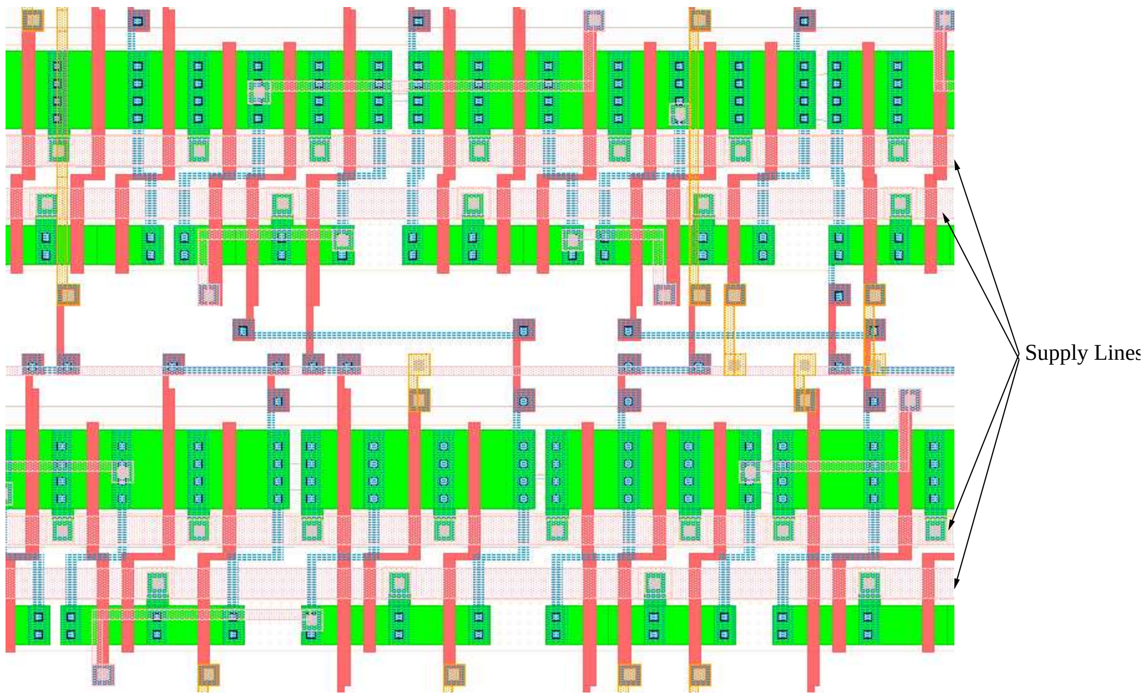


Figure 4.2: Supply lines between PMOS and NMOS diffusion

When supply lines are performed between diffusion strips, the second metal layer must be used because connections between P and N diffusion strips are done using the first metal layer.

4.2.3.2 *Supply lines over diffusion*

In this style, the supply lines are disposed over the diffusion. Thus, the V_{DD} signal is easily connected to the PMOS diffusion and the GND is also easily connected to the NMOS diffusion. Figure 4.3 shows an example of this style of supply disposition.

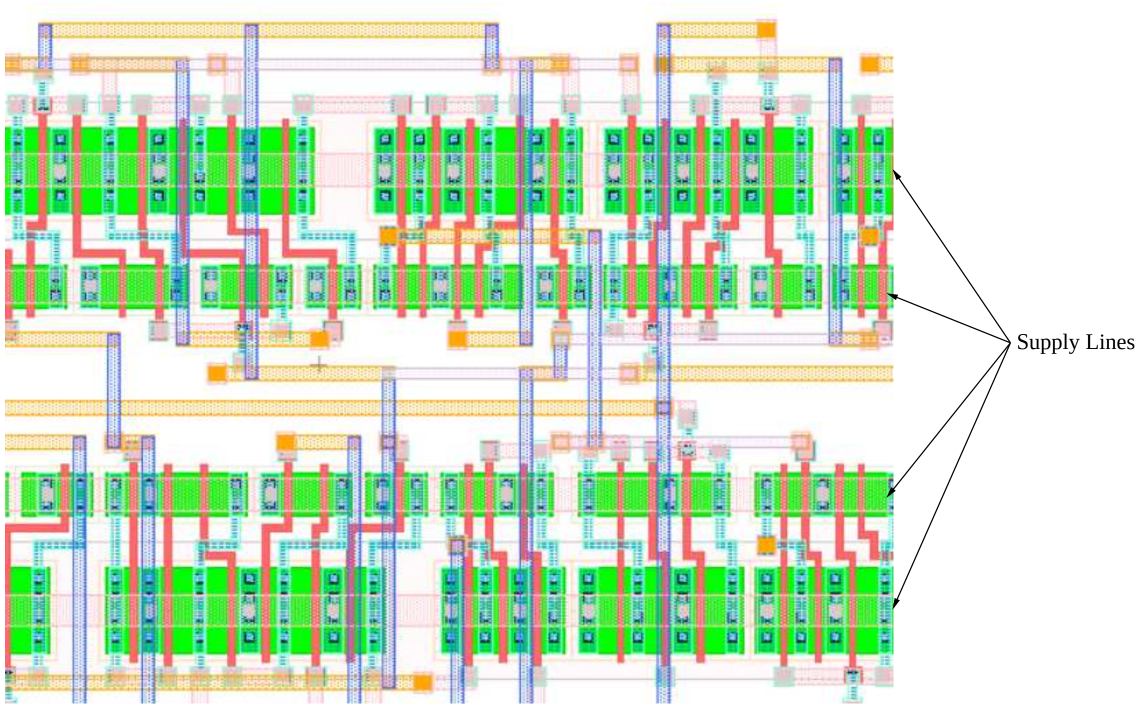


Figure 4.3: Supply lines on PMOS and NMOS diffusion

The body tie insertion must be implemented in the diffusion gaps. At the moment when the layout is generated, the insertion of body ties is done increasing the spacing between diffusions.

As the style shown above, the second metal layer must be used to distribute the supply lines.

4.2.3.3 *Supply lines between rows*

This supply disposition style is widely used by layout developers and automatic layout generators. The V_{DD} and GND distribution is done horizontally to the row as the previous styles. Figure 4.4 shows this supply lines disposition.

The supply lines distribution is done between rows using the first metal layer of the technology and it is propitious to the layout implementation mainly for two reasons:

1. Rows Mirroring: If supply lines are implemented between rows, adjacent rows can be placed mirrored in order to share the supply lines and to reduce the circuit area.

2. Metal Layers: When the first metal layer is used for the power supply distribution, superior layers can be used to connect nets (full over-the-cell style).

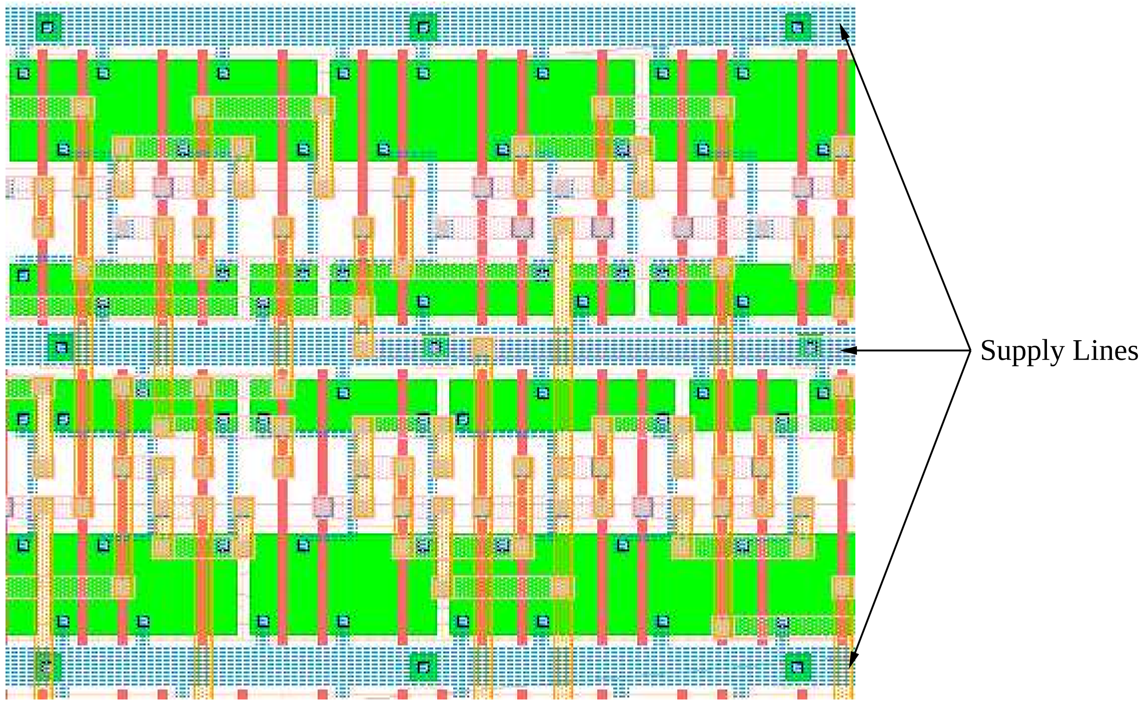


Figure 4.4: Supply lines between rows

4.3 The Placement Problem

Logic Gates placement consists on finding the position of each logic cell within the circuit surface. The main goal of the placement algorithms is to guarantee the routability. Thus, the placement algorithm tries to place together interconnected logic gates reducing the wire length (HENTSCHKE, 2002).

Hentschke also emphasizes the congestion as an important aspect for the routability of the circuit. The placement algorithm must be able to avoid congestion areas in order to increase the efficiency of routers and to guarantee the routability. It is assumed that congestion areas are parts of the circuit that there are demands for connections greater than the area available for the routing.

An example of placement algorithm is the quadrature. In (MORAES; REIS; LIMA, 1997; MORAES; VELASCO, 2000), it is reported the quadrature placement algorithm in which the circuit is divided into horizontal and vertical directions generating quadrants. These quadrants are processed line-by-line in the case of horizontal partition and column-by-column when vertical partition is performed.

The quadrature placement presented in this work is efficient because of the pin propagation. Thus, logic cells with common signals are placed in adjacent quadrants to reduce the wire length.

Mango Parrot is the placement tool presented by Hentschke. This tool uses many algorithms and parameters and it offers these facilities to the users. Thus, the algorithms are very flexible and the users are able to explore the functionality of the features available in the tool. Some examples are:

- Number of iterations;

- Temperature (Used in Simulated Annealing);
- wire length estimate.

The placement in the Mango Parrot starts with a constructive placement that may be random, Plic-Plac or Cluster Growth. After the placement is refined by an interactive algorithm such as Simulated Annealing, Force Direct Placement and Greedy. It is possible to implement a final refinement for congestion treatment.

4.4 The Routing Problem

The routing problem consists of interconnecting points of the circuit that have been assigned positions as a solution of the placement. The specification of a routing problem consists on the position of the terminals, the netlist that indicates which terminals should be interconnected and the area available for routing in each layer (GEREZ, 1998).

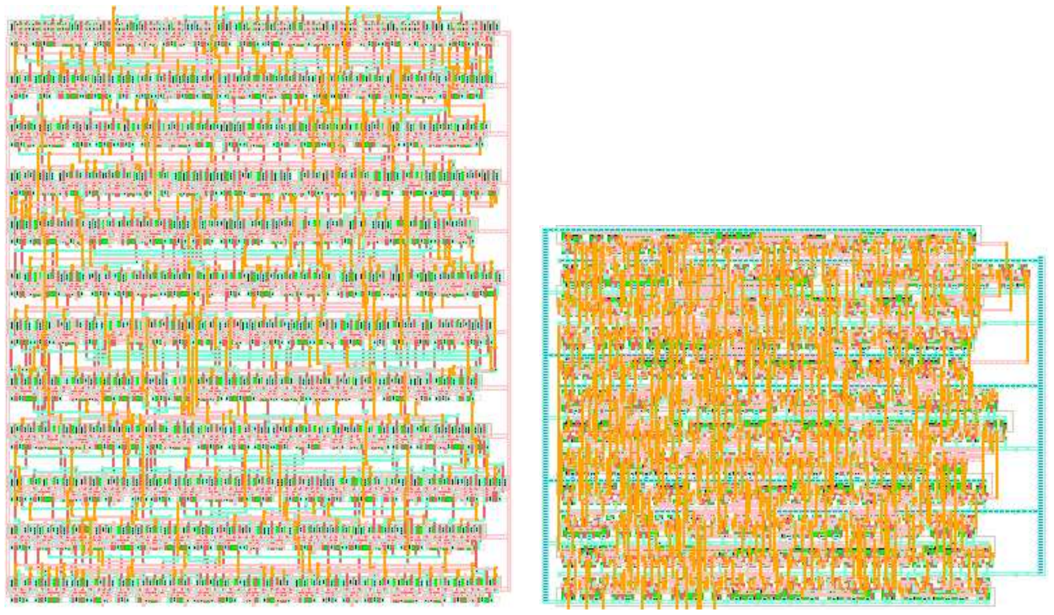
Gerez also separates the routing problem in *area routing* problem and *channel routing* problem. Besides that, settings of some parameters define different routing problems that can require specific techniques to solve them. The main parameters are:

- *Number of layers*: The number of layers available for routing depends on the technology. Nowadays, technology can offers to the designer six or more metal layers.
- *Gridded or gridless routing*: In gridded routing wires must run along points in a grid with uniform space while in gridless routing the wires may have different widths. Gridless routing algorithms are able to lead better the routing problem while gridded routing algorithm demand smaller computational requirements.
- *Position of terminals*: Terminals are sometimes located on the boundary of the cells and sometimes they are inside the cells. Besides, in some problems the position of the terminals are fixed and in other the router can move the terminals inside a restricted area.
- *Obstacles*: Sometimes part of the area in one or more layers is blocked.

Full Over-the-cell (FOTC) routing is presented in (JOHANN, 1994). In this work, Johann presents a new algorithm called LEGAL applied to the FOTC problem. FOTC routing is a type of area routing in which all connections are performed over the layout.

Figure 4.5 shows a comparison between a circuit which the routing problem is solved with the use of channel routing (4.5(a)) and with the use of FOTC routing (4.5(b)). In this example the placement of the cells is exactly the same, emphasizing the area reduction when using the second strategy.

In addition to the occupied area reduction presented by the FOTC routing, the reduction of the wire length can signify a great improvement to a layout concerning reduction of the electrical parasitics of circuit delay. This can be verified in the chapter 6 which comparisons between these strategies are presented.



