Design of CMOS Cell Libraries for Minimal Leakage Currents

Master's Thesis

by

Jacob Gregers Hansen, s973741

August 13th., 2004

Supervisor: Flemming Stassen

Project number: 55
Informatics and Mathematical Modelling
Computer Science and Engineering
Technical University of Denmark

Preface

Preface

This report is part of the results from the master's thesis project 'Design of CMOS Cell Libraries for Minimal Leakage Currents' conducted at Informatics and Mathematical Modelling (IMM), Computer Science and Engineering division (CSE), Technical University of Denmark (DTU) from February to August 2004.

This project was conducted as a part of three independent, but collaborative master's thesis. The original idea for this work was conceived by Peter Østergaard Nielsen from Vitesse Semiconductor Corporation, Denmark.

I would like to thank my colleagues Martin Hans and Michael Kristensen for inspiring cooperation. Further I would like to thank Alberto Nannarelli for valuable insights and the administrative staff of IMM for helping me speed up the project work.

Jacob Gregers Hansen, Copenhagen 2004.

Abstract

Abstract

Leakage due to scaling down CMOS device sizes will be the major power consumption source in cell based IC design in a few years. This work addresses the problem of this leakage, investigating the possibilities of utilizing alternative logic families instead of static CMOS for the creation of a low leakage cell library. For this purpose, MTCMOS, CPL and Domino logic are investigated for leakage characteristics and are found unusable for low leakage design.

Using cell libraries of small logic cells for IC design is found to be the major reason for much of the leakage. Synthesizing without cell boundaries by building larger cells reduces the leakage problem greatly. A new synthesis flow and cell library is proposed.

Keywords: Low leakage CMOS, CPL, Domino, MTCMOS, MacroCMOS, Synthesis for low leakage design.

Resumé

Lækstrømme forårsaget af de evigt krympende transistorstørrelser vil om få år være den største kilde til effektforbrug i CMOS cellebaseret IC design. Mulighederne for at anvende andre logikfamilier end statisk CMOS til design af et lavlæk cellebibliotek bliver i denne opgave undersøgt. Tre mulige kandidater, MTCMOS, Domino og CPL, bliver undersøgt og findes ubrugelige til lavlækdesign.

Anvendelsen af cellebiblioteker af små, logiske celler findes at være årsagen til meget af lækket. Der foreslås i stedet en ændring af synteseværktøjer mod at syntetisere designs uden grænser mellem cellerne ved at sammenbygge logikken til større celler, under anvendelse af et foreslået cellebibliotek.

Stikord: Lavlæk CMOS, CPL, Domino, MTCMOS, MacroCMOS, Syntese af lavlæk design.

Contents

CHAPTER 1

INTRODUCTION

Contents

The aim of this chapter is to describe the problem that this work intends to solve. The development of MOS transistors, synthesis tools and cell libraries is described to introduce the origin of the leakage current problem. Possible solutions to the leakage current problem are presented forming the basis for the objectives set in this work. Last, an overview of this report is given.

1.1 Invention of MOSFET transistors

For the past two decades Complementary Metaloxide Silicon (CMOS) technology has played an ever more important role in the integrated circuits industry. Not that MOS field-effect transistor (MOSFET) technology is new. It was already proposed in 1925 by J. Lilienfield[1], but problems with materials prevented production attempts of MOSFET transistor. The research of MOSFETs gave birth to bipolar transistors, which were easier to produce and became the dominant transistor technology for decades.

Further research in silicon processing yielded the silicon planar process, which made MOSFET devices possible around 1960. Single-polarity p-type transistors were favored until the emergence of nMOS silicon-gate technology in 1971. The first patents of CMOS gates were filed in 1967 by Fairchild Semiconductor Research and Development patenting the CMOS concept and three basic gates: the inverter, the nand-gate and the nor-gate.

Though more complex to design, CMOS devices had one great advantage: low power consumption. The first CMOS inverters dissipated nanowatts of power compared with milliwats for pMOS or bipolar devices. So, CMOS was initially used for low power devices such as watches. In the pre-LSI days when circuitry built with CMOS technology consumed much more area than pMOS or bipolar circuitry, CMOS was primarily used where power and not area was the critical parameter. But as device sizes shrunk and technology improved to support larger chip sizes, more circuitry could be built into every chip, diminishing the area concerns and raising the need for low power circuitry, especially since the device density skyrocketed. CMOS transistors and the static CMOS logic family were the answer, and they still are. Static CMOS is today the best preferred technology for IC design in terms of power dissipation, area and operational speed.

1.2 Synthesis of cell based designs

Designing circuits in the early days was done by skilled full-custom hardware designers, laying out as much as thousands of transistors per working day. But as the number of devices per chip grew exponentially over time, this design flow was no longer feasible. Automation was needed, and the first synthesis and place & route tools saw the day.

Synthesis tools have come a long way from mere scripting of small logic blocks to the powerful synthesis tools of today, capable of synthesizing abstract, high-level coded designs into RTL-level netlists of predefined logic cells and their interconnects. These synthesis tools include algorithms for numerous optimization techniques enabling the automation of a great variety of optimizations for power dissipation, area, operational speed etc. With numbers of devices approaching hundreds of millions on a chip, optimizations such as retiming or clock gating have become infeasible to do manually. Synthesis tools have truly become indispensable.

The modern synthesis tools and synthesis methodologies originate from a time when the main problem was utilizing the chip surface's ever growing potential for devices efficiently. For this purpose, higher level hardware description languages such as Verilog and VHDL were invented and synthesis tools were created. The task of synthesizing designs written in these languages is done by breaking the problem into smaller problems until a level of boolean functions on a RTL level is reached. The synthesis tool then matches the boolean expression against boolean functions supplied by a cell library consisting of a variety of cells implementing boolean functions in logic hardware. The synthesis tool imports timing, power and area specifications from the cell library and optimizes the design according to cost functions defined by the design engineer. Hereby, very large designs can be implemented and optimized to a certain extent without the design engineer ever laying out a single transistor.

1.2.1 Cell libraries

This design flow requires a cell library of predefined logic circuits implementing a selected range of logic functions, characterized for power, area and timing. Further, a model for incorporating wires between internal nodes is required for a complete timing verification. Defining the set of logic functions the cell library is offering, and accurately modelling and simulating electrical characteristics of logic gates of transistors is the job of the cell library designer.

Using cells built with static CMOS logic eases the workload of both the cell library designer and the synthesis tool as static CMOS cells are stable and predictable enough to be cascaded like putting Lego-blocks together. Further, if a logic function does not match the wanted boolean expression entirely, a few inverters or smaller gates are fitted in regards to timing requirements. If a path is too slow, a cell with higher drive is put instead of the slower one. Static CMOS cells will always work, but might be slower than expected. Designing conservatively for the worst case will eliminate most errors.

Connecting blocks like Lego-blocks has a few costs, though. A small area overhead in comparison with full-custom design must be expected since not all functions are present in the cell library. Further, dynamic power consumption suffers a bit from this procedure due to the area or logic functional overhead. Nonetheless, architectural design decisions and incorporating new optimization algorithms have reduced the power consumption, and area is no longer a critical parameter due to the process developments.

All in all, using a synthesis tool and a static CMOS cell library is a very efficient way to build VLSI systems with minimum penalties. But, a problem has been lurking in the future and will soon become the major problem of the integrated circuits industry the years to come. The problem is power consumption due to leaking devices. This consumption does not depend on activity or operational speed, but rather the shear number of leaking devices in the circuitry. And that number is increasing exponentially.

1.3 The problem of leakage currents

Leaking devices will soon be the major concern of the IC design industry. As device sizes have been scaled down to keep up the exponential growth of device density and to enable lower supply voltages, reducing the dynamic power dissipation, MOSFET transistors are beginning to conduct current when they are in 'off'-mode. MOS transistors have always conducted a bit of current in their 'off'-mode, but until recent years the problem has not been big enough to get worried about. When the industry embraces the new sub-100nm technologies though, these currents will be the reason for almost half of the total power dissipation in an integrated circuit.

Lowering power consumption is critical for further improvements for operational speed in high-speed applications and for low-power consumption in battery-supplied applications such as cellular phones. Reducing the unwanted currents, called *leakage currents* or simply *leakage*, is vital for further growth in IC designs.

Leakage has two components: Subthreshold leakage and gate-oxide leakage. Subthreshold leakage consists of source-drain currents when the transistor is supposed to be non-conducting. These currents are now flowing through the substrate of the transistors due to effects near the active regions of transistors that heavily depend on the length of the transistor gate. Gate-oxide leakage comes from currents tunnelling through the very thin oxide layer between gate and source, drain or bulk. Clearly, both types of leakage depend on the device sizes, and also depend on the voltages at the terminals. Further, altering the doping of the substrate, the threshold voltage, V_{th}, can be changed enabling the design of low leakage transistors with higher V_{th} values. Though, high-V_{th} have weaker drive and will deteriorate the speed of the circuitry.

1.4 Possible solutions

A leaking transistor can be perceived as a switch with a parallel resistor. Putting the switch in 'off' mode, the resistor keeps on drawing currents past the switch (see Figure 1.1). Hereby a integrated circuit becomes a vast array of parallel resistors leaking between the voltage rails.

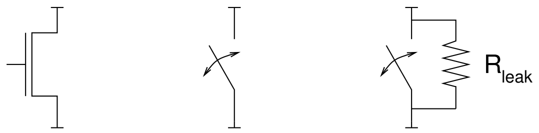


Figure 1.1: *The transistor as a switch and as a leaking device.*

With this picture in mind, the problem of current day synthesis tools and small static CMOS cells become clear. Using large numbers of small gates containing very few transistors each is the cause of the problem. This is the manner in which the number of leaking resistors (transistors) is maximized and the resistance on each path is reduced to the minimum. The resistance can be increased by using high-V_{th} low leakage transistors, but these transistor reduce the speed of the circuitry.

A solution to this problem could be to go back to the decision of selecting static CMOS as logic family. If devices had been leaking as much two decades ago as they will do within a few years, small cell static CMOS might not have been selected as the logic family of the future. Instead other interesting logic families might have prevailed. In this work different logic families will be discussed and two, Domino logic and Complementary Pass-transistor Logic, have been selected for closer low leakage evaluation.

Another solution is found in a characteristic of leakage currents: As the leakage power dissipation is not dependent on activity, but is an ever present power dissipation source, cutting off power to inactive regions may save quite large fractions of the total power dissipation. This is especially interesting for applications that do only operate in a small fraction of the time. Therefore, this concept is taken under evaluation in this work.

The third solution presented here came through a study of transistor characteristics. Connecting transistors in series (stacking), which will be shown to decrease leakage considerably, will be proven to be a good solution to the problem. Building larger logic blocks on-the-fly in the synthesis process and including optimization algorithms for leakage reductions can yield very large savings in the power budget. This approach will require changes in the synthesis process and complete redesign of current cell libraries. Changes to the synthesis process of today and a new cell library are proposed in this work.

A fourth, and very well explored possible solution, is to replace all transistors with high-V_{th} (low leakage) transistors, which will postpone the leakage problem for quite some years. This is though only possible when adequate time slack is available, since low-leakage transistors are slower by nature. Therefore, this work is based in the area where time requirements are just met or met by a fraction of the paths in the design. This is the setting for this work: Reducing leakage currents where slow, high-V_{th} are not possible to use, or only usable to some extent.

1.5 Objectives for this work

The main objective of this Master's Thesis is thus to evaluate logic families alternative to static CMOS for the creation of a low leakage cell library. This is achieved through implementations of simulation cases utilizing the selected logic families, followed by simulation case comparison and evaluation of the characteristics of the logic families. For this purpose a static CMOS library of cells is designed and simulated to perform as a basis for comparison. This library of cells is minimized in number of cells in to order to explore the limitations of standard cell IC design, but still serves as a fair comparison set for further logic family evaluation.

It will be shown in Chapter 6 that the logic family evaluation does not give an indication that an alternative logic family could prevail over static CMOS. Therefore, the objectives are expanded to explore how transistor characteristics can be taken into account when designing static CMOS cells for low leakage. Whether combining areas of logic into larger blocks, built on-the-fly by the synthesis tool, can prove to be an effective way of reducing leakage power dissipation is then the main objective.

This new synthesis process requires synthesis tools to be altered and cell libraries to be completely redesigned. This design area is explored and a proposal for a new cell library and synthesis tool will be presented in Chapter 8.

This work is carried out as an independent work in collaboration with two other Master's projects: Architectural Aspects of Design for Low Static Power Consumption by Martin Hans[2], and Incorporating Leakage Current Considerations in Logic Synthesis by Michael Kristensen[3].

The official project description is placed in Appendix A.

1.6 Overview of the report

To enable easy reading of this report, the specific organization of the report is given here. As has already been seen from the opening pages of the report, only chapter and section titles are for clarity given in the contents list. Further contents of the individual chapters may be found in the beginning of each chapter.

This work spans over a number of research areas that all affect each other. These areas include: Design of cell libraries, transistor technology, logic family design, power modelling and synthesis. To evaluate logic families for cell library design, three areas are particularly important. These areas are: Design of cell libraries, leakage current simulation and theory of power consumption, and logic family design. These three areas will be described in the

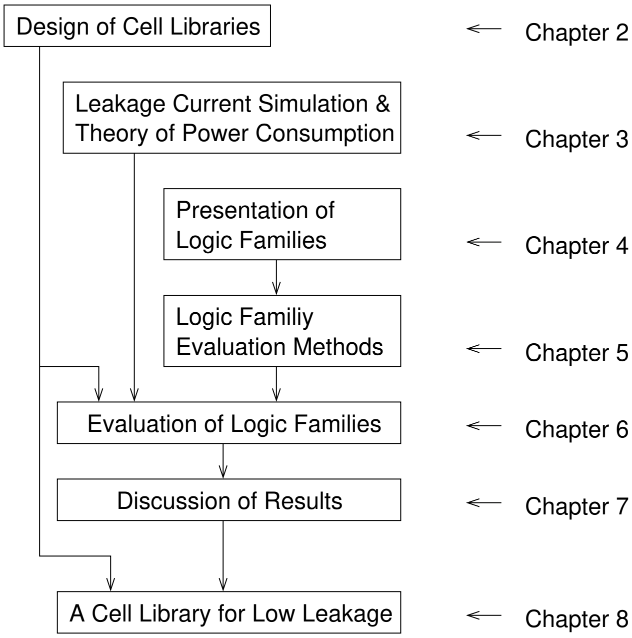


Figure 1.2: *Flow of the report.*

first three chapters of this report to form the basis for the evaluation work described in the following chapters.

The flow of this report is depicted in Figure 1.2. This figure will be repeated at the beginning of each chapter with markings showing the placement of the specific chapter in the entire report flow. Here follows a short description of the contents of the eight following chapters.

Chapter 2 introduces the design of cell libraries and the synthesis flow of today and discusses the future of cell libraries taking the rising problem of leakage current into account. The contents of cell libraries and the process of cell library design are explored.

Cell library design requires accurate simulation of electrical characteristics of logic cells. For this purpose **chapter 3** gives a investigation of how the power consumption of integrated circuits is simulated, with special focus on the leakage currents in CMOS designs as device sizes grow smaller. This chapter also introduces the transistors models and simulation approaches.

Alternative logic families are presented in **chapter 4** which investigates logic families through a short survey of the characteristics of each logic family in terms of power and ease of design. Based on this discussion a number of target logic families are selected for evaluation. How the logic families are evaluated is presented in **chapter 5**, which also describes how fair comparisons are achieved between logic families.

Chapter 6 presents the simulation work based on techniques described in chapters 3 and 5, and the results from the work.

Chapter 7 evaluates the results from all simulations and describes why static CMOS is chosen for the creation of the low leakage cell library. The new cell library is presented in **chapter 8**, which also describes the changes in the synthesis flow that are required to be done in order to use the library. **Chapter 9** concludes on the work and presents topics for future work and projects.

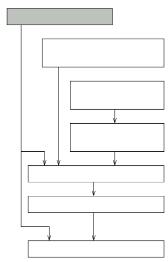
Hereafter follows the appendices. The contents and numbering of the appendices will be clarified when referred to in the report.

CHAPTER 2

DESIGN OF CELL LIBRARIES

Contents

Designing cell libraries requires an understanding of how a cell library is used by synthesis tools in order to asses what information it must contain and how the information must be structured. This chapter presents what an available cell library contains and discusses how timing, power, area etc. of logic cells is represented in the cell library and how this information is used by a synthesis tool.
Clearly, the cell library constitutes the interface between the physical world and the logical synthesis world. Yet, only a fraction of the possible logic functions are present in a cell library for practical reasons. The benefits of having cell libraries versus the drawbacks that this interface imposes are discussed.



2.1 The role of cell libraries

A cell library of today plays three key roles in the synthesis process. Firstly, it supplies the synthesis tool with a list of cells implementing logical functions from which the synthesis tool can pick and build larger functions. The cell library also delivers area, timing and power characteristics of the cells to enable the synthesis tool to optimize the design in respect to design goals set by the designer. Figure 2.1 depicts the flow.

Secondly the cell library contains all the information needed by the place & route tool to create a floorplan of the design optimized to certain constraints set by the designer. The place&route tool can then import technology specific wireload models supplied by the cell library and create a netlist of the entire design. This netlist in unison with the cell library implements a model of the design including locigal function, area, power and timing for both cells and interconnects (from wireload models). This can be used to verify the design by backannotation to the synthesis tool. Supplying good wireload models is the third role of the cell library.

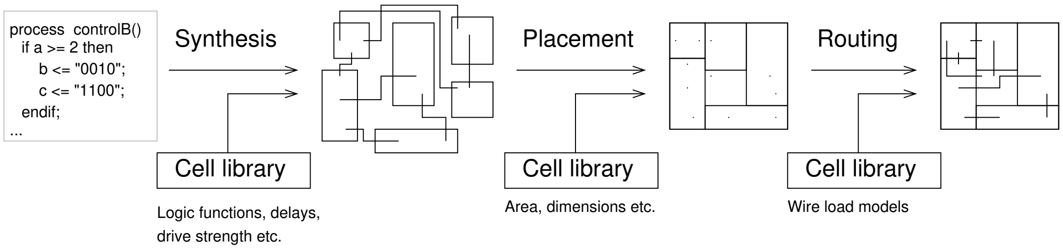


Figure 2.1: *Synthesis, placement and routing using data from a cell library.*

From this description of the roles of the cell library it is evident that the cell library needs to include the following:

- A compilation of cells including information of: Logic function, area, timing, dynamic and leakage power consumption
- Wireload models for both synthesis and place & route
- The physical layout of the cells for the place & route tool
- A library of symbols and other graphics for the graphic interfaces of all tools etc.

Since this work is about characterization of logic cells in terms of power consumption and timing, the term 'cell library' here refers to the first two points in unison. The STM $180nm$ DKHCMOS8D[4] cell library available at IMM/DTU will serve as example of a cell library.

2.2 *The contents of cell libraries*

The $180nm$ cell library available for this project contains both high-speed and low-leakage cells. This cell library will here serve as example to illustrate the design of cell libraries process. The cell library uses the LIBERTY file format, which will be described here. A sample of the cell library is included in Appendix B on page 93[2.1]. All references to actual tables and values are pointed at Appendix B

The LIBERTY file format contains two parts: General definitions and models followed by the cells in the cell library. The first part contains:

- Global values such as temperature, unit declarations, and settings for the synthesis
- Wire load models for wires formulated by resistance, capacitance, slope, area and fanout length
- Wire load selection criteria defining which wire load model to use depending on area
- Templates for propagation delay lookup tables with input net transition and output capacitance as parameters
- Templates for power dissipation lookup tables with input net transition and output capacitance as parameters

These values are printed for the synthesis tool to inform the tool under which assumptions the simulations of the cells have been done, and how the following electrical specifications of the cells are to be read. The cells follow hereafter. The description of the cells contain these data:

[2.1] All information in this sample has been manipulated in structure and values for copyright protection purposes.

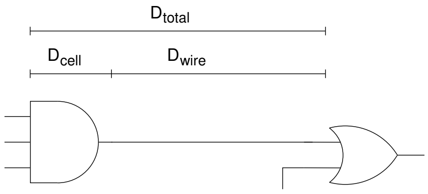


Figure 2.2: *Total gate delay split into cell and wire delay.*

Scalar values:

- Area
- Average leakage power
- Logic function
- Maximum capacitance

Lookup tables:

- Input dependent leakage power values
- Switching power, both for rising and falling transition
- Rise and fall output delay
- Rise and fall output transition time

With these values the synthesis tool is able to calculate the total area consumption, timing of the circuit with statistical wire loads, and the power dissipation with random inputs. Doing place & route and backannotating the design with input value information produces a realistic picture of whether the timing requirements of the circuit are met, and a reasonably good power dissipation prediction.

2.2.1 Modelling propagation delay in cell libraries

In the LIBERTY cell library format delays are modelled as gate delays and wire delays. The delay model used[5] can be expressed as:

$$D_{total} = D_{cell} + D_{wire} \tag{2.1}$$

The delay is modelled as the sum of the cell delay and wire delay (Figure 2.2). The cell delay is the time from a input value transition reaches 50% of its final value till the output of the cell has changed to 50% of its final value. This is depicted in the left hand side of Figure 2.3.

The propagation delay depends on the slope of the input value transition and the total capacitance on the output. The lookup tables for gate delay is therefore a table with capacitance and input value transition slope as parameters.

The delay of wires is read from lookup tables with resistance and capacitance as parameters, to model what delay that wire causes. A number of wire load models are available modelling a variety of wire lengths and capacitive loads on these. Statistical area dependent models are used to evaluate which wire load model is to be used for each wire. Backannotating the real wire length improves the accuracy of the model, and until it is done the delay models rely only on statistical, and possibly very conservative, wire delay models.

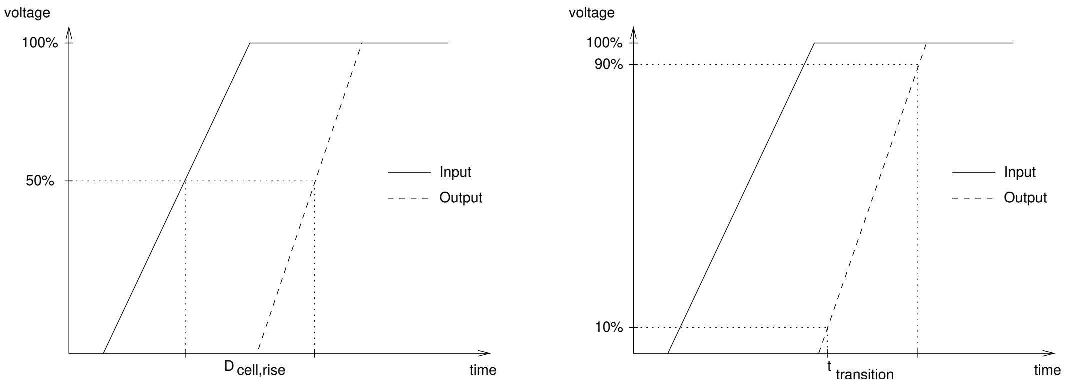


Figure 2.3: *Rise time and rise transition of a cell.*

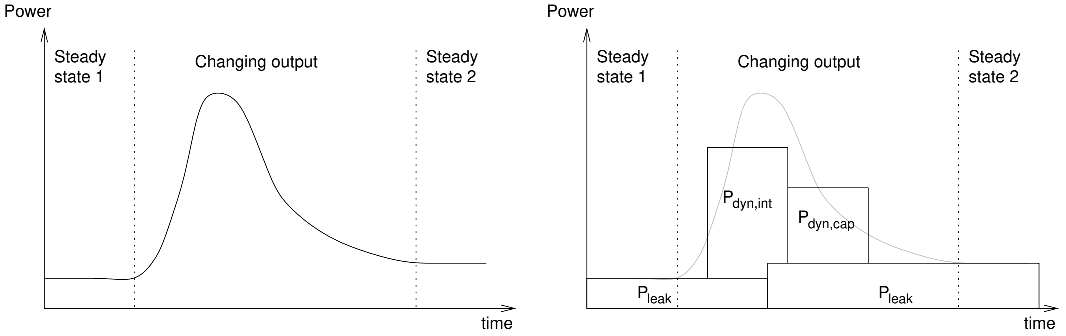


Figure 2.4: *Power consumption before, during and after an output transition. A cell library representation.*

2.2.1.1 *Calculation of Total Propagation Delay*

Calculation of the total delay is performed in three steps:

1. Calculate total output capacitance: Wire capacitance + total gate input capacitance
2. Look up the rise/fall-time of the cell using the calculated output capacitance and the input transition time as parameters
3. Add wire delay. This is calculated from adding the wire load model to the output transition

2.2.2 *Modelling power dissipation in cell libraries*

Modelling power dissipation in the cell library is done by dividing it into three categories: Input dependent leakage, P_{leak}, internal dynamic switching power $P_{dyn,int}$ and dynamic power consumption due to charging and discharging of output capacitances $P_{dyn,cap}$ [5]. Figure 2.4 illustrates a probable power dissipation over time of an output transition of a cell.

The peak of the power consumption graph on the left is due to internal power consumption such as charging/discharging of internal nodes and short circuit switching power. The slower falling slope after the peak is due to the capacitive wire or gate load on the output. There is quite some overlap, off course. Before and after the output transition two different input state dependent leakage currents are responsible for the entire power consumption in these regions.

A model of the power consumption is shown on the right hand side of Figure 2.4. $P_{dyn,int}$ depends on the slope of the input transition. A low slope causes increased short circuit power consumption. $P_{dyn,cap}$ naturally depends on the capacitive load on the output, which totals the wire load capacitance and the total input capacitance of connected logic gates.

Denoting the frequency of output signal transitions (the toggle rate) by TR the entire power model can be expressed in one relation:

$$P = P_{leak} + P_{dyn,int} + P_{dyn,cap} = h(v_{i,0}, v_{i,1}...) + E_{switch} * TR + E_{cap} * TR \tag{2.2}$$

h is an input state dependent leakage power function of the input state, where $v_{i,j}$ is the *j*'th input value to the *i*'th cell. This value is read from the input state dependent leakage power lookup table (*leakage_power*). If input values are unknown the default leakage power value is used.

E_{switch} is the switching energy required to change output state due to a transition from one to another input state. This value is looked up in the *rise_power* or *fall_power* lookup tables. The internal power consumption depends on the input transition time and the total output capacitance, which are the parameters for the lookup tables.

The last component is $P_{dyn,cap}$ which depends only on the output capacitance. This factor is summed into E_{switch} for practical reasons.

2.2.2.1 Calculating power consumption

The calculation of the power consumption follows in three steps for each cell:

1. When a input transition occurs: Determine what output transition the input transition causes and lookup the rise or fall power consumption for that transition
2. Then, lookup the leakage power consumption caused by both input vectors and add a average of these values to the total power consumption
3. If no input transitions occur, just lookup the leakage of the cell and add it to the total power consumption

Leakage power dissipation as a function of input states requires the leakage to be expressed in lookup tables with input vectors as parameter. The leakage at any moment can then be expressed as the total sum of leaking gates according to their respective input states. If input states are unknown, an average value read from the cell library is used.

2.3 Synthesis of cell based designs

How to represent area, power and propagation delay for each cell is described above. These values can be derived by either simulation of a full-custom design of the cells or by electrical simulation of a transistor netlist in for example SPICE. Yet, before these simulations can begin, one need to decide which cells to put in the cell library. To evaluate this, a look is taken on the synthesis process.

Figure 2.5 presents a simplified synthesis case where an abstract problem is synthesized into logic cells. Here the add-function is broken down into sub-problems iteratively until a level of boolean expressions is reached. No further synthesis or optimizations can be done without a cell library.

The cell library supplies a range of logic functions for the synthesis tool to pick from. In Figure 2.5 the synthesis tool picked a cell matching the 'Carry'-expression perfectly (1). If this cell was not available, the synthesis tool would have to go back to the boolean expression level and reorder the logic to fit smaller cells from which the larger one could be built(2).

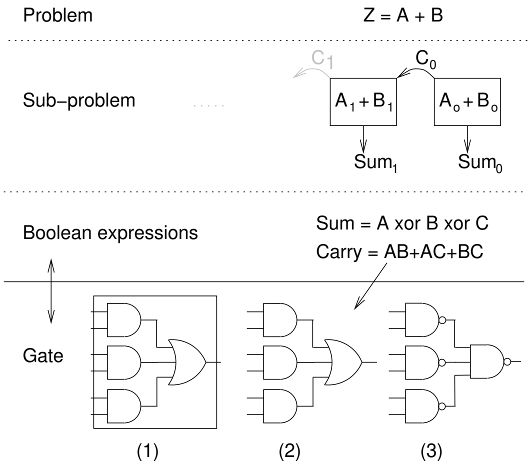


Figure 2.5: *A synthesis flow of mapping an abstract problem into logic cells.*

Optimization then follows in several steps. Possibly some of the paths through the increased levels of logic depth are not fast enough and must be compensated by increasing the drive strength of the gate. If this is still not enough to meet the timing requirements, logic optimizations must be done to improve speed. Since NAND-gates are typically faster than AND/OR-gates, the NAND-gates replaced the AND/OR-gates(3) in the right hand side of Figure 2.5.