(a) Channel routing (b) Full over-the-cell routing

Figure 4.5: Comparison in area between channel routing and full over-the-cell routing

4.5 Experiences on Layout Generation

Layouts can be developed under different strategies and the consequences of area and performance of the circuit are directly related to the layout characteristics.

This section reports two experiences on layout generation. Characteristics are related to the supply lines and the placement of the contacts of the cell terminals. The advantages and disadvantages of the techniques are also discussed in this section.

The main purpose in both experiences is to reduce the diffusion area of transistors and internal and external routing of the cells. The success of the technique to generate layouts reflects directly in compact circuits and in the final delay.

4.5.1 First Experience

Figure 4.6 shows an automatic generated layout under some preliminary specified characteristics. In this first experience, the layout presents as main characteristics supply lines over the transistors and contacts to transistor outside the row.

As reported in section 4.2.3 the strategy to distribute the supply lines over the diffusions has as advantage the possibility to connect easily transistor drain and source signals to V_{DD} and GND. Disadvantages related to this type of supply lines disposition is the fact that the placement of body ties demand the increase of the gaps between adjacent diffusions strips.

This supply line style demands the second layer of metal and it reduces the routability of circuits, mainly when FOTC routing is aimed. For example, a technology with three metal layers can be routed only if it is used channels between the rows. The advantages of the FOTC routing cannot be applied in this case because in most cases only the third

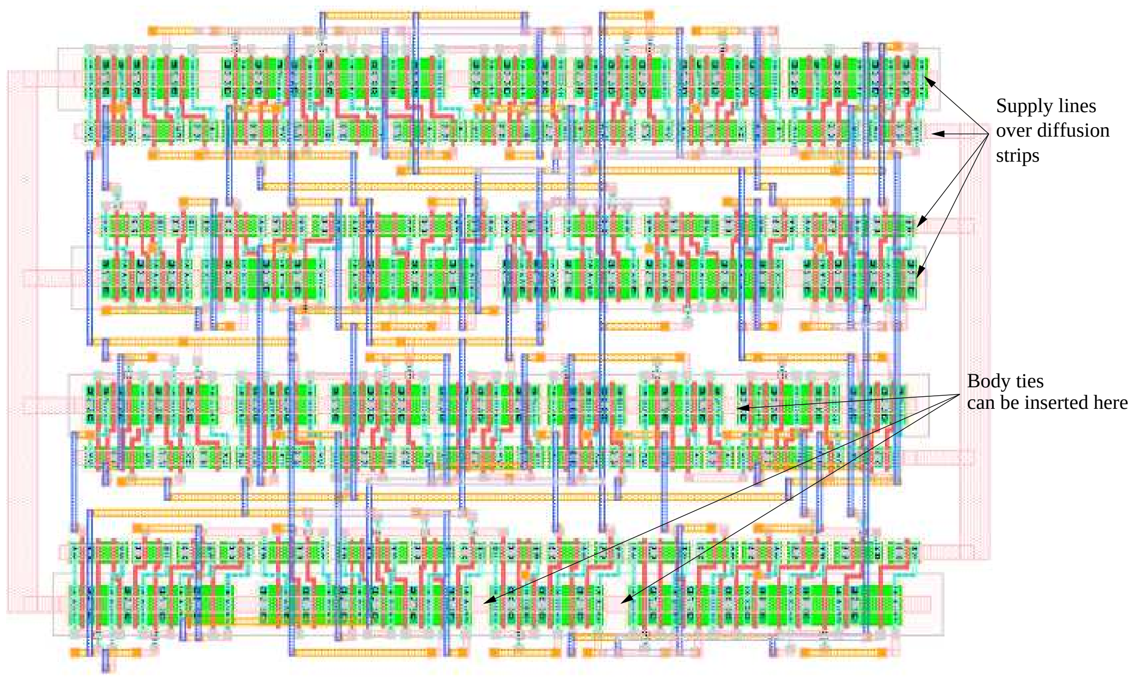


Figure 4.6: An Example Resulting of the First Layout Generation Experience

metal layer can be placed over the rows.

The rows mirroring can present good characteristics to layout because of the sharing of supply lines. In this type of supply line distribution, the advantages of the supply lines sharing cannot be used.

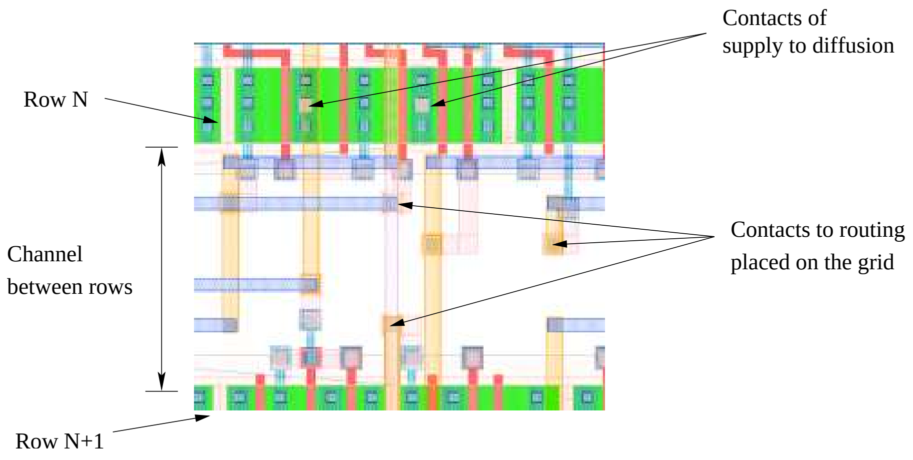


Figure 4.7: Other Characteristics about the Layout Shown in Figure 4.6

The figure 4.7 shows a part of the figure 4.6 in more details. It is shown the connection between the source of transistors to the supply lines and the channel demanded by the terminal contacts. Contacts to routing must be placed on a grid used by the routing step. This discrete spacing demanded by the router results in a big area in the channel between rows.

Taking into account these disadvantages, a new experience was developed and the obtained results are reported in section 4.5.2. In this experience, layout characteristics able to perform FOTC are implemented and full integration with the grid router is developed.

4.5.2 Second Experience

As previously reported, this second experience on layout generation was developed based on the disadvantages obtained from an initial automatic layout generation strategy (section 4.5.1).

The attempt for generating compact layout in order to obtain small as possible diffusion areas and wire length demanded basically two main modifications on the layout strategy:

1. Power supply lines between rows between adjacent rows (at the boundary of rows);
2. Contacts to routing inside the row.

Power supply lines between rows make possible the rows mirroring and the sharing of this lines. Another advantage of this type of supply line distribution is its implementation in the first metal layer. Thus, there is a reduction in routing obstacles in the second metal layer becoming easier the circuit routing.

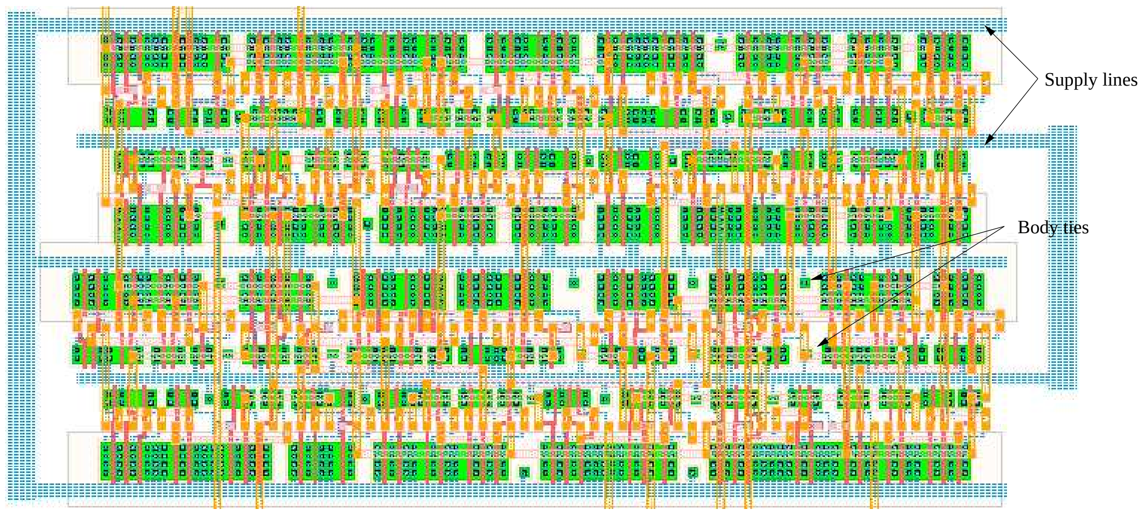


Figure 4.8: An Example Resulting of the Second Layout Generation Experience

In figure 4.8 it is shown an automatic generated layout based on the second experience. The supply line distribution is performed in the first metal layer and contacts can be inserted in the gaps between diffusion strips.

The layout shown in figure 4.8 is presented in more details in figure 4.9 and 4.10. The routing is hidden in these figures in order to present better visibility of the layout.

In figure 4.9, it is shown the supply line in the first metal layer and it is possible to conclude that the spacing between two adjacent rows is related to the size of the supply line. In comparison with the first experience, this spacing is reduced because of the placement of the contacts inside the rows.

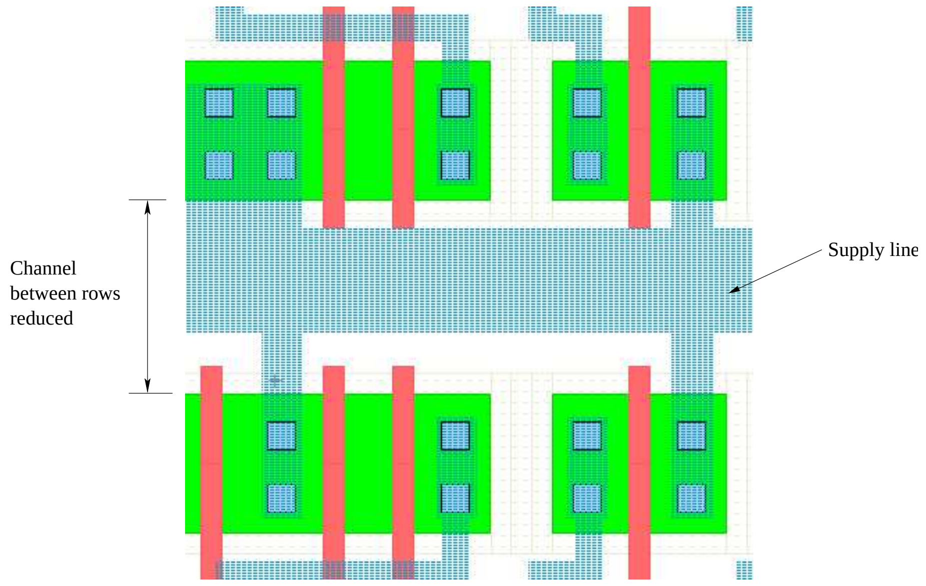


Figure 4.9: Supply line and Channel between rows about the Layout Shown in Figure 4.8

Figure 4.10 presents some important characteristics of this strategy. Contacts to routing are inserted always inside the row and placed on the routing grid. In addition, when it is necessary for increase the diffusion area of a transistor, only a diffusion area connected to V_{DD} or GND is increased.

This strategy increases only the source of transistors is done trying to reduce the delay of the circuit. Thus, parasitics capacitances and resistances minimized in the drain of the transistors present better delay to the circuit.

Complex Gates

SCCG (Static CMOS Complex Gates) may implement specified logic function more efficiently than a set of simple gates as $NANDs$ and $NORs$. In addition, a complex functional logic gate implementation occupies less area and provides in general small propagation delay than a small set of simple gates needed to implement the same function.

In standard cell libraries, a set of useful SCCG functions is implemented in order to improve the advantages of the complex gates in the layout. On the other hand, other types of complex gates may be generated increasing the number of cells into the library. Thus, only a set of complex gates is generated.

Automatic custom layout generation is able to implement any kind of complex gates related to any logic function. Thus, this approach takes full advantage of the good qualities of the complex gates.

The layout implementation of simple and complex gates are very similar, mainly if it is used a style based on the linear matrix style. Differences in the layout of simple gates in comparison with complex gates lies only on the presence of extra connections between the source/drain of transistors.

In Figure 4.11 it is presented two implementations of complex gates automatically generated. The complex gate shown in figure 4.11(a) implements the function $S = !((AB) + (CD))$ and figure 4.11(b) represents the function $S = !(D + (EF) + (A(C +$

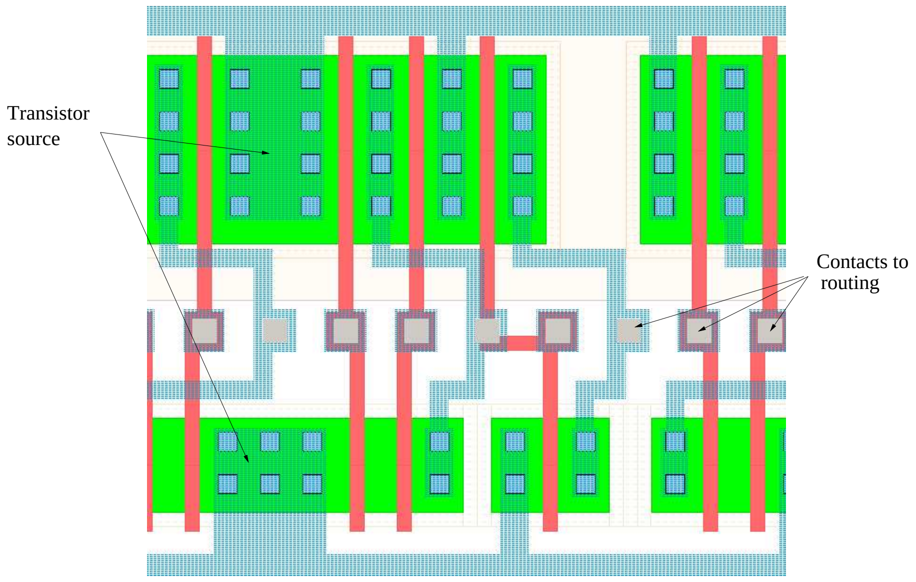


Figure 4.10: Other Characteristics about the Layout Shown in Figure 4.8

$B))))$.

In these layouts, complex gates are implemented as simple gates and connections between transistors source/drains are connected using the second metal layer over the transistors.

4.6 Layout Validation

Layouts can be validated on simulation but it can result in high costs to the designer because of the long time needed for simulation even in circuits with some few thousand of transistors.