2.3.1 *The cell library/synthesis tool interface*

From the example above it is evident that selecting logic cells for a cell library can be done in different ways. An analysis of the most common cells could be conducted and the cell library could be built with these cells, small as large.

Another way is to ignore the larger cells and build a large variety of smaller cells with widely different drive strength, gate delay etc, so that larger functions can be synthesized with minimum overhead.

A third way is to build large cells with both inverted and non-inverted inputs and outputs. These multi-purpose cells could be used in many places, reducing the need for other cells which allows for more complex cells to be put in the cell library.

No matter what approach is taken to selecting the cells, only a limited number of these cells are feasible to put in a cell library. This is mainly due to the shear simulation and design time it requires to design by hand and simulate cells. Looking into the cell library available in this project, 777 different cells are present. More than 80% of these cells are drive buffers, repeaters and inverters in different sizings. Sorting these out, 157 unique combinational logic cells remain. The distribution of the number of cells versus the number of inputs is depicted in Figure 2.6. It is clear, that designers behind the cell library have chosen a mix of a good deal of rather small, three- or four-input cells, and added a smaller number of commonly used larger cells.

2.3.1.1 *Limits of cell libraries*

This interface of supplying the synthesis tool with only a limited number of cells clearly has some disadvantage. First, it cannot contain all logic functions, so smaller cells have to be cascaded. Secondly, as all cells are not available with inverted/non-inverted inputs, inverters have to be put in numerous places. This is a further important as the number of cells and logic depths increase.

Thirdly, if a cell is just a bit too slow or too fast no improvements can be done, and the synthesis tool has to redesign the logic expression, if no slightly faster cell is available. A

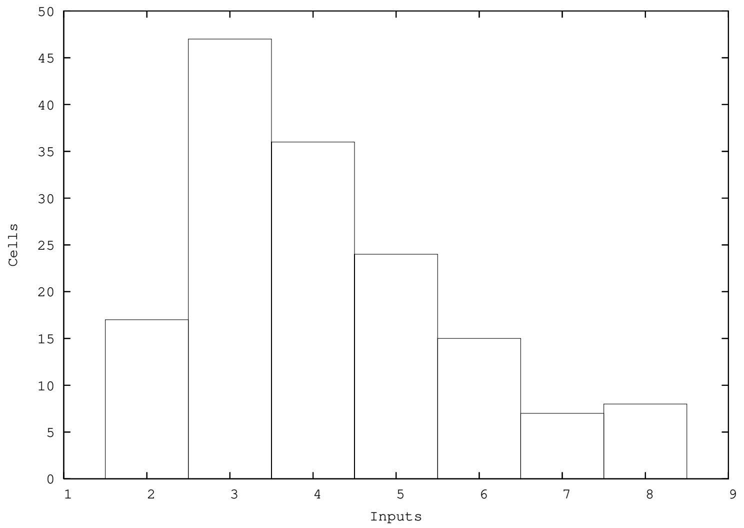


Figure 2.6: *Distribution of the number of cells versus number of inputs.*

fourth reason is, that for low leakage applications the cell library with a fixed number of cells is not good either. The possibilities of reducing leakage are hereby limited to replacing high-speed, high-leakage cells with reduced-speed, low-leakage cells. In many cases there is not enough time slack for this replacement, and high-leakage cells are therefore necessary. The limitations of using cell libraries will be further discussed in Chapter 8.

2.4 Implicit cell library contents

There are some characteristics, that are not expressed explicitly in the cell library, that the synthesis tool needs to be aware of in order to synthesize utilizing the cell library. First of all the synthesis tool needs to know the characteristics of the logic family with which the cell library has been constructed. For a given logic family there are limitations and issues that must be considered:

Connection of cells Can combinational logic be built simply by connecting logic cells like Lego blocks only taking the timing (sum of propagations delays) into account? Or do cells alter their electrical characteristics dependent on the characteristics of the previous logic stage?

Value stability Can signals be assumed to remain stable in value as long as the cells are fed with supply power an input values are stable? Or are there dynamic characteristics of the logic family that prevent this assumption? A notion of drive strength and drive limitations has to be formulated for each logic family.

Clocking issues Are cells simple logic functions or do they need a clock signal requiring the synthesis tool to build logic considering the timing of the clock for each cell?

These considerations are defining the way the synthesis tool has to synthesize a given design to a cell library built on a given logic family. Other considerations are:

Leakage current Do cells leak the same amount of current with all possible input combinations or can power be saved by building the logic utilizing statistical information in order to put as many cells in their low leakage state as long as possible?

Power versus speed What are the tradeoffs for the given logic family when it comes to power versus speed? Is high speed and low power impossible to achieve at the same time? And what does it cost in terms of area to pursue?

These considerations have to be done for the given library of logic cells and the results be built into the synthesis tool cost functions and synthesis operation style.

2.4.1 The static CMOS cell library

The $180nm$ static CMOS library available at the department is a fully characterized cell library in terms of the above mentioned issues. The synthesis of static CMOS is the topic of numerous papers.

Since static CMOS circuits both produce the output values and drive the value by connecting the output pins to either V_{DD} or V_{SS}, the task of the synthesis tool in terms of logic synthesis is reduced to combining the cells to form the correct larger logic blocks. Determining the output load of all cells and selecting cells with given drive strength tells the synthesis tool the total propagation delay of all paths in the design. If the delay is larger than the allowed value, the synthesis tool can either reorder the logic blocks, select faster cells or cells with more drive strength to boost the speed of the path. No specific connection considerations are needed with static CMOS.

Furthermore, since the static CMOS drives the output actively, outputs remain stable as long as the cells is fed by power and stable input signals. Static CMOS is not a dynamic or clocked (hence the name 'static') family so the synthesis tool can do the synthesis in respect to timing by just verifying that the critical path of combinational logic between two registers is no longer than the clock period.

The ease of synthesis with static CMOS is one of the key features that helped static CMOS become the most widely used logic family in VLSI design. Static CMOS Cell libraries can be derived from simulation of the electrical characteristics gates and wires. Gates are modelled by transistor netlists and wires are included as simulations of statistical RC wire loads. Together these are capable of implementing simple logical functions and their interconnects. This approach of pre-defining static CMOS cells implementing simple logical functions and pre-determining the electrical and delay characteristics lists benefits as very fast synthesis, pre-testable cells, pre-layout statistical wireload estimation and in general faster optimization by re-synthesis.

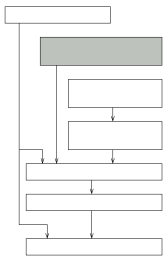
Yet, as this approach is very good for static CMOS, it may not be feasible for other logic families. Some logic families are not suitable for the static CMOS approach of connecting layer by layer of logic within time bounds. And cells implemented with certain other logic families do not preserve their logic output values over time. It is evident that the interface between cell library and synthesis tool has to be reevaluated when other logic families are taken into consideration.

CHAPTER 3

LEAKAGE CURRENT SIMULATION AND THEORY OF POWER CONSUMPTION

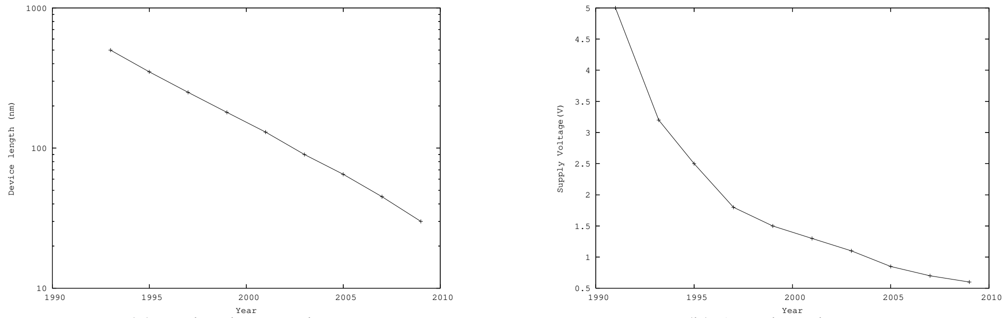
Contents

The aim of this chapter is to describe the effect of scaling down MOS devices on the dynamic and leakage power consumption. Projections of the future in terms of device sizes, supply voltages and power estimations are presented and used to estimate the magnitude of the leakage problem in the future.
Evaluating the leakage of logic gates is done through simulation with HSPICE. Transistor model cards used for these simulations are presented, and an introductory study of the effect of stacking transistors is given. Since stacking will be shown to have a great effect on the leakage, considerations for utilizing this and other facts for the design of low leakage gates are presented in the end of this chapter.



3.1 *Scaling device dimensions*

For the purpose of increasing performance and density, and lowering the power consumption, MOS devices have been scaled for more than 30 years. With more than 30% improvement in delay times per technology generation, a doubling of microprocessor performance



(a) Technology node (b) Supply voltage

Figure 3.1: *Projected development in device sizes and supply voltage [9]*

have been achieved every two years [6]. To keep power consumption down, supply voltages have been lowered. Hence, the transistor threshold voltage (V_{th}) has to be scaled accordingly to maintain the high drive current and to maintain the performance improvement of 30% per technology generation dictated by Moore´s Observation[3.1]

Power consumption of integrated circuits has become the major technical problem of the semiconductor industry. This problem has to be dealt with at all levels to make the exponential growth in device density possible in the future. So far, large achievements in reducing the power dissipation has come from voltage scaling and parallelizing designs to preserve computational speed. Voltage scaling is very effective due to the power dissipations quadratic dependency of the supply voltage. Total power consumption can be expressed in this equation[7]:

$$P = P_{dynamic} + P_{static} = ACV^2 f + VI_{leak} \tag{3.1}$$

This equation expresses that the total power dissipation originates from two main sources: 1) Dynamic power dissipation, that includes the charging and discharging of capacitances and 2) Static power dissipation produced by leaking devices. Dynamic power also includes switching power dissipation, which is often expressed[8]:

$$P_{sc} = (\beta/12)(V_{DD} - 2V_T)^3(\tau/T) \tag{3.2}$$

Taking a look at the computational speed versus voltage supply this equation comes in handy[7]:

$$f \sim \frac{(V - V_{th})^\alpha}{V} \tag{3.3}$$

The term α is an experimentally derived constant, that for current technology is approximately 1.3.

Combining equation 3.1 and 3.3 it is evident why voltage scaling is so effective. The computational speed of a circuit decreases approximately linear with decreasing voltage, but the power consumption drops quadratically with decreasing voltage supply. Therefore, halving voltage supply and doubling hardware in parallel preserves computational speed and decreases dynamic power consumption by around 50%. Projected supply voltages and device sizes are depicted in figure 3.1.

Though, leaking devices causing static power consumptions have become just as power hungry as the dynamic sources of power dissipation. Equation 3.1 states that the static

[3.1] Moore’s Law is an inaccurate name for the law since it is not a mathematical (or legislative) law at all. Moore´s Observation, which it is more accurately called in many sources, depends on a survey of the development of integrated circuits versus time. As this relationship cannot hold forever, Moore´s Law is best called Moore´s Observation.

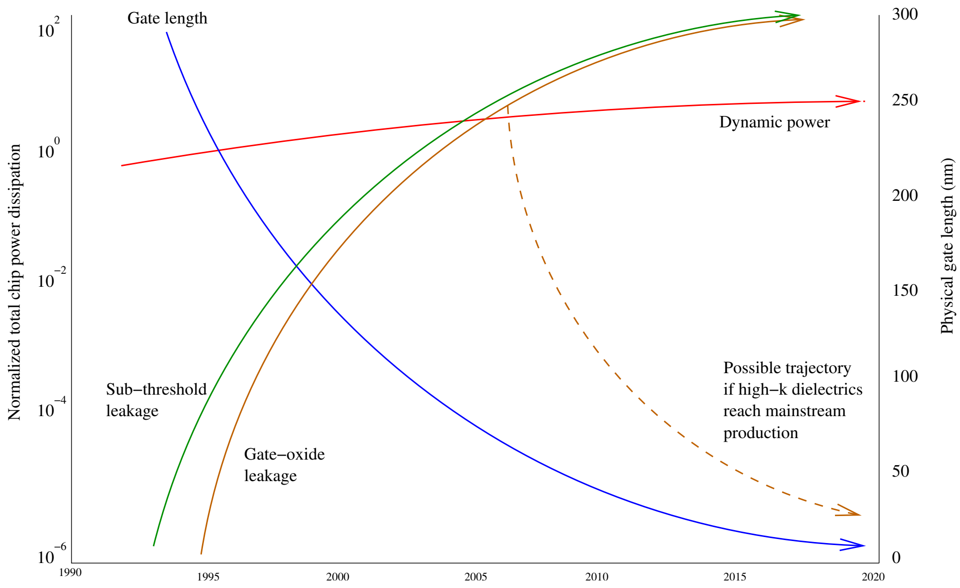


Figure 3.2: *Total chip dynamic and static power dissipation trends assuming doubling of on-chip devices every two years. Based on the International Technology Roadmap for Semiconductors[10] and [7]*

power dissipation depends linearly on the voltage supply, which may lead to the interpretation that static power consumption puts an end to voltage scaling. This is not entirely correct since the term I_{leak} is exponentially dependant on the supply voltage as well, which is why voltage scaling and hardware doubling still works in many cases in the future for lowering total power consumption[2].

Yet, as hardware is doubled and devices are leaking, the leakage power dissipation grows to be the major fraction of the total power dissipation. Figure 3.2 shows projected dynamic and leakage power dissipation together with projected device sizes. The leakage component is broken in to two contributors:

- Subthreshold leakage, I_{subth}, which is the drain-source current when the transistor is in its non-conducting state.
- Gate-oxide leakage, I_{gate}, is the total amount of leakage currents through the gate oxide due to tunnelling etc.

Figure 3.3 depicts subthreshold leakage and gate-oxide of a leaking nMOS-transistor.

The right hand side of Figure 3.3 shows paths of gate-oxide leakage. Gate-oxide leakage is not modelled in this work, but will be discussed shortly in section 3.2.3, where reasons for leaving out gate-oxide leakage are given. The term leakage or I_{OFF} in this work refers to subthreshold leakage currents only.

There are numerous further ways of reducing the total power consumption. Clock-gating, bus-encoding and switching activity reduction schemes are a few. All of them tar-

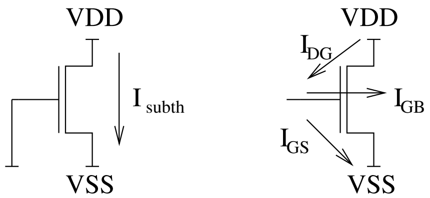


Figure 3.3: *Subthreshold leakage and gate leakage of an nMOS transistor.*

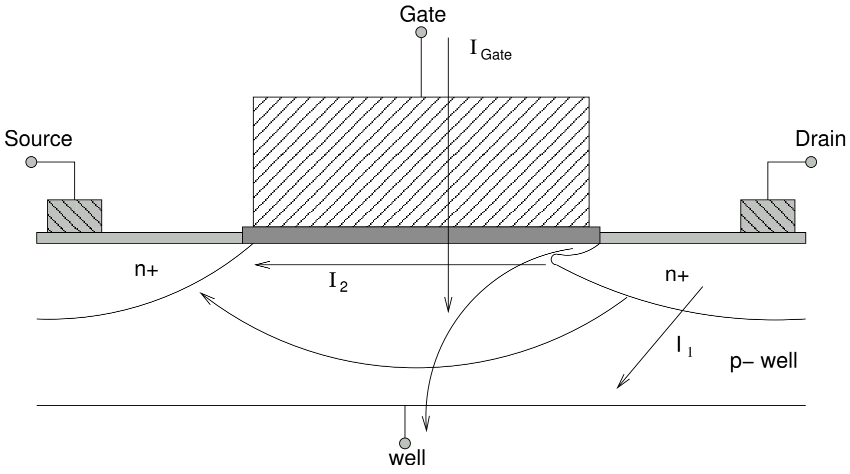


Figure 3.4: *Summary of leakage currents mechanisms.*

get primarily the dynamic power consumption though. The scope here is mainly leakage currents and only the leakage part of the total power consumption will be discussed. Yet, when a solution is presented it is discussed whether the solution causes increased dynamic power consumption.

3.2 The effect of device dimension scaling on leakage currents

I_{OFF} is influenced by threshold voltage (V_{th}), the physical dimensions of the channel, channel/surface doping, drain/source junction depth, gate oxide thickness and V_{DD} [6].

Scaling down V_{th} increases the leakage drastically due to the weak inversion state leakage which is a function of V_{th} and is not due to the transistor channel length. Leakage in long channels are dominated by drain-well and well-substrate reverse biased *p-n* junctions.

3.2.1 p-n junction reverse bias current

When building structures with layers of doped silicon and electrically charging them, currents will unavoidably leak through the silicon. From drain and source regions a reverse bias p-n junction leakage current flows into the well region (Figure 3.4, I_1). This current has two main components: Firstly, the minority carrier drift near the edge of the depletion region and secondly the electron-hole pair generation in the depletion region. Both components are heavily dependent on the doping level of the source and drain regions. When heavily doped drain/source regions together with short-channel-effect enhancements, such as halo-doping [11] are used, p-n junction reverse bias currents increase significantly.

3.2.2 Subthreshold leakage

The most severe of all leakage currents in deep submicron devices is the subthreshold leakage current[6]. When the gate voltage drops below V_{th} a weak inversion conduction current is still present in the MOS transistor (Figure 3.4, I_2). Ideally the MOS transistor should be nonconducting as the gate voltage reaches below V_{th}, but instead the subthreshold current decreases exponentially with decreasing gate voltage. This forms a linear slope with $\log I_{subth}$ as function of V_{th}, see figure 3.5. Evidently the subthreshold current at zero gate voltage increases exponentially as V_{th} is decreased.

Considering a *n*-channel transistor with source connected to ground, $V_g < V_{th}$ and drain-source voltage $| V_{ds} | \geq 0.1V$ the almost entire voltage drop occurs over the reversed bias

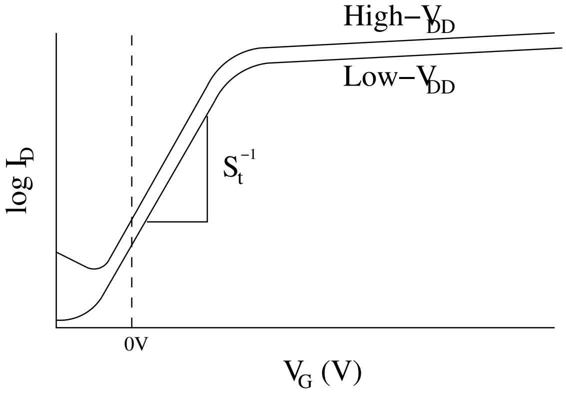


Figure 3.5: *Drain current versus gate voltage at two different drain voltages.*

substrate-drain *p-n* junction. Under these conditions the electrostatic potential variations are very small and the electric field formed by the gate is negligible, causing the number of mobile carriers to be small. In this case the drift component of the subthreshold drain-to-source current is negligible and subthreshold conduction is dominated by the diffusion current. The carriers move by diffusion along the surface causing a current which is exponentially dependant on the gate voltage.

The weak inversion current can be expressed by:

$$I_{ds} = \mu_0 C_{ox} \frac{W}{L}(m-1)(v_T)^2 \times e^{\frac{V_g - V_{th}}{m v_T}} \times (1 - e^{\frac{-v_{DS}}{v_T}}) \tag{3.4}$$

where

$$m = 1 + \frac{C_{dm}}{C_{ox}} = 1 + \frac{(\varepsilon_{si}/W_{dm})}{\varepsilon_{ox}/t_{ox}} = 1 + \frac{3t_{ox}}{W_{dm}} \tag{3.5}$$

The threshold voltage of the transistor is denoted V_{th} and the thermal voltage $v_{th} = KT/q$. C_{ox} is the gate oxide capacitance and μ_0 is the zero bias mobility. *K* is the Boltzmann constant, *T* is the temperature in Kelvin and *q* is the electron charge. *m* is the subthreshold swing coefficient or body effect coefficient for the transistor. W_{dm} is the maximum width of the depletion layer and t_{ox} is the thickness of the gate oxide. C_{dm} and C_{ox} are the depletion layer capacitance and the capacitance of the insulator layer.

From equation (3.4) it can be seen that the subthreshold current is independent of the drain-source voltage for V_{DS} larger than just a few v_T. This seems counter-intuitive, since one would expect the drain-source voltage to have great impact in the leakage current. Equation (3.4) does not hold for small devices due to effects such as drain-induced barrier lowering and body-effect, and is merely printed here to show the leakage currents dependency of gate width, length and gate voltage in longer devices. It confirms that the leakage grows exponentially with V_g. This dependency is expressed in the subthreshold slope (S_t) which described the inverse slope of the linear part of the I_{subth}/V_{th}-graph (figure 3.5).

$$S_t = (\frac{d(log_{10} I_{ds})}{dV_g})^{-1} = 2.3\frac{mkT}{q}(1 + \frac{C_{dm}}{C_{ox}}) \tag{3.6}$$

A low value of the parameter S_t is desirable since it expresses the amount of voltage, the gate voltage has to be reduced in order to reduce subthreshold leakage a certain factor. Or in other words, how easily (and to which extent) leakage can be reduced. S_t values for a bulk CMOS process are typically around 80 to 120 mV per decade. The value of S_t can be improved by lowering the oxide thickness or lowering the substrate doping, increasing the maximum depletion layer width.

In the following sections the most dominant effects causing deviations from equation (3.4) leading to altered leakage in small devices will be described. These effects are drain-induced barrier lowering and the body effect. Further effects are 'narrow width effect' and

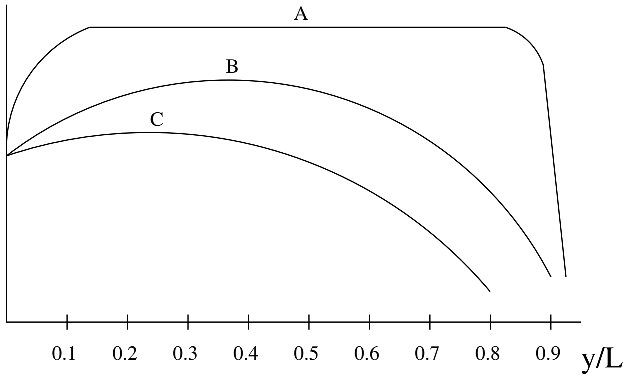


Figure 3.6: *Energy bands at the surface versus distance normalized to the channel length L from source to drain. Curve A depicts a long-channel device, curve B a short-channel device. Curve C represents a short-channel device with high drain bias.*

'V_{th} roll-off'. These effects will not be discussed here, as they are determined by the device sizes alone and is not altered by reconfiguring transistors.

3.2.2.1 Drain-induced barrier lowering

In long devices the drain and source regions are far enough apart for the electrical field and depletion regions induced into the device by these regions to have any impact in the threshold voltage. Hence the threshold voltage is almost independent of the channel length and drain bias. In a short-channel device, on the other hand, source and drain depletion width and source-drain potential have great effect on the energy band bending over a considerable portion of the device. Threshold voltage and thereby subthreshold currents of short-channel devices vary with the drain bias. This effect is called drain-induced barrier lowering (DIBL).

Figure 3.6 depicts three different energy bands near the surface of a long device (A) and two short devices (B and C), charged by relative low drain-source voltage except (C) which is driven by higher voltage. The threshold voltage equals the maximum energy level a charge carrier has to achieve to move between the source and drain terminals.

It is evident that decreasing channel lengths reduces the threshold voltage. Increasing drain voltage causes further V_{th} lowering in the short-channel device, but does not affect the long-channel device. This is due to the flatness of the curve in the middle (or the high slopes near drain and source), which originates from the extension of the non-affected area under the gate.

Ideally DIBL does not change the S_t-slope, but it reduces V_{th}. Higher surface and channel doping can reduce the DIBL effect [6]. DIBL certainly has to be taken into account when designing new technologies as supply voltage lowering not only slows the circuits down, but counters the DIBL-effect and raises the threshold voltage which further slows circuits down. This is especially important when considering multi-V_{th}-designs [12]. The effects of DIBL is shown on Figure 3.7.

3.2.2.2 Body effect

Devices built from numerous MOS transistors are typically made on a common substrate. All MOS transistors therefore share the same substrate and hence the same substrate potential $V_{substrate}$. Yet, as transistors are connected in series to form gating functions, it is no longer possible to guarantee the same source potential for all transistors. Source to substrate voltage (V_{SB}) may increase along the chain of transistors when moving along the chain away from V_{SS}. This increase in V_{SB} widens the bulk depletion region and increases

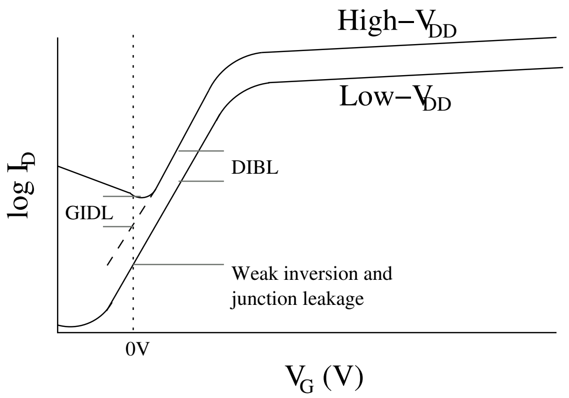


Figure 3.7: *Leakage current contributors as function of gate length.*

the threshold voltage. This effect is known as the body effect. The following equation expresses the threshold voltage equation [1, 6]:

$$V_{th} = V_{fb} + 2\psi_B + \frac{\sqrt{2\varepsilon_{st} q N_a (2\psi_B + V_{sb})}}{C_{ox}} \tag{3.7}$$

where V_{fb} is the flat band voltage, N_a is the doping density in the substrate, and $\psi_B = (KT/q)\ln(N_a/n_i)$ is the difference between the Fermi potential and the intrinsic potential on the substrate. Looking at the V_{th}'s dependency of the bulk-source potential, it is evident that the V_{th} is more sensitive to V_{bs} with high bulk doping concentrations. The substrate sensitivity can be expressed as: [6]

$$\frac{dV_{th}}{dV_{bs}} = \frac{\sqrt{\varepsilon_{st} q N_a / 2(2\psi_B + V_{sb})}}{C_{ox}} \tag{3.8}$$

At zero V_{sb} the substrate sensitivity is C_{dm}/C_{ox} equal to $m - 1$ which explains, why m is also referred to as the body effect coefficient.

3.2.2.3 *Modelling subthreshold leakage*

The entire subthreshold leakage current including weak inversion, DIBL and body effect can be expressed by the following equation. [6, 13]

$$I_{subth} = A \times e^{\frac{1}{m v_T}(V_G - V_S - V_{th0} - \gamma' \times V_S + \eta \times V_{DS})} \times (1 - e^{\frac{-v_{DS}}{V_T}}) \tag{3.9}$$

where

$$A = \mu_0 C'_{ox} \frac{W}{L_{eff}} (v_T)^2 e^{1.8} e^{\frac{-\Delta V_{th}}{\eta v_T}} \tag{3.10}$$

V_{th0} is the zero bias threshold voltage. For small values of V_{sb} the body effect is nearly linear with respect to V_s, so the body effect is represented here as $\gamma' V_s$. The DIBL coefficient is denoted η, and C_{ox} is the gate oxide capacitance, μ_0 is the zero bias mobility and m is the subthreshold swing coefficient for the transistor. The term ΔV_{th} is introduced here to account for the transistor-to-transistor leakage variations [6].

Equation (3.9) and (3.10) are the equations used in this project to model subthreshold leakage. The same equations are used in the transistor models[13], which will be used in the simulation work.

3.2.3 Gate leakage

To keep up the electrical field strength under the gate as voltages are being scaled down, gate oxide thickness has continuously been reduced. This causes the gate to become leaky, leaking current into drain, source and bulk dependant on the voltages in these nodes. Gate-oxide leakage, or simply gate leakage, totals all the direct tunnelling currents from gate to source, drain and bulk (Figure 3.4 and 3.3). The thin oxide further introduces leakages such as gate-induced drain leakage (GIDL), which increases the leakage from drain to gate as gate voltage reaches 0V, and further increases the leakage exponentially if gate voltage should drop below 0V [14]. GIDL is incorporated in Figure 3.7

Gate leakage will be dominant as device sizes hit $65nm$ [15] around the year 2007 [7]. Yet, this form of leakage is projected to get under control by using high-k dielectric materials under the gate (Figure 3.2).

Different materials such as ZrO_2 and Ta_2O_5 have been investigated for this purpose, but unfortunately the bandgap reduces with increasing permittivitty, which is why materials with very high-k values tend to cause leakage due to thermal emissions [9]. Until new materials have been investigated, gate leakage will be a major contributor to the leakage problem.

Modelling gate leakage is very difficult, and the main work done in this field is based on statistical models based on measurements from real processes [16]. Various models are presented in many papers, and no uniform model can be derived from these. This is partly because gate-oxide tunnelling is a quantum-mechanical process impossible in classical physics and not entirely understood yet[14].

A good estimation of gate leakage can be obtained by multiplying a statistical gate leakage per transistor width of the process by the total width of transistors in the design. This method is suggested in [7]. But one must bear in mind, that not all devices produce gate leakage due to aspects of the transistor configuration. A transistor with the same voltage on all terminals does not produce leakage currents. A stack of nMOS transistors for instance with gate voltages at $0V$ will not all leak from drain to gate. Especially not if another stack of nMOS transistors is pulling the output low. Hence, gate leakage is dependant upon input combinations.

This work does not include gate leakage evaluations as the only way to reduce this form of leakage is through changing the materials in the fabrication process. Reconfiguring transistors and changing logic families cannot reduce gate leakage sufficiently to be worth going for. Therefore, the SPICE models have been selected not model leakage.

In the evaluation of dynamic logic families, gate leakage plays a big role in the construction of these. Therefore a statistical model is formulated from the statistical data presented in this chapter. The stacking effect on gate leakage described just above will be evaluated where it might yield beneficial results in terms of numbers of leaking devices.

3.3 Leakage current modelling using HSPICE

All simulations of propagation delays and leakage currents in this work were done using *Synopsys ® HSPICE*. [3.2] Transistors are modelled with the Berkeley Predictive Technology Model (BPTM[17]) model cards compliant with the Berkeley Short-channel IGFET Model version 3 (BSIM[18]) model. This section gives brief information about the simulation process using HSPICE, BPTM and BSIM.

3.3.1 The Berkeley Predictive Technology Model

The BSIM model is on the homepage described as a physics-based, accurate, scalable, robustic and predictive MOSFET SPICE model for circuit simulation. In the literature it is

[3.2] Synopsys HSPICE version 2004.03 with AvanWaves 2004.03 as graphical interface

Process	L_{eff}	T_{ox}	V_{th-n}	V_{th-p}	R_{dsw-n}	R_{dsw-p}	V_{DD}
70nm LL	38nm	16Å	0.30V	-0.35V	150 $\Omega/\square$	280 $\Omega/\square$	1.0V
70nm HS	38nm	16Å	0.15V	-0.16V	150 $\Omega/\square$	280 $\Omega/\square$	1.0V
180nm LL	100nm	40Å	0.4V	-0.4V	450 $\Omega/\square$	250 $\Omega/\square$	1.2V
180nm HS	100nm	40Å	0.25V	-0.25V	450 $\Omega/\square$	250 $\Omega/\square$	1.2V

Figure 3.8: Model card parameters for $70nm$ and $180nm$ LL and HS transistors

frequently used as basis for circuit simulation and is widely used by most semiconductor manufacturers world wide[18]. For the BSIM model a range of BPTM transistor model cards is available in device sizes $180nm$ down to $70nm$. On the BPTM site a generator for model cards is offered, that can produce model cards with user specified parameters. The parameters are:

- L_{eff}, effective gate length.
- T_{ox}, gate oxide thickness.
- V_t, threshold voltage.
- R_{dsw}, drain/source parasitic resistance.

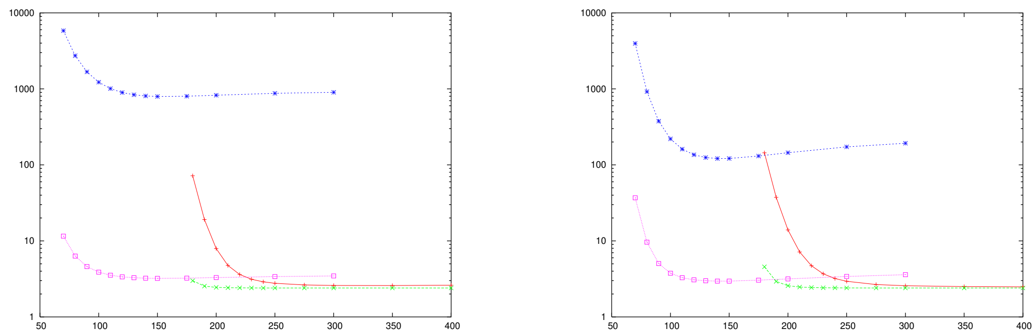
Estimating these four parameters enables the generation of nMOS and pMOS model cards for any process within some limits specified by the generator.

In this work four nMOS/pMOS-pairs of transistor model cards have been generated this way. A high-speed (low-V_{th}) and a low-leakage (high-V_{th}) pair, both in $180nm$ and $70nm$ versions. The value of V_{th} was for the $180nm$ high-speed process copied from the STM DKHCMOS8 cell library and for the $70nm$ high-speed(HS) process taken from [19]. For the low-leakage (LL) versions the maximum V_{th} allowed by the BPTM model card generator were selected. Values recommended by BPTM for T_{ox} and R_{dsw} were used. Table 3.8 shows selected model parameters.

To enable sufficient current drive V_{th} is often set to be $V_{DD}/4$ [19]. In the low-leak transistors in Table 3.8 this design rule-of-thumb has been altered to be $V_{DD}/3$ to further enhance the low-leakage performance of the LL transistors. All model cards created for this project is attached in Appendix C.

3.3.2 *Predicting the future with BPTM model cards*

To give an impression of the difference in leakage currents in $180nm$ and $70nm$ technologies, figure 3.9 was produced through SPICE simulation of the eight transistors. Figure



(a) Leakage vs. gate length of nMOS transistors (b) Leakage vs. gate length of pMOS transistors

Figure 3.9: *Leakage in pico-Amps (pA) of nMOS and pMOS transistors. Both $180nm$ and $70nm$ transistors versus device length in nm. The top two lines represent HS transistors, and LL transistors bellow.*

3.9(a) shows the leakage of the $180nm$ (blue and purple) and $70nm$ (red and green) nMOS transistors in HS and LL versions.

The difference in leakage is very clear. The minimum sized $70nmLL$ transistor leaks $13pA$, and $5871pA$ for the $70nmHS$. In the $180nm$ case the leakages are $2.5pA$ and $70pA$ respectively. The pMOS $70nm$ transistor leaks $3956pA$ and $39pA$ in HS and LL versions respectively. For the $180nm$ transistors the leakages are $145pA$ and $4.5pA$ in HS and LL versions respectively.

The difference is very clear. The leakage of a $70nmHS$ transistor is a factor of 84 higher than the $180nmHS$ nMOS transistor. The difference between HS and LL transistors is even more expressed in $70nm$ technology than in $180nm$ technology.

Surprisingly, through simulation it was found that the pMOS transistor (except the $70nmHS$ case) leaks more than the corresponding nMOS transistor. The literature states, that the opposite should be the case. All BPTM models seem to have this behavior.

The leakage currents do not decrease exponentially with long device sizes. After $2*L_{min}$ the leakage seems to increase a bit and flatten out at a certain level. This is due to the derivation of V_{th} which depends on a number of either experimentally or calculatory approximated factors[20]. The model cards therefore have maximum accuracy near minimum device sizes.

3.3.3 *Assumptions*

To enable fair comparison between logic families, the surrounding circuitry behaves according to a set of assumptions given here:

- Input values reach perfect ($0V$ or V_{DD}) value and are noise free.
- Voltage supply lines are perfect in voltage values and do not swing when power is drawn from them.
- Outputs of the circuit under test drive a capacitor equal to ten times the gate capacitance for the given technology.

The first two assumptions prevent logic families coping miserably with low quality input values and voltage supplies to perform equally miserably. Clearly, when designing circuitry utilizing these logic families, steps would be taken to improve input and voltage supply voltage level stabilities. All simulations are done assuming room temperature (25 degrees Celcius).

3.3.4 *Device sizes*

Since no design rules for a $70nm$ process could be found, the minimum width of a nMOS transistor was adopted from [21] and linearly scaled with device size. The same approximation lies behind other device sizes that could not be located in the literature. The minimum width of a pMOS transistor is set to $1.5 * W_{min,n}$[3.3] to balance designs for maximum speed.

Whenever a width or length of a device is mentioned in this work, it refers to *n* times the minimum width W_{min} or minimum length L_{min}, respective to whether it is a pMOS or nMOS device. Table 3.10 shows these sizes.

[3.3] This figure is approximated from $\sqrt{\frac{\mu_n}{\mu_p}}$, which is the typical way to balance the widths [22]. In this work there is no clear reason to alter this relation.

Feature	Description	Calculus	DS = $70nm$	DS = $180nm$
$L_{min,n}$	Minimum gate length nMOS	1 * DS	$70nm$	$180nm$
$L_{min,p}$	Minimum gate length pMOS	1 * DS	$70nm$	$180nm$
$W_{min,n}$	Minimum gate width nMOS	(280/180)* DS	$108nm$	$280nm$
$W_{min,p}$	Minimum gate width nMOS	1.5*(280/180) * DS	$162nm$	$420nm$
AS_n	Area of source nMOS	(400/180) * DS * $W_{min,n}$	$16800nm^2$	$120000nm^2$
AS_p	Area of source pMOS	(400/180) * DS * $W_{min,p}$	$25200nm^2$	$180000nm^2$
AD_n	Area of drain nMOS	(400/180) * DS * $W_{min,n}$	$16800nm^2$	$120000nm^2$
AD_p	Area of drain pMOS	(400/180) * DS * $W_{min,p}$	$25200nm^2$	$180000nm^2$
PS_n	Perimeter of source nMOS	2*((400/180)*DS+$W_{min,n}$)	$526.8nm$	$1359.2nm$
PS_p	Perimeter of source pMOS	2*((400/180)*DS+$W_{min,p}$)	$634.8nm$	$1639.2nm$
PD_n	Perimeter of drain nMOS	2*((400/180)*DS+$W_{min,n}$)	$526.8nm$	$1359.2nm$
PD_p	Perimeter of drain pMOS	2*((400/180)*DS+$W_{min,p}$)	$634.8nm$	$1639.2nm$

Figure 3.10: Feature sizes of transistors with device sizes (DS) $180nm$ and $70nm$.

3.4 *The leakage of logic gates*

The entire discussion above about leakage in transistors was focused on a single non-conducting transistor connected with maximum voltage drop across it. Due to the exponential dependency of I_{OFF} to V_{DS}, leakage is very much changed when several non-conducting transistors are put in series, as the V_{DS} voltage drop is shared by the serialized transistors.

Forming logic gates depends upon configuring transistors in parallel and in serial, and the gate will leak depending on how these transistors are connected. Hence, designing logic gates for low leakage must take into account the effects of staking transistors.

3.4.1 *Stacking of transistors*

The leakage through series-connected transistors in a stack with more than one non-conducting device reduces the leakage by at least an order of magnitude [9]. As device sizes are diminishing, and thereby the DIBL effect increases, the stacking effect increases. Therefore, the stacking factor, defined as the ratio between the leakage of a single versus a stack of transistors, will increase in the future. Stacking transistors is a promising way of reducing leakage. Firstly due to the stacking effect itself, and secondly because a stack of non-conducting transistors will have increasing source voltages the closer they are placed to the output (at V_{DD}), which increases the body effect, reducing leakage further.

Estimating the leakage of a stack of transistors is rather simple if the transistors are homogenous and in one unbroken line. But when using different transistor sizes and connecting other paths midways in the stack, the task becomes quite difficult. In [13] a promising pseudo-algorithm is given for estimating leakage. Using HSPICE and the BSIM3 model, the stacking effect is modelled by an iterative approach.

Figure 3.11 shows different transistor configurations leaking into ground. The difference between one single and two transistors in series is evident. The difference is nearly a factor of nine. As expected the right-most configuration leaks twice the amount of the second configuration. The two configurations with three transistors show the importance of placing transistors correctly in a stack.

Due to body effects it is better to place the single transistor near ground (near V_{DD} for pMOS transistors) than near the output, which might seem counterintuitive. Voltages on the midway of the stacks are written on the figure. These voltages show the great effect of the body effect, since the voltages indicate much higher resistance in the upper transistors. These low voltages cause the rightmost of the three-transistor configurations to be superior in terms of leakage compared with the leftmost.

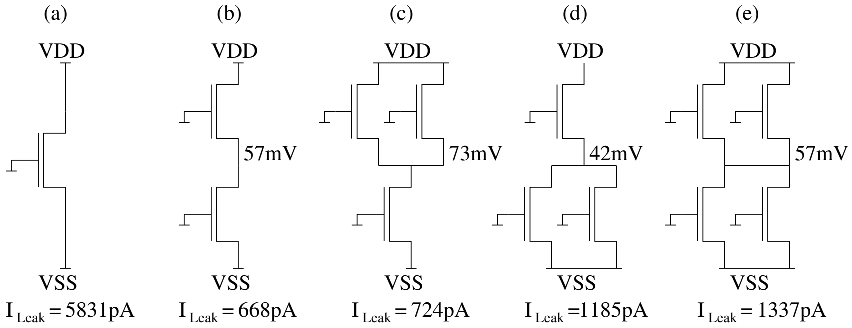


Figure 3.11: *Leaking stacks of 70nm HS transistors. The voltage denotes the voltage measured in the middle of the stack when full V_{DD}=1V is supplied to the stack.*

3.4.2 Leakage as function of input combinations

The leakage of a gate depends heavily upon input combinations[13]. The difference between the leakage in different input states can be orders of magnitude. Figure 3.11 serves a good example here. Assigning the input values 0 and 1 on configuration *b)*, the configuration leaks as configuration *a)*. Assigning logic 1s to one of the two topmost transistors in configuration *e)* makes this configuration leak twice the leak of configuration *a)*. This leakage is a factor of 8.7 larger than if all inputs in configuration *e)* were set to logic zero.

Comparing again *a)* and *e)*, and assigning zero's to the lower transistors and random inputs to the top transistors, another leakage determining factor is evident. In 50% of the time, configuration *a)* will leak through one transistor. Configuration *e)* will leak through two parallel transistors in 75% of the time. This makes configuration *e)* much more leaky than *a)*. Taking the average of the leakage, configuration *e)* leaks more than configuration *a)* by a factor of 2.79. Configuring transistors for low leakage is clearly beneficial.

3.5 Designing for low leakage

The characteristics of transistor described in this chapter can be used to design circuitry for low leakage. The following list provides the key points of this chapter for use in low leakage design. The list describes characteristics of nMOS transistors. pMOS transistors have equal characteristics.

1. Transistors near the output are most affected by the DIBL effect, lowering their V_{th}.
2. Transistors that are not directly connected to V_{SS} are affected by the body-effect increasing their V_{th}.
3. The leakage of a transistor depends heavily upon the gate length and V_{th}.
4. Stacking of transistors reduces the leakage greatly.
5. The leakage of a gate depends on the input state

When design a region of circuitry (a logic gate for example) placing as few transistors near the output(*1*) and V_{SS}(*2*) as possible will reduce the leakage. This can be achieved by reconfiguring the transistors in the stacks.

Reducing the leakage can be done by increasing the gate length(*3*) or V_{th} of the transistors. This reduces the drive strength of the transistor, so available time slack must be available.

Leakage can also be saved by using building logic gates with an increased number of transistors in series (in stacks) (*4*). A larger with high stacks of transistors will therefore leak less than a cascade of smaller cells in many cases.

Since the leakage is input dependent (*5*), a gate can be supplied with a low leakage input vector when it is inactive.

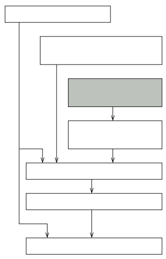
These considerations help selecting logic families for leakage evaluation, which is the topic of the following chapter.

CHAPTER 4

PRESENTATION OF LOGIC FAMILIES

Contents

The aim of this chapter is to give a short survey of logic families. Based on this survey the logic families for further evaluation are selected. These families are CPL and Domino logic. These logic families together with MTCMOS and static CMOS are further introduced and their benefits and drawbacks in terms of power consumption, ease of design, and characteristic features such as robustness to voltage swings and process variations are discussed.



4.1 Logic selection criteria

Logic families have been developed for a number of purposes. Some have been developed with focus on increased speed, reduced power or reduced area. Furthermore, logic families aimed at avoiding heavy peak current loads on the voltage supplies or aimed at reducing the noise emissions from the circuitry etc. have been developed to solve or reduce problems encountered in the IC design world.

Many of them have had their era, which ended when new problems were encountered that other logic families were better to cope with, making them better overall. Improvements in the IC fabrication process have solved or introduced problems, changing the circumstances for the choice of logic family.

For a long time static CMOS was the logic family of choice in overall terms of power, area and speed. But, since the leakage problem will only grow in the future, this choice may have to be reconsidered. One topic of this work is selecting and evaluating alternative logic families that may experience a come-back in the main industry due to the rising leakage problem.

As described in the introduction and explained more closely in Chapter 3 the leakage problem can be narrowed down to a coarse relation:

$$P_{leak} = V_{DD} \sum_n I_{path(n)} = V_{DD}^2 \sum_n \frac{1}{R_{path(n)}} \tag{4.1}$$

In this equation n is the total number of paths from V_{DD} to V_{SS}, and R_{leak} is the steady state equivalent average resistance of the leaking paths considering all possible input combinations.

Since V_{DD} is predefined when designing cell libraries, only two factors remain to adjust. These factors are n and R_{leak}, the number of paths between the voltage sources and the resistance on these paths. Therefore, the logic families in question in this work either:

- **Reduce** the number of leaking paths
  - This can be achieved for example by reordering logic to serialize transistors and reduce parallel transistors constructs.
- **Increase** the equivalent resistance of the paths
  - Stacking transistors, adding high-V_{th} in series etc. are ways of doing this.
- **Derive** low-leakage input vectors
  - Since the leakage depends on input values, the input vectors causing the least leakage can be applied in inactive periods.

In this chapter a short survey of logic families will be given, discussing which of them are interesting in respect to the three topics above. Thereafter a presentation of static CMOS logic and the selected three design styles; Complementary Pass-Transistor Logic, Domino and MacroCMOS, are presented and evaluated for leakage current characteristics.

4.2 *Survey of logic families*

Before selecting the logic families for evaluation, a survey was conducted. The logic families in this survey are here presented in three categories: Static, differential and dynamic logic families. Representative logic families are here described by category.

4.2.1 *Static logic styles*

Static CMOS is evaluated in this work to form basis for comparison with other logic families. By building small static CMOS cells of logic with simple logic functions, not much more than changing the V_{th} of the transistors can be done. Voltage sources can be disconnected from the logic to reduce leakage in periods of no activity. MTCMOS[23] (Multiple Threshold CMOS) is a way of doing this, adding power routing transistors in series with regular static CMOS logic[4.1]. This is quite interesting since battery powered equipment is typically inactive for most of the time. Therefore MTCMOS is investigated in this work.

Replacing the pull-up or pull-down network with a weak pMOS or nMOS transistor is the design style of Pseudo-n(p)MOS (Figure 4.1). The weak transistor will cause the block to leak less with high output for a pMOS transistor, since the output voltage can drop

[4.1] There seems to be quite some alternating interpretations of this abbreviation. Multiple-Threshold CMOS, MTCMOS, is interpreted both as the power routing scheme as used here in this work, and the idea of altering the threshold voltage by changing the bulk potential.

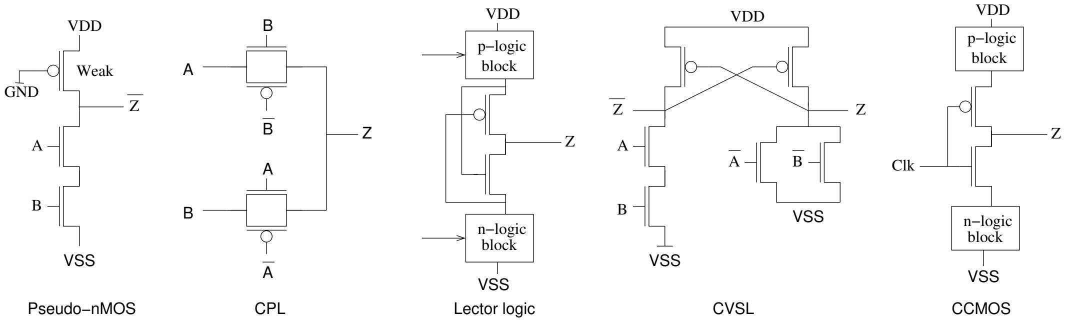


Figure 4.1: *Logic AND-gates designed with: Pseudo-nMOS, Lector logic, CVSL and CCMOS logic.*

without changing the logic function of the block. But, as the leakage of the following block is dependant of the quality of the input value, this will cause the following logic block to leak severely.

Complementary Pass-Transistor Logic (CPL) is very interesting since it reduces the need for connections to the voltage sources and maximized the number of transistor in series. CPL is selected for evaluation.

Lector logic[24] is an enhancement to static CMOS where a cross coupled pair of pMOS and nMOS transistors is put between the pull-up and pull-down networks(Figure 4.1). This puts an extra non-conducting transistor in series with all paths when signals are at a steady value. Yet, this method increases the propagation delay. Scaling up transistors to overcome this overhead will increase the leakage, and it is doubtful how big the benefit of adding a single transistor in series could be.

As described, low leakage design is obtained by increasing the resistance of the paths in the design and generally reducing the number of them. This can be achieved by replacing small logic blocks by larger more complex blocks. Chapter 2 describes the limitation of cell libraries of blocks consisting of a limited number of logic gates. Breaking this boundary by designing cells on the fly can reduce leakage by enabling the design of logic cells fully customized taking leakage current considerations into account. This is investigated. The improved static CMOS 'logic family' is here called MacroCMOS.

4.2.2 Differential logic styles

Cascade Voltage Switch Logic (**CVSL**) [1] is a logic style where two complementary nMOS switch structures are constructed and then connected in a pair of cross-coupled pMOS pull-up transistors(Figure 4.1). All inputs are needed in both inverted and non-inverted form and the pair of pull-down networks doubles the hardware needed. Yet, CVSL gates can be built to be very fast as the pull-down load on the nMOS network is minimized when leaving out the complementary pull-up network. The pull-up network can also be clocked to reduce power consumption, making this gate a dynamic gate. In this way the CVSL gate practically becomes two complementary Domino gates with double hardware and double power consumption, which is why Domino must be more efficient than CVSL.

MOS Current Mode Logic (MCML)[25] is like CVSL built from a pair of pull-down networks and a resistor replacing the pMOS pull-up transistors. A constant current is drawn through the pair of nMOS networks [26], which flows through one of the networks conditionally to the inputs forming a very fast gate.

In general differential logic is not interesting in this work. It requires doubling of outputs for inverted and non-inverted inputs/outputs which will double the number of leaking paths. One could argue, that the currents flowing are not leakage, but dynamic current since the current is used for the fast changing of output values. Yet, reducing the total

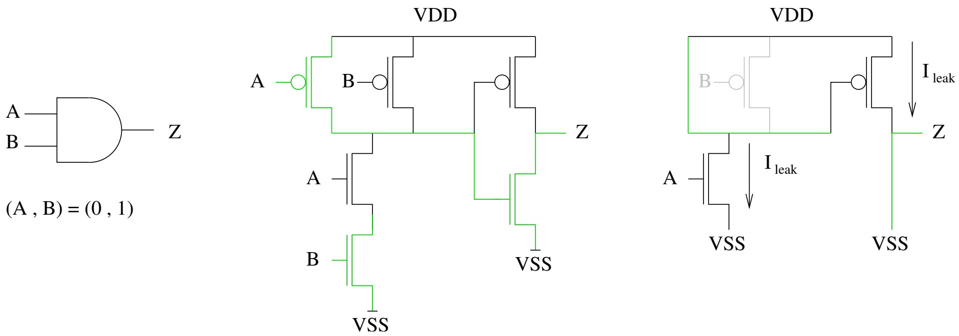


Figure 4.2: *A static CMOS AND-gate. Logic symbol, transistor netlist and leakage.*

power consumption is goal here, so reducing the leakage causing an increased dynamic power consumption beyond the reduction in leakage power consumption is not useful.

4.2.3 Clocked and dynamic logic styles

In the dynamic logic domain there are three very interesting logic families. These are Clocked CMOS (CCMOS), Domino Logic and NP Domino Logic (Zipper CMOS).

CCMOS is built from regular static CMOS with added serial clocking transistors near the output to disconnect the output from the pull-up and pull-down logic networks. The clocking transistors will not introduce much overhead in dynamic power consumption, especially if the clock signal is only turned off in inactive periods. Leakage current is reduced by the stacking effect of inserting the clocking transistors in series. Yet, the speed of the circuit is affected by these transistors.

Domino Logic on the other hand is not affected negatively by the clocking transistors, but these transistors are the key to the very fast operation of the Domino gates. The leakage of the gates can be reduced by these transistors and the speed be utilized for further leakage current reductions. Further, cascading Domino logic blocks alternating between nMOS- and pMOS-implementations reduce the need for inverters. This is called NP Domino Zipper Logic. Domino Logic is therefore selected as a target logic family for evaluation.

The selected logic families for evaluation are then: static CMOS, MTCMOS, CPL, Domino and MacroCMOS which will be discussed further in the following sections.

4.3 Static CMOS logic

Circuitry in static CMOS logic is built using a p-MOS pull-up network driving the output high when the inputs are at certain values and a n-MOS pull-down network driving the output low at all other input combinations. A boolean function is implemented by configuring the p-MOS and n-MOS logic networks in a way that there exists one or more paths from either V_{DD} or V_{SS} to the output through conducting transistors. The computation of the output value and the driving of this output is thereby done by the same transistors. This is illustrated in Figure 4.2 where the green parts illustrate conducting paths in the network.

Having transistors both doing the computation of the output value and the driving of the output (hence the name 'static') is a great advantage of static CMOS since it greatly improves the circuits robustness to noise and irregularities in supply voltages. Internal nodes and outputs are strongly driven by short paths to the voltage sources enabling fast circuitry that is rather insensitive to input and voltage source noise. This eases the design of cell libraries and coding of synthesis tools and leaves the chip designer with a minimum of technology considerations in the design phase.

This great advantage though is growing to be the greatest disadvantage of the static CMOS family in terms of leakage power dissipation as dimensions grow smaller. At all times either the $p-$ or $n-$MOS network is conducting leaving only a semi-nonconducting $n-$ or $p-$MOS network blocking the path between the voltage rails for every path.

This is illustrated in the right hand side of Figure 4.2 where the conducting transistors have been removed and the non-leaking transistor grayed out. The leakage currents of this AND-gate flows through two single transistors forming high leakage in comparison with the logical effort of the gate. To keep up computational speed, more complex logical functions are built from connecting a number of small gates. This causes the number of paths needed from V_{DD} to V_{SS} to rise introducing further leakage.

Trying to avoid leakage currents, one could design the circuitry of a static CMOS cell using high-V_{th} transistors, which inherently reduces the computational speed of the cell. Another way is to size up transistors responsible for most of the leakage. Both solutions lead to slower circuits, and an improvement to the leakage current problem that is dependant on how much speed one would be willing to sacrifice. Hence, static CMOS is not the optimal logic family for high-speed low-leakage circuit design.

The leakage of a cell is dependent on the input values to the cell[27]. Deriving a low-leakage input vector is possible through simulation and applied in inactive periods of time. Special latches can be designed to produce the low leakage input vector when a 'inactive'-signal dictates it[28]. Yet, deriving these input vectors can be highly time consuming, and the results depend on the logic depth. As logic depth increases the likelihood decreases that a good low-leakage input vector causing all cells to leak minimally can be found.

Summing up the key benefits and drawbacks of static CMOS:

Advantages:	Fast, robust, easy to design and synthesize
Disadvantages:	High leakage, low-leak input vectors difficult to derive

4.4 MTCMOS

MTCMOS, which in this work refers to the cutting off power supply concept, will be evaluated for possible incorporation in current cell libraries. Adding power routing transistors inside every cell and adding a 'on' signal to the cell will allow for design, where the synthesis tool can derive a controller to turn specific cells off, and on during operation. Cells can also be connected to the same 'on'-signal to enable for turning on/off regions of logic.

In theory, this could reduce the leakage problem in inactive periods of operation considerably. The leakage power consumption savings must be so large, that a controller can be build consuming less power than the power saved. Further, adding transistors in series with power rails may increase the propagation delay of the cells. This delay overhead must not exceed the delay overhead of using a low-leakage cell, or else a low-leakage cell would be preferred due to ease of design, no controller overhead etc.