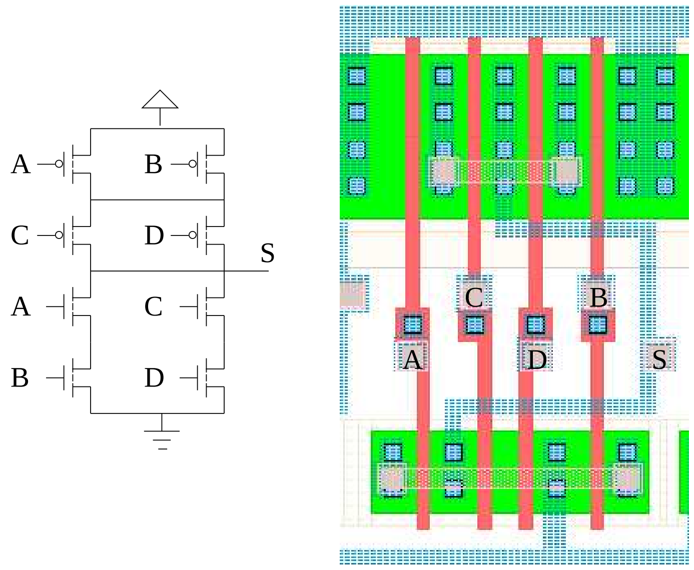
An important strategy to layout validation used in the industry of circuits is to perform a comparison between the developed layout and the its initial schematic. This procedure is faster than a simulation because electric characteristics may not be included and patterns are not necessary.

Automatic custom layouts can be validated using this strategy. The initial specification to the layout generation is a transistor netlist and a generation of a schematic from this netlist is very simple with commercial tools. An example of a useful layout versus schematic tool is the Cadence Diva^{TM} LVS. This tool is used to validate the layouts in this work. As part of the Cadence Virtuoso^{TM} custom design platform, Diva LVS is able to check, identify and correct layout connectivity and physical and logical mismatch errors.

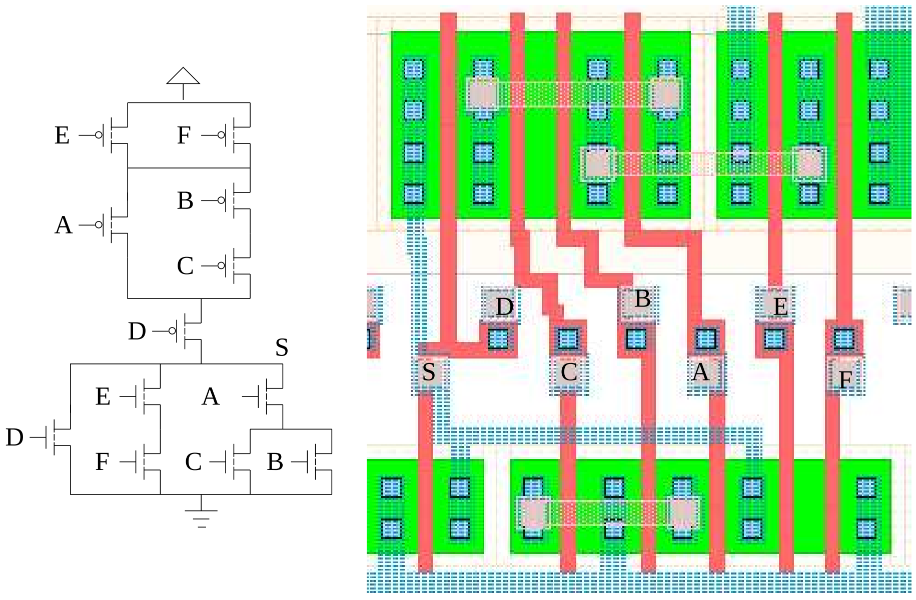
Figure 4.12 shows the used flow to validate layouts. Layouts are automatic generated starting with a initial transistor netlist.

Once the layout is generated, it is inserted into Cadence and a extraction is done in order to generated some layout information demanded by the LVS tool. The CDL (Circuit Description Language) description is also inserted into Cadence to generate the schematic.

After that, the layout versus schematic can be performed. This procedure generates several information about the extracted layout and the schematic and makes a comparison



(a) Schematic and Layout of $S =!((AB) + (CD))$



(b) Schematic and Layout of $S =!(D + (EF) + (A(C + B)))$

Figure 4.11: Examples of Complex Gates Automatically Generated

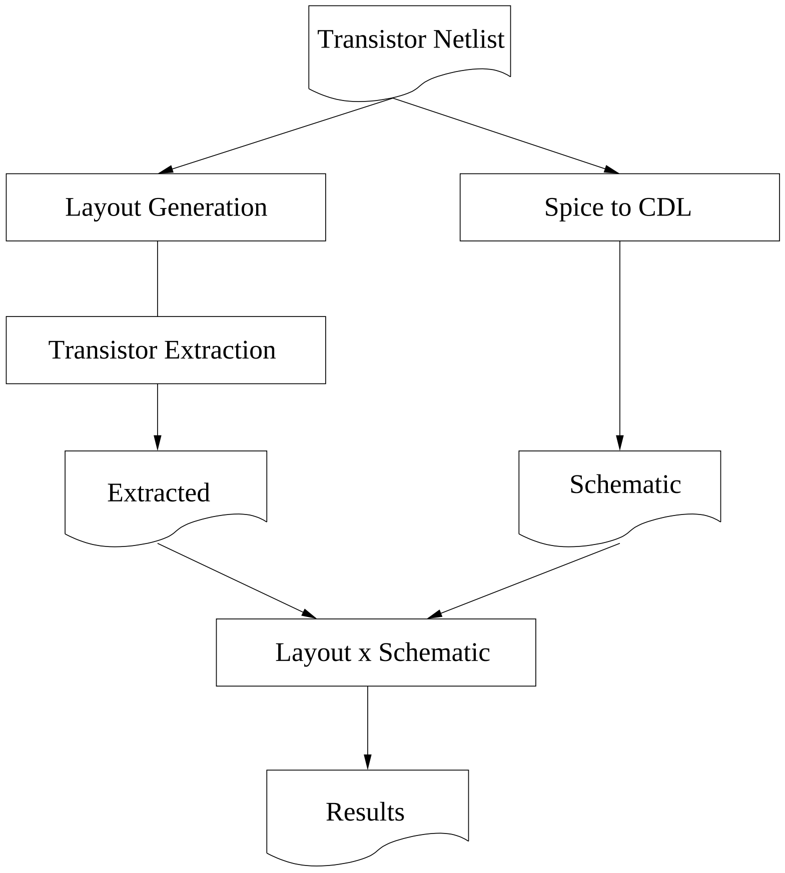


Figure 4.12: A Layout versus Schematic Flow

based on their characteristics.

By analyzing the resulted report, the designer is able to known whether the generated layout is implemented as the initial specification described in the transistor netlist.

4.7 Conclusion

Layout characteristics are very important due to the effort of the designers to generate circuits with the best possible compromise between area, delay and power according to project specifications.

The full automatic layout generation is highlighted in this chapter in which transistor placement, effects of the diffusion area on the circuit delay and the power supply distribution are excelled.

Besides, some experiences obtained by the effort to implement a automatic generation tool are discussed. These experiences are base to the development of a new strategy presented in chapter 5.

5 A NEW STRATEGY TO LAYOUT GENERATION

5.1 Introduction

A new strategy to layout generation is presented in this chapter. This strategy is based on the experiences reported in the chapter 4. A tool called *Parrot Punch* was developed as result of the proposed strategy. The Parrot Punch design flow is described in section 5.2. A preliminary work is presented in (LAZZARI et al., 2003) where a layout generation strategy is given for three metal layer CMOS technologies.

Parrot Punch associated with *Tictac::Sizing* are able to generated circuits optimized in delay. The transistor sizing is applied to the layout using the folding technique. In section 5.3 it is presented a new folding algorithm developed by Bastian (BASTIAN, 2003).

A new grid router is presented in section 5.4. A new method to avoid the blocking in maze router is implemented in this router. The method consists on divide blocking situations in three types and solves the problem slicing the layout.

5.2 Layout Generation Strategy

This section presents the strategy used to generate layouts based on the second experience. This layout style is used because of the advantages shown in the section 4.5.2.

In figure 5.1 it is presented the main characteristics of a layout generated by *Parrot Punch*. It is shown the folding technique, internal connections in metal2, the body tie insertion in the diffusion gaps, the attempt for using minimal allowed values to transistor spacing and input/output contacts between PMOS and NMOS transistors.

Figure 5.2 shows the automatic layout strategy. In this flow it is presented the layout generation separated in some steps. Based on this flow, *Parrot Punch* is able to generate full custom automatic layouts as follow specified.

5.2.1 Input Files and Parsers

For generating the layout four input files are needed. These files contain basic information used in the layout generation. A set of parsers is able to load these information and to store them into the data structure. The input files are:

- Parameter defined by the user: User parameters consist on some definitions used in the layout generation as supply line characteristics and transistors width. These user characteristics influence on each logic gate in the circuit.

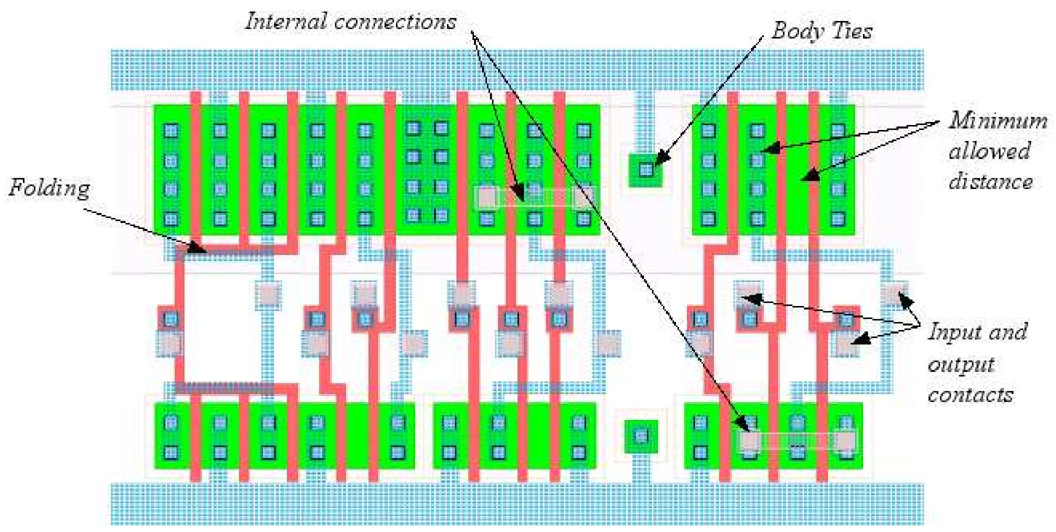


Figure 5.1: An example of Layout generated by Parrot Punch

- Design Rules: Design rules are the technology rules furnished by the silicon foundry. They are basically minimal values to layers, metal enclosures to contacts and vias and minimal spacing between identical layers. The layout generation tries to use minimal values given by the technology whenever it is possible.

- Circuit Description: A SPICE-like netlist is used as input. This netlist is a set of logic gates defined by sub-circuits. Each sub-circuit in the netlist represents a logic gate.

- Placement: Layouts are generated in which logic gates are placed in rows. Thus, the placement input is a list of logic gates laid in rows. Placement is done separated from the layout generation. Consequently, any placement tool can be used to realize this task. The placement tool must be able to use an area estimate of each logic gate in the circuit because this information is not know in this step. Usually, Mango Parrot (HENTSCHKE, 2002) is used.

5.2.2 Transistors placement

Once logic gates and their position in the layout are known, the layout can be generated. Transistors placement is the first step in the circuit generation flow. This step consists on searching for the Euler path in order to choose the position to each transistor in a row. This algorithm is responsible for ordering transistors in the rows in such a way that PMOS and NMOS transistors with common gate signal are easily connected using only the polysilicon layer.

The Euler algorithm is applied to a set of transistors (a logic gate) in which each transistors has an associated weight (JOHANN, 2003). Figure 5.3 shows the algorithm used in the transistor placement. The value of the weight is attributed to a transistor concerning the estimated position of the nearest point of this net in the circuit. Thus, a transistor connected to a gate in its left has more chance to be placed on the left side

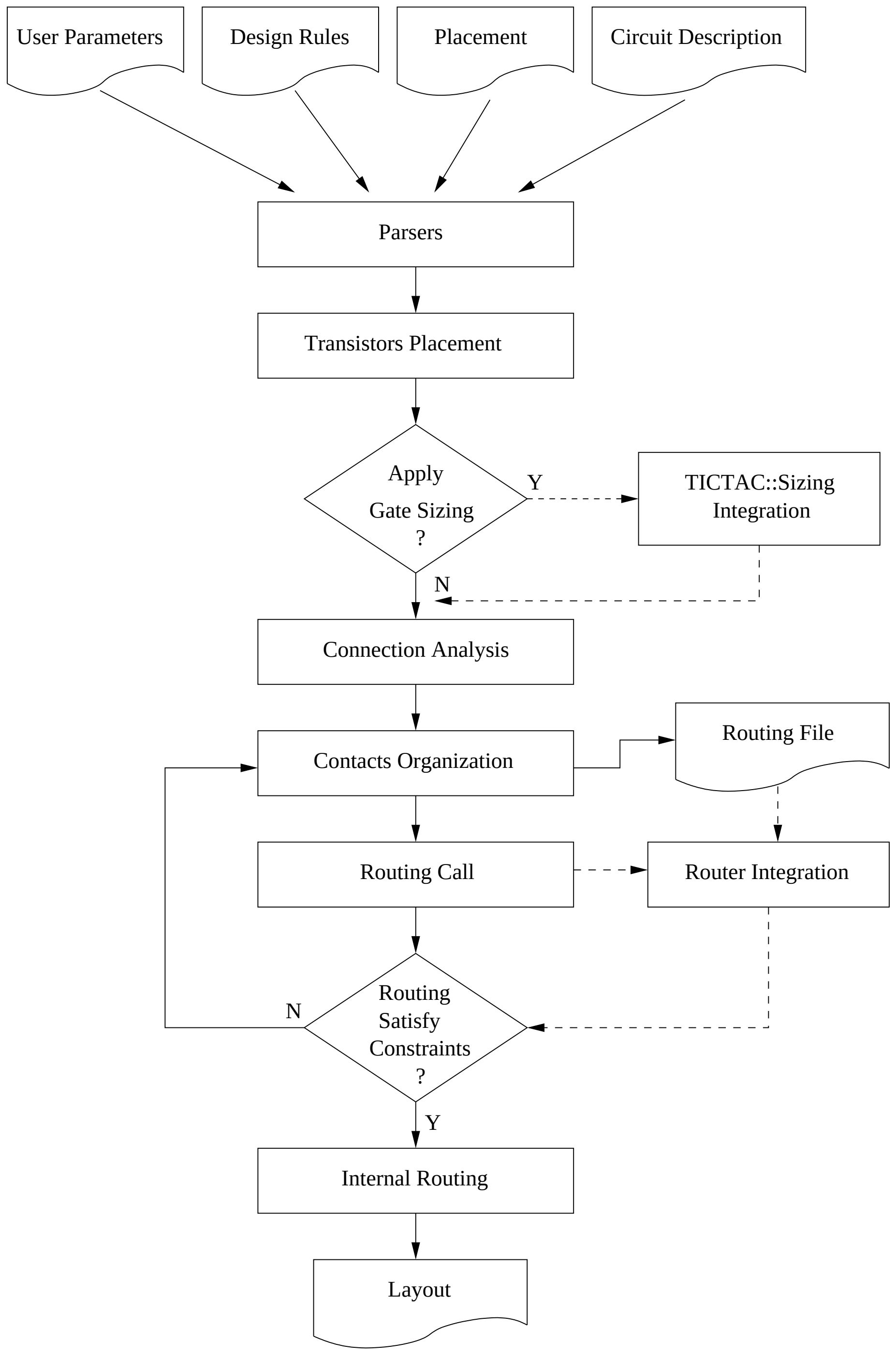


Figure 5.2: Automatic Layout Generation Strategy

```
weight(transistor t, gate G) {
  if connection_is_on_the_left_of_this_gate(G)
    return number_of_transistors_on_the_left_size(T)
  else if connection_is_on_the_right_of_this_gate(G)
    return number_of_transistors_on_the_left_size(T)
  else
    return 0
}

trans_place(gate G) {
  foreach transistor_list L in euler_path_to_gate(G) do
    gate_weight[L] = 0
    foreach transistor T do
      gate_weight[L] = gate_weight[L] + weight(T, G);
    end
  end
  return smaller_gate_weight(gate_weight);
end
```

Figure 5.3: Algorithm for transistor placement

because its weight is smaller. This is performed in order to reduce the wire congestion over the transistors of a gate.

5.2.3 The Gate Sizing Tool Integration

In section 3.5 it is shown a tool called *TICTAC:Sizing* able to analyze a circuit and to realize gate sizing based on some constraints demanded by an automatic layout generation tool. The integration between *TICTAC:Sizing* and *Parrot Punch* is done in such a way that transistors width given by the sizing tool can by easily applied in the layout generation.

Once transistors are sized, a list of gates and their transistors width is generated. This list is used to apply the gate sizing through the gate folding technique. This is explained in more details in section 5.3.

This step is optional to the layout generation. If the circuit was not optimized by the *TICTAC:Sizing* the layout generation can be performed without any restriction.

5.2.4 Connection Analysis

The connection analysis step consists on the analysis of any internal net of each gate in the circuit. This is necessary because of the correctly estimation of the internal connection locations and the area occupied by the gates.

Internal gate connections and their number of transistors are responsible by the area occupied by this gate. Once these connections are generated only after that the routing, connection analysis and the estimate of where these connections will be implemented in such a way the other steps can be realized without problems.

This step is very trivial in simple gates in which only the output signal must be analyzed. In complex gates, many other internal connections may exists and the task to predict where is the best location to each connection is not an easy task.

In complex gates, connections between transistors source/drain are implemented over the transistors. These connection estimative demanded by complex gates are passed as obstacles to the routing of the circuit.

5.2.5 Contacts Organization

Contacts organization consists on generate the sequence of input and output contacts in a gate. The result of the transistor placement tool is verified to both PMOS and NMOS positions and contacts are inserted in the layout.

Contacts are inserted inside the row and they are placed aligned with the routing grid. Figure 5.4 shows the grid spacing definition which all metal layers are evaluated to result in the real grid spacing without violating design rules.

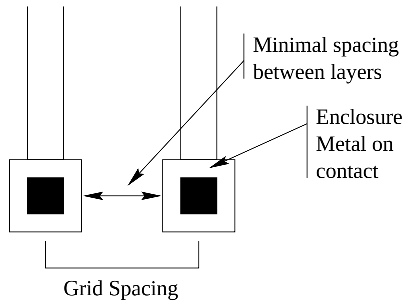


Figure 5.4: Grid router spacing

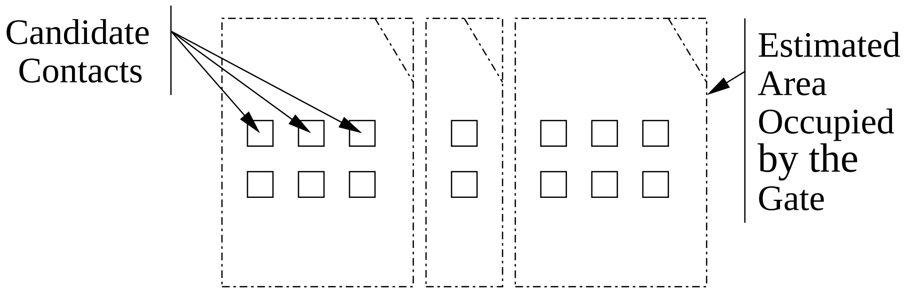


Figure 5.5: Candidate positions to gates input and output terminals

In order to generate better circuit routing, a set of candidates positions to the input and output contacts are passed to the router. With these set of positions the router are able to choose the best position to implement a connection. Figure 5.5 illustrates the estimated area occupied by a logic gate and the set of candidate positions to input and output terminals.

5.2.6 The Routing Integration

After the contacts organization is performed, the routing of the circuit can be done. An external router performs the routing. Thus, any router can be used to realize this task. Usually, the *Worm Grid Router* is used to route the circuits. This router is totally integrated with *Parrot Punch* and its advantages are presented in section 5.4.

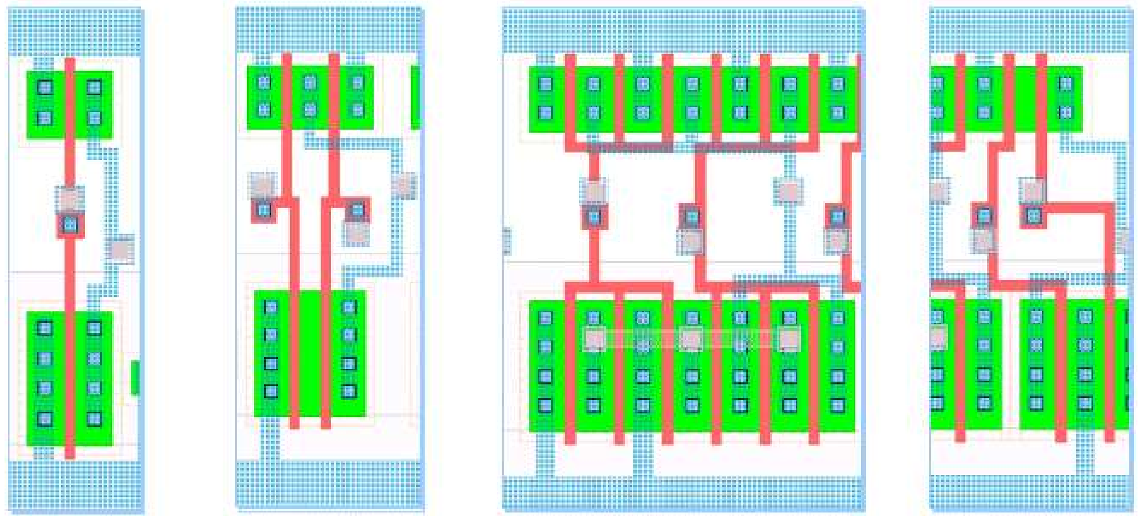


Figure 5.6: Four situations on the poly routing algorithm

When the routing cannot be completed and if the problem is related to the contacts position the contacts organization step can be realized again until the routing problem can be solved. This means that not only the rows can be moved during the routing step but the position of the input and output terminals also can be changed.

5.2.7 Row Internal Routing

When this step is realized, the positions of the contacts are placed and they can be moved in function of the result of the routing. Transistors are also placed into the layout but if there are spaces in the diffusion gaps, it allows that transistors can be moved in these areas.

The internal routing is divided in two parts. First, polysilicon nets are connected and after, source/drains transistors areas are implemented.

Polysilicon Routing

It consists on connecting transistors and contacts placed by the router that shares the same signal.

In figure 5.6, four possible situations are shown according the position of the transistor and the contact inside the row. These situations are evaluated and the polysilicon routing is performed. They are:

1. If the contact and the transistor are aligned by the axis X, they can be connected by a straight line;

2. If the contact and the transistor are not aligned by the axis X but there are not obstacles between the gate and the contact;

3. When adjacent transistors are connected to the contact (folding technique);

4. If the contact and the transistor are not aligned by the axis X but there are obstacles

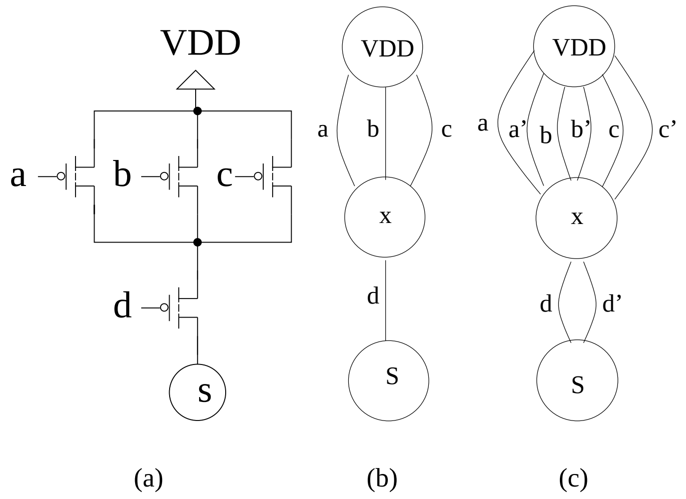


Figure 5.7: Schematic and Graph of a logic gate

between the gate and the contact, the river routing algorithm is used to connect the gate and the contact.

Source/drain connections

These connections are implemented with the first and the second metal layers. The first metal layer is always used in wires between P and N diffusion strips. The second metal layer is only used when the function logic demands connections over the transistors (i.e. in complex gates).

5.3 Transistor Folding

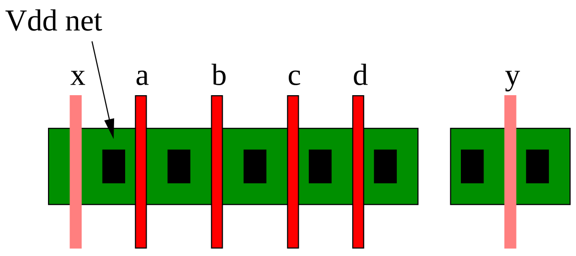
In timing-driven layout synthesis, transistor sizes tend to be significantly different from each other and thus the use of conventional layout approaches can cause inefficient area utilization (KIM; KANG, 1997).

The folding technique consists on breaking a large transistor into smaller ones, connected in parallel and placed continuously with diffusion sharing. The folding technique is especially important in the case of row-based layout. Different transistor sizes may cause non-uniform cell heights, which may lead to significant waste of area.

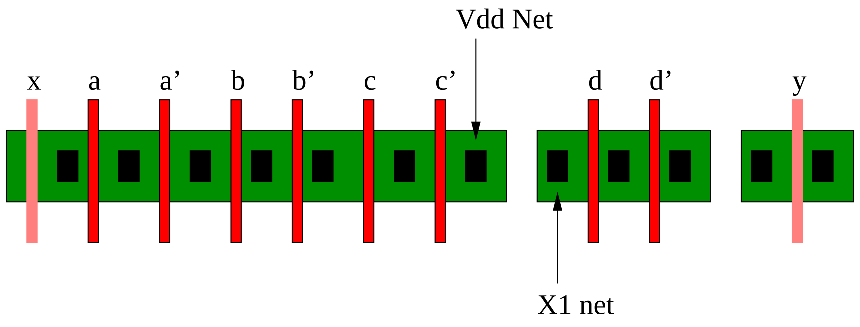
The basic problem of folding is that it affects the diffusion sharing. When a transistor with source and drain nets (S, D) has to be folded by an odd number of legs, the nets at its ends remain (S, D). The problem appears when the transistor should be folded by an even number of legs. In this case, the end nets depend on the transistors orientation: (S, S) if the transistor is placed unflipped, and (D, D) if the transistor is placed flipped. At this point, the diffusion sharing will certainly become affected.

This folding algorithm intent to look at every folded transistor as a unique transistor, applying the folding over a given transistor placement without either modifying this placement or inserting new diffusion breaks in the row.

Figure 5.7 exemplify the folding technique. In figure 5.7(a) its presented the PMOS part of a logic gate. This schematic is converted to a graph in figure 5.7(b). Figure 5.7(c)



(a) P part of a logic gate



(b) Duplicated transistors to improve folding

Figure 5.8: An Example of the Folding technique

presents a graph in which all transistors are duplicated.

Figure 5.8 presents the layout view of the logic gate presented in figure 5.7. Transistors x and y means transistors of other logic gates placed on left and right.

The P part of the gate is illustrated in figure 5.8(a) where the V_{DD} net is shared with the gate placed on the left. Figure 5.8(b) shows the duplication of the transistors and the new organization of the transistors.

These figures exemplify the presence of gaps in the diffusion blocks after folding. In this example it is impossible to find a path between the net V_{DD} and the logic gate output without gaps (the gates sequence must be maintained).

There are two ways to eliminate the gaps in the diffusion blocks:

- *Odd number of transistors*: It is possible to avoid the gaps on diffusion increasing the number of folded transistor to an odd number (i.e. 3 parts). Thus, the gap is eliminated due the additional transistor (c'' in the example). This method reduce the width of the row but increases the number of transistors switching.

- *Transistor connected to the supply line*: Another method is to insert a transistor connected to the supply in order to create a opened transistor.

Bastian (BASTIAN, 2003) developed a folding technique applied to automatic custom layout generators. The technique finds the optimal folded transistors arrangement for a given transistor placement, given desired transistors sizes and given initial PMOS and NMOS transistors sizes.

The developed algorithm introduces a new concept in which transistors are classified according even and odd multiplicity. An even transistor is a transistor that, after resized, gives a path to return to a node. An odd transistor is a transistor that, after sized, gives a path to go to another node.