MTCMOS will be further explored in section 5.2.1.

4.5 CMOS Domino logic

Logic blocks in CMOS Domino logic are built from an nMOS pull-down network that is precharged through a clocked pMOS transistor driving the drain region of the nMOS network high in half the clock phase, denoted the precharge phase[29]. Thereafter the capacity of the network is discharged through a series nMOS clocking transistor conditionally discharging the network drain region. The output of the domino logic gate is driven by an inverter that is dynamically fed by the drain region of the nMOS network. This makes the domino logic a non-inverting logic with dynamically held output values driving output inverters. Figure 4.3 (left) depicts a Domino logic AND-gate.

During the evaluation phase of the clock the nMOS network can at most make one transition (from logic '1' to logic '0') allowing the output inverter to shift from logic 0 to

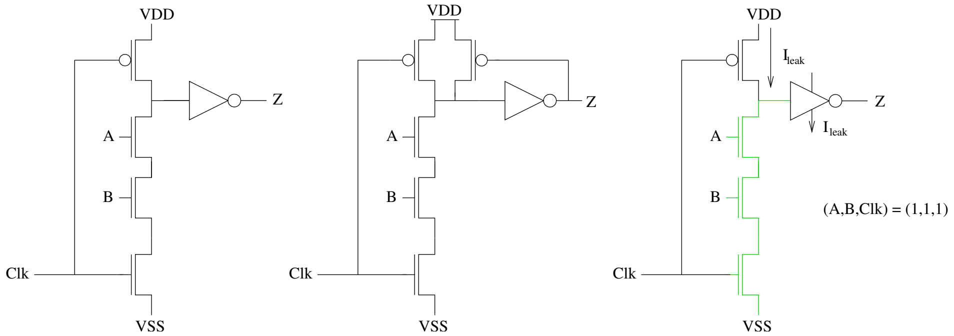


Figure 4.3: *A Domino AND-gate. Basic gate, bleeder transistor added and leakage.*

logic 1. Any number of domino logic gates can therefore be cascaded provided that all gates can evaluate within the evaluate phase of the clock.

Domino logic is designed for speed. The precharging and discharging of internal nodes, which might seem a waste of power, is the price of achieving very fast cascaded logic. After the precharge period the pMOS precharge transistor stops conducting reducing the load on the nMOS discharge transistor to only pulling down the capacitance of the nMOS network.

The primary stage of domino logic will draw some short circuit current as both clocked transistors change state at the same time, but the following stages will have their precharge transistor fully non-conducting when the (conditionally only-rising-edge) input values arrive. Hereby the pull-down load on the nMOS-network and clocking transistor is minimized causing input values to propagate very fast through the stages driven by fast inverters[4.2].

4.5.1 *Trading speed for low leakage*

The speed of Domino logic can be utilized to gain advantages in terms of leakage power dissipation. Firstly, the pMOS-transistor has half the clock period to pull up the network to V_{DD}. This does not require a very strong transistor, and according to the length of the clock period this transistor can be sized to be just adequate to pull up the network. This will reduce the leakage considerably, as either a high-V_{th} transistor can be used, or a low-V_{th} transistor, both sized to leak the least.

Secondly, reducing the drive strength of the nMOS pull-down transistor until the speed of the gate matches the same gate in static CMOS, will reduce the leakage even further. The resulting gate consists of a nMOS network equivalent to the nMOS network of the static CMOS gate with added low-leakage pull-up/down transistors in series. Due to stacking effects this cell will leak less in the precharge phase. Depending on the logic function of the gate and the clock period, the gate will leak more or less than static CMOS in the evaluate, due to the pMOS network replaced by the pMOS clocking transistor.

Since Domino logic is a dynamic logic depending upon a capacitive charge to hold the state of the gate, leaking devices may cause failure in the device. Especially gate leakage is a problem for dynamic logic blocks, as the two gate regions of the inverter in Figure 4.3 will leak and alter the dynamically held voltage node on their gates. This can be helped by adding a state-holding transistor (a so called bleeder or keeper device[30]), which naturally has costs both in terms of dynamic and leakage power dissipation.

[4.2] The analogy here is, that the capacitances in the cascaded Domino logic blocks are discharged like falling Domino bricks, that can only fall, and only be raised (precharged) by hand.

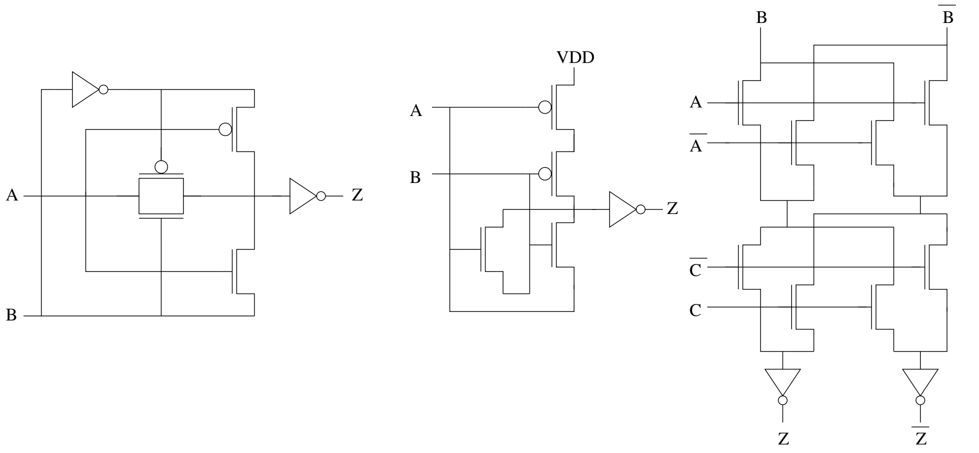


Figure 4.4: *CPL XOR gates in three versions: Yano's 2-input, Wang's 2-input and a 3-input [31].*

Advantages:	Fast Leakage current reductions easy to obtain
Disadvantages:	Increased dynamic power consumption Sensitive to process variations Difficult to design due to clocking issues

4.6 Complementary Pass-Transistor logic

Static and dynamic structures have been described above. These structures are based on the idea of either 'evaluate and drive' or 'precharge, evaluate and drive dynamically'. Both types of structures draw power from the voltage supply to drive outputs causing dynamic power dissipation in every logic depth level. Static structures draw current from the voltage supply to drive the outputs by directly connecting them the appropriate voltage supply. Dynamic structures draw current from the supply in order to form an electric charge large enough to drive the output dynamically for a period of time.

An alternative way is to avoid connecting outputs statically or dynamically to the voltage supply, but to drive the outputs by connecting them conditionally to the inputs. This is the concept of complementary pass-transistor logic (*CPL*). This approach was developed to save dynamic power, area and increase speed [31], but was later superseded by static CMOS again [32]. This was due to the increased number of nodes and transistors required to produce both inverted and non-inverted signals causing higher wiring overhead as well. [32]. In this work the leakage current issues of CPL makes it an interesting contender for low power design once more.

Opposite static CMOS where transistors gates are driven by inputs, CPL utilizes the conditional conductance of a transistor to conduct logic input values. Therefore pass-transistor gates are built of nMOS and/or pMOS transistors with sources typically connected to the inputs and drains to outputs or internal nodes, and transistor gates are controlled by input values or by internal nodes. Pass-gates are typically drawn horizontally to emphasize the flow of logic values from inputs(left) to outputs(right), as can be seen from Figure 4.4.

Driving outputs by the inputs inherently limits the need for connections to either V_{DD} or V_{SS} as many of the output values can be derived from conducting input values directly. In some cases, though, connects are still needed. Regions of logic supplied by a number of inputs can only produce output values that are present in the set of inputs. Hence, a region supplied by logic zero's alone will have no voltage source to produce a logic one on the output. For an AND-gate for example this not a problem since the output value is always represented in the input values, but a NAND-gate is not possible to design without inverting the output.

Furthermore, when driving logic values one must use an appropriate (nMOS or pMOS) transistor to be able to drive the value well enough, which raises the need for inverted input values. Hence, a choice must be made between producing the inverted input values needed for an implementation that is not connected to the voltage supplies, or saving the inverters by making connections to V_{DD} and V_{SS} to drive logic values.

4.6.1 Possible problems with CPL

The speed of the gate is naturally dependent of the drive strength of the previous gate, the load on the output and the drive strength of the gate itself. One could speculate that reducing the number of supply connections, the three factors mentioned before will all worsen reducing the speed of the gates exponentially as logic depth increases. Yet, as no power is lost by pulling the intermediate outputs high or low using voltage supplies, a high output drive is not needed.

Considering a chain of logic blocks in static CMOS, a change in input value traverses through the chain forming a sharp edged wave. This is due to the amplification of each inverter step giving a strong logic '0' or '1' as function of the input voltage. Considering for comparison a long chain of blocks in CPL with reduced number of voltage source connects, a small change in input value will change the output, propagating right through, without the need to wait for each step to trigger and change output value. In this way the output changes slowly (low slope), but might reach the final value as fast as static CMOS.

This is clearly beneficial in terms of leakage due the low number of leaking paths, but the low slope of the output causes the last stage to consume large amounts of dynamic power due to switching currents. Furthermore, when drive buffers or inverters are left out to save leakage, the quality of internal signals are inherently reduced and get prone to external noise which cause gate voltages to fluctuate and cause subthreshold leakage.

This problem is increased in magnitude when cascading CPL gates. The multiple voltage drops over consecutive transistors add to the reduction of signal quality and might even reduce the stability and render outputs unusable even after a number of cascaded steps without driving buffers[31]. This fact and the fact that designing CPL gates is more complicated task, both in terms of formulating logic expressions and in terms of power modelling, are the two main concerns of utilizing CPL for low leakage cell design.

CPL will be evaluated in more depth in section 5.2.2.

Advantages:	Reduced number of connects to voltage sources improve leakage Dynamic power consumption can be reduced by CPL in certain cases
Disadvantages:	Difficult to design High speed requirements may prevent leakage current reductions Low quality and unstable signals may cause leakage and even failure

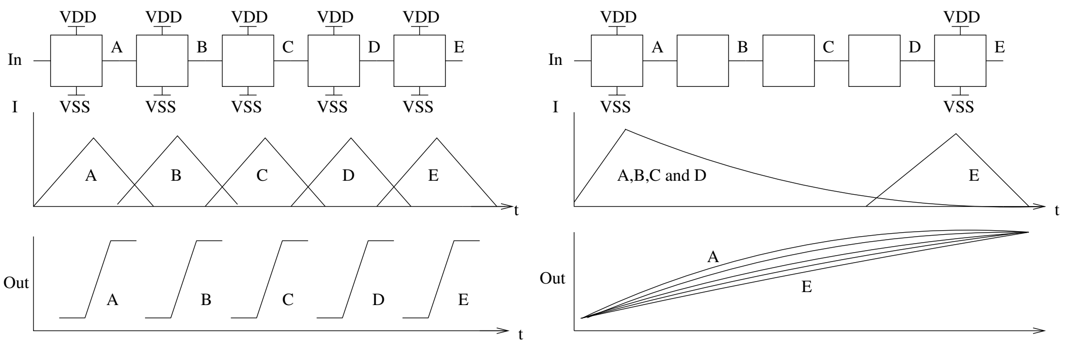


Figure 4.5: *Power characteristics of a traversing input values. Static CMOS to the left with switching current peaks. To the right a CPL implementation with fewer connects to voltage sources.*

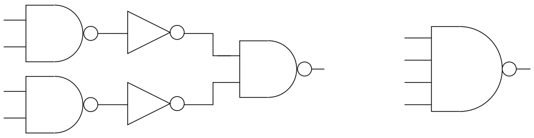


Figure 4.6: *4-input NAND-gates in two versions: cascaded small gates and one larger gate.*

4.7 MacroCMOS

MacroCMOS presented in this work is not a classic logic family itself, but rather a proposed improvement for lower leakage in static CMOS cell based designs. As found earlier the boundaries between logic cells in cell libraries limit the possible optimizations that can done in terms of leakage power. Synthesizing hardware without these boundaries enables the construction of larger customized logic blocks that will have greatly improved leakage current characteristics.

In the beginning of this chapter it was found that leakage current reduction is achieved by reducing the number of paths and/or increasing the resistance on these paths. Both can be done by replacing a number of smaller gates with larger, more complex gates forming the same logic function, which is the concept of MacroCMOS. Larger gates can be designed much more leakage power efficient, because of the stacking effect, improved logic optimization possibilities and the utilization of gained speed for low leakage.

4.7.1 Larger cells for transistor stacking

As described in Chapter 3, the stacking of transistors can be utilized to reduce leakage by orders of magnitude. This is due to the fact that stacking non-conducting transistors decreases the leakage exponentially because of I_{OFF}'s exponential dependency of V_{DS}. By applying random input values to the transistors the number of non-conducting transistors will be statistically higher the more transistors that are stacked reducing leakage.

Figure 4.6 shows a small example of the concept. Here the inverted output of two 2-input NAND-gates drive a third NAND-gate to form a larger NAND-gate. Taking a look at the transistor configurations it is evident that a high number of short paths exist drawing considerable amounts of leakage current. The same figure shows the larger NAND-gate as a stand alone device, and here the number of leaking paths is greatly reduced. Further, the path of nMOS transistors contain four nMOS devices which will reduce leakage drastically in most of the 16 possible input states. Measuring with HSPICE with random inputs, the large NAND-gate consumes leakage power more than a factor of ten less than the equivalent small gate implementation.

4.7.2 Logic optimizations for low leakage

The second effect of using larger, fully customized cells is the possibility of further logic optimizations. Using the same example from Figure 4.6 one will notice that the two inverters have been cancelled out through logic optimization, and only four pMOS pull-up paths are needed. This reduces leakage a large amount, since inverters and single transistors are the most leaking devices.

Most cell libraries contain a basic four-input NAND-gate, so the example seems a bit far fetched. But the inspection of the $0.18\mu m$ STM cell library available at IMM/DTU showed that from the 777 cells of the library only 157 different logic blocks are available. The rest are duplicates with altered drive strengths, drive buffers in various drive strengths etc. Most of these cells have a maximum of five inputs and larger cells are typically special purpose like eight-input multiplexors etc.

Larger functional blocks are built from either smaller blocks or by adding a few inverters on the inputs of a larger cell, which is costly in the leakage current budget. Hence, a more elaborate and very real example must be devised to prove the benefits of logic optimizations when ignoring cell logic boundaries.

Logic optimizations are not always possible though. During the work of this project it became apparent, that some logic gates perform badly when built into a larger cell. Chapter 8 elaborates more on this subject.

4.7.3 Utilizing speed for leakage reduction

Once more figure 4.6 serves as an example. Building the four-input NAND-gate and comparing the timing of the gate to the timing of the cascaded two-input case, the larger cell will have improved pull-up propagation delay due to the directly connected single pMOS transistors. This can be utilized to save leakage. Sizing these transistors to be adequately weak, leakage current is reduced in the (1,1,1,1)-input state.

In a more elaborate example paths can be found that have lower pull-up or pull-down delay than other parallel paths. Transistors on these paths can be sized to reduce leakage, or even replaced by low leaking high-V_{th} transistors.

In some cases it is not possible to utilize speed for leakage reduction since the derived larger cell is slower than the equivalent smaller cell implementation. If no reasonable improvements can be done to improve speed without increasing the leakage beyond the leakage of the smaller cell implementation a larger cell implementation is not feasible. Chapter 8 discusses this further.

Generally, this approach seems optimal since it uses a well known static logic family that is easy to design and test. It carries all the benefits of static CMOS and remedies to a certain extend the leakage current problem. Further, many synthesis approaches can be reused and design engineers do not need to change their work flow. Fully customized cells can be built in a post synthesis process, which allows for the reuse of most synthesis tools and design methods, including architectural considerations. This will also be discussed further in Chapter 8.

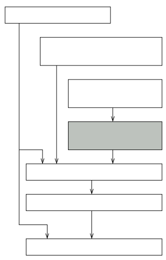
Advantages:	Numerous possible optimizations for low leakage No loss in speed through optimizations Low leakage input vectors easy to define due to low logic depth
Disadvantages:	Cell library of predefined logic function blocks not possible

CHAPTER 5

LOGIC FAMILY EVALUATION METHODS

Contents

Evaluation of the logic families requires great care taken when devising fair and comparable simulation cases. The same care has to be taken when designing a fair and average-case set of static CMOS logic gates to enable fair comparison. Furthermore, the results from the simulation cases need further treatment in order to give comparable values. This chapter describes the considerations done for designing simulation cases and generating a static CMOS set of gates for comparison. Then, the steps of building and optimizing the logic blocks built with the selected logic families is described. After these general remarks specific implementation remarks are given for each logic family.



5.1 Logic families comparison

Comparing logic families is a delicate task. Firstly, every logic family has its characteristic pros and cons when utilizing the family in certain design styles or even building specific logic blocks. Secondly, the design space of logic gates is vast. All transistors can be scaled in gate length and width and be connected in many alternative ways forming the same logic function. Furthermore, the building of larger logic functions from smaller logic gates can be done in numerous ways, which adds to the size of the design space for a logic function with speed and power (and area) as critical design parameters.

Comparing logic families by comparing specific example logic functions can evidently only be done with success when taking great care of the selection of the logic functions to be implemented. Logic functions for simulation must be selected not to favor certain logic families, and timing and speed requirements must also be defined to emphasize a fair comparison. Furthermore, comparisons with the static CMOS logic family can only be done when the logic families are being compared to fair average-case implementations of logic functions in static CMOS. Devising these implementations is the first task.

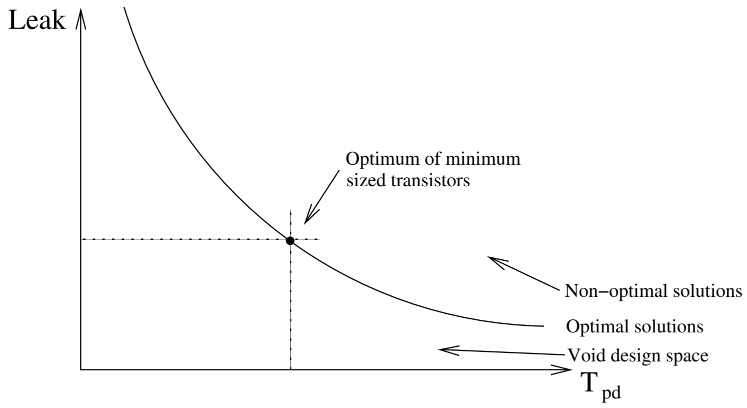


Figure 5.1: *Speed/power design space with the optimal curve as a boundary between non-optimal and impossible solutions.*

5.1.1 A static CMOS basis for comparison

The need for a fair comparison basis is evidently shown by the following example. An example logic gate utilizing the static CMOS logic family that is so poorly designed that any logic family will prevail over static CMOS is easily devised. Consider a static CMOS NOR-gate with one very wide and one very long nMOS transistor. This gate will draw large leakage currents in its high output state and it will have a large worst case propagation delay due to the slow long-channel nMOS transistor. This gate can be built in any logic family with better performance in terms of leakage power and speed if, that is, the static CMOS nor-gate is designed poorly enough. That implies that great care must be taken when designing the basis for comparisons.

This basis itself must be an optimum solution regarding a defined cost function with speed, power, etc. as function parameters. For comparison a set of gates must be designed to have a fair relation between speed and leakage power, not giving great advantage to either of the two, which may rule out specific logic families. This means that a tradeoff between speed and power is needed, which lies on the optimal curve in the speed/power design space.

5.1.1.1 Optimal curves

In the vast 'speed/power'-design space of implementation solutions three regions can be defined. Figure 5.1 illustrates this. For every propagation delay (inverse speed) an optimum solution can be found in terms of low leakage power, and vice versa, i.e. if a maximum propagation delay is defined (typically by the clock speed) there exists an optimal low-leakage solution for this given logic family. On the other hand if a maximum leakage current limit is defined, then a minimum propagation delay exists, defining the optimal solution. Both solutions are present as a point on the optimal solution curve. The space above the curve represents all non-optimal solutions. The space below the curve is void, and no solution can be found here (or the curve would not be a optimal solution curve).

Defining a set of gates to represent the static CMOS logic family in comparisons is done by first deciding the device sizings of the gates. This is found as a point on the optimum solution curve. One optimum solution is building logic gates with all minimum sized transistors. This solution is not the best solution in terms of either leakage power or speed, but it is represented on the optimal solution curve.

This can be argued by looking at the characteristics of MOS transistors. Starting out with minimum sized transistors one could improve speed by increasing the width of the transistors, but this will increase the leakage of the gate as well. Or one could reduce leakage by increasing the gate lengths, but this will reduce the drive strength and thereby the speed

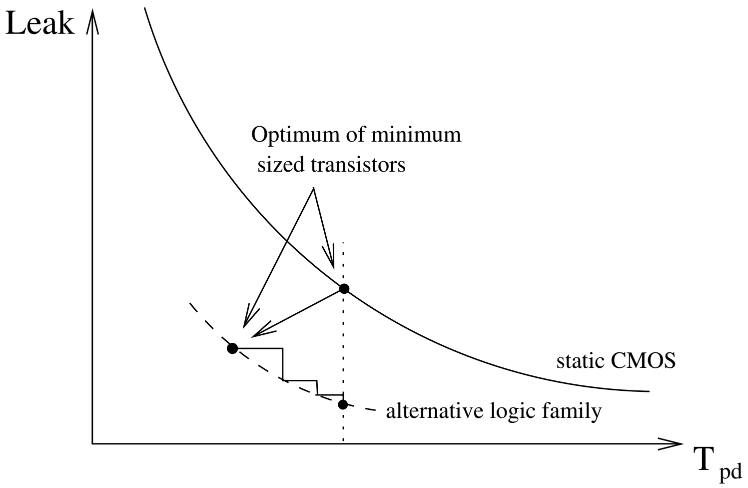


Figure 5.2: *Finding an optimal solution in the speed/power design space for a given logic family compared with static CMOS.*

of the gate. If one would define, that the maximum propagation delay and leakage current of the gate must be the equivalent to the minimum sized transistor gate, no improvements can be made. Therefore this solution is on the optimal solution curve.

One could argue that both the transistor width and length could be increased to improve the solution, but that solution would be the same as returning to an older technology with larger device sizes. Therefore the minimum sized static CMOS gates are used as a basis of logic family comparison.

5.1.1.2 Selecting a set of logic gates

Clearly not all possible logic gates can be simulated in reasonable time, and that would not be necessary either. As stated in [33], 20 logic gates are necessary to form a valid comparison set of gates. The work done in [33] is based upon minimizing a cell library by excluding the logic gates that were used least often when synthesizing a set of simulation circuits. This library of only 20 logic cells including flip-flops forms a complete cell library reduced to only 20 cells with minimum delay and power dissipation overhead. The 20 cells described in this paper is used here as comparison basis representing the static CMOS logic family.

5.1.2 Logic family comparison steps

Comparison between logic families is now possible by building the selected logic blocks utilizing the given logic families with minimum sized transistors. For each logic family this gives a point in the speed/power-graph. This point will be situated on the optimal solution curve for the given logic family, due to the utilization of minimum sized transistors. If this point lies below the curve for static CMOS, proof has been found that this logic family is better for the implementation of the specific logic block.

Further improvements to the implementation can be done decreasing the leakage of the gate by paying some speed. This is illustrated in Figure 5.2. Here an implementation has been proved to be better than the equivalent implementation in minimum sized static CMOS logic. The zigzagged curve represent iterative improvements to the implementation towards the goal of minimum leakage under the same time constraint as for the static CMOS logic implementation. Drawing a curve through the iteratively achieved solutions yields a piece of the optimal solution curve of this logic family for this specific logic function.

The above described procedure is the approach taken in the analysis of logic families in this work. First a basis is built in static CMOS (minimum size) and then the equivalent is

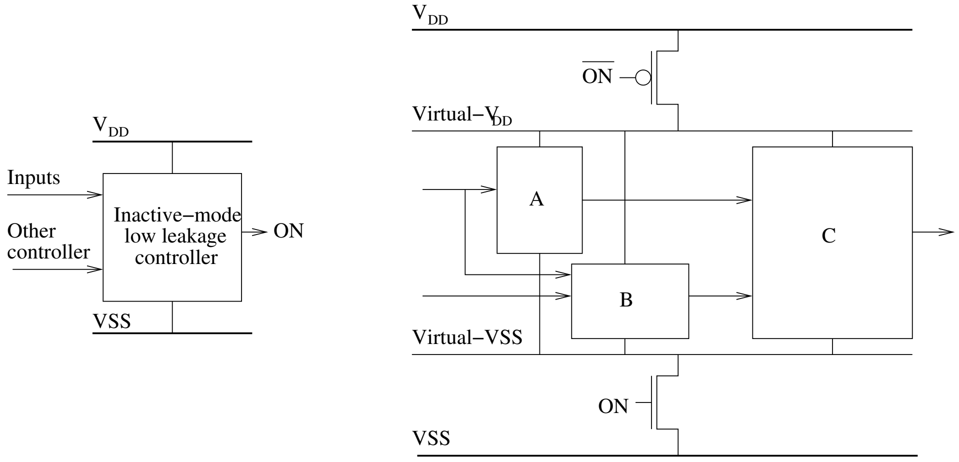


Figure 5.3: *A inactive-mode low leakage controller, controlling the power supply to virtual supply voltage rails powering logic blocks (A, B and C).*

built in the logic family under evaluation. If the achieved solution is better than the static CMOS solution in terms of speed, an iterative approach is taken to utilize the gained time slack slowing down the logic block and achieving less leakage power consumption.

On the other hand, if the solution should prove to be worse, steps are taken to improve the speed taking the leakage current into consideration. Hence, the propagation delay of the logic block is the critical parameter, and the leakage current is derived as the product of improvements done under the strict timing limit. The final solutions can then be compared directly in terms of leakage current.

5.2 Logic family specific simulation approaches

In this section the approaches to simulate the specific logic families together with the technique of cutting off power supply in inactive periods are described. General considerations and remarks about the expected results are also given here.

5.2.1 Cutting off power supply

Leaking devices cause power consumption not only when the circuitry is working, but also in inactive periods, as described in Chapter 3. The fraction of the total power consumption that is caused by leakage current is dependent upon the utilization of the circuit, i.e. the percentage of the time a given circuitry is working. For a system with parts with low utilization, this fraction grows quite high.

An obvious method of leakage current reduction is to cut off the power to inactive regions of logic by routing the voltage supplies through transistors controlled by a designated 'inactive-mode low leakage controller'. This is often called MTCMOS in the literature[23]. Using transistors to cut off power to large areas of logic causes large voltage swings and possible failure when re-activating the circuitry. To avoid this, small regions of logic can be cut off and reactivated independently using a more complex controller and several cut off transistor stages.

Figure 5.3 shows the concept. The controller can listen to inputs or other controllers, such as a controller of an input queue, to be able to decide when to put the logic blocks into sleep mode.

5.2.1.1 *Voltage supply swings*

The virtual supply voltage rails, from which the logic block will be drawing power, will naturally be affected by the voltage drop over the two transistors. This voltage drop is dependant on the current drawn by the logic block, which leads to swings in the virtual supply voltage when the logic block is working. These swings impact the propagation delay of the logic block causing increased delay. Increasing the drive strength by sizing up the width of the supply voltage transistors reduce the voltage swings and the voltage drop, but inherently causes increased leakage when power is to be cut off from the logic block.

Increasing the length of the power supply transistors reduces the leakage, but reduces the drive strength of the transistors further increasing the virtual supply voltage swings. Using high-V_{th} transistors may reduce the leakage without causing a too severe increase in propagation delays that may be remedied by sizing up the width of the transistors and still saving leakage power overall.

It is clear that a study of the effects of adding circuitry for cutting off power supply must be conducted. The width, length and threshold voltage of the transistors feeding the virtual supply voltage rails are the parameters for this study. The outputs are propagation delays and leakage current measurements for a set of logic blocks designed for simulation purpose.

The set consists of two simulation cases, which will be shown to be sufficient for this study. In the first case the logic block is represented by a resistor simulating a leaking circuit. This rather simple case enables comparisons of the effectiveness of adding the supply voltage transistors in different sizings without taking the dynamic characteristics of the logic block into consideration.

This forms the basis for the second case where a logic block consisting of a 'NAND-NOR' structure is simulated for propagation delay and leakage current. Comparing the results from this case to the simplified resistor case helps locate characteristics originating from this specific (NAND-NOR) simulation case that might invalidate the general conclusion derived from this simulation.

5.2.2 *Complementary pass-transistor logic*

Investigations of low power logic styles reported in the literature are mainly based on full-adder designs [32]. A full-adder consists of a 3-input XOR gate and an 6-input AND-OR structure. This forms a good basis design for exploring logic style dependent benefits and drawbacks.