Figure 5.7(a) is taking as example in order to explain the proposed algorithm. Assuming that a previously execution of the Euler path algorithm finds the sequence a, b, c and d, the following steps are realized in order to apply the folding technique:

1. *Passing by transistor a*: it is odd, because it gives a path to go from V_{DD} to $N1$ (go from one node to another);
2. *Passing by transistor b*: now the analyses is made to the pair of transistor a and b. This pair of transistors together are even, because they give a path to go from V_{DD} to V_{DD}. In other words, they give a path to return to V_{DD} node. So, transistors a and b are classified as even transistors;
3. *Passing by transistor c*: it is odd, because it gives a path to go from V_{DD} to $N1$.
4. *Passing by transistor d*: it is odd, because it gives a path to go from $N1$ to S. Just at this part of the algorithm transistor c is marked as an odd transistor.

5.4 The Worm Grid Router

Worm Grid Router is a maze router based on the algorithm of Lee-Moore (LEE, 1961). A maze router uses a surface represented as a grid where each point in the grid has connection to the adjacent points (GEREZ, 1998). The worm router was developed to be integrated with the layout generator Parrot Punch.

The maze routing is divided in two phases in order to connect source and destination points with the smallest possible path:

1. *Expansion*: This phase enumerates the points on the grid surface with the distance of the source point. This phase is executed until the destination point is found;
2. *Traceback*: Implements the connection by following any path with decreasing labels.

It is guaranteed that the maze router always find the shortest as possible path between a source and destination point for a given connection. However, in a circuit with several connections, a connection may block other connections taking the routing impossible. This highlights the use of other techniques applied to the maze router in order to perform complete routing to any circuit.

The main characteristics of the Worm router are the following:

- Grid Maze-based algorithm;
- 1-to-N routing layers;
- Independent directions to any routing layer (Horizontal, Vertical and Diagonal);
- Fully integrated with Parrot Punch.

The Worm router uses a technique based on slices to open horizontal and vertical spaces on the grid to guarantee the completeness of the routing. The slices on the circuit are performed in such a way that all terminals over the grid are spaced.

Some experiences were developed whence only the row related to the blocking connection was spaced but it generates problems in other connections. Thus, when a slice in the grid is opened, all points in a column or a line are separated.

In order to perform the routing, three block situations are responsible to invalidate the routing. Solving each one of these blocking problems, a router is able to complete route any circuit.

5.4.1 First Situation: Blocking on the Source Point

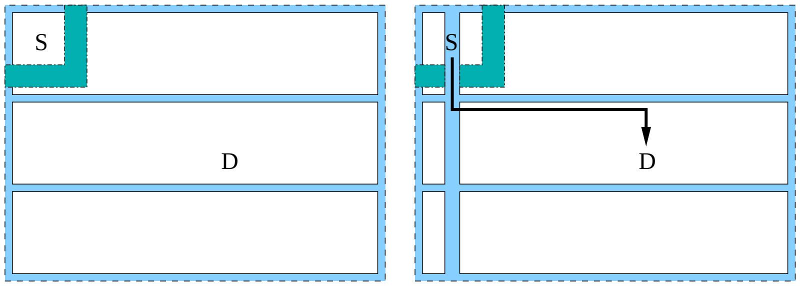


Figure 5.9: First Situation: Blocking on the Source Point

Figure 5.9 shows the first situation which the source point is blocked. Opening a vertical space in the column of the source point can solve this situation. In this blocking situation, horizontal wires are maintained unaltered but vertical wires are spaced in such a way a new vertical path can be used to route the blocked net.

5.4.2 Second Situation: Blocking on the Destination Point

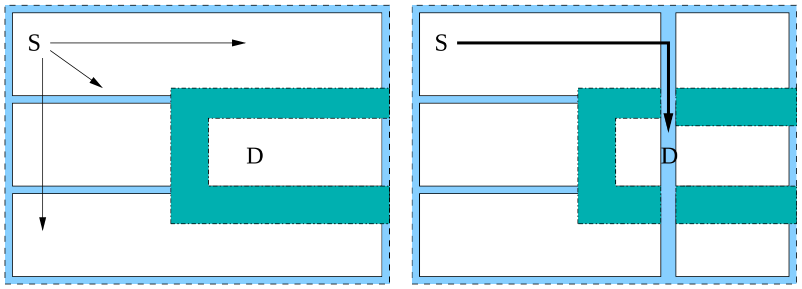


Figure 5.10: Second Situation: Blocking on the Destination Point

Figure 5.10 shows the second situation which the destination point is blocked. Opening a vertical space in the column of the destination point can solve this situation.

As the first blocking situation, horizontal wires are maintained unaltered and vertical wires are spaced in order to connect source and destination points.

5.4.3 Third Situation: No path between Source and Destination

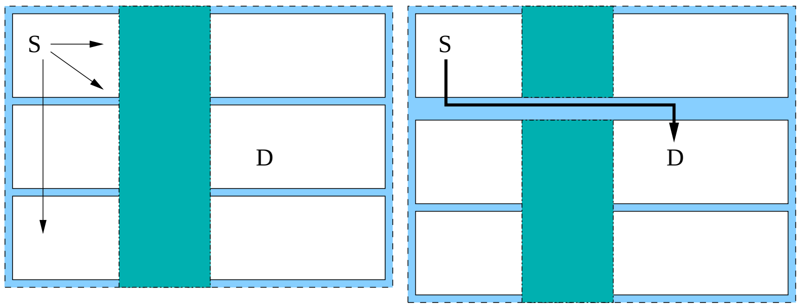


Figure 5.11: Third Situation: No path between Source and Destination

Figure 5.11 shows the third situation which any point (source and destination) is block but the connection cannot be performed because of a congestion between the points. When this situation is found, rows where source and destination points are presented are space to complete the routing of the blocked net.

5.5 Conclusion

A new layout generation strategy is presented in this chapter. The main goal of this strategy is to generate compact and timing optimized layouts as well as a designer can develop manually.

Some good characteristics of the proposed strategy are:

1. Euler path algorithm with weights in the transistors associated to the routing of the circuit. This reduces the complexity of the routing over the layout;

2. Integration with a gate sizing tool and a new folding technique;

3. A routing technique applied to a maze router algorithm to achieve complete circuit routing;

Parrot Punch is an automatic custom layout generator based on this strategy. Layout characteristics are based in layout experiences shown in section 4.5. The main characteristics are:

1. *Supply lines*: The supply lines are generated shared by adjacent rows in metal 1;

2. *Reduced transistors drain/source areas*: Minimal values obtained by the design rules are used whenever it is possible;

3. *Internal routing*: Connections of internal nodes are performed in metal 1 whenever it is possible. Thus, the second metal layer is usually free to perform the circuit routing;

4. *Complex Gates*: Complex gates can be easily generated. Only extra nets are generated over the transistors in metal 2;

5. *Folding*: The Folding technique applies discrete sizing aiming at small area penalties.

These characteristics show that Parrot Punch is a tool able to deal with the layout generation problem. All steps on the layout generation process are developed in such a way generated layouts are very similar to layouts developed by hand.

6 EXPERIMENTAL RESULTS

6.1 Introduction

The main goal of this chapter is to report results obtained from the comparison between the strategy described in chapter 5 and other well known methods to layout generation. Results were obtained by the utilization of logic optimization tools as Cadence PKS and Berkeley SIS and Cadence tools for parasitics extraction and simulation.

In section 6.2 it is presented comparison in area occupation, circuit delay and power consumption between layout generated by TROPIC3 and Parrot Punch. Section 6.3 shows some results obtained from the comparison between Parrot Punch layouts and a very commercially used tool based on the standard cell approach.

Besides, results on layout optimization are reported in section 6.4. In this section, some circuits were optimized in the TICTAC:Sizing tool and the layouts were generated based on the gate sizing results.

The technology used in these circuits is the AMS 0.35μm with 3 metal layer.

6.2 Comparison with the automatic custom generator TROPIC3

As previously reported, TROPIC3 (MORAES; REIS; LIMA, 1997) is an automatic custom layout generator able to generate circuits based on a transistor netlist. In this section is reported a comparison between layouts generated by Parrot Punch and TROPIC3.

Layouts were generated in both tools, checked and extracted by CADENCE Diva^{TM}. Pattern applied to the primary inputs were randomly generated and the electric simulation were performed using the CADENCE Spectre^{TM}.

Delay analysis was performed based on the propagation delay as reported in section 3.4.1. Power consumption were measure based on the RMS consumption on the V_{DD} source of the circuit.

Figures 6.1, 6.2 e 6.3 show the circuits C432, C499 e C880 generated by Parrot Punch and TROPIC3. It is possible to note that the layouts generated by Parrot Punch are smaller than layouts generated by TROPIC3. The reason of these results is related to the routing strategy. In Parrot Punch layouts, the FOTC strategy is used in which routing channels are not used.

Table 6.1 shows the area occupation in Parrot Punch and TROPIC3 circuits. It is possible to note that there is a difference between the occupied area of the layouts in the same technology.

The main reason of these values is due to the routing strategy. In TROPIC3, routing is

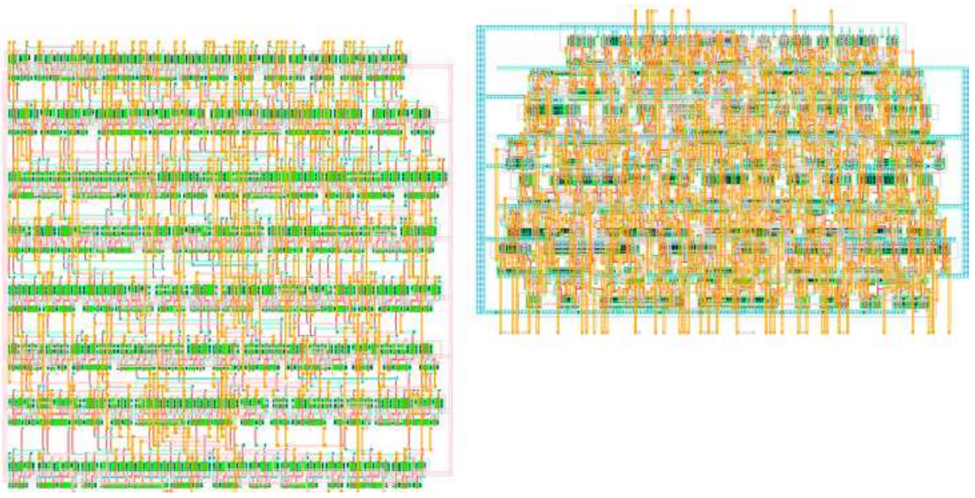


Figure 6.1: TROPIC3 (left) and Parrot Punch (right) layouts of the C432 circuit

Table 6.1: Area Occupation in TROPIC3 and Parrot Punch layouts (μm^2)

Bench	No. Trans	Parrot Punch	TROPIC3	Gain (%)
c17	24	495.6	756.6	34.4
c432	804	20782.2	35443.2	41.3
c499	1556	54000.8	86224.3	37.3
c880	1802	59925.6	96014.4	37.5
c1355	2308	74613.9	111784.8	33.2
c1908	3482	116397.0	181575.6	35.8
c6288	10112	303376.8	578082.0	47.5
Average				38.14

performed basically in routing channels between p and n diffusions while in Punch Parrot the routing is generated over the rows whenever it is possible.

Results present between 33.2% and 47.5% of area reduction layouts generated by Parrot Punch when compared with layouts generated with TROPIC3. The average in the area occupation is around 38%.

Table 6.2 presents results given by electric simulation. The table shows comparison between delay and power consumption of layouts. Results related to delay of circuits show an average gain of around 22% when compared to TROPIC3 layouts. These results are obtained due to the difference of routing strategy and the attempt for optimizing drain/source areas on Parrot Punch strategy.

As reported in previously chapters, the effort for layout optimization is employed in all steps of the layout generation strategy proposed in this work. In addition, the reduction on the wire lengths obtained by the FOTC routing reduces the delay of the wires.

Power consumption in Parrot Punch layouts has an average gain of 4% over TROPIC3 layouts. In these circuits, it was applied the same patterns to simulation and the switching activity is the same. Thus, the power consumption must be almost the same in layouts generated by both tools.

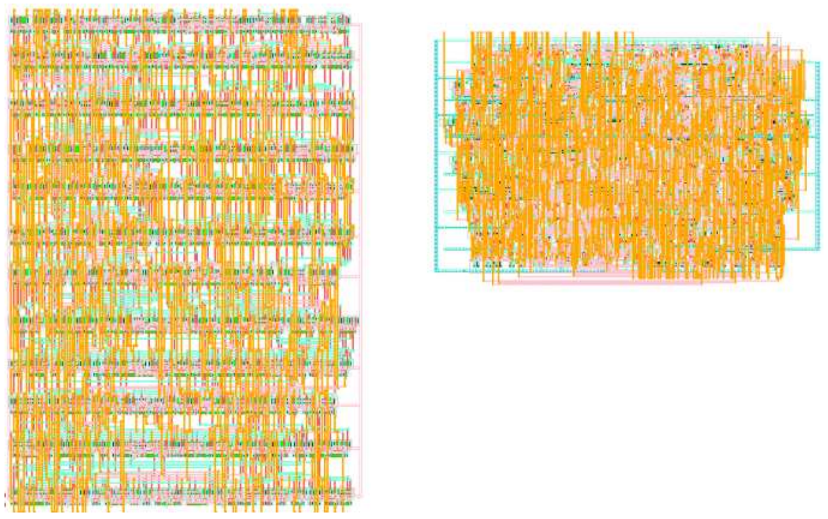


Figure 6.2: TROPIC3 (left) and Parrot Punch (right) layouts of the C499 circuit

Table 6.2: Delay (ns) and Power Consumption (mW) in TROPIC3 and Parrot Punch Layouts

Bench	No.	Parrot Punch		TROPIC		Gain (%)	
	Trans	Delay	Power	Delay	Power	Delay	Power
c17	24	0.47	0.42	0.55	0.45	15.1	12.0
c432	804	0.78	15.31	1.07	16.73	26.9	9.3
c499	1556	0.31	18.61	0.42	17.8	26.0	-4.0
c880	1802	1.73	17.67	2.22	17.2	22.0	-2.0
c1355	2308	0.50	26.38	0.64	29.0	21.1	9.0
c1908	3482	2.67	35.1	3.66	45.56	27.0	22.0
c6288	10112	4.28	205.1	5.45	201.9	21.4	-1.5
Average						22.75	4.47

The analysis of the power consumption in the circuit C1908 shows great difference between the layout generated by Parrot Punch and TROPIC3. This difference does not appear in other layouts. Verifying the simulation waveforms, glitches were found with more frequency in the layout generated by TROPIC3. It is believed that these glitches are responsible by the greater power consumption in the TROPIC3 layout.

6.3 A Comparison with the Standard Cell Approach

Traditional layout development is based on standard cell libraries. Silicon Ensemble is an example of tool used to place and route circuits based on the standard cell methodology.

In figure 6.4 it is presented the design flow used in the layout comparison between Parrot Punch and the Standard Cell approach. In order to develop a comparison between area occupation and circuit delay, layouts were generated with Parrot Punch and the electric simulation were performed. After that, the obtained circuits delay was used as constraint

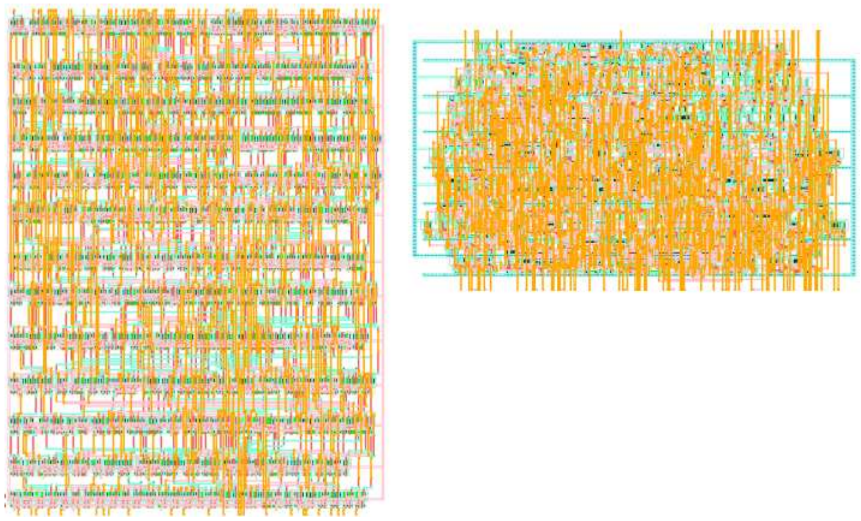


Figure 6.3: TROPIC3 (left) and Parrot Punch (right) layouts of the C880 circuit

to the logic optimization in the Cadence PKS and the standard cell-based layouts was generated in the Cadence Silicon Ensemble.

The circuit **7seg** was generated by Parrot Punch. Simulations were performed and the obtained delay was 0.79ns. This delay was applied to the Cadence PKS and the layout was generated with Cadence Silicon Ensemble. The resulting area is 6133μm^2 and 7831μm^2 to Parrot Punch and Silicon Ensemble layouts, respectively.

Table 6.3: Comparison between STD CELL and Parrot Punch layouts

Bench	No.	STD.CELL		Parrot Punch		Gain (%)
	Trans	Delay	Area	Delay	Area	Area
7Seg	294	0.77	7831	0.79	6133	22%
csa8	288	2.93	6500	3.19	7730	16%
bw	628	1.78	16251	1.8	15979	2%
alu2	1390	4.13	41260	4.02	44585	8%

The same method was used in the circuits **csa8**, **bw** and **alu2**. In these circuits, the occupied area in Parrot Punch layouts was around 16%, 2% and 8% smaller than layouts generated with the Standard Cell approach. Table 6.3 shows the obtained results.

6.4 Layout Optimization

A folding algorithm is presented in section 5.3. Parrot Punch uses this algorithm in order to apply the gate sizing in circuits. *TicTac::Sizing* was applied to some circuits to perform timing optimization.

Table 6.4 shows the gate sizing applied to the circuit csa8. A delay reduction of 30% was required to the TicTac::Sizing. To implement the sizing in the layout, 9 gates were sized where 25 PMOS transistors and 37 NMOS transistors were increased. Due to logic limitations, the obtained delay reduction is 20% with an area increase of 3%.

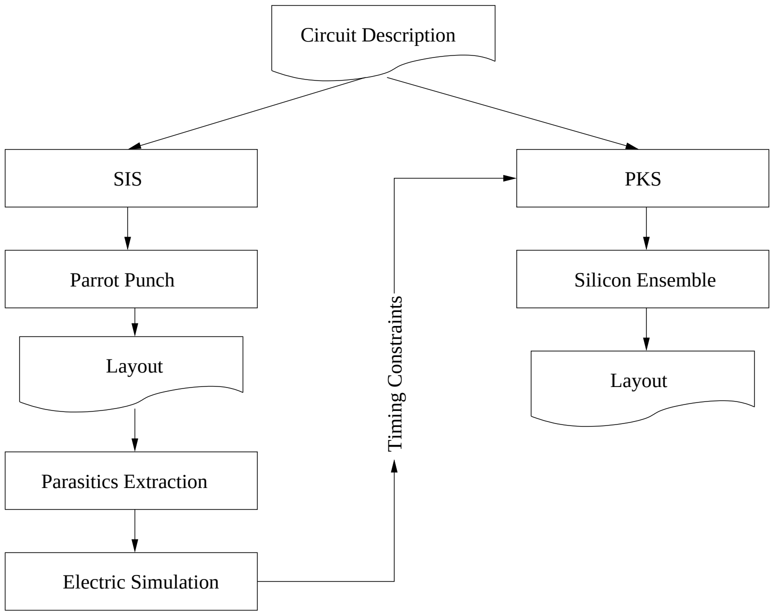


Figure 6.4: Comparison between Standard Cell layouts and Parrot Punch Layouts

Table 6.4: Layout optimization in the circuit csa8

	Optimization	
Delay Red. Req.	N.opt.	30%
Area	6500μm^2	6695μm^2
Delay	3.19ns	2.56ns
Power	0.004w	0.007w
Sized gates	-	9
Sized P-trans	-	25
Sized N-trans	-	37

In the circuit bw (table 6.5), three delay requirements were demanded to TicTac::Sizing. In this circuit, the maximum delay reduction was 25% and the occupied area increases 4%, 6% and 10% to the delay reduction requirements of 10%, 20% and 30%, respectively. The number of sized gates increases in exponential way. When 10% of delay reduction is required, only 4 gates are sized. On the other hand, 23 gates are sized to 30% delay reduction requirement but only 25% are obtained.

The circuit alu2 has as characteristic the large number of connections. These connections influence on the area occupied by the circuit. It is possible to note in the results that the sized circuits present small area occupation related to original circuit. This is due to routing congestion.

The main characteristic of the *worm router* (section 5.4) is to slice the circuit when points in the circuit cannot be connected. The folding technique reduces the contact concentrations in a region of the circuit. Thus, the number of slices is reduced and the occupied area is smaller.

These examples show what is presented in figure 3.18. The delay of circuits can be

Table 6.5: Layout optimization in the circuit bw

	Optimization			
Delay Red. Req.	N.opt.	10%	20%	30%
Area	$16068 \mu m^2$	$16710 \mu m^2$	$17032 \mu m^2$	$17674 \mu m^2$
delay	1.8ns	1.63ns	1.45ns	1.36ns
Power	0.010w	0.009w	0.012w	0.012w
Sized gates	-	4	8	23
Sized P-trans	-	4	12	42
Sized N-trans	-	2	13	30

Table 6.6: Layout optimization in the circuit alu2

	Optimization			
Delay Red. Req.	N.opt.	10%	20%	30%
Area	$41260 \mu m^2$	$40434 \mu m^2$	$37546 \mu m^2$	$39197 \mu m^2$
delay	4.02ns	3.57ns	3.51ns	3.53ns
Power	0.018W	0.018W	0.018W	0.020W
Sized gates	-	5	8	22
Sized P-trans	-	19	26	59
Sized N-trans	-	13	24	56

reduced while the structure of a circuit can be changed to support this delay requirement. After that, delay constraints have no effect in the timing optimization but the area increases significantly.

6.5 Conclusion

In this chapter it is presented the results obtained from the comparison of the proposed method for layout generation with other well-known methods. The results show that the Parrot Punch is an evolution in comparison with TROPIC. Both tools use the strategy to generate blocks and circuits without the use of library cells and the main goal of these tools is to generate layouts similar to the manual development.

A comparison with the standard cell approach was also performed. Silicon Ensemble is used by the industry of integrated circuits and the obtained results show that the automatic full custom generation is able to generate layouts similar to the manually developed.

Some examples are presented in order to show the efficiency of the gate sizing algorithm by the utilization of the folding technique. Examples show the increase of transistors width and the attempt to reach to a required delay reduction with minimum increase in the occupied area.

7 CONCLUSION

In this work it is presented the effort to generate automatic static CMOS custom layouts as well as a designer can do manually. Besides, the layout generation has as main goal to implement circuits optimized to area and timing.

Area optimization means in this case smaller as possible diffusion areas, spacing between rows reduced and routing fully over the cells. On the other hand, timing optimization consists on be able to apply sizing on the transistors to reduce the critical delay of the circuit.

The standard cell approach is widely used because of the predictability. This predictability is offer due to cells are characterized before synthesis. In addition, different versions of each cell are produced to generated several solutions and to achieve a range of solutions. According the demand for area, timing or power in the design, the best cell in the library is used to generate the layout.

In a design in which timing is aimed, fast cells are chosen to be used by the synthesis tool. Other solution to the synthesis is the buffers insertion. When chosen faster gates cannot reduce the critical delay or the efficiency of the optimization is better with the insertion of a buffer in its output, the synthesis tool can modify the gates structure to apply this technique.

Different from the standard cell approach, an automatic full custom tool generates all transistors and connection on demand. Thus, any gate can be generated based on a transistor netlist.

The space of solution related to layout optimization is greater in this methodology due to the possibility to apply other techniques in the layout. A technique very discussed in this work and reported in the literature is the gate sizing.

In order to achieve a good strategy to layout generation, techniques and layout characteristics were searched in the literature. This study is reported in the previous chapters of this work.

In chapter 2 it is presented some basic concepts about layout generation. It is briefly given characteristics about abstraction levels of design starting of a high level design but emphasizing the physical design level. Even when ASIC is aimed, techniques on different abstraction levels can be used in order to generate accurate integrated circuits given the circuit specification.

Chapter 3 shows a study related to design situations where performance are involved. It is presented some cost models and techniques to layout optimization. Efficient area and delay modeling results in accurate techniques to layout optimization.

Three techniques for optimizing layouts are reported in this chapter:

- Gate sizing
- Buffer insertion
- Gate cloning

Several works are reported and the association between gate sizing and buffer insertion obtains the best results. Thus, gate sizing reduces the circuit delay by adjusting the gate sizes and their drive strengths and input capacitances. On the other hand, buffer insertion achieves circuit delay reduction by speeding up the timing critical signals.

Placement and routing techniques are also reported in order to situate the layout generation in other areas of the physical synthesis. Placement algorithms are directly related to routing in such a way it tries to reduce the congestion of wire in whole circuit surface improving complete routing. Besides, routing techniques are developed to find the best as possible connections (timing, area, etc) between points in the circuit and to guarantee that all nets are routed.

In chapter 5 it is presented a strategy to automatic full custom layout generation. As previously reported, the main goal of this strategy is to generate compact and timing optimized layouts as good as a designer can be develop manually.

Parrot Punch is the automatic full custom layout generator developed to validate the proposed strategy. The main characteristics of the proposed technique to layout generation are:

- Generation of any static CMOS gates;
- Supply lines between rows and routing fully over the cell to reduce the area occupied by the circuit;
- Transistors ordering using weight in order to reduce routing congestion;
- Candidate points to each transistor are passed to routing in order to improve best routability;
- Complex gates can be easily generated;
- Folding technique by the integration with a gate sizing tool.

Results obtained from the comparison between the proposed strategy and other known methods to layout generation. First, comparisons between layouts generated by TROPIC3 and Parrot Punch are performed in which area occupation, delay and power consumption are analyzed.

Due to the difference of routing technique, results vary between 33.2% and 47.5% of reduction on the occupied area between layouts generated by Parrot Punch and TROPIC3. In TROPIC3, routing channels are used while Punch Parrot generate the routing over the rows whenever it is possible.

The reduction of the wire length in Parrot Punch layouts and the effort for layout optimization results in better delay of these layouts. Results related to delay of circuits show an average gain of around 22% on TROPIC3 layouts.

Results obtained by the comparison with the standard cell approach shows that gain in area occupation is between 2% and 22%. However, these results can be improved by the development of new algorithms able to reduce the gap between automatic layout generation and full custom layouts.

7.1 Future Works

The proposed strategy can aggregate many other improvements in order to enrich this work. Some of them are following reported.

- Generate transmition gates, tri-state gates and to study the possibility to generate dynamic logic;
- Study a way to choose the best width to transistors given a specific circuit. The tool must be able to analyze the circuit and to detect the best width thinking in terms of occupied area, timing and the folding technique;
- Use of lines with 45% to obtain better results in the technique of slices in which the routing could be done with less penalties compared to the adopted technique;
- Antenna effects (or plasma induced gate oxide damage) must be detected and avoided by an automatic layout generator. In order to guarantee that a circuit works, this problem can be detected and solved in the layout generation time;
- Choose a real design to develop a layout, starting from a high level until the layouts. In this flow, optimization must be done in each one of the abstraction levels. The main idea is to validate a design flow and to guarantee that a layout can be developed and manufactured.

REFERENCES

ALIDINA, M.; MONTEIRO, J.; DEVADAS, S.; GHOSH, A.; PAPAEFTHYMIOU, M. Precomputation-based sequential logic optimization for low power. In: CONFERENCE ON ADVANCED RESEARCH IN VLSI, 16., 1994. **Proceedings. . .** [S.l.: s.n.], 1994. p.430–444.

BASTIAN, F. B. **A new parametrizable transistor folding algorithm applied to an automatic full-custom layout generation tool**. Porto Alegre, 2003. Informal discussion.

CAMPOSANO, R.; WOLF, W. **High-Level VLSI Synthesis**. [S.l.]: Kluwer Academic Publishers, 1991. 390p.

CHEN, W.; HSIEH, C.-T.; PEDRAM, M. Simultaneous Gate Sizing and Fanout Optimization. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2000, San Jose, California. **Proceedings. . .** [S.l.: s.n.], 2000. p.374–378.

CHEN, W.; HSIEH, C.-T.; PEDRAM, M. Simultaneous Gate Sizing and Placement. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.19, n.2, p.206–214, 2000.

CONG, J.; SARRAFZADEH, M. Incremental physical design. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2000, San Diego, California, USA. **Proceedings. . .** New York: ACM Press, 2000. p.84–92.

COUDERT, O. Gate Sizing - A General Purpose Optimization Approach. In: EUROPEAN DESIGN AND TEST CONFERENCE, 1996, Paris, France. **Proceedings. . .** [S.l.: s.n.], 1996. p.214–218.