The XOR gate is in many logic families impossible to improve on, i.e. very few optimizations can be achieved by transistor reconfiguration. Due to the input passing nature of CPL, XOR gates can be designed using very few transistors, which is one of the key benefits of CPL. The AND-OR structure can be optimized in different ways when utilizing different logic families, so this structure is too an interesting design for logic family evaluation. The full-adder will be used in all logic family evaluations.

5.2.2.1 *CPL design styles*

In CPL quite a few ways of designing XOR gates are possible. Figure 5.4 depicts three different implementations. The implementation to the left is a mix of static CMOS and pass-gates. This design includes two inverters, which are expensive in terms of leakage.

Wang's XOR gate in the middle of Figure 5.4 is a true CPL gate including only one inverter for driving the output value. Eliminating the inverter yields a XNOR gate with weak pull-up with inputs A=1 and B=1 [32].

The third XOR-gate is 3-input true CPL XOR-gate[31]. This gate is built of only nMOS transistors, and as the pull-up of these transistors grows more limited as the number of

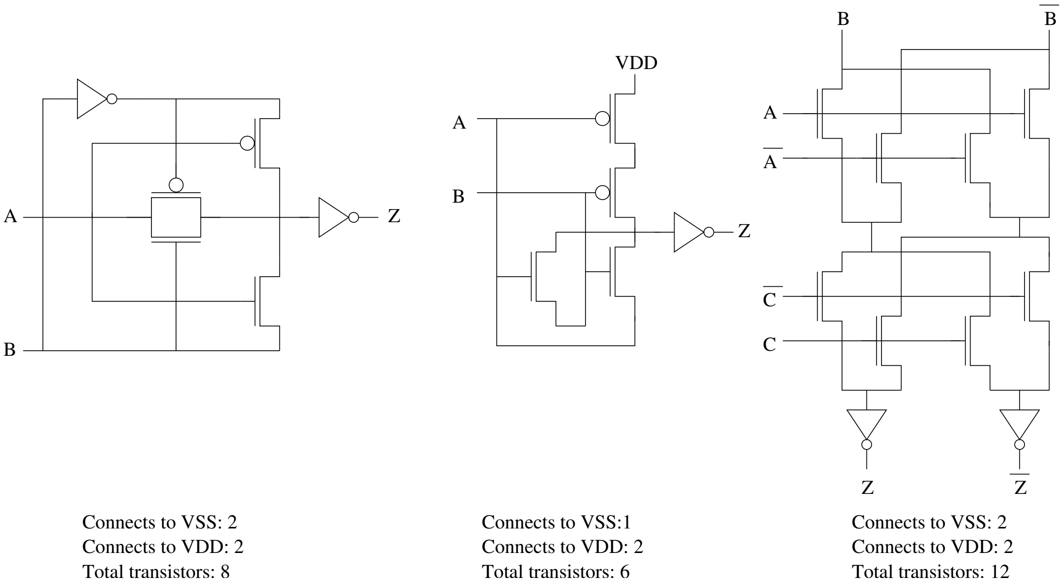


Figure 5.4: *Three different implementations of XOR gates in complementary pass-transistor logic with different number of connections to the power rails.*

transistors in series increases, a better implementation can be built adding pMOS transistors to form pass-gates instead of single nMOS pull-up/pull-down transistors.

In the evaluation of CPL all three different types of XOR implementations were simulated and both speed and leakage power characteristics were explored. The speed gain from changing static CMOS to CPL would have been used to further decrease the leakage of the CPL gates. Though, due to very poor results from this analysis, no further simulation was done in CPL.

From the XOR case it was determined that the weakly driven signals cause more leakage power consumption than what was saved due to reduced number of supply connections. This will be described further in Section 6.3.

5.2.3 Domino Logic

The investigation of Domino logic relies on two key evaluations. First, it is determined to which extent the clocking transistors in a Domino block can be designed to reduce the leakage in the block in both clock phases. Secondly, gate leakage is taken into account as measures must be taken to guarantee the functionality of Domino blocks under the presence of leaking gates.

5.2.3.1 Transistor scaling for low leakage

Scaling a Domino block down in speed to match static CMOS by scaling the clocking transistors to save leakage is done in a series iterative steps. First the pMOS transistor is scaled to be exactly strong enough to pull up the stage. This is shown in Figure 5.5 in the *Precharge* phase of the clock. The arches a, b and c represent a too strong, an appropriate and a too weak pull up respectively. It is, off course, not possible to pull up entirely to V_{DD} within a given clock phase, so another required minimum value must be set. Here, the pull-up is required to pull-up to $V_{th}/8$, which was found to be achievable without too severe impact on leakage through the pMOS device. The clock frequency is set to $1GHz$.

Secondly, the nMOS pull-down transistor is sized to be exactly strong enough. This is shown in the *Evaluate*-phase of Figure 5.5. Here arches d, e and f represent a too weak, an appropriate and a too strong pull-down respectively. The pull-down nMOS transistors

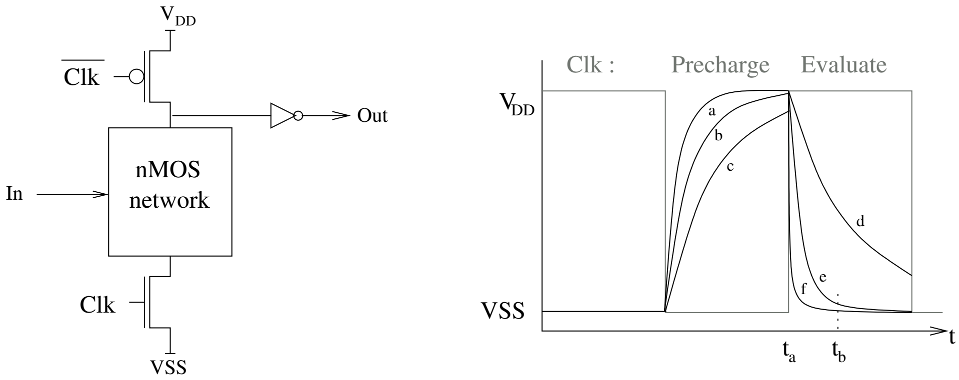


Figure 5.5: *A Domino block with clocking transistors. The pull-up and pull-down of the block.*

does not have the entire *Evaluate* clock phase to pull down, as following Domino blocks are waiting for the output. The maximum pull-down time including the propagation delay of the output inverter is set to be the propagation time of the corresponding static CMOS gate. The maximum propagation delay without the inverter delay is shown on Figure 5.5 as $t_b - t_a$.

After the nMOS transistor has been sized for minimum leakage, pull-up and pull-down times are checked again, to verify operation again with the added capacitive load caused by the larger nMOS device. A optimal solution is found by iteratively sizing the two transistors. The design chosen for simulation is again the full-adder, which will enable easy comparison with CPL and static CMOS implementations.

5.2.3.2 Simulating gate leakage

Gate leakage, although generally not included in this work, will have a very bad influence on the performance of dynamic logic. When gates in the output inverter start leaking, the dynamically held input to the inverter must be helped by a bleeder transistor to keep the high signal value. Designing a MOS device to keep an internal nodes voltage value very near V_{DD} is a tradeoff between keeping a high quality signal value using a large pMOS device causing little subthreshold leakage in the inverter, but large amounts of leakage through the rest of the Domino block, or using a smaller device causing the opposite effects.

Clearly, gate leakage causes further leakage when trying to remedy the effects of gate leakage. In this study the gate leakage will be approximated by an resistor connected from ground to the dynamically held nodes. The analysis described above in the previous section then repeated with the added current source.

5.2.4 MacroCMOS gates

The evaluation of MacroCMOS gates is done in three steps. First, the full-adder is again used for comparison reasons. Then a block is built to show that the benefits of logic optimization without cell boundaries is a powerful way of reducing leakage. Further, the third simulation case explores the decreases in leakage that building larger cells for transistor stacking may bring.

5.2.4.1 The full-adder

The concept of MacroCMOS is to form larger gates from either smaller gates or direct boolean expression synthesis to reduce leakage through logic optimizations and stacking of transistors. The full-adder design is not optimal to show the benefits from using MacroCMOS due to the parallelism of the full-adder design, that includes two entirely disjoint components. It will still be designed for comparison purposes.

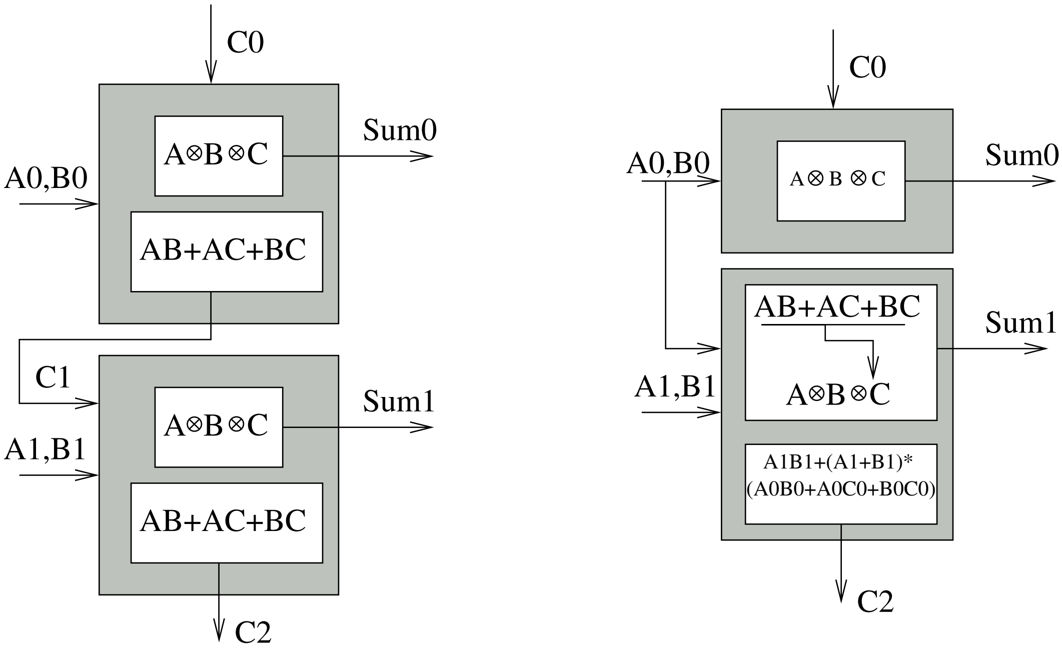


Figure 5.6: *A 2-bit standard full-adder and a leakage improved MacroCMOS 2-bit full-adder.*

Figure 5.6 shows a possible way to construct a larger block from the smaller ones. Two full-adders have been joined into one block by including the carry-computation in the sum-computation in the next stage. The carry *C1* does not exist anymore, but the corresponding evaluating networks have been incorporated in the 3-input XOR gate in the next stage.

The component calculating *C2* becomes somewhat larger since *C1* does not exist, so the carry *C2* must be determined from the four input values and *C0*. Comparing the *C2* carry generator to the original one, it is evident that a AND function (*) is introduced which is good in terms of leakage because this implies chaining transistors in series. The full-adder is selected to be used in the evaluation of MacroCMOS for comparison.

5.2.4.2 Logic optimization

To investigate what logic optimizations can be done to a circuit when the limitations of cell libraries are ignored, a logic block is devised for simulation. The STM cell library available at IMM/DTU offer a modest number of larger cells with a maximum of five or six inputs. These cells are inherently not optimal in terms of leakage due to the fact that they have been designed for common purpose usage. Therefore a logic block matching a cell in a common cell library is applied with inputs that enable logic optimization in MacroCMOS, but not possible with current cell libraries. This optimization is not possible with the current cell libraries, but only when manufacturing cells on-the-fly.

A larger logic block, for example, connected with the same input connected to more than one input terminal could be rebuild to reduce leakage. This is done by reconfiguring the transistors, removing superfluous transistors and resizing other transistors to reduce leakage while still keeping the original timing of the gate.

5.2.4.3 Larger cells for transistor stacking

Evaluating the benefits from building larger cells for stacking is a delicate matter since the results will depend heavily of the particular simulation case. As will be shown in section 6.5 the leakage per input of a XOR-gate increases when replacing a larger XOR-gate with smaller ones in cascade, while the opposite is true for a NAND-gate.

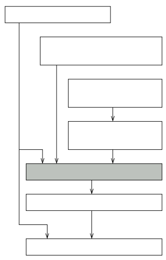
This implies that a randomized case consisting of a variety of different gates in cascade is needed where optimization can be done, and from which general optimization methods can be derived. Here, an 11-input gate with total of 9 distinct inputs will be examined for this purpose.

CHAPTER 6

EVALUATION OF LOGIC FAMILIES

Contents

This chapter describes the evaluation of target logic families through the simulation cases presented in Chapter 5. The results from each evaluation will be discussed. The following chapter contains a comparative discussion of all the simulation results. The files used for simulation are included in the attached disk. Appendix F gives a short outline of the contents of the disk.



6.1 Static CMOS

For the purpose of comparison between the selected logic families, the minimized set of 20 logic cells described in [33] was implemented and simulated with HSPICE.

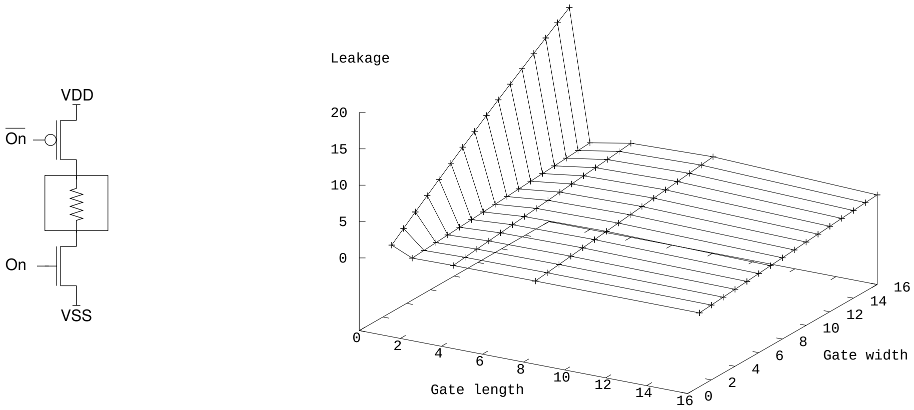


Figure 6.1: *Leakage current of a 36.5 Ohm resistor driven by virtual voltage supply transistors.*

Leakage currents were measured as steady state leakage current drawn from the voltage supply through the circuit for every possible input. The average leakage current was then calculated under the assumption that every input value combination is equally frequent.

The input vectors causing minimum and maximum leakage current were recorded together with the corresponding leakage current to enable derivation of low leakage input vectors at a later stage. To measure propagation delays of the circuits the worst case shift from one to another input vector causing maximum output delay was predicted by hand and investigated by simulation.

Further descriptions of the 20 cells, including logic functionality, transistor netlists and simulation results, is printed in Appendix D.

6.2 Cutting off power supply

As described in section 5.2.1 cutting off power supply to inactive regions of logic MTCMOS style will reduce the effect of leaking devices in the total power budget. The investigation of the effects of doing so is explored in this section. The technique is explored through two cases, a resistor case and a NAND-NOR case, which is described in the following sections.

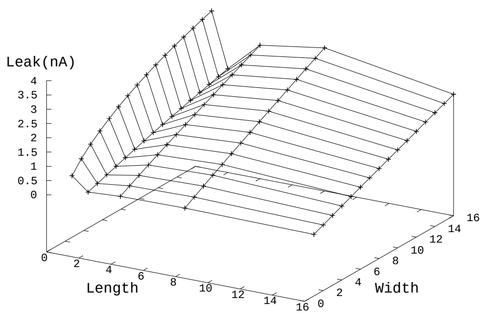
6.2.1 The resistor case

The investigation of the possibility of cutting off power supply to inactive regions of logic begins with routing power to a linear resistor through virtual voltage supply transistors and simulating for a range of transistor widths and lengths. Figure 6.1 shows the leakage current (in nano-Amps) of a 36.5 $M\Omega$ resistor driven by $70nm$ HS supply transistors sized in the range 1 to 16 times W_{min} and L_{min}. The resistor value of was selected by approximating the resistance of the NAND-NOR-structure as:

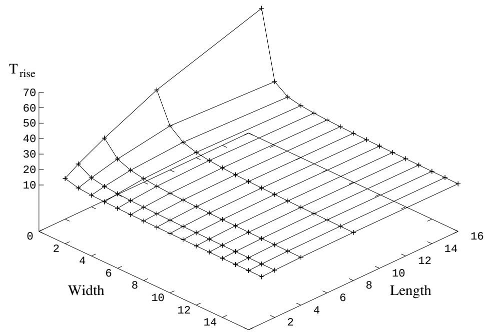
$$R(V) = \frac{U}{I_{Leak}} = \frac{1V}{2I_{Leak-nand} + I_{Leak-nor}} = \frac{1V}{2 * 7.9nA + 11.65nA} = 36.5M\Omega, V = V_{DD} \tag{6.1}$$

Figure 6.1 depicts the leakage current as function of length and width of the voltage supply transistors. The width is set to in the range 1 to 16 times W_{min} and length steps through 1, 2, 4, 8, and 16 times L_{min}. The leakage current grows approximately linearly with transistor width as expected and is reduced non-linearly with increasing gate length.

For a given gate width and increasing gate length it is easily seen that the leakage drops off sharply when increasing the gate length to 2 times L_{min} and it is expected that the leakage drops further with increased gate length. This change is not visible from figure 6.1 though due to the characteristics of the transistor model cards as described in section



(a) Leakage



(b) Propagation delay

Figure 6.2: Leakage current and rising edge propagation delay of a nand-nor structure driven by virtual voltage supply transistors.

3.3.2. Instead the curve becomes slightly bend upwards with medium long gate lengths, as shown in figure 3.9 on page 31.

6.2.2 *The Nand-Nor case*

The resistor is now substituted by a 2-input NOR gate driven by two 2-input NAND gates. This NAND-NOR structure can be cascaded to explore deeper logic depths, but it was found that a no further conclusions could be drawn that way other than that the propagation delay problem increased in magnitude.

First the leakage current through the circuit was measured for the same range of gate lengths and widths as in the resistor case. Figure 6.2(a) depicts this. The same relation between gate length, width and leakage current can be observed. The bending of the curve here seems much more pronounced than in the resistor case. This is due to the decreased leakage current in general. The decrease of leakage current comes from the chaining of the non-linear transistors, where the current drawn is exponentially falling with decreasing V_{DS}, see equation (3.9). Figure 6.2(a) in comparison with figure 6.1 shows that the NAND-NOR example does not show characteristics that could invalidate further exploration of this technique.

Adding transistors in series alters the timing of the circuit. Even though the power routing transistors are set in conducting mode for a large period of time, and thereby the voltages on the drain regions of these transistors are pulled to V_{SS} and V_{DD} respectively, the timing of the circuit is changed. This is due to the current drawn by the logic blocks making the virtual voltage supplies swing in voltage. This behavior is examined by measuring propagation delay of the NAND-NOR stage fed by power routing transistors in the before used sizes. Figure 6.2(b) presents this propagation delay. [6.1]

6.2.2.1 *Added delay*

The minimum propagation delay is achieved with the maximum width of 16 times W_{min}. The propagation delay is then 102 to 108 ps for gate length varying between 1 and 16 times W_{min}. Scaling down the gate width to W_{min} increases the propagation delay to the range 140 to 646 ps. The NAND-NOR stage without power routing transistors has a total delay of 72 ps according to Table D.1 in Appendix D. [6.2]

Routing power through transistors is evidently not without cost in terms of timing. Hence, saving leakage power utilizing power routing transistors comes down to a tradeoff between speed and power in the end.

[6.1] Please note that the graph is rotated in comparison with the leakage graph to show the curvature of the graph.

[6.2] Falling delay of a nand2-gate, 28 ps, plus the rising delay of a nor-gate, 44 ps. This has been verified by simulation.

Circuit	Width	Length	Prop. delay	Leakage
Regular voltage supply	-	-	72 ps	27.45 nA
Cut off voltage supply HS	6*W_{min}	4*L_{min}	126 ps	1.06 nA
Cut off voltage supply LL	6*W_{min}	4*L_{min}	135 ps	0.0147 nA
Difference HS			+ 75%	- 96.1%
Difference LL			+ 87.5%	- 99.95%

Figure 6.3: Propagation delay and leakage at different power routing transistor widths and lengths.

Comparing data from figure 6.2(b) and 6.2(a) a good tradeoff is selected at the point (W,L) = (6,4). Here the timing is in the flat area of the timing graph and the leakage current is in the low range of the leakage graph. Table 6.3 presents this situation with propagation delay in 'on' mode and leakage current in 'off' mode. The same simulation was conducted with low-leakage (LL) power supply transistors. The general conclusions are the same, but the specific results are inherently different.

It is clear that the great reduction in leakage current has a cost of speed of the circuit. As described, this is due to swings in the virtual voltage sources as the logic block draws power. Figure 6.4 shows the extent of the voltage swing for power routing transistors of minimum dimensions. As the output is driven high, the voltage difference between the virtual V_{dd} and ground gets as low as 0.67V, which is a 33% decreased supply. Sizing the width of the supply transistor up to 4 times W_{min} decreases this swing to 9%.

The regular V_{dd} can be raised to compensate for this voltage swing, and this was done in the tradeoff situation mentioned in Table 6.3. The supply voltage was increased until the timing of the circuit was back to its original level. At $V_{dd} = 1.24V$ the timing was comparable to original timing of the circuit. This increase in supply voltage leads to an increase of the leakage of the circuit in 'on' mode from $27.4nA$ to $53.4nA$ and to increased dynamic power consumption, which will not be pursued here further.

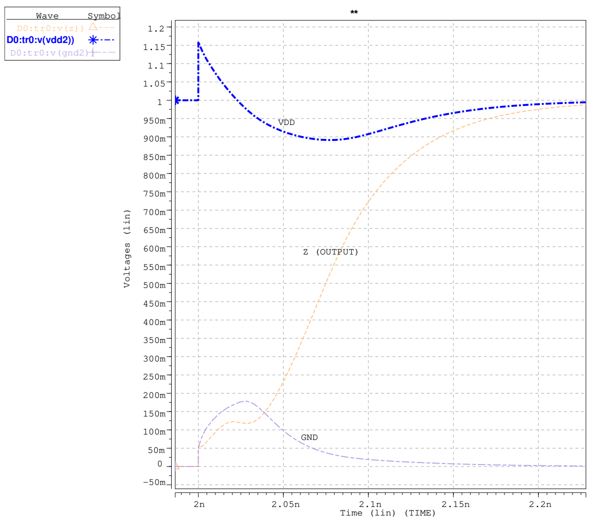
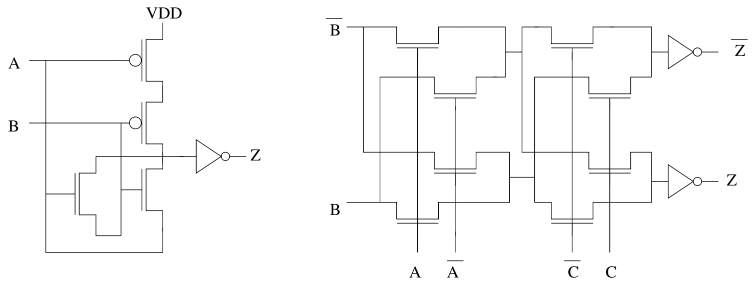


Figure 6.4: *Virtual ground and virtual Vdd voltage swings under rising edge shift of a nand-nor structure.*

6.2.3 Discussion of results

Clearly, cutting off power can reduce the leakage current of logic blocks, but it does come at some expenses. The timing is unavoidably affected, demanding a tradeoff between speed and leakage power to be made. Great improvements to the leakage problem can be made



Wang's 3−input CPL XOR gate

3−input CPL XOR−gate

Figure 6.5: *Wang's 2-input and a 3-input[31] CPL XOR-gate.*

this way when the timing allows for it, but achieving a solution with minimum timing overhead is difficult without scaling the power cutoff transistors into the very large.

The timing overhead can be sought to be remedied by increasing the supply voltage. Yet it was shown that a very high increase ($\tilde{2}4\%$) was needed to restore the circuit to full speed. This voltage increase causes the entire circuit to leak more and consume considerably more dynamic power. If enough time slack was available in the design, a more feasible way of leakage current reduction would be to utilize the time slack for replacing high-speed with low-leakage transistors. Using only one of the two power routing transistors to reduce the effects on the propagation delay only solves the problem partially and reduces the low leakage benefits.

The leakage power consumption savings only apply in inactive mode. This greatly limits the usage of this method, but as it introduces no power overhead (other than the power consumed by the controller) nothing can be lost (other than computational speed). Care has to be taken when designing this way, though. Supply voltage swings may reduce the noise margins of the gate resulting in failure in the worst case. [27, 34]

Summing up, adding power cutoff transistors reduces the leakage current in the circuitry, but has drawbacks:

- Added area is needed for transistors, controller and routing
- Added delay for all circuits with reasonably sized power supply transistors
- No power saving in 'on' mode
- A customized controller is needed consuming hardware designers time and possibly introducing design faults
- Unstable supply lines may introduce failure

6.3 *Complementary pass-transistor logic*

The investigation of CPL begins with design and simulation of the two different 3-input XOR gates described in [32] and [31]. The layout of the two selected implementations was earlier depicted in figure 6.5. These two designs will here be denoted as *Wang's* and *Yano's* XOR gates, due to the author (Yano) of the paper where the design were adopted from [35]. A 3-input XOR gate is built from two 2-input Wang's XOR gates in cascade. Wang's 2-input XOR gate will be examined here followed by an analysis of the properties of the cascaded Wang's XOR gate.

For comparison, a 2-input and 3-input static CMOS gates were simulated for propagation delays and leakage current. Leakage was measured both with and without the internal inverters. Results are shown in Table 6.1.

Family	Logic gate	Tpd rise	Tpd fall	Leakage	Leakage w/o inverters
Static CMOS	2-input XOR	58 ps	121 ps	17.7 nA	7.9 nA
Static CMOS	3-input XOR	320 ps	160 ps	25.37 nA	10.6 nA

Table 6.1: *2- and 3-input XOR gates in static CMOS cells simulated with 70 nm HS BPTM model cards.*

6.3.1 Wang's XOR gate

Table 6.1 sets the design boundaries for the CPL designs. Regarding these boundaries Wang's 2-input XOR gate was implemented and simulated. The results from this analysis is shown in Figure 6.2. Leakage is in this table the total leakage of the gate without the leakage of driving inverters on inputs. The leakage of $9.9nA$ is quite surprising as there are no connections to V_{SS}. The leakage flows from V_{DD} to ground through the inputs.

As the inputs are ideal voltage sources (as described in Section 3.3.3), HSPICE draws any amount of current from this node to ensure perfect input voltage levels. By closer inspection it becomes evident that with the (0,0)-input combination, two nMOS transistors in parallel are leaking causing high leakage levels. In the (0,1)-input situation the gates of the inverter is driven by a weak logic '1' which causes the inverter to leak considerably. Since the main part of the leakage of the gate originates from leakage through the nMOS transistors, these were sized up in length within the propagation delay limits. This design is the 'improved' design in Figure 6.2. Leakage was reduced by almost a factor of four.

Family	Logic gate	Tpd rise	Tpd fall	Leakage w/o output inv.
CPL	Wang's 2-input XOR	52 ps	91 ps	9.9 nA
CPL	Impr. Wang's 2-input XOR	72 ps	121 ps	2.56 nA
CPL	Wang's 3-input XOR	72 ps	102	26.2 nA
CPL	Impr. Wang's 3-input XOR	108 ps	159 ps	5.05 nA

Table 6.2: *2-input and 3-input XOR gates in static CMOS cells simulated with 70 nm High Speed BPTM model cards.*

Cascading CPL XOR gates will enhance the overall speed of the circuit in comparison to static logic families, as described in Section 4.6. Cascading two 2-input Wang's XOR gates to form a 3-input XOR gate needed in the full-adder investigates this theory. The gained time slack was utilized by sizing up the inverters to reduce leakage power dissipation. Results from these simulations are also shown in Figure 6.2.

As theorized the speed penalty of cascading XOR gates is very low and only raises the falling-edge propagation delay by a little more than 10% in the worst case. The 'improved' version is slowed down to reduce leakage power dissipation by a factor of more than five.

6.3.1.1 Non-ideal input voltage sources

Reduction in leakage of a factor of four to five is a considerable achievement. Yet, the leakage in these circuits originate from 'voltage source to input'-leakage which must be considered more thoroughly.

Since inputs are ideal voltage sources, the input values are always guaranteed to be the specified value and stable. This assumption is not valid from a real input, since this input is being driven by other logic circuits. The assumption is valid, though, as long as no power is drawn from the input source. No current is drawn when:

- The circuit is in a stable state,
- Inputs are only connected to transistor gates that do not draw currents to drive signal values and
- Transistors gates do not leak, see Section 3.2.3.