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, November 1997.

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

CRÉMOUX, S.; AZÉMARD, N.; AUVERGNE, D. Path resizing based on incremental technique. In: ISCAS, 1998, Monterey, USA. **Proceedings. . .** [S.l.: s.n.], 1998.

DASH, R. K.; PRAMOD, T.; VASUDEVAN, V.; RAMAKRISHNA, M. A transistor level placement tool for custom cell generation. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, 13., 2000, New York, NY, USA. **Proceedings...** New York: ACM Press, 2000. v.8, p.3–7.

DETJENS, E.; RUDELL, R.; SANGIOVANNI-VINCCENTELLI, A.; WANG, A. Technology mapping in MIS. In: ICCAD, 1987. **Proceedings...** [S.l.: s.n.], 1987. p.116–119.

FERREIRA, F.; MORAES, F.; REIS, R. LASCA - Interconnect Parasitic Extraction Tool for Deep-Submicron IC Design. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, 13., 2000. **Proceedings...** [S.l.: s.n.], 2000. p.327–332.

FISHBURN, J. P. Clock Skew Optimization. **IEEE Transactions on Computers**, Washington, DC, USA, v.39, n.7, p.945–951, 1990.

FRAGOSO, J. **WTROPIC - um Gerador Automático de Macro Células CMOS Acessível via WWW**. 2001. 89p. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

GAJSKI, D.; DUTT, N.; WU, A.; LIN, S. **High Level Synthesis - Introduction to Chip ans System Design**. [S.l.]: Kluwer Academic Publishers, 1992. 359p.

GEREZ, S. H. **Algorithms for VLSI Design Automation**. [S.l.]: John Wiley and Sons, 1998. 326p.

GÜNTZEL, J. L. **Geração de circuitos utilizando matrizes de células pré-difundidas**. 1993. 174p. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

GÜNTZEL, J. L. **Functional Timing Analysis of VLSI Circuits Containing Complex Gates**. 2000. Thesis (Doctor in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

GÜNTZEL, J. L.; FREITAS, D.; FLORES, A.; REIS, R. Analysis of a Sea-of-Cells Assignment Too. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 8., 1993, Campinas, São Paulo, Brazil. **Anais...** [S.l.: s.n.], 1993. p.XII.13–XII.15.

GÜNTZEL, J. L.; REIS, R.; FLORES, A.; JOHANN, M. A Novel Approach for ASIC Layout Generation. In: MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, 38., 1995, Rio de Janeiro. **Proceedings...** New York: IEEE Circuits and Systems Society, 1995. v.2, p.719–794.

GÜNTZEL, J. L.; RIBAS, R. P.; REIS, R. MARCELA - Uma Nova Abordagem para Pré-difundidos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 1991, Belo Horizonte. **Anais...** [S.l.: s.n.], 1991. p.534–543.

GUPTA, A.; HAYES, J. P. CLIP: integer-programming-based optimal layout synthesis of 2d cmos cells. In: IEEE TRANSACTIONS ON DESIGN AUTOMATION OF ELECTRONIC SYSTEMS, 2000, New York, NY, USA. **Proceedings...** New York: ACM Press, 2000. v.5, p.510–547.

HACHTEL, G. D.; SOMENZI, F. **Logic Synthesis and Verification Algorithms**. [S.l.]: Kluwer Academic Publishers, 1996. 564p.

HASHIMOTO, A.; STEVENS, J. Wire routing by optimizing channel assigment within large apertures. In: DESIGN AUTOMATION CONFERENCE, 1971. **Proceedings...** [S.l.: s.n.], 1971. p.155–159.

HENTSCHKE, R. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. 2002. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

IMAN, S.; PEDRAN, M. POSE: power optimization and synthesis environment. In: DESIGN AUTOMATION CONFERENCE PROCEEDINGS, 33., 1996, New York, NY, USA. **Proceedings...** New York: ACM Press, 1996. p.21–26.

JACOBS, E. T. A. F. **Power Dissipation and Timing in CMOS Circuits**. 2001. 164p. Thesis (Doctor in Computer Science) — Technische Universiteit Eindhoven.

JACOBS, E. T. A. F.; BERKELAR, M. Gate Sizing Using a Statistical Delay Model. In: DESIGN, AUTOMATION AND TEST IN EUROPE, 2000, Paris, France. **Proceedings...** New York: ACM Press, 2000. p.283–291.

JIANG, Y.; SAPATNEKAR, S.; BANJI, C.; KIM, J. Interleaving Buffer Insertion and Transistor Sizing into a Single Optimization. **IEEE Transactions On Very Large Scale Integration Systems**, [S.l.], v.6, n.4, p.625–633, 1998.

JIANG, Y.; SAPATNEKAR, S.; BANJI, C.; KIM, J. Combined Transistor Sizing with Buffer Insertion for Timing Optimization. **IEEE Custom Integrated Circuits Conference**, [S.l.], p.605–608, 1998.

JOHANN, M. **Roteamento sobre células transparentes**. 1994. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

JOHANN, M. **Search of the Euler Path with Weight Associated to Transistors**. Porto Alegre, 2003. Informal discussion.

JOHANN, M.; REIS, R. MARTE - MAze RouTer Environment. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 8., 1993, Campinas, São Paulo, Brazil. **Anais...** [S.l.: s.n.], 1993. p.XII.37–XII.39.

KANEKO, M.; JIALIN, T. Concurrent cell generation and mapping for CMOS logic circuits. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.247–252.

KIM, J.; DU, D. H. C. Performance Optimization by Gate Sizing and Path Sensitization. **IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.17, n.5, p.459–462, 1998.

KIM, J.; KANG, S. M. An Efficient Transistor Folding Algorithm for Row-Based CMOS Layout Design. In: DESIGN AUTOMATION CONFERENCE, 1997, Anaheim, California, USA. **Proceedings...** New York: ACM Press, 1997. p.456–459.

KONG, J.-T.; HUSSAIN, S. Z.; OVERHAUSER, D. Performance Estimation of Complex MOS Gates. **IEEE Transactions on Circuits and Systems - Fundamental Theory and Applications**, [S.l.], v.44, n.9, p.785–795, 1997.

KONG, J.-T.; HUSSAIN, S. Z.; OVERHAUSER, D. Performance Estimation of Complex MOS Gates. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.17, n.5, p.459–462, 1998.

KU, D.; MICHELI, G. D. **High Level Synthesis Of ASICs Under Timing and Synchronization Constraints**. [S.l.]: Kluwer Academic Publishers, 1992. 294p.

LAZZARI, C.; DOMINGUES, C. V.; GÜNTZEL, J. L.; REIS, R. A. L. A New Macrocell Generation Strategy for Three Metal Layer CMOS Technologies. In: VLSI-SOC, 2003, Darmstadt, Germany. **Proceedings...** [S.l.: s.n.], 2003. p.143–147.

LEE, C. An Algorithm for Path Connections and its Applications. **IRE Transactions on Electronic Computers**, New york, v.EC-10, p.346–365, Septembre 1961.

LEFEBVRE, M.; MARPLE, D.; SECHEN, C. The Future of Custom Cell Generation in Physical Synthesis. In: DESIGN AUTOMATION CONFERENCE, 34., 1997, Anaheim, California, USA. **Proceedings...** New York: ACM Press, 1997. p.446–451.

LIN, H.-R.; HWANG, T. Dynamical Identification of Critical Paths for Iterative Gate Sizing. In: ICCAD, 1994. **Proceedings...** [S.l.: s.n.], 1994. p.481–484.

LOPEZ, A.; LAW, H. S. A dense Gate Matrix Layout Method for MOS VLSI. **IEEE Transactions on Electron Devices**, [S.l.], v.ED-27, n.8, p.1671–1675, 1980.

LOWE, K. S.; GULAK, P. G. A Joint Gate Sizing and Buffer Insertion Method for Optimizaing Delay and Power in CMOS and BiCMOS Conbinational Logic. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.17, n.5, p.419–434, 1998.

LUBASZEWSKI, M. **Geração Automática de Lógica Aleatória Utilizando a Metodologia TRANCA**. 1990. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

MAHESHWARI, N.; SAPATNEKAR, S. S. Gate Size Optimization for Row-based Layouts. In: MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, 38., 1995. **Proceedings...** [S.l.: s.n.], 1995.

MAZIASZ, J. R.; HAYES, J. P. Layout optimization of CMOS functional cells. In: DESIGN AUTOMATION CONFERENCE, 1987. **Proceedings...** [S.l.: s.n.], 1987. p.551–554.

MAZIASZ, J. R.; HAYES, J. P. Exact width and height minimization of CMOS cells. In: DESIGN AUTOMATION CONFERENCE, 1991. **Proceedings...** [S.l.: s.n.], 1991. p.487–483.

MONTEIRO, J.; RINDERKNECHT, J.; DEVADAS, S.; GHOSH, A. Optimization of combinational and sequential logic circuits for low power using precomputation. In: CONFERENCE ON ADVANCED RESEARCH IN VLSI, 16., 1995. **Proceedings. . .** [S.l.: s.n.], 1995. p.430–444.

MORAES, F. **TRAGO - Síntese Automática de Leiaute para Circuitos em Lógica Aleatória**. 1990. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

MORAES, F. **Synthèse Topologique de Macro-Cellules en Technologie CMOS**. 1994. Thesis (Doctor in Computer Science) — Université Montpellier II, France.

MORAES, F.; REIS, R.; LIMA, F. **An Efficient Layout Style for Three-Metal CMOS Macro-Cells**. [S.l.]: Chapman & Hall, 1997. 415–426p.

MORAES, F.; VELASCO, A. J. Deterministic versus Non-Deterministic Placement Algorithms for Automatic Layout Synthesis Tools. In: DCIS, 2000. **Proceedings. . .** [S.l.: s.n.], 2000.

NAKAGAKI, T.; YAMADA, S.; FUKUNAGA, K. Fast optimal algorithm for CMOS functional cell layout based on transistor reordering. In: ISCAS, 1992. **Proceedings. . .** [S.l.: s.n.], 1992. p.1697–1700.

NEVES, J. L.; FRIEDMAN, E. G. Optimal Clock Skew Scheduling Tolerant to Process Variations. In: DESIGN AUTOMATION CONFERENCE, 33., 1996. **Proceedings. . .** [S.l.: s.n.], 1996. p.623–628.

NGUYEN, D.; DAVARE, A.; ORSHANSKY, M.; CHINNERY, D.; THOMPSON, B.; KEUTZER, K. Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization. In: INTERNATIONAL SYMPOSIUM ON POWER ELECTRONICS AND DESIGN, 2003, Seoul, Korea. **Proceedings. . .** New York: ACM Press, 2003. p.158–163.

ONG, C.; LI, J.; LO, C. GENAC - an automatic cell synthesis tool. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION CONFERENCE, 26., 1989, Las Vegas, Nevada, USA. **Proceedings. . .** New York: ACM Press, 1989. p.239–244.

OTTEN, R.; CAMPOSANO, R.; GROENEVELD, P. Design Automation for Deep-submicron - present and future. In: DESIGN, AUTOMATION AND TESTE IN EUROPE CONFERENCE AND EXHIBITION, 2002, Washington, DC, USA. **Proceedings. . .** [S.l.]: IEE Computer Society, 2002. p.650–657.

PRADHAN, D.; CHATTERJEE, M.; SWARNA, M.; KUNZ, W. Gate-level synthesis for low-power using new transformations. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, 1996. **Proceedings. . .** [S.l.: s.n.], 1996. p.297–300.

RABAEY, J. M. **Digital Integrated Circuits - A Design Perspective**. [S.l.]: Prentice Hall, 1996. 702p.

RAVINDRAN, K. **Automatic Clock Skew Scheduling**. Available at <http://www.eecs.berkeley.edu/∼kaushikr/ee219b_paper.pdf>. Visited on November, 2003, Berkeley, USA.

REIS, A. I. **Geração de Células Transparentes**. 1993. 128p. Dissertation (Master in Computer Science) — UFRGS - Universidade Federal do Rio Grande do Sul, Porto Alegre.

REIS, A. I.; REIS, R. A. L. TRAMOII - Proposta para um Gerador de Leiaute Baseado em Células para Circuitos CMOS Digitais com Dois Níveis de Metal. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 1991, Jaguariúna. **Anais. . .** [S.l.: s.n.], 1991. p.90–99.

REIS, A. I.; REIS, R. A. L.; AUVERNE, D.; ROBERT, M. Library Free Technology Mapping. **VLSI: Integrated Systems on Silicon, IFIP TC10 WG10.5 International Conference in Very Large Scale Integration**, [S.l.], p.303–314, August 1997.

REIS, R. A. **TESS - Evaluates Topologique Predictif pour la Génération Automatique des Plans de Masse de Circuits VLSI**. 1983. Thesis (Doctor in Computer Science) — Institut Polytechnique de Grenoble, France.

REIS, R. A New Standard Cell CAD Methodology. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1987, Portland. **Proceedings. . .** [S.l.: s.n.], 1987. p.385–388.

ROY, K.; MUKHOPADHYAY, S.; MEIMAND, H. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. **Proceedings of the IEEE**, New York, v.91, n.2, p.305-327, Feb.2003.

SANTOS, C.; WILKE, G.; LAZZARI, C.; GÜNTZEL, J. L.; REIS, R. A Transistor Sizing Method Applied to an Automatic Layout Generation Tool. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2003. **Proceedings. . .** [S.l.: s.n.], 2003. p.303–307.

SAPATNEKAR, S.; RAO, V.; VAIDYA, P. A Convex Optimization Approach to Transistor Sizing for CMOS Circuits. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1991. **Proceedings. . .** [S.l.: s.n.], 1991. p.482–485.

SARRAFZADEH, M.; BOZORGZADEH, E.; KASTNER, R.; SRIVASTAVA, A. Design and analysis of physical design algorithms. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2001, Sonoma, California, USA. **Proceedings. . .** New York: ACM Press, 2001. p.82–89.

SATO, K.; KAWARABAYASHI, M.; EMURA, H.; MAEDA, N. Post-Layout Optimization for Deep Submicron Design. In: DESIGN AUTOMATION CONFERENCE, 33., 1996, Las Vegas, Nevada, USA. **Proceedings. . .** [S.l.: s.n.], 1996. p.740–745.

SERDAR, T.; SECHEN, C. Automatic datapath tile placement and routing. In: DESIGN, AUTOMATION AND TEST IN EUROPE, 2001. **Proceedings. . .** [S.l.: s.n.], 2001. p.13–16.