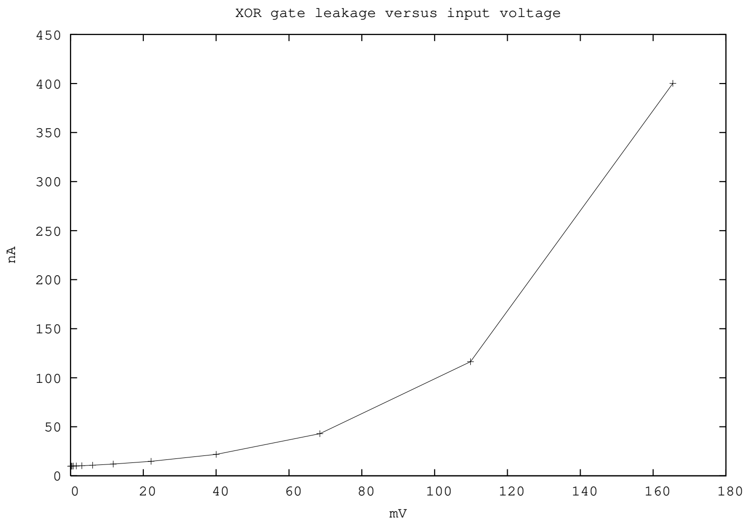


Figure 6.6: *Leakage current of a 2-input Wang's XOR gate with self-induced, alternated input value on one input.*

Since Wang's XOR gate uses the inputs to drive the input to the inverter high, leakage current can flow out through the inputs when they are at their low value. Therefore a ideal input voltage source is not a realistic assumption in this case. Input values will be driven by logic on the input, which has a certain I_{DS}/V_{out}-characteristic, i.e. depending on the current the logic has to conduct the input value changes in voltage.

Substituting the ideal input voltage source by a voltage source with a resistor in series simulates a more real input source in the steady state. By altering the resistance value the input values alternates. Figure 6.6 depicts the leakage current of a 2-input Wang's XOR gate as function of input value. The main reason for this increase is, that the input both drives the nMOS transistor disconnecting it from the pMOS pull-up network and as pull-down voltage source. Adding some resistance to the input increases the input value above V_{SS} (due to the leakage current through the resistor) which increases the conductance of the nMOS transistor leading again to increased leakage and so forth. This happens for both inputs.

6.3.2 Yano's XOR Gate

The Yano 3-input XOR gate is a gate driven only by input values and has no connections to either V_{DD} or V_{SS}. Output values are inverted to form the correct values and to drive following logic circuits. This gate was implemented and simulated, and leakage current values are depicted in Table 6.3. The three columns contain leakage current measurements from input inverters (II), output inverters (OI) and current drawn from inputs (In) in the steady state. The high leakage current originates from the weak pull-up of nMOS-transistors causing the output inverters to operate close to the boundary to the cutoff region. At around $800mV$ on the gates of the output inverters, these inverters leak considerably.

Introducing pass-gates instead by adding pMOS transistors remedies this situation. The main problem is then the current drawn from the inputs. This might have been reduced by sizing up transistor length, but as it was found, that the gate was slower than the static CMOS implementation, nothing can be done about the leakage problem.

Family	Logic gate	II Leak	OI Leak	In Leak
CPL	Basic 3-input XOR	6.6 nA	110.2 nA	9.1 nA
CPL	3-input XOR with pass-gates	6.4 nA	2.3 nA	9 nA
CPL	Impr. 3-input XOR	-	-	-

Table 6.3: *Propagation delay and leakage of a 3-input Yano's XOR gate in three implementations: Basic nMOS gate, pass-gate implementation and leakage reduced implementation using pass-gates. 70 nm High Speed.*

6.3.3 Discussion of results

In the Yano XOR gate there are no connections to voltage sources and the output is connected to the gates of inverters, so no leakage current should be possible internally in the gate. Yet by closer inspection of the circuitry it becomes apparent that there are paths from the node B to node $\overline{B}$ containing only one nMOS transistor. No matter what value A might assume, one nMOS transistor will be conducting and one not. The same applies to the value C. Picking two random discreet values for the inputs A and C and disregarding the conducting transistors, the Yano's XOR gate becomes 4 nMOS-transistors in parallel driven by inverters of opposite output values. This is the explanation of why the CPL gate leaks more than the equivalent static CMOS implementation. Adding pMOS-transistors to form pass-gates just increases the problem as the leakage through these transistors adds to the sum of leakage.

Wang's XOR gate suffers from somewhat the same problem. Using input values to drive outputs directly causes the leakage from the few, but present, voltage sources to affect input values causing further leakage. So in general it can be concluded that removing voltage sources and using input values to drive internal nodes affects the internal signal value stability and causes inverters and drive buffers to leak considerably. CPL reduces the need for transistors due passing of input values, but this in terms increases the leakage through paths that are maybe not so easily identified. Leaking paths are even harder to predict if CPL were to be used for cell based design, as the leakage source and drain in many cases will placed in two different cells.

In general it must be concluded that:

- Signal value variations due to passing of input values are hard to control causing considerable leakage
- Increasing the number of transistors in series further alters signal values causing increased leakage
- The speed gained by CPL does not give enough time slack to improve the logic gates sensitivity to signal variations

Due to these conclusions no further work on complementary pass-transistor logic was done. The other half of the full-adder, the AND-OR circuitry, is by inspection predicted to perform equally bad in terms of leakage power dissipation.

6.4 Domino logic

The investigation of Domino logic begins with the full-adder design. The XOR and AND-OR structures were designed in static CMOS for comparison. Two versions has been made available. First the CyHP based version built from the 20 smaller gates. Secondly, as the full-adder components are typically available in any cell library, the components were also designed as compound gates an simulated for leakage and propagation times. The results are shown in Table 6.4.

Family	Logic gate	Tpd rise	Tpd fall	Leakage
CyHP CMOS	3-input XOR	242 ps	179 ps	35.4 nA
Static CMOS	3-input XOR	320 ps	160 ps	25.37 nA
CyHP CMOS	6-input AND-OR	-	95 ps	16.95 nA
Static CMOS	6-input AND-OR	-	99 ps	25.37 nA

Table 6.4: *Static CMOS and CyHP based compound gates for full-adder design.*

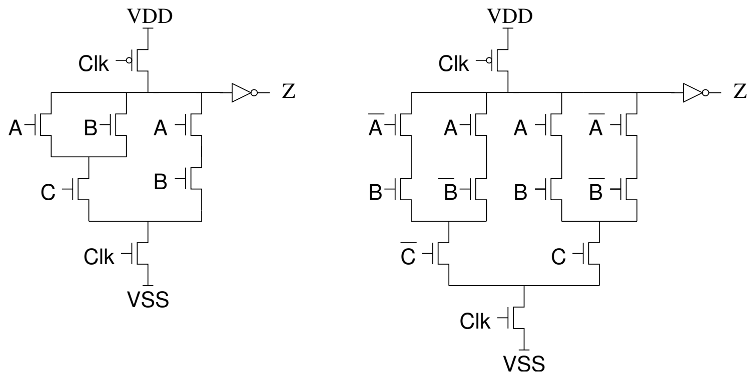


Figure 6.7: *Transistor netlists of the AND-OR (left) and 3-input XOR Domino logic gates.*

6.4.1 The Domino XOR block

The XOR gate was implemented in a nMOS pull-down Domino block and simulated. Figure 6.7 depicts this block. First the basic gate was investigated. The leakage of the block was clearly much reduced and the pull-down propagation time was less than the equivalent time of the static CMOS gate. Table 6.5 shows results from this simulation.

The time slack was utilized to size up the length of the pMOS device to $4.5 * L_{min}$. The nMOS device could, within time bounds, be scaled to $1.4 * L_{min}$. This design is called the *improved* design. The leakage hereby was reduced by up to a factor of 16.

Further, by instead using low-leakage clocking transistors the leakage could further be reduced. The leakage of this optimized gate is around $4.9pA$ which is a factor of 3,000 less than the static CMOS implementation.

Family	Logic gate	Tpd-fall	Leak, Clk=0	Leak, Clk=1
Domino	3-input XOR	-	2.23 nA	3.31 nA
Domino impr.	3-input XOR	159 ps	0.715 nA	0.198 nA
Domino LL	3-input XOR	158 ps	0.0049 nA	0.0048 nA

Table 6.5: *Propagation delay and leakage of a 3-input Domino XOR gate in three implementations: Basic nMOS gate, improved gate sized to match the timing of static CMOS, and the same approach with low-leak transistors instead.*

6.4.2 The Domino And-Or block

Using a nMOS Domino logic block for the AND-OR case (Figure 6.7) yields equally good results. The simulation run is essentially the same as for the XOR logic block. First a basic implementation proved to be superior in time, so the time slack was used to built the *improved* lower leakage design by transistor sizing. Thereafter low-leakage transistor replaced the high-speed clocking transistor, and even better results were achieved. The results are shown in Table 6.6

Family	Logic gate	Tpd-fall	Leak, Clk=0	Leak, Clk=1
Domino	6-input AND-OR	-	2.22 nA	2.93 nA
Domino impr.	6-input AND-OR	65 ps	0.743 nA	0.197 nA
Domino LL	6-input AND-OR	98 ps	0.0166 nA	0.0456 nA

Table 6.6: *Propagation delay and leakage of a 6-input Domino AND-OR gate in three implementations: Basic nMOS gate, improved gate sized to match the timing of static CMOS, and the same approach with low-leak transistors instead.*

6.4.3 Gate leakage

Domino logic by the above simulations almost seems too good to be true. And unfortunately it is. The reason is, that the simulations do not incorporate the fact, that the gates that are driven by the dynamically held pre-output node are leaking and quickly drain the capacitively charge held there.

Without incorporating a gate leakage model in the simulations of the two designs above this was not a problem since the subthreshold leakage was minimal and did not affect the dynamically held node. But considering gate leakage steps have to be taken to ensure, that the dynamic gate can hold its state the entire *evaluate* clock phase with leaking gates.

6.4.3.1 Estimating Gate Leakage

As described in section 3.2.3 many different models for gate leakage can be found in the literature. The models are typically based on a statistical study from a given process, from which a model has been formulated by exponential regression. These models contain factors specific for the given process. Hence, since no analysis could be found with the same model parameters and supply voltages as used here in this work, these models do not apply in this case.

Instead, a rather crude model can be formed from the knowledge, that around the year of introduction of $70nm$ processes, the total gate leakage will be equally large as the total subthreshold leakage. A design built with the simulated CyHP library with maybe 40% registers will have a average subthreshold leakage of around $4nA$ per transistor in (non-conducting state) in the design.

One study, though, is very interesting in this respect. The paper [36] incorporates gate leakage models for a $70nm$ process into BPTM transistor models and simulates for gate leakage. The gate leakage printed in the paper is $50nA$ per nMOS transistor with $V_{DD} = 1V$ and T_{ox}=10Å. The gate leakage decreases with an order of magnitude for each added 2Å gate-oxide thickness or each added $0.3V$ to V_{DD}.

The transistor models in this work have 16Å gate-oxide thickness. This is oxide thickness not allow for much voltage scaling. In a real process, the T_{ox} must be assumed to be thinner. Furthermore, process variations can easily cause several Ångström variations in the oxide thickness causing up to several orders of magnitude [37] increase in the gate leakage.

Using a process with $1V$ supply voltage and gate-oxide thickness 10 will then leak $50nA$ per transistor. Process variations can increase this problem by more than an order of magnitude, since only a process variation in the gate-oxide of 2Å is required to cause this. Assuming no process variations the gate leakage still causes major problems for dynamic logics.

6.4.3.2 The Domino XOR Gate With Leaking Gates

To evaluate the impact of leaking gates on the total leakage of Domino gates, a resistor (R_{leak}) was connected with the output and ground, like in Figure 6.9. Assuming gate leakage of either the $4nA$ estimated in this work, or the $50nA$ from [36], the resistor value be-

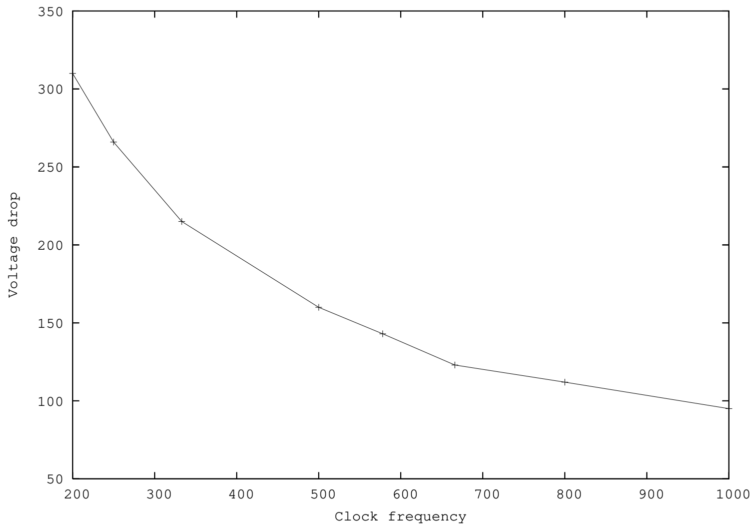


Figure 6.8: *Voltage change(mV) of a dynamically held output as function of clock frequency(MHz).*

comes $125M\Omega$ and $10M\Omega$ respectively, when two transistor gates are driven by the dynamically held output.

Precharging the dynamically held output to V_{DD} and applying a non-pulldown input vector, the effect of leakage can be measured in the end of the *evaluate* phase. Figure 6.8 shows the voltage drop, the dynamically held node experiences, as function of clock frequency. Naturally, as the clock frequency is increased, the voltage drop decreases due to the shortened time the output has to be held high dynamically.

The leakage currents of $8nA$ and $100nA$ are simulated by the R_{leak} resistor, and the 3-input XOR with the achieved leakage improvements is examined again. The dynamically held node can be kept high by using a bleeder transistor or simply by a resistor. The resistor will can be sized very precisely to match the leakage.

Calculating the resistance value follows these steps: The leakage is set to $100nA$ and the maximum voltage drop is relaxed from $V_{DD}/8$ to $V_{DD}/4$ to ease the design of the bleeder device. With these values the resistor becomes:

$$R_{pull-up} = \frac{V_{DD}/4}{I_{leak}} = \frac{0.04V}{100nA} = 4 * 10^5\Omega \tag{6.2}$$

In the case, where the nMOS network is supposed to pull-down, the resistor will then leak $1V/40.000\Omega = 2.5\mu A$, which is unacceptable. An alternative way is to use a bleeder transistor, that can be turned off, when the output is at certain levels. This is depicted on Figure 6.9. This turns the logic family into a semi-static family, though.

The design of this transistor is rather difficult. The transistor has to be able to deliver $100nA$ at a drain-source voltage of $40mV$. This transistor has to be quite strong to achieve this. Though, the transistor must not be too strong to prevent the nMOS network from being able to pull-down. Either a very wide transistor is needed or a ultra-low V_{th} transistor is needed. Both will leak considerably.

Here, a simulation setup was made consisting of the 3-input XOR gate with $1GHz$ clock frequency and the maximum voltage drop of $40mV$. A low-V_{th} transistor was used as bleeder transistor and sized to match the required drive strength at $100nA$ at $40mV$ drain-source voltage.

First, the bleeder transistor was measured to be pulling high adequately at the device sizes $L = 9 * L_{min}, W = 1 * W_{min}$. Adding this transistor causes the pull-down nMOS transistor to be inadequate and therefore was sized up to $W = 4 * W_{min}$. The pull-up pMOS transistor was no longer capable of pulling high, so the length of that transistor had to be reduced to $L = 3.5 * L_{min}$.

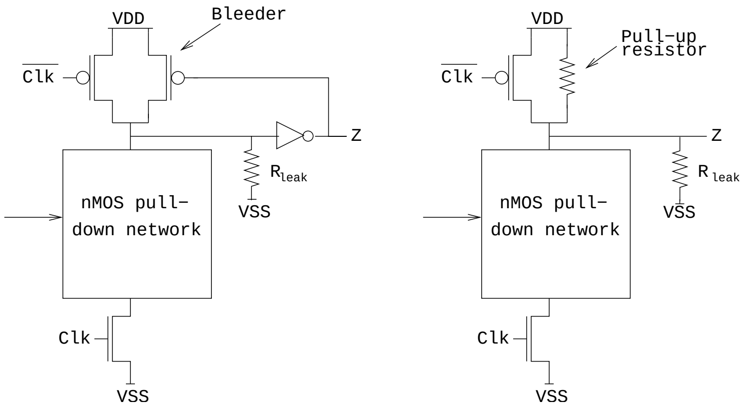


Figure 6.9: *Adding a resistor to the output simulates leaking gates. Possible solutions could be to add a pull-up transistor or resistor.*

With the resistor connected the gate leaks around $100nA$, naturally. This leakage is though the worst case leakage, which all Domino gates are not experiencing. Removing the resistor the leakage remains around $50nA$. This is partly due to the altered clocking transistors, and also due to the output inverter. As the bleeder and clocking pMOS device leaks into the gate region of the output inverter, the voltage on the gates increases and causes high leakage.

6.4.4 Discussion of results

Domino logic seems very promising in the first part of this analysis with no leaking gates. The clocked operation of the dynamic logic family allows for low leakage transistors to be put in series with all paths. Operational speed is initially faster and can be lowered with large benefits to leakage current reductions.

Yet, when gate leakage is introduced, Domino logic is not usable. Dynamic logic families are inherently not built for driving outputs in longer than very short periods of time and only with very limited currents. Adding leaking gates to a dynamically held node requires a keeper device to keep voltage values stable, which requires the clocking transistors to be resized to perform correctly. This causes these transistors to leak considerably more than the case with no gate leakage.

In Chapter 3 the arrival of high-*k* dielectrics is predicted (Figure 3.2) to reduce the gate leakage problem to be negligible. Until new dielectrics have been introduced in the process, dynamic logic families are not feasible for deep-submicron design. Further issues not covered in this work affect Domino logic. If Silicon-on-Insulator is used, dynamic circuits become very sensitive to parasitic capacitances in the circuitry, especially on the bulk-side of the gate (the parasitic bipolar effect, PBE). To reduce this effect, keeper transistors are needed to guarantee a full pull-down/up on all nodes in the precharge clock phase[38]. These transistor will also leak, further disproving the usage of dynamic logic style for low leakage design.

6.5 MacroCMOS

The evaluation of MacroCMOS follows in three steps. First, the full-adder is implemented in MacroCMOS fashion. Secondly, a circuit showing the possible logic optimization when building cells on-the-fly. The third evaluation consists of a large block, that is optimized to match the best case Synopsys implementation.

Before these analyses were done, a proof-of-concept simulation run was completed. The results from this survey are described in the end of this section.

6.5.1 The full-adder

The full-adder as described in section 5.2.4 was implemented in transistor netlists. The evaluation consisted of three implementations: A CyHP for comparison, a large cell implementation and the MacroCMOS implementation.

The large cell implementation was here done for comparison since a typical cell library would contain cells for the full-adder. These consisted of the 4-input XOR and the AND-OR structure in two big cells. The MacroCMOS cell was implemented as described in section 5.2.4.

The evaluation of the three circuits follow the same steps as for the other logic families. The results from this analysis is presented in Table 6.10.

Design	Average leakage	Saved leakage
CyHP	52.7 nA	-
Large cells	46 nA	12.7%
MacroCMOS	35.2 nA	33.2%

Figure 6.10: *Results from the full-adder analysis of MacroCMOS*

6.5.2 Logic optimizations

A rather small cell was built to demonstrate the logic optimizations that are not possible when using a cell library. The small six-input cell has the logic function as depicted in Figure 6.11*(1)*. The fact that *C* is connected to two different inputs led to great reductions in leakage.

If this cell did not exist in the cell library (for this simplified example), the six-input version*(1)* or a NAND-equivalent*(2)* would have to be used. The transistor netlist of the six-input version is presented as *(3)* in Figure 6.11 and the MacroCMOS implementation is on the right hand side, *(4)*.

The logic optimizations are clear. The pMOS transistor with *C* on the gate can be sized up in length or replaced by a LL transistor since it easily matches the two chains in pull-up delay. The nMOS device with *C* on the gate is in series with all pull-down paths. As shown in Figure 3.11 on page 34 this structure leaks far less than the original pull-down structure. The results from this analysis are presented in Table 6.12.

The inverter leakage is not included in the total leakage since it depends on the specific example whether the inverter is needed in different implementation styles. If the equal inverting gate was used as example, the inverter would have been needed in the CyHP case and not the MacroCMOS case. With or without inverters, MacroCMOS leaks less than the CyHP implementation.

It is quite evident, that larger cells save leakage. Including knowledge of the input probabilities internal optimizations can be done saving leakage power.

This is quite a small example to demonstrate logic optimization with. A larger example is given in the following section. Logic optimization is further discussed in Chapter 8.

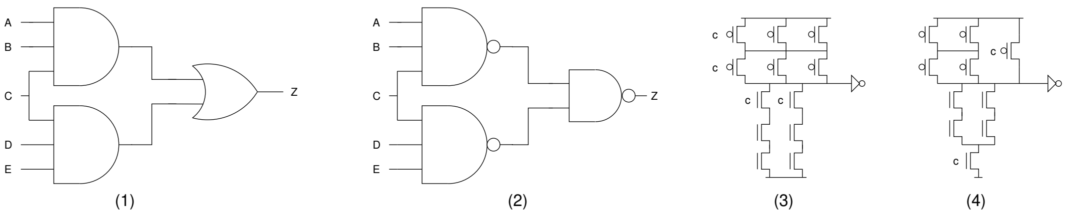


Figure 6.11: *1: The small gate. 2: CyHP implementation. 3: Transistor netlist of the 6-input cell library cell. 4: Optimized transistor netlist.*

Design	Average leakage	Minimum leakage	Inverter leakage
CyHP	11.62 nA	8.6 nA	-
6-input large cell	4.9 nA	0.69 nA	4.7 nA
MacroCMOS	3.8 nA	0.37 nA	4.7 nA
Saved:	58%/67%	92%/96%	-

Figure 6.12: *Results from logic optimization beyond cell boundaries.*

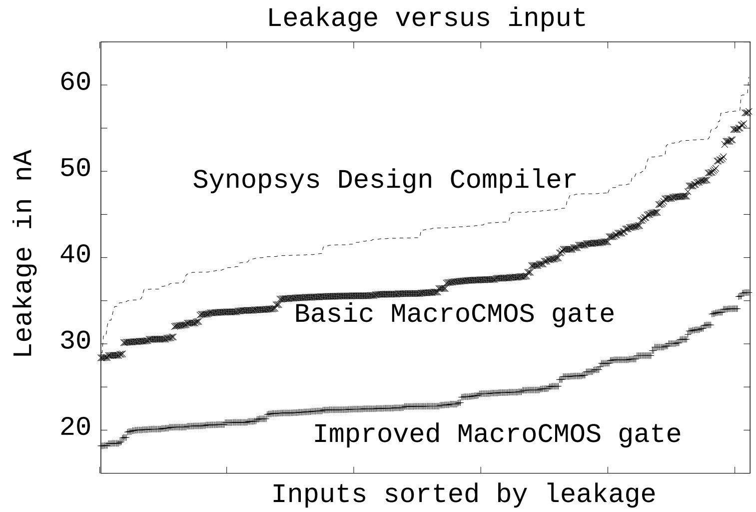


Figure 6.13: *1: The small gate. 2: CyHP implementation. 3: Transistor netlist of the 6-input cell library cell. 4: Optimized transistor netlist.*

6.5.3 *Larger cells for MacroCMOS*

A large logic block was built to show the benefits of MacroCMOS. Due to the size of this block and the number of steps of optimizing the block, the description of the block and the optimizations are placed in Appendix E.

The logic block was built from a cascade of randomly selected smaller, 2- and 3-input cells, and random inputs were assigned to the primary logic level. This produced a 9-input logic block with seven different logic gates included.

First the leakage was evaluated with the CyHP library. Then Synopsys Design Compiler was used to do logic optimizations on the circuit. Design Compiler's solution was then built in transistor netlists and simulated for leakage and timing.

The equivalent MacroCMOS implementation was then built to compete with Design Compilers best logic optimization. The MacroCMOS cell was built to match the timing of the Design Compiler derived circuit.

The leakage current of the 9-input gate was measured with all 512 possible input value combinations. These leakages are presented in figure 6.13 sorted by the leakage value. It is evident, that even the basic MacroCMOS gates leaks less than the best possible implementation with the cell library.

Further results from this analysis are shown in figure 6.14. The results are normalized to the results from the basic gate. The optimizations done the MacroCMOS are not complete since automation would have been needed for this task. Only the obvious sources of leakage was removed. A better solution can be attained by automation. The complete analysis of this circuit can be found in Appendix E.

The optimizations done in this example were only focussed on the combinational logic without the inverters. Clearly, the inverters could be part of the optimization process where some of the time slack can be dedicated to reducing the very leaky inverters. Comparing the performance of MacroCMOS to Synopsys Design Compiler can be done by removing

Design	Saved avg. leakage	Saved min. leakage
Basic gate	(66.53 nA)	(45.73 nA)
Synopsys optimization (HS)	33.5%	38.1%
MacroCMOS basic	42.7%	38.3%
MacroCMOS opt. for low leakage	63%	74.2%

Figure 6.14: *Results of the MacroCMOS and Synopsys optimization of a larger cell.*

Design	Average leakage	Minimum leakage
Synopsys optimization (HS)	24.87 nA	13.3 nA
MacroCMOS opt. for low leakage	14.63 nA	1.4 nA
Leakage reduction	41%	89.5%

Figure 6.15: *The leakage reduction of using MacroCMOS versus Synopsys Design Compiler. Without inverters.*

the leakage from the inverters and comparing the leakages of the two implementations. This way only the optimized bits are compared. The results from this comparison is shown in figure 6.15.

6.5.4 Limitations of MacroCMOS

Not every logic block can be built with MacroCMOS to save leakage. To prove this, a variety of NAND and XOR gates were build with minimum sized transistors and simulated for leakage currents. The average leakage current of the gates are presented in table 6.16.

It is evident, that a NAND-gate reduces in leakage when the number of inputs is increased. Yet, due to the complexity of the XOR gate, this gate increases in leakage per input when larger XOR gates are build.

Gate/Inputs:	2	3	4	8
NAND	3.9 nA	4.5 nA	2.1 nA	0.68 nA
XOR	15.4 nA	-	91.4	-

Figure 6.16: *Average leakage of a NAND and XOR gate with minimum sized transistors.*

6.5.5 Discussion of results

The evaluation of MacroCMOS was done with three example circuits. The first was the full-adder that though not very suited for a MacroCMOS implementation proved to reduce the leakage by around 30% in comparison with the large cell implementation.

The second example was the 6-input gate that could be optimized due to a redundancy in the input values. Here, this simple optimization which reduced the logic depth by one lead to 23% leakage reduction in the average case compared to a typical library cell and 47% reduction in the minimum leakage input state. Comparing with the CyHP implementation, the reductions were 67% and 96% respectively.

The third and final example was the large 9-input gate. Here, MacroCMOS proved to be comparable to the Synopsys Design Compilers best implementation even before MacroCMOS optimizations had begun. After optimizing a few places in the gate, the leakage was reduced to half of the Synopsys implementation. Ignoring the inverters, which were not optimized, the MacroCMOS cell leaked nearly a factor of 10 less.

It was not possible to optimize the cell fully by hand. For this task automation is needed. In this example just a few transistors were optimized to match the timing of the Synopsys version of the gate. By further inspection of the transistor netlists presented in Appendix E is becomes clear, that there still are redundant transistors, which have not been sorted out in the manual optimization process. These would most definitely have been optimized away in an automated process, which both saves the leakage of these transistors, and allows for other transistors to be scaled for lower leakage. It is believed by the author, that an even much better result could have been achieved given enough time to derive a automated process.

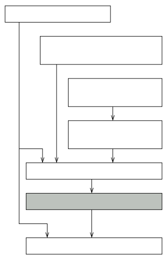
MacroCMOS will be discussed further in the following chapters.

CHAPTER 7

DISCUSSION OF RESULTS

Contents

This chapter will present the key reasons for the selection of the logic family for implementation of a cell library. Results from the previous simulations will be discussed briefly to determine whether or not general conclusions can be drawn from the example simulation cases.



7.1 Results

The results from the simulations are presented here in short and the candidate for a cell library implementation is selected based on these considerations.

7.1.1 MTCMOS

Cutting off power to a region in periods of no activity proved a good solution to reduce leakage. A factor of 2000 and even more can be saved depending on the amount of speed one would be willing to sacrifice. The factor of 2000 came with a delay penalty of 87.5%.

An implementation built with low-leakage transistors can be sized to match this performance. This implementation would not need a controller, that cannot be switched off, or extra hardware. Therefore, MTCMOS did not prove to be better than an existing LL/HS cell based implementation.

7.1.2 Complementary pass-transistor logic

From the analysis of CPL for low leakage applications a list of problems emerged. Reducing the number of connections to the voltage rails make the signals sensitive to noise and in general weakly driven. This causes inverters and other driving units to leak considerable.

The concept of having multiple stages after each other without voltage rail connections reduces the leakage due to the left out connections, but causes the reduced voltage value quality and thereby leakage.

Furthermore, as the same signal is used as input value and voltage source, the circuitry becomes very sensitive to process variations and long wires, both causing non-ideal connections between the logic blocks.

An XOR gate that matched the speed of the equivalent CMOS gate was built with a leakage reduction of around 50%, but this gate was proven to be very sensitive to variations in input value levels. Introducing gate leakage would further have increased these problems. These problems will apply to any circuit built with CPL logic.

In general, is not a possible to design for low leakage using the CPL logic family. In this work it is not explored whether CPL can be utilized to further decrease the leakage of a design built from low-leakage transistors. It can be speculated that LL transistors are not so sensitive to the described effects. Yet again, one would probably choose to increase V_{th} even further for this purpose instead.

7.1.3 Domino logic

Domino logic, or many dynamic logic families in general, are very interesting in low leakage terms. The division of the clock-phase into a precharge and a evaluate phase allows for very low-leakage implementations. The increased speed of these logic families can be used for further reducing the leakage.

In this work very good results were presented. That is, until gate leakage was introduced. Preserving the dynamically held node disallows the clear separation between the clock phases and thereby the pull-up and pull-down logic.

In the analysis a quite potent gate leakage of $100nA$ was applied. If a smaller gate leakage was applied, the result would have been the same though[7.1]. The problem was not the magnitude of the gate leakage, but the fact, that the bleeder transistor would have to operate at very low source-drain voltages, requiring a transistor with high drive. So, in the nA-range this is not feasible.

When new high-*k* dielectrics have been fully developed, dynamic logics should definitely be reconsidered for low leakage design.

7.1.4 MacroCMOS

The design style proposed in this work is MacroCMOS. The analysis here totals three example implementations. It is usually difficult to prove something in general from a few examples. Yet, the examples show general optimizations that are not possible with current cell libraries.

The full-adder example showed, that this design which is very parallel and not very optimal for MacroCMOS could be built with around 33% leakage reduction. The six-input AND-OR gate showed that logic optimization without a static cell library enables optimizations for low leakage. Further, it proved that larger cells leak less than smaller.

The nine-input MacroCMOS cell design proved, that in many cases a randomly generated logic block can be built with the same delays and with far better leakage reductions than using current synthesis tools and cell libraries. This is not always true, it is proven also. The XOR gate is better left out of a larger block in many cases. The synthesis tool must explore design space in every case to search for the best solution.

[7.1] The case with $8nA$ gate leakage was simulated for verification. Equally bad results were encountered.

7.2 The chosen candidate for cell library implementation

The theory that stacking transistors by building larger blocks of logic seems to be proven by the examples given. Large blocks can be built just to be just as fast as cascaded smaller blocks reducing the leakage of the total circuitry.

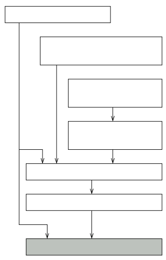
MacroCMOS is therefore chosen to be the candidate for building a cell library. Building cells on-the-fly requires a new synthesis process, or at least some changes in the synthesis flow. These changes and the possible optimizations possible when using MacroCMOS will be discussed in the next chapter.

CHAPTER 8

A CELL LIBRARY FOR LOW LEAKAGE

Contents

This chapter describes the new synthesis flow proposed in this work. This new synthesis flow sets new requirements to the cell libraries. A cell library designed to match the requirements set by the synthesis tool is proposed.
With the MacroCMOS cell library and synthesis tool a range of optimization algorithms can be devised to take advantage of this new synthesis paradigm. A number of possible optimizations enabled by using MacroCMOS style synthesis is presented in the end of this chapter.



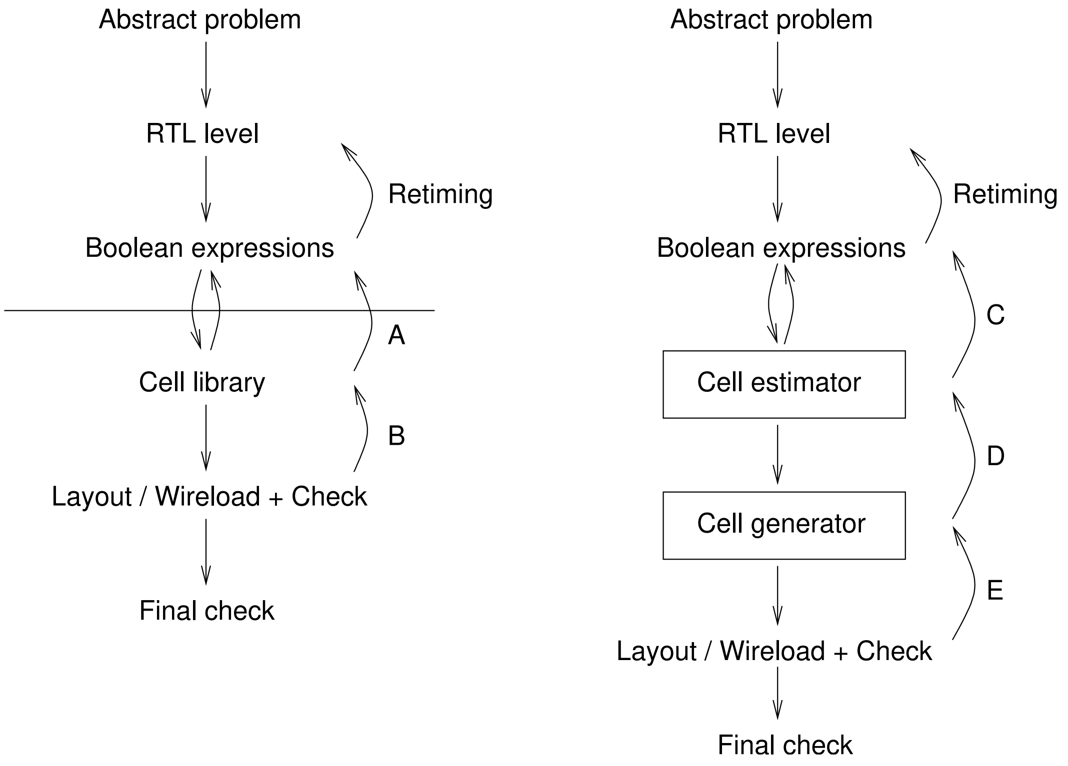


Figure 8.1: *Current and MacroCMOS synthesis work flows.*

8.1 Synthesis of MacroCMOS

Since the MacroCMOS cells are created on-the-fly, additional steps are required in the synthesis process. Figure 8.1 presents a possible synthesis flow of a current synthesis tool, and the changes required to alter this flow to a proposed MacroCMOS cell synthesis flow.

8.1.1 Current synthesis

In a current synthesis flow the abstract problem is broken down into sub-problems until an RTL level of boolean expressions is reached. The full-adder example given in Figure 2.5 demonstrates this. The design is in the boolean expression step represented by graphs of logic. Breaking these graphs into sub-graphs that can be mapped down into logic cells is done iteratively using the logic functions supplied by the cell library.

Since cells cannot be scaled during synthesis, but only replaced by the limited number of equivalent cells with other drive strength, this step can be quite time consuming. One could speculate that a cell library with an infinite number of cells would be perfect for this task. Yet, as the design space increases dramatically when the number of cells is increased, this might not be beneficial in terms of synthesis time consumption.

If no solution can be found at this level, the synthesis tool must go back and reorder the logic (Figure 8.1,*A*). When a solution is found, the cells are laid out by a place&route tool and wire loads are determined. If this solution does not meet the timing requirements, maybe a single cell could be changed (*B*). Else, one would have to go back to either rewriting the design code or change the design parameters given to the synthesis tool.

The main boundary here is the interface between the synthesis tool and the cell library. If a cell can not be found, that matches the requirements, a complete reordering of the logic is required. Further, the possibilities for logic optimizations for low leakage are, as discussed, very much limited.

8.1.2 Proposed synthesis flow

The task of synthesizing a cell and simulating it for electrical characteristics is a heavy task. And, as the synthesis tool has to explore the design space for a leakage power optimal design, this task becomes infeasible to complete in reasonable time.

Instead a cell estimator and cell generator is proposed. This estimator uses a cell library of predefined structures to estimate a variety of different implementations of the region of logic to find an optimum solution. The boolean expression level supplies a logic function, output drive strength and delay limits for each output to the cell estimator. The estimator then creates an optimum solution due to a cost function of leakage, dynamic power, area etc., regarding the requirements.

The cell estimator does not fully simulate the cell, but predicts what the partition of the logic, the transistor configuration and transistor sizings produce the best solution under the requirements given. With conservative predefined structures and wire load models, a conservative solution can be found, that is not the optimum solution, but very likely a near-optimum solution. If no solution can be found, the synthesis has to reorder, retime or regroup the logic (*C*).

A cell generator is then used to lay out the transistor netlists designed by the estimator. Only these optimal solutions have to be fully laid out. If the finished cell does not meet the requirements, the cell generator must be invoked to regenerate the cell with even more conservative parameters (*D*).

After simulation of electrical characteristics, the entire design is laid out and wire loads are added. A check is performed to validate the design against the design constraints. Smaller adjustments can be performed by invoking the cell generator (*E*). If more severe adjustments have to be made, earlier steps have to be invoked again.

8.1.2.1 Infinite loops

At all levels it must be ensured, that infinite loops do not occur. That is, if two conflicting adjustments are done alternatingly forever, and no convergence towards a solution is possible.

Caching is a powerful tool to fight this problem. By caching earlier solutions the tools can be made to prevent these solutions from been retried. Another way is to limit the number of retries every level in the synthesis flow is allowed to do.

Caching the transistor lists from the cell estimator after a cell has been finished can speed up the cell generation when the cell can be reused in a future design. The cell generator still needs to be invoked though, to get the optimum low-leakage cell and to take new circumstances into account.

8.1.2.2 Post-synthesis resynthesizing for low leakage

The proposed synthesis flow requires a complete restructuring of current synthesis flows. Another alternative way is possible, though. Synthesizing a design with a current synthesis tool produces a netlist of logic gates for the place & route tool.

This netlist could be read by a post-synthesis tool, that locates areas of logic for the design of larger low-leakage cells. This tool can read from the cell library the area, timing etc. for the cells in question, and then generate a lower leaking cell matching these characteristics. Clearly, this does not produce as good a solution as the proposed flow, but it will require less alternations to a current synthesis flow.

8.2 The MacroCMOS cell library

Defining a cell library with smaller predefined structures might overcome the problem of slow full custom synthesis, and allow for deeper exploration of the design space for better

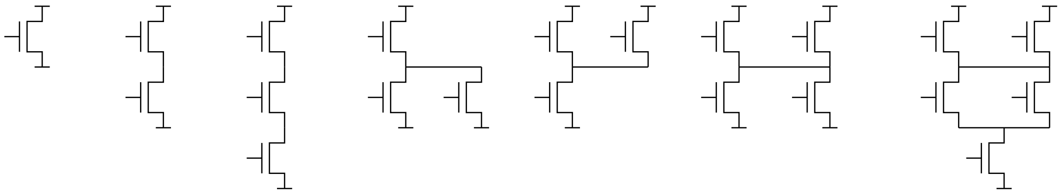


Figure 8.2: *Pull-down graphs in the MacroCMOS cell library.*

solutions.

A possible way is to include predefined pull-up and pull-down networks in a new cell library, from which larger cells can be built. It is infeasible to design a static cell library with all possible logic functions. But allowing the synthesis tool to combine predefined networks, either serially or in parallel, all possible logic functions can be evaluated for leakage, propagation delay etc. This forms a cell library with virtually infinite cells. A few of the structures are depicted in figure 8.2.

The number of structures needed in this library is only a small fraction of all possible logic functions, as these can be built from combining networks. How these networks must be described in the cell library for use in this synthesis process is explored through the requirements set to the cell library by the cell estimator and the cell generator.

8.2.1 Data required by the cell estimator

The job of the cell estimator is to express the logic function using the structures available in the cell library. Leakage can be evaluated very closely with the transistor netlist completed. Switching power and propagation delay associated with all input transitions together with cell area is estimated by the cell estimator.

To complete these tasks the cell estimator requires these data in the cell library for each structure:

- Logic function, as definition of pull-up/pull-down expression
- Structure area, a statistical value
- Drive strength formulated by tables of pull-up/down delay according to output capacitance and resistance parameters
- Total capacity on all outputs
- Leakage as function of all input states

Furthermore, the cell estimator needs wire load models for external wires. Internal wire loads are included statistically in the values described above. This can be accepted as most of the wire load lies in the external wires. The external wire load models could well be implemented like in the Liberty format.

8.2.2 Data required by the cell generator

The cell generator does the job of producing the actual cell designed by the cell estimator. This process must be completed through laying out the transistors according to layout rules for the specific process, which must be included in the cell library. For simulation, accurate transistor models for the specific transistors must be included also. The cells real area and leakage is determined in this step.

To improve the speed of the layout process, either earlier cells can be reused from a cache, or predefined and already laid out structures which only need slight modification can be reused.

Internal wires have to be included in this step. During simulation of a cell, wire models of internal wires are needed. External wire loads can be added in two ways. Either the wire loads are added as statistical models in this step and the cell is simulated. Or the wire loads are first added after all cells have been laid out, and then the circuit is fitted through simulation.

A good solution would be to use the statistical wire load models to size the transistors and then lay them out. When all have been laid out final adjustments can be made by retrieving the actual wire loads from the design. No matter where in the process wire loads are added, the cell library is required to include wire load models.

Further considerations, such as noise sources, rules about layout etc. which are not covered here, need to be incorporated in the cell library as well.

From this description of the task of the cell generator it is evident, that the cell library is further required to include the following:

- Layout design rules for the given process
- Transistor models for simulation
- Internal wire load models
- Global values such as voltages, temperatures, operating conditions etc.

8.2.3 The total of new requirements to cell libraries

Comparing the new cell library to the library described in Chapter 2, a list of changes is needed. These changes are presented here in two sections: The general contents and the cell specific contents. First the general values:

Current cell libraries	**A MacroCMOS cell library**
Wire load models	Wire load models. Both internal and external.
Timing look up tables with input net transition and output capacitance as parameters	Timing lookup tables with input net transition and output load formulated as output capacitance and resistance
Power look up tables with input net transition and output capacitance as parameters	Statistical power lookup tables with load and input net transition as parameters

For each cell/structure these changes are needed:

Current cell libraries	**A MacroCMOS cell library**
Area per cell	Area per network, statistical
Average leakage power	Average leakage power according to device sizes
Logic function	Pull-up/down function
Input dependent leakage values	Input dependent leakage values
Switching power, rising and falling transition	Switching power due to load, either rising or falling according to network. Statistical
Output delay	Output delay according to capacitive and resistive load
Output transition time	Output transition time according to internal and external load

8.2.4 Modelling propagation delay in a MacroCMOS cell library

Modelling propagation delay can be done by approximating the worst case load on a transistor stack by a capacitative load through a resistor. Including tables of resistance and capacitance in the cell library, the cell estimator can evaluate the worst case pull-up/pull-down delay by estimating the load and then looking up the propagation delay.

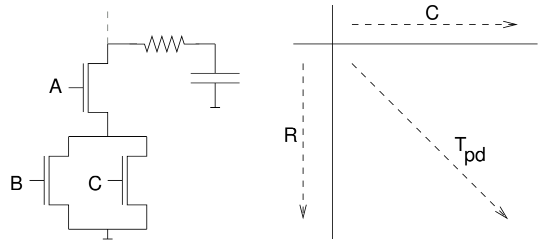


Figure 8.3: *Modelling propagation delay.*

Figure 8.3 gives the general idea. For each structure with different transistor sizings a table of propagation delays is required. Interpolation between equal structures with different transistor sizings is possible if the difference between sizings is reasonably small.

When the cell has been assembled in the cell estimator, wire loads are added, and propagation delays are checked again. More factors such as input transition time etc. are included in the cell generator step, which is done by real simulation. The tables need to be conservative enough to ensure, that the cell generator can generate the cell by the specifications from the cell estimator.

8.2.5 Modelling power consumption in a MacroCMOS cell library

Energy dissipation per switch is in the cell estimator evaluated by a conservative guess formulated by the number and sizes of transistor gates. Further, each structure might carry a statistical model with capacitive and resistive load as parameters to model, how much short circuit power one can expect to be drawn through the structure. In general is dynamical power consumption easiest to evaluate through simulation.

Modelling leakage currents can be done through a lookup table for each structure. For each input state a leakage current is given in the table. Leakage currents can easily be modelled with lookup tables as long as the structures are not connected in series. With serially connected structures, the leakage is the same or less as the least leaking structure in total average.

8.2.6 Layout of MacroCMOS cells

The layout is automated by the cell generator. Yet, not every transistor netlist with any sizings can be lain out. Transistors cannot be unlimitedly wide for example. If a standardized approach to layout is taken and the space between the voltage rails is constant, transistors will have to be broken into several pieces. Figure 8.4 demonstrates a place with narrow width between the voltage rails. Here an inverter is broken into four transistors to be able to fit in.

In a real implementation of a MacroCMOS cell library a list of design limits must be derived to guide and limit the cell generator to reasonable design.

8.3 Optimizing a design for low leakage with MacroCMOS

With a synthesis process and cell library in place, the task is to devise how low leakage designs can be built efficiently. A range of possible optimizations will be investigated here.

8.3.1 Input optimizations

As seen in section 6.5.2 building cells with knowledge of the input connections can save leakage power. This was due to two reasons. First, due to redundancy transistors could be

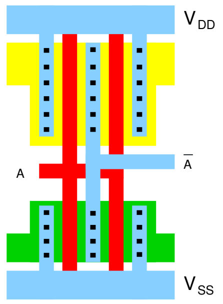


Figure 8.4: *Layout of a 4*W_{min} inverter with limited space.*

removed, minimizing the paths between the voltage rails. Secondly, the increased speed allowed for transistors to be resized reducing the leakage.

Knowledge of inputs values can also be useful in the synthesis for low leakage. If a working model of a design in a HDL is available, statistical information about signal values can be extracted. The synthesis tool at the boolean expression level hand over statistical information to the cell estimator. If no model is available, statistical information can be extracted directly from the design. An eight-input AND-gate can be assumed to produce a logic zero on the output for the major part of the time.

This statistical information can be used to help structure the logic blocks beneficially. If a signal is typically ’0’, the nMOS transistor that is fed by this signal should be placed closest to V_{SS}, due to the leakage dependency of the structure of transistors (Figure 3.11).

8.3.2 *Internal scaling*

Appendix E contains a larger example of building a MacroCMOS block. Here, the leakage reductions were achieved through stacking of transistors and sizing them to make propagation delays equal for all signals. LL transistors were also possible to use.

This scaling of transistors is not possible if not enough available time slack is available. In the example the timing was first violated, but through sizing up critical transistor was matched to the original cell. Building larger cells often introduces fast and slow paths in parallel. The slow ones can be sized to match the long in speed, and hence reduce leakage. These can be sized to be even more low leakage than regular high-V_{th} transistors.

Further sizing can be done as a cell can be built to match the timing requirements exactly, and not just be within timing bounds.

8.3.3 *Structural considerations*

Not every block can be built from smaller blocks for leakage reductions with success. Figure 8.5 depicts three attempts to build a larger block from a stochastically chosen logic block A and a 2-input cell.

With the AND-gate leakage will be reduced considerably due to added transistor near V_{DD}. The nMOS transistor near V_{SS} will add some leakage when the block A tries to pull up and X is a logic one. This is though only in a fraction of the time, and the nMOS transistor can be sized to leak less regarding the total delay. The same arguments can be used with the OR-gate.

The XOR-gate does not show this benefit. The cell has to be implemented by copying the pull-up/down (Pu/Pd) networks and form duals of the logic. The logic increases in number of transistors and dependent on the logic function of A this can be good or bad in terms of leakage. If A is an OR-gate it is beneficial (shown in the 9-input cell example) to build a compound gate, but if A is another XOR gate, the cells are better left as two 2-input cells (as shown from the proof-of-concept in section 6.5.4).

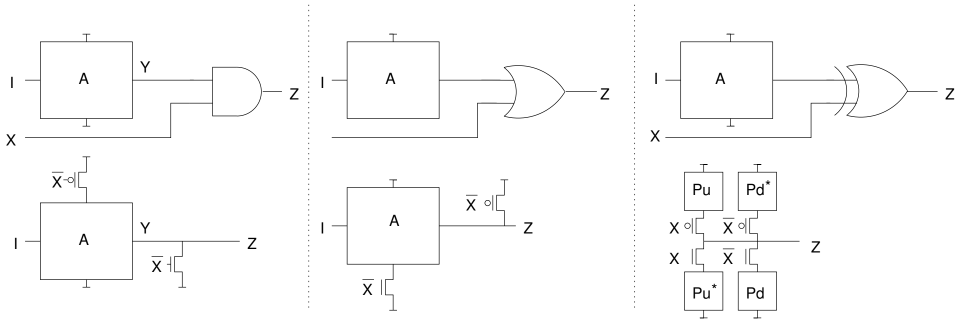


Figure 8.5: *Building a larger block from any logic block and a smaller block.*

Evaluation of each specific case is needed.

In general dividing large logic depths into smaller areas to build MacroCMOS cells can have great effect on the leakage of the final design. In section 6.5.3 (and Appendix E) it was beneficial to include the OR-gate in the XOR-gate, but not the AND-gate or other constructs.

8.3.4 Trading time slack for low leakage

With a static cell library of high-speed (HS) and low-leakage (LL) cells, leakage power optimizations are typically done by replacing HS cells with the corresponding LL cells. Since LL cells have higher propagation delays, this procedure is only possible when enough time slack is available.

Figure 8.6 presents this. On the left hand side of the figure the distribution of all path delays is shown in black. The red line indicates the percentage of low-leakage cells. As the path delay increases, the possibilities of replacing high-speed cell with corresponding low-leakage version diminish due to timing issues. Therefore, most LL cells will be placed on low delay paths.

The right hand side of Figure 8.6 represents the same concept. Here the paths have been sorted by path delay and are presented with the largest delays horizontally in the top of the figure. When timing allows for it, a cell is replaced with a LL cell. If more time slack is available this procedure is repeated until all cells are low-leakage cells. In the figure two replacements are depicted.

Three regions are of interest here. The region *A* represents paths which are somewhat too fast for their timing requirement, but not fast enough to use LL cells. The regions *B* and *C* represents paths where all, or a maximum number of, cells have been replaced by LL cells, but they still have some time slack available for optimization.

When the difference in propagation delays of HS and LL cells supersede the available time slack, no optimizations can be done when using a cell library of static cells. If cells could be scaled to match the time slack, a great deal of leakage would be saved. Since the drive of a transistor scales linearly with gate length (equation 3.4 on page 27) and the leakage scales exponentially (equation 3.5 on page 27), scaling up the transistor gate length to match the timing requirement causes the leakage to exponentially.

8.3.5 Gaining time slack

The full-adder example from Chapter 3 showed that building logic blocks together in MacroCMOS blocks can produce logic blocks that are faster than the original one. This time slack can be used to lower the leakage of the cell by using LL transistors or scale the length up of some or all transistors.

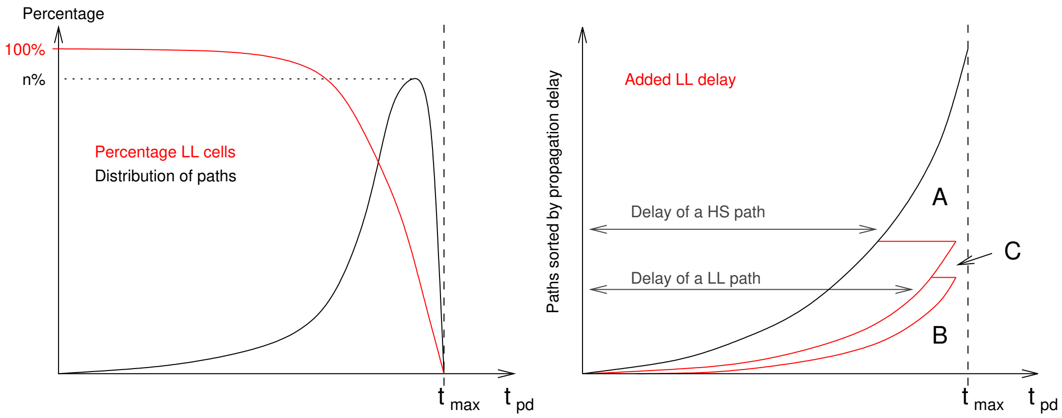


Figure 8.6: *Distribution of path delays with percentage of LL cells (in red). On the right hand side: All paths sorted by path delay. The delay overhead of using LL cells (in red).*

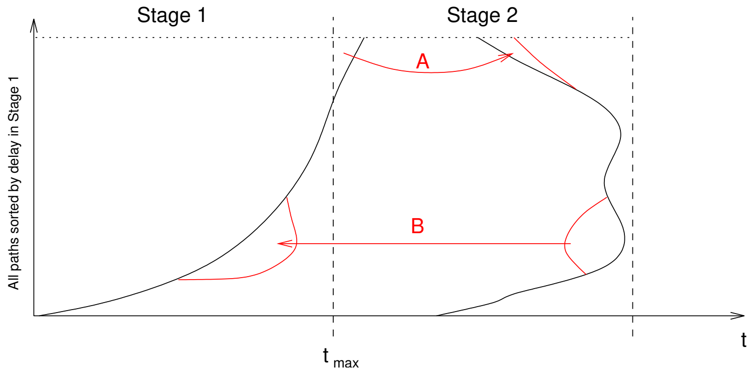


Figure 8.7: *Retiming to meet timing requirements (A). Further retiming to balance delays in two pipeline stages(B)*

Another way of gaining time slack is by retiming. Retiming is for many purposes such as dividing logic between pipeline stages to meet timing requirements, and it can also be used to equalize the time slack on both sides of a pipeline register for example.

By moving a part of the delay from a stage that barely meets the timing requirements to a stage with available time slack the time slack on both sides of the register is balanced (Figure 8.7). This enables further leakage reductions by trading time slack for low leakage. This can only be done if the structure of the logic allows for it.

8.4 Further issues

This work has covered MacroCMOS in areas of design, synthesis and cell library design. Many other issues must be taken into consideration when planning new cell libraries and synthesis tools. Some of the considerations these considerations will be discussed briefly here.

8.4.1 Physical synthesis

It was not possible to find any synthesis flow and cell library that is similar to MacroCMOS. Physical synthesis tools are not new, and a variety of proposed synthesis flows are presented in the literature. Yet, the literature is mostly concerned about physical synthesis for circuit verification[39] or for logic optimizations[40, 41, 42]. Both of them are not aimed at low leakage design. But, they do prove that physical synthesis is possible.

8.4.2 Gate leakage

As described in Chapter 3 gate leakage is dependent on the voltages at the terminals of the transistors. An exponential dependency of the gate-drain, gate-source and gate-bulk voltages must be expected. With current decreasing gate-oxide thicknesses and without improved oxides there is not much to do about the gate leakage else than changing the voltages.

In a stack of all non-conducting transistors the entire V_{DD} voltage drop is shared by the transistors according to the configuration and sizing of the transistors. Therefore, the gate leakage per transistor can be expected to be lower for larger stacks of transistors. If random input values are applied gates might begin to leak in different directions in and out of the stack causing more leakage. Yet, a small cell will typically leak maximally through the gate at all input states. Therefore, a larger gate must be considered to be less gate leaking than equivalent small cells.

8.4.3 Dynamic power consumption

Dynamic power consumption is not covered in this work. In Chapter 3 dynamic power is modelled as the total power consumption dissipated by charging and discharging capacitances plus the short circuit power consumption.

Depending on the logic function built with MacroCMOS the number of transistors either increases or decreases. So, it is difficult to predict whether MacroCMOS cells will consume more or less dynamic power due to charging capacitances. But, with decreasing device sizes the capacitive load due to wires will supersede the gate capacitive load. Therefore, the gate capacitance will become less important.

What is becoming more important is power dissipation due to short circuit currents in the switching period. Building larger cells that in general contain higher transistor stacks, this short cut current may be minimized. This is due to the fact, that input value transitions may arrive at different times causing more transistors to be in their non-conducting mode at all times.