SHAMS, A. M.; BAYOUMI, M. A. A novel high-performance CMOS 1-bit full-adder cell. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, [S.l.], v.47, p.478–481, May 2000.

SUNDARARAJAN, V.; SAPATNEKAR, S.; PARHI, K. K. Fast and Exact Transistor Sizing Based on Iterative Relaxation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.21, n.5, p.568–581, May 2002.

SUTHERLAND, I.; SPROULL; HARRIS. **Logic Effort - Designing Fast CMOS Circuits**. [S.l.]: Morgan-Kaufman, 1999. 239p.

SYLVESTER, D.; KAUL, H. Power-driven challenges in nanometer design. **Design and Test of Computers IEEE**, [S.l.], v.18, p.12–21, November 2001.

TAVARES, R. **A method for Routing Circuits before the Layout Generation**. Porto Alegre, 2003. Informal discussion.

UEBEL, L. F. **Verifica cão de Timing em Circuitos VLSI CMOS Digitais**. 1995. 222p. Dissertation (Master in Computer Science) — Instituto de Informática, UFRGS, Porto Alegre.

UEHARA, T.; CLEEMPUT, W. Optimal Layout of CMOS Functional Arrays. **IEEE Transactions on Computes**, [S.l.], v.C-30, n.5, p.305–312, May 1981.

WANG, C. H. An optimal transistor-chaining algorithm for CMOS cell layout. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1989. **Proceedings...** [S.l.: s.n.], 1989. p.344–347.

WANG, C. H. An optimal transistor-chaining algorithm for CMOS cell layout. **IEEE Transactions on Computer-Aided Design**, [S.l.], p.781–786, 1990.

WANG, C. H. An efficient layout style for two-metal CMOS leaf cells and its automatic synthesis. **IEEE Transactions on Computer-Aided Design**, [S.l.], p.410–423, 1993.

WANG, Q.; VRUDHULA, S. Multi-level logic optimization for low power using local logic transformations. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1996. **Digest of Technical Papers...** [S.l.: s.n.], 1996. p.270–277.

WEINBERGER, A. Large scale integration of MOS complex logic: a layout method. **IEEE Journal of Solid State Circuits**, [S.l.], p.182–190, December 1967.

WESTE, N.; ESRAGHIAN, K. **Principles of CMOS VLSI Design - A Systems Perspective**. [S.l.]: Addison-Wesley Publishing Company, 1993. 317–357p.

WILKE, G.; GüNTZEL, J. L.; BYSTRONSKI, M.; PINTO, A. C.; REIS, R. Finding the Critical Delay of Combinational Blocks by Floating Vector Simulation and Path tracing. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 15., 2002. **Proceedings...** [S.l.: s.n.], 2002. p.277–282.

WIMMER, S.; PINTER, R. Y.; FELDMAN, J. A. Optimal chaining of CMOS transistors in a functional cell. **IEEE Transactions on Computer-Aided Design**, [S.l.], p.795–801, 1987.

APPENDIX A AUTOMATIC LAYOUT GENERATION DESIGN RULES

Design rules are a set of rules used in the layout generation. They are basically minimal values to layers, metal enclosures to contacts and vias and minimal spacing between identical layers. All design rules used in the Parrot Punch layout generation is described below.

S1P1P1	Minimum Gate Spacing,
S1DFP1	Minimum POLY1 spacing to DIFF,
E1P1DF	Minimum POLY1 extension of GATE,
E1DFP1	Minimum DIFF extension of GATE,
E1DNP1	Minimum NDIFF extension of GATE when butted to PDIFF,
E1DDP1	Minimum PDIFF extension of GATE when butted to NDIFF,
W2CT	Fixed CONT size,
E1M1CT	Minimum MET2 enclosure of CONT,
E1DFCT	Minimum DIFF enclosure of CONT,
E1P1CT	Minimum POLY1 enclosure of CONT,
S1CTP1	Minimum DIFFCON spacing of GATE,
S1CTDF	Minimum CONT spacing of DIFF,
W2VI	Fixed VIA size,
E1M1VI	Minimum MET1 enclosure of VIA,
E1M2VI	Minimum MET2 enclosure of VIA,
S1M2M2	Minimum MET2 spacing to MET2,
W2P1	Minimum GATE length,
W2DF	Minimum GATE width,
W2V2	Fixed VIA2 size,
S1M1M1	Minimum MET1 spacing to MET1,
E1M2V2	Minimum MET2 enclose of VIA2,
S1CTCT	Minimum CONT spacing,
E1INDF	Minimum NPLUS extension of DIFF,
E1IPDF	Minimum PPLUS extension of DIFF,
E1WNDP	Minimum NTUB enclosure of PDIFF,
S2M2M2	Minimum MET2 spacing to WIDE_MET2,
W1M1	Minimum MET1 width,
W1M2	Minimum MET2 width,

W1M3	Minimum MET3 width,
W1M4	Minimum MET4 width,
W1M5	Minimum MET5 width,
W1M6	Minimum MET6 width,
W1M7	Minimum MET7 width,
W1M8	Minimum MET8 width,
W1M9	Minimum MET9 width,
W1M10	Minimum MET10 width,
S1DFDF	Minimum DIFF spacing to DIFF,
S1DNWN	Minimum NDIFF spacing to NTUB,
S1VIVI	Minimum VIA spacing to VIA,
S1V2V2	Minimum VIA2 spacing to VIA2,
S1M3M3	Minimum MET3 spacing to MET3,
E1M3V2	Minimum MET3 enclosure of VIA2,
E1M3V3	Minimum MET3 enclosure of VIA3,
W2V3	Fixed VIA3 size,
E1M4V3	Minimum MET4 enclosure of VIA3,
S1M4M4	Minimum MET4 spacing to MET4,
W2V4	Fixed VIA4 size,
E1M4V4	Minimum MET4 enclosure of VIA4,
S2P1P1	Minimum GATE spacing to GATE,
E1M5V4	Minimum MET5 enclosure of VIA4,
S1M5M5	Minimum MET5 spacing to MET5,
W2V5	Fixed VIA5 size,
S1IPIP	PPLUS spacing to PPLUS,
S1ININ	NPLUS spacing to NPLUS,
W2V6	Fixed VIA6 size,
W2V7	Fixed VIA7 size,
W2V8	Fixed VIA8 size,
W2V9,	Fixed VIA9 size,
W2V10	Fixed VIA10 size

APPENDIX B GERAÇÃO AUTOMÁTICA DE LEIAUTES DE CIRCUITOS CMOS ESTÁTICOS VISANDO DIMINUIÇÃO DE ATRASO E CONSUMO

Ferramentas de CAD (do inglês, Computer Aided Design) são usados desde o início da microeletrônica. Estas ferramentas têm fundamental importância devido à necessidade de obtenção de produtos competitivos no mercado. Como novas tecnologias estão disponíveis cada vez mais freqüentes, é indispensável o uso de ferramentas de CAD no desenvolvimento de circuitos.

Outro ponto de influência no uso de ferramentas de CAD está na complexidade de sistemas VLSI. Pequenos circuitos comerciais podem conter centenas ou milhares de transistores, tornando difícil o desenvolvimento manual.

Em circuitos estado-da-arte e nas próximas gerações, as geometrias tornam-se menores, freqüência de relógio aumentam e interconexões **on-chip** tornam-se mais importantes (CONG; SARRAFZADEH, 2000). Ainda, problemas em síntese física estão tornando-se mais complexos e ferramentas de EDA (do inglês, Electronic Design Automation) são essenciais para resolver problemas em projetos atuais (SARRAFZADEH et al., 2001).

Em tecnologias de 0.13μm ou menores, circuitos integrados são mais susceptíveis a problemas de antena e eletro-migração. Lidar com estes problemas requer cuidadoso planejamento (OTTEN; CAMPOSANO; GROENEVELD, 2002). Isto enfatiza a necessidade de ferramentas de EDA aptos para implementar e validar circuitos integrados.

Tradicionalmente, projetos a nível físico baseiam-se em bibliotecas de células. Bibliotecas de células oferecem alguma previsibilidade ao projeto devido a células caracterizadas antes da síntese. Entretanto, Bibliotecas de células possuem limitado número de células. Isto impõem algumas restrições na síntese de leiaute. Ainda, diferentes versões de cada célula são exigidas com o objetivo de atender características de atraso e potência, aumentando o número de elementos em bibliotecas para centenas de células.

Uma alternativa para a geração de leiaute baseada em células é a geração automática parametrizada. Nesta metodologia não são usadas células de uma biblioteca. Cada elemento (transistores, contatos e conexões) é gerado baseado em padrões de leiaute que são intrinsecamente programados em algoritmos. Ainda, Geração automática parametrizável de leiautes é flexível para criar leiautes otimizados de acordo com a posição onde estão inseridos.

A geração de leiaute aparece com a matriz Weinberger (WEINBERGER, 1967). Outros trabalhos relacionado com geração automática de leiautes são apresentados por Lopez e Law (LOPEZ; LAW, 1980) e Uehara e Cleemput (UEHARA; CLEEMPUT, 1981).

Os primeiros introduzem um estilo de leiaute conhecido como *gate matrix*, enquanto os últimos apresentam o estilo de leiaute chamado *linear matrix*. Ambos os trabalhos foram originalmente desenvolvidos tendo em mente a geração de células. Gerados de leiaute atuais são baseados no estilo linear matrix e são aptos a gerar módulos com centenas de transistores.

Na geração de leiautes parametrizáveis, ferramentas de estimativa e análise devem oferecer a previsibilidade. Uma ferramenta de análise deve lidar com estimativas de leiautes e gerar informações acuradas sobre o circuito.

Em (GÜNTZEL, 2000) são apresentados importantes conceitos sobre análise de atraso em circuitos VLSI. Güntzel relata que a verificação de atraso tem como objetivo determinar se restrições de atraso impostas ao projeto podem ser satisfeitas ou não. Mais especificamente, verificação de atraso diz respeito a estimar o atraso crítico do circuito a freqüência máxima de operação.

A precisão na verificação de atraso é completamente dependente da precisão dos modelos de circuitos usados. Por modelos de circuitos entende-se não somente o modelo de atraso físico usado para quantificar o atraso de cada componente, mas também os modelos para computar os componentes de atraso no circuito e seus atrasos.

A Análise de atraso associada à geração de leiaute pode aumentar a eficiência de projetos de circuitos. Além disso, a análise de atraso pode ser usada para prover otimização de leiaute através de técnicas de dimensionamento de transistores e inserção de buffers.

O espaço de soluções relacionado à otimização de circuitos é muito grande nesta metodologia devido à possibilidade de aplicarias várias técnicas ao leiaute. Dimensionamento de transistores é a técnica para otimizar cada transistor individualmente no circuito, baseado em certas características. Estas características referem-se a largura de transistores, relações entre as portas lógicas e suas implicações no atraso do circuito.

Inserção de buffers consiste em inserir buffers no circuito com o objetivo de distribuir a capacitância de determinada porta lógica para os buffers, reduzindo o atraso de um caminho.

Santos et al. apresenta em (SANTOS et al., 2003) uma ferramenta de dimensionamento de transistores apto a realizar otimização em atraso. A técnica utiliza uma ferramenta de análise de atraso para estimar os caminhos longos e aplicar dimensionamento de transistores nos caminhos críticos do circuito.

O número de transistores está diretamente relacionado com a potência consumida do circuito. Assim, a otimização lógica objetivando redução no número de transistores associada à geração de leiaute apta a gerar um grande número de funções complexas pode reduzir no consumo de potência de um circuito.

A otimização de leiaute é muito importante no desenvolvimento de circuitos integrados. Nenhum circuito é projetado sem análises do ambiente onde ele será utilizado. Dependendo da aplicação, características como área, desempenho e consumo de potência são estudados para gerar melhores resultados.

Neste trabalho é apresentada uma pesquisa sobre síntese física e a implementação de um gerador automático parametrizável de leiaute chamado de Parrot Punch. A pesquisa está relacionada com estratégias de leiaute e os efeitos na ocupação de área e no atraso do circuito. Como resultado é apresentada uma nova estratégia para a geração de leiaute na qual redução em área e atraso é objetivada.

A estratégia de geração de leiaute aplica técnicas de otimização de atraso por meio da integração com uma técnica de dimensionamento de transistores. Isto é realizado com a aplicação de *folding* que dimensiona os transistores.

As principais características da técnica de geração de leiautes proposta são as seguintes:

- Geração de qualquer lógica CMOS estática;
- Linhas de alimentação entre as bandas e roteamento sobre as células com o objetivo de reduzir a área ocupada pelo circuito;
- Ordenamento de transistores usando pesos para reduzir o congestionamento sobre as células;
- Portas complexas podem ser facilmente geradas;
- Uso da técnica de folding pela integração com uma ferramenta de dimensionamento de transistores.

Comparações entre leiautes gerados pela ferramenta TROPIC3 e Parrot Punch foram realizados na qual a ocupação em área, atraso e consumo de potência foram comparados. Devido as diferenças nas técnicas de roteamento, resultados variam 33.2% e 47.5% de redução na área ocupada entre os leiautes gerados pelo Parrot Punch em relação aos gerados pelo TROPIC3. No TROPIC3, canais de roteamento são utilizados enquanto Parrot Punch gera circuitos com roteamento sobre as bandas sempre que possível.

A redução no comprimento das conexões em leiautes gerados pelo Parrot Punch e o esforço pela otimização de leiautes resultam em melhor atraso nestes circuitos. Resultados obtidos relacionados ao atraso dos circuitos mostram um ganho médio de 22% sobre os circuitos gerados pelo TROPIC3.

Resultados preliminares obtidos pela comparação com a metodologia Standard Cell mostra ganho em ocupação em área de 2% a 22%. Apesar disso, estes resultados podem ser melhorados pelo desenvolvimento de novos algoritmos aptos a reduzir a distância entre a geração automática de leiautes e layout totalmente parametrizáveis desenvolvidos manualmente.

Como trabalhos futuros, a proposta apresentada deve agregar outras características com o objetivo de enriquecer o trabalho. Alguns deles são:

- Geração de portas de transmissão, portas tri-state e o estudo da possibilidade de geração de lógica dinâmica;
- Estudar uma maneira de escolher a melhor largura de transistores dado um circuito específico. A ferramenta deve estar apta a analisar o circuito e detectar a melhor largura pensando em relação a área ocupada, atraso e a utilização da técnica de folding;
- Utilização de linhas de 45^o para obter melhores resultados no roteamento do circuito;
- Efeitos de antena devem ser anulados na geração de leiautes. Para que se garanta o funcionamento de um circuito, este problema deve ser detectado e resolvido em tempo de geração do leiaute.