A fact that may counter this expectation is, that larger cells will probably produce low output transition slopes. These outputs, routed into the next cell, will bring the following transistors in semi-conducting mode in much of the time. Yet, as a stack is built from an increased number of devices, the total resistance on the paths keeps the switching current down. The 'MOS device degradation' section elaborates on this subject.

8.4.3.1 Switching activity

The output switching activity of a larger cell must be expected to be lower than the total switching activity of a cascade of smaller cells. This is due to the missing internal nodes that for an input vector transition do not switch numerous times before all previous levels of logic have stabilized at their final levels. Further, the increased propagation delay of a larger cell in comparison with a single smaller cell dampens glitches in the circuit. This further reduces the switching activity.

Even though a switch in output state is bound to be more expensive in power the reduced switching activity and robustness to glitches will counter this effect.

8.4.4 MOS device degradation

Due to the low rising and falling output transition slopes, MOS devices are in a semi-conducting mode for an increased period of time. In current small-cell designs this has a bad effect on the MOS devices since increased wear and electromigration are effects of these increased currents.

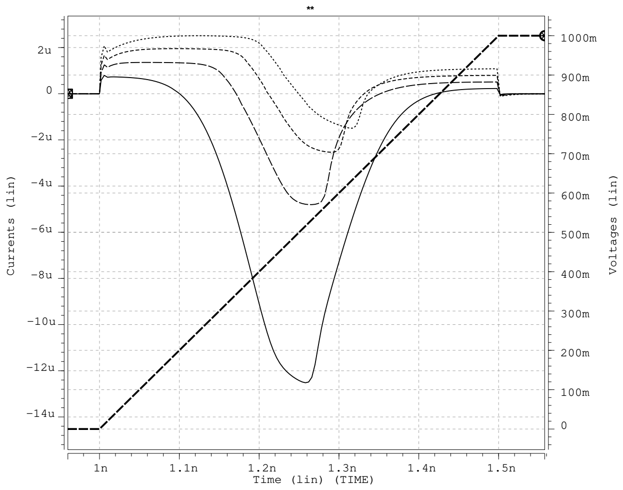


Figure 8.8: *I_{Vdd} for 1-, 2-, 3- and 4-device stacked inverter.*

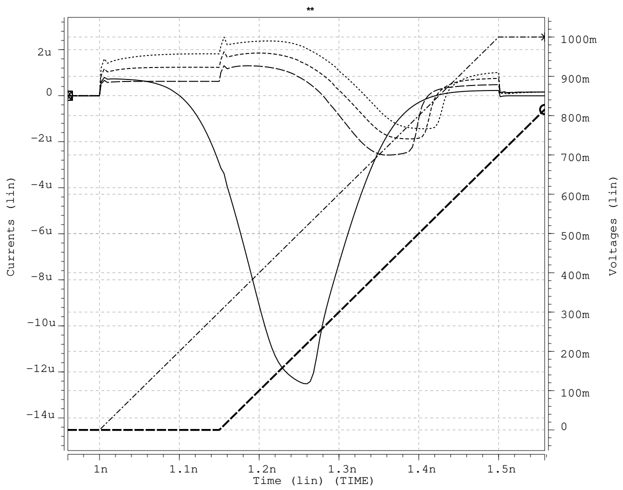


Figure 8.9: *I_{Vdd} for 1-, 2-, 3- and 4-device stacked inverter. The device pair closest to the output has a 150ps time shifted input.*

Low input slopes do not necessarily cause massive short circuit currents though. A $70nm$ HS inverter was simulated with an input transition slope of $\frac{1V}{500ps} = 2V/ns$. The same experiment was done building an inverter with two, three and four devices in series in both the pull-up and pull-down networks. The currents drawn from V_{DD} is depicted on Figure 8.8.[8.1]

It is evident, that the more devices that are placed in series, the lower peak current is flowing through the stack. Furthermore, as input signals arrive at different time points the devices will be in different conducting states at all times. Figure 8.9 shows the same four stacks with the device pair nearest to the output being driven by the same input, just delayed by 150ps. The single-inverter is driven by the normal non time-shifted input.

The results from this analysis did not show an indication that larger cells increase switching currents. Therefore, in combination with the fact, that larger cells reduce the switching activity, MOS device degradation is no more a problem in MacroCMOS than in regular CMOS.

[8.1]The currents are negative in value since HSPICE measures it as 'current into the node V_{DD}

Chapter 9

Conclusion and Future Work

Contents

9.1 Conclusion

The main objective of this work was to evaluate possible logic families other than static CMOS for low leakage design. This task was completed in a series of analyses.

The effects on leakage of scaling down device sizes was explored and rules of thumb for low leakage design of gates were presented. Based on these leakage considerations, a survey of logic families was conducted and MTCMOS, CPL and Domino logic were selected for closer leakage evaluation.

MTCMOS proved to be unusable since the delay overhead of adding the power routing transistors matches the overhead of using low-leakage transistors instead, which is an equally good and more design friendly approach. CPL failed due to reduced signal quality on internal nodes causing more leakage than gained by removing connections to the voltage supply rails.

Domino logic proved to be very good at reducing the subthreshold leakage. Yet, when gate leakage was taken into consideration, the benefits were lost as a keeper device would have to be added to maintain the dynamically held node.

The proposed design style, MacroCMOS, was investigated through three example simulation cases. MacroCMOS was found to be more efficient in reducing the leakage than a current synthesis tool with a current cell library. This was proven for both smaller and larger cells, that are not present in the cell library.

Through the study of transistor characteristics it was found that the main reason for the magnitude of the leakage problem is the usage of static cell libraries and current synthesis tools. The cell libraries offer only a limited number of cells, and typically these cells only have a small number of inputs. Assuming that larger logic functions can be built from these small cells without much overhead is not correct when including leakage considerations. Furthermore, the limited interface between synthesis tool and cell library consisting of a limited list of logic functions prevents many of the optimizations needed for low leakage design. Small cell synthesis for low leakage is not feasible in the future.

For the synthesis of MacroCMOS logic a new synthesis flow and cell library was proposed. Optimizations for low leakage such as logic optimizations, internal scaling, structural considerations and the efficient utilization of time slack for low leakage were presented and proven to work through the examples given. Retiming for low leakage was presented here also. Furthermore, it was discussed how the entire time slack available can be used for lowering the leakage of a circuit.

Although a logic family could not be found to replace static CMOS and change the way low leakage design is done, this work demonstrated a new way of using static CMOS for low leakage. Incorporating more logic in every (larger) cell and benefiting from the optimizations now made possible proved to be a viable way to reduce the leakage problem in the future.

Changing the design flow towards utilization of an alternative logic family than static CMOS would have had great costs. Not only synthesis tools and cell libraries needed to be changed, but also the designers would have to adjust their work flow and their architectural knowledge of IC design. Therefore, continuing the design flow in static CMOS with in MacroCMOS style preserves much of the work that has been done in the areas of optimizations on the architectural and synthesis levels.

The static CMOS logic family is generally recognized as the best overall performing logic family in terms of power consumption, area and timing. This work has concluded that static CMOS still will be the best performing logic family in the future even when the leakage problem is taken into consideration. Yet, the small cell based synthesis flow will have to be rethought incorporating aggressive leakage current reduction schemes, such as the MacroCMOS design style.

9.2 Future work

From this work a number of future work topics arises. Here, a short list of topics are presented. This list is not complete; more topics will arise when further work has been completed.

- Logic optimizations for low leakage presented in this work is not an exhaustive analysis of the research area. Further logic optimizations and even more elaborate improvements or transistor reconfigurations can be done. This area can be explored to improve the efficiency of an automated full custom synthesis flow like MacroCMOS.
- The implementation of a synthesis tool, or post-synthesis tool, and cell library are future work topics.
- Optimization algorithms for fast full custom synthesis taking leakage into consideration is an area of research for the future.
- Fast layout and accurate simulation of gates built on-the-fly is also an interesting topic.
- When high-*k* dielectric materials have been fully implemented in productions dynamic logic style could be reevaluated for low leakage applications.

A PROJECT DESCRIPTION

Number: 55
Master's Thesis Project:
Title: Design of CMOS cell libraries for minimal leakage currents
Student: Jacob Gregers Hansen
Period: 17.02.2004 - 13.08.2004
Project description:

Objectives
The objective of this MSc thesis work is to investigate optimal design under the presence of static gate leakages, and to device how design rules and trade-offs are altered.

Description
A main concern during the design of System-on Chips (SOCs) is the power budget, especially battery supplied systems are considered. In general, dynamic and static contributions constitute total power dissipation.
Dynamic power is primarily consumed by the information processing in the charging and discharging of internal capacitances. As such, dynamic power consumption is proportional to these capacitances, the switching frequency and the supply voltage. Static power consumption, on the other hand, is caused by leakage currents while the circuit is idle, i.e. not performing computations.
One key attraction of CMOS is negligible static power consumption. However, with decreasing device sizes this property is no longer satisfied due to subtreshold conduction. The reason for this is that for smaller devices, supply voltages are reduced. For speed, this in turn forces a reduction in threshold voltages. As a consequence, transistors are no longer turned off satisfactorily, i.e. drain currents contributes significantly to power losses in the transistor non-conductive state. For a $0.13\mu m$ process, the static losses may constitute almost 50% of the total power consumption.
The issue has been addressed by offering libraries of gates and cells in both low-VT and high-VT versions. This offers the option of fast, low-VT cells with high static power losses where timing is critical, and a slower, high-VT design for other parts. Traditional synthesis tools do not offer the means to optimize for multiple-VT libraries to reduce static power consumption. The solution, using such known synthesis tools, consists of synthesizing a design using a low-VT library, under the constraint that timing and performance requirements are met. Then, in a post-synthesis phase, the back-annotated circuit is analyzed with respect to power consumption and the circuit modified, replacing low-VT by high-VT library cells whereever possible. The update process does not involve any re-synthesis steps.
This thesis work addresses the design of logic families using different transistor configurations to realize libraries representing alternatives in the speed-power design space and under various technologies. This includes the generation of a 90nm library or better from an existing library, and the characterization of this library for use in a synthesis tool. The thesis work will be performed in parallel with two other MSc thesis works in a collaborative but independent effort. One work focusses on the incorporation of static power consumption metrics in the synthesis process, while the other work concentrates on the architectural aspects of multiple-VT libraries.

Supervisor: Flemming Stassen

B A Cell Library in the Liberty Format

B.1 General definitions, settings and units

```
/*****************************************************
 Synopsys Technology File genstf version 5.7.1
 Options used:
   force_incr                      on
...
 Process values given:
   Library nominal:                0.5
...
 Voltage values given:
   Library nominal:                1.6
   Tech-file best case:            1.8
...
 Temperature values given:
   Library nominal:                -40.0
...
 *****************************************************/
library( CORELIB8DLL ) {
   delay_model : table_lookup;
/*****************************************************/
   time_unit                : "1ns";
   voltage_unit             : "1V";
...
   slew_lower_threshold_pct_rise : 10 ;
   slew_upper_threshold_pct_rise : 90 ;
   input_threshold_pct_fall : 50 ;
...

/********************************************************/
wire_load(maxarea_000980) {
      resistance  : 0.00023
      capacitance : 0.00018
      slope       : 9.35
      area        : 0
      fanout_length( 1 , 9.35)
}

...
wire_load_selection(default_by_area){
      wire_load_from_area( 0 , 980 , maxarea_000980)
      wire_load_from_area( 980 , 4540 , maxarea_004540)
... }
lu_table_template( table_1 ) {
  variable_1 : input_net_transition ;
  variable_2 : total_output_net_capacitance ;
  index_1 (" 0.01, 0.06, 0.3, 1.2, 2.4 ");
  index_2 (" 0.003, 0.048, 0.24, 0.72, 1.44 ");
} power_lut_template( power_table_1 ) {
  variable_1 : input_transition_time ;
  variable_2 : total_output_net_capacitance ;
  index_1 (" 0.01, 0.06, 0.3, 1.2, 2.4 ");
```

```
  index_2 (" 0.003, 0.03, 0.09, 0.18 ");
}
```

B.2 Cell specific data

```
/*******************************************************/
/*-------------------------------------------------------------

2 Input AND, 1x Drive
-----------------------------------------------------------*/
cell(AN2LL) {
  area : 22.48 ;
  cell_leakage_power : 35.41850 ;
  leakage_power() { /* A_F_Z_F */
    value : 24.24400 ;
    when  : "A*B" ;
  }  ... Leakage at the other input combinations
  pin(Z) {
    direction : output ;
   function : "A*B";
    max_capacitance : 0.15400 ;
    internal_power() {
      related_pin : "A" ;
      /* A_R_Z_R */
      rise_power(power_table_1) {
        values( "0.01535, 0.05314, 0.13548, 0.26114", \
                "0.01586, 0.05335, 0.13505, 0.26018", \
                "0.01584, 0.05230, 0.13602, 0.26183", \
                "0.01401, 0.05567, 0.13743, 0.26302", \
                "0.02585, 0.06101, 0.15350, 0.26486" );
      }
      /* A_F_Z_F */ -  fall_power(power_table_1) { [fall power values]}
    }
    timing() {
      related_pin : "A" ;
      timing_sense : positive_unate ;
      /* A_R_Z_R */
      cell_rise(table_2) {
        values( "0.06745, 0.13468, 0.24035, 0.51625", \
                "0.08550, 0.15714, 0.30764, 0.53527", \
                "0.15755, 0.22440, 0.38087, 0.60772", \
                "0.32411, 0.50778, 0.55775, 0.78351", \
                "0.51021, 0.60055, 0.75015, 0.47572" );
      }
      rise_transition(table_2) { [rise transition values]
      }
      /* A_F_Z_F */
      cell_fall(table_2) {
        values( "0.08885, 0.15204, 0.27230, 0.55115", \
                ....
      }
      fall_transition(table_2) { [fall transition values]
      }
      timing_label : "A_Z" ;
[everything is repeated for input B] } /* end of library */
```

C Model Cards for Simulation

```
***-------------------------------------------------------------------------
***  BPTM 0.18, 0.13, 0.10 and 0.07 micron technologies
***-------------------------------------------------------------------------
***
*** This library of model cards was
*** created by Jacob Gregers Hansen on 17 April 2004 ***
*** This library of transistor models contains MOSFETs based on the
*** Berkeley Predictive Technology Model parameters / technology cards.
*** No responsibility is assumed for the use of the information stated
*** ***
```

C.1 180nm High-Speed BPTM Model Cards

```
***---------------------------------------------------------------------
***          BPTM 0.18 micron high speed technology   Vtn=0.25 Vtp=-0.25
***---------------------------------------------------------------------

.LIB    BPTM180HSN_LIB .SUBCKT nmosths  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +               L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0 .param  WOTn='9.35338e-08-(4.19715e-15/l)-(1.50197e-14/w)'

M1      Drain Gate Source Bulk   BPTM180HSN_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM180HSN_TYP NMOS +Level = 49

+Lint = 4.e-08 Tox = 4.e-09 +Vth0 = 0.3999 Rdsw = 250

+lmin=1.8e-7 lmax=1.8e-7 wmin=1.8e-7 wmax=1.0e-4 Tref=27.0 version
=3.1 +Xj= 6.0000000E-08         Nch= 5.9500000E+17 +lln= 1.0000000
lwn= 1.0000000              wln= 0.00 +wwn= 0.00
ll= 0.00 +lw= 0.00                  lwl= 0.00
wint= 0.00 +wl= 0.00                  ww= 0.00
wwl= 0.00 +Mobmod=  1                binunit= 2
xl=  0 +xw=  0                    binflag=  0 +Dwg= 0.00
Dwb= 0.00

+K1= 0.5613000               K2= 1.0000000E-02 +K3= 0.00
Dvt0= 8.0000000             Dvt1= 0.7500000 +Dvt2= 8.0000000E-03
Dvt0w= 0.00                 Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 1.6500000E-07          W0= 0.00 +K3b= 0.00
Ngate= 5.0000000E+20

+Vsat= 1.3800000E+05       Ua= -7.0000000E-10          Ub=
3.5000000E-18 +Uc= -5.2500000E-11        Prwb= 0.00 +Prwg= 0.00
Wr= 1.0000000               U0= 3.5000000E-02 +A0= 1.1000000
Keta= 4.0000000E-02         A1= 0.00 +A2= 1.0000000
Ags= -1.0000000E-02         B0= 0.00 +B1= 0.00

+Voff= -0.12350000          NFactor= 0.9000000          Cit= 0.00
+Cdsc= 0.00                Cdscb= 0.00                 Cdscd= 0.00
+Eta0= 0.2200000           Etab= 0.00                  Dsub=
0.8000000
```

```
+Pclm= 5.0000000E-02       Pdiblc1= 1.2000000E-02      Pdiblc2=
7.5000000E-03 +Pdiblcb= -1.3500000E-02   Drout= 1.7999999E-02
Pscbe1= 8.6600000E+08 +Pscbe2= 1.0000000E-20     Pvag= -0.2800000
Delta= 1.0000000E-02 +Alpha0= 0.00              Beta0= 30.0000000

+kt1= -0.3700000           kt2= -4.0000000E-02         At=
5.5000000E+04 +Ute= -1.4800000           Ua1= 9.5829000E-10
Ub1= -3.3473000E-19 +Uc1= 0.00                 Kt1l= 4.0000000E-09
Prt= 0.00

+Cj= 0.00365               Mj= 0.54                    Pb= 0.982
+Cjsw= 7.9E-10             Mjsw= 0.31                  Php= 0.841
+Cta= 0                    Ctp= 0                      Pta= 0
+Ptp= 0                    JS=1.50E-08
JSW=2.50E-13 +N=1.0                     Xti=3.0
Cgdo=2.786E-10 +Cgso=2.786E-10            Cgbo=0.0E+00
Capmod= 2 +NQSMOD= 0                 Elm= 5
Xpart= 1 +Cgsl= 1.6E-10             Cgdl= 1.6E-10
Ckappa= 2.886 +Cf= 1.069e-10             Clc= 0.0000001
Cle= 0.6 +Dlc= 4E-08                Dwc= 0
Vfbcv= -1 .ENDS .ENDL

.LIB    BPTM180HSP_LIB .SUBCKT pmosths  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +               L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0

M1      Drain Gate Source Bulk   BPTM180HSP_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM180HSP_TYP PMOS +Level = 49

+Lint = 3.e-08 Tox = 4.2e-09 +Vth0 = -0.42 Rdsw = 450

+lmin=1.8e-7 lmax=1.8e-7 wmin=1.8e-7 wmax=1.0e-4 Tref=27.0 version
=3.1 +Xj= 7.0000000E-08         Nch= 5.9200000E+17 +lln= 1.0000000
lwn= 1.0000000              wln= 0.00 +wwn= 0.00
ll= 0.00 +lw= 0.00                lwl= 0.00
wint= 0.00 +wl= 0.00                ww= 0.00
wwl= 0.00 +Mobmod=  1               binunit= 2
xl= 0.00 +xw= 0.00 +binflag=  0              Dwg= 0.00
Dwb= 0.00

+ACM= 0                    ldif=0.00                   hdif=0.00
+rsh= 0                    rd= 0                       rs= 0 +rsc=
0                    rdc= 0

+K1= 0.5560000             K2= 0.00 +K3= 0.00
Dvt0= 11.2000000            Dvt1= 0.7200000 +Dvt2= -1.0000000E-02
Dvt0w= 0.00                 Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 9.5000000E-08          W0= 0.00 +K3b= 0.00
Ngate= 5.0000000E+20

+Vsat= 1.0500000E+05       Ua= -1.2000000E-10          Ub=
1.0000000E-18 +Uc= -2.9999999E-11        Prwb= 0.00 +Prwg= 0.00
Wr= 1.0000000               U0= 8.0000000E-03 +A0= 2.1199999
```

```
Keta= 2.9999999E-02         A1= 0.00 +A2= 0.4000000
Ags= -0.1000000             B0= 0.00 +B1= 0.00

+Voff= -6.40000000E-02      NFactor= 1.4000000          Cit= 0.00
+Cdsc= 0.00                Cdscb= 0.00                 Cdscd= 0.00
+Eta0= 8.5000000           Etab= 0.00                  Dsub=
2.8000000

+Pclm= 2.0000000           Pdiblc1= 0.1200000          Pdiblc2=
8.0000000E-05 +Pdiblcb= 0.1450000       Drout= 5.0000000E-02
Pscbe1= 1.0000000E-20 +Pscbe2= 1.0000000E-20     Pvag=
-6.0000000E-02        Delta= 1.0000000E-02 +Alpha0= 0.00
Beta0= 30.0000000

+kt1= -0.3700000           kt2= -4.0000000E-02         At=
5.5000000E+04 +Ute= -1.4800000          Ua1= 9.5829000E-10
Ub1= -3.3473000E-19 +Uc1= 0.00                Kt1l= 4.0000000E-09
Prt= 0.00

+Cj= 0.00138               Mj= 1.05                    Pb= 1.24
+Cjsw= 1.44E-09            Mjsw= 0.43                  Php= 0.841
+Cta= 0.00093              Ctp= 0                      Pta=
0.00153 +Ptp= 0                    JS=1.50E-08
JSW=2.50E-13 +N=1.0                     Xti=3.0
Cgdo=2.786E-10 +Cgso=2.786E-10            Cgbo=0.0E+00
Capmod= 2 +NQSMOD= 0                Elm= 5
Xpart= 1 +Cgsl= 1.6E-10            Cgdl= 1.6E-10
Ckappa= 2.886 +Cf= 1.058e-10            Clc= 0.0000001
Cle= 0.6 +Dlc= 3E-08               Dwc= 0
Vfbcv= -1 .ENDS .ENDL
```

C.2 180nm Low-Leakage BPTM Model Cards

```
***------------------------------------------------------------------
***          BPTM 0.18 micron low leakage technology   Vtn=0.4 Vtp=-0.4
***------------------------------------------------------------------

.LIB    BPTM180LLN_LIB .SUBCKT nmostll  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +               L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0 .param  WOTn='9.35338e-08-(4.19715e-15/l)-(1.50197e-14/w)'

M1      Drain Gate Source Bulk   BPTM180LLN_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM180LLN_TYP NMOS +Level = 49

+Lint = 4e-08 Tox = 4e-09 +Vth0 = 0.5499 Rdsw = 250

+lmin=1.8e-7 lmax=1.8e-7 wmin=1.8e-7 wmax=1.0e-4 Tref=27.0 version
=3.1 +Xj= 6.0000000E-08         Nch= 5.9500000E+17 +lln= 1.0000000
lwn= 1.0000000              wln= 0.00 +wwn= 0.00
ll= 0.00 +lw= 0.00                  lwl= 0.00
wint= 0.00 +wl= 0.00                  ww= 0.00
wwl= 0.00 +Mobmod=  1                binunit= 2
xl=  0 +xw=  0                    binflag=  0 +Dwg= 0.00
Dwb= 0.00

+K1= 0.5613000               K2= 1.0000000E-02 +K3= 0.00
Dvt0= 8.0000000             Dvt1= 0.7500000 +Dvt2= 8.0000000E-03
Dvt0w= 0.00                 Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 1.6500000E-07          W0= 0.00 +K3b= 0.00
Ngate= 5.0000000E+20

+Vsat= 1.3800000E+05       Ua= -7.0000000E-10          Ub=
3.5000000E-18 +Uc= -5.2500000E-11        Prwb= 0.00 +Prwg= 0.00
Wr= 1.0000000               U0= 3.5000000E-02 +A0= 1.1000000
Keta= 4.0000000E-02         A1= 0.00 +A2= 1.0000000
Ags= -1.0000000E-02         B0= 0.00 +B1= 0.00

+Voff= -0.12350000          NFactor= 0.9000000          Cit= 0.00
+Cdsc= 0.00                Cdscb= 0.00                 Cdscd= 0.00
+Eta0= 0.2200000           Etab= 0.00                  Dsub=
0.8000000

+Pclm= 5.0000000E-02       Pdiblc1= 1.2000000E-02      Pdiblc2=
7.5000000E-03 +Pdiblcb= -1.3500000E-02   Drout= 1.7999999E-02
Pscbe1= 8.6600000E+08 +Pscbe2= 1.0000000E-20     Pvag= -0.2800000
Delta= 1.0000000E-02 +Alpha0= 0.00             Beta0= 30.0000000

+kt1= -0.3700000           kt2= -4.0000000E-02         At=
5.5000000E+04 +Ute= -1.4800000           Ua1= 9.5829000E-10
Ub1= -3.3473000E-19 +Uc1= 0.00                 Kt1l= 4.0000000E-09
Prt= 0.00

+Cj= 0.00365               Mj= 0.54                    Pb= 0.982
+Cjsw= 7.9E-10             Mjsw= 0.31                  Php= 0.841
+Cta= 0                    Ctp= 0                      Pta= 0
```

```
+Ptp= 0                    JS=1.50E-08
JSW=2.50E-13 +N=1.0                    Xti=3.0
Cgdo=2.786E-10 +Cgso=2.786E-10           Cgbo=0.0E+00
Capmod= 2 +NQSMOD= 0                 Elm= 5
Xpart= 1 +Cgsl= 1.6E-10             Cgdl= 1.6E-10
Ckappa= 2.886 +Cf= 1.069e-10             Clc= 0.0000001
Cle= 0.6 +Dlc= 4E-08                Dwc= 0
Vfbcv= -1 .ENDS .ENDL

.LIB    BPTM180LLP_LIB .SUBCKT pmostll  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +                L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0

M1      Drain Gate Source Bulk   BPTM180LLP_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM180LLP_TYP PMOS +Level = 49

+Lint = 3e-08 Tox = 4.2e-09 +Vth0 = -0.57 Rdsw = 450

+lmin=1.8e-7 lmax=1.8e-7 wmin=1.8e-7 wmax=1.0e-4 Tref=27.0 version
=3.1 +Xj= 7.0000000E-08         Nch= 5.9200000E+17 +lln= 1.0000000
lwn= 1.0000000              wln= 0.00 +wwn= 0.00
ll= 0.00 +lw= 0.00                 lwl= 0.00
wint= 0.00 +wl= 0.00                 ww= 0.00
wwl= 0.00 +Mobmod=  1                binunit= 2
xl= 0.00 +xw= 0.00 +binflag=  0              Dwg= 0.00
Dwb= 0.00

+ACM= 0                    ldif=0.00                  hdif=0.00
+rsh= 0                    rd= 0                      rs= 0 +rsc=
0                    rdc= 0

+K1= 0.5560000             K2= 0.00 +K3= 0.00
Dvt0= 11.2000000            Dvt1= 0.7200000 +Dvt2= -1.0000000E-02
Dvt0w= 0.00                 Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 9.5000000E-08          W0= 0.00 +K3b= 0.00
Ngate= 5.0000000E+20

+Vsat= 1.0500000E+05       Ua= -1.2000000E-10          Ub=
1.0000000E-18 +Uc= -2.9999999E-11        Prwb= 0.00 +Prwg= 0.00
Wr= 1.0000000               U0= 8.0000000E-03 +A0= 2.1199999
Keta= 2.9999999E-02         A1= 0.00 +A2= 0.4000000
Ags= -0.1000000             B0= 0.00 +B1= 0.00

+Voff= -6.40000000E-02      NFactor= 1.4000000          Cit= 0.00
+Cdsc= 0.00                Cdscb= 0.00                Cdscd= 0.00
+Eta0= 8.5000000           Etab= 0.00                 Dsub=
2.8000000

+Pclm= 2.0000000           Pdiblc1= 0.1200000          Pdiblc2=
8.0000000E-05 +Pdiblcb= 0.1450000        Drout= 5.0000000E-02
Pscbe1= 1.0000000E-20 +Pscbe2= 1.0000000E-20     Pvag=
-6.0000000E-02        Delta= 1.0000000E-02 +Alpha0= 0.00
Beta0= 30.0000000
```

```
+kt1= -0.3700000           kt2= -4.0000000E-02        At=
5.5000000E+04 +Ute= -1.4800000          Ua1= 9.5829000E-10
Ub1= -3.3473000E-19 +Uc1= 0.00                Kt1l= 4.0000000E-09
Prt= 0.00

+Cj= 0.00138              Mj= 1.05                    Pb= 1.24
+Cjsw= 1.44E-09           Mjsw= 0.43                  Php= 0.841
+Cta= 0.00093             Ctp= 0                      Pta=
0.00153 +Ptp= 0                   JS=1.50E-08
JSW=2.50E-13 +N=1.0                    Xti=3.0
Cgdo=2.786E-10 +Cgso=2.786E-10            Cgbo=0.0E+00
Capmod= 2 +NQSMOD= 0                Elm= 5
Xpart= 1 +Cgsl= 1.6E-10            Cgdl= 1.6E-10
Ckappa= 2.886 +Cf= 1.058e-10            Clc= 0.0000001
Cle= 0.6 +Dlc= 3E-08               Dwc= 0
Vfbcv= -1 .ENDS .ENDL
```

C.3 130nm High-Speed BPTM Model Cards

```
***--------------------------------------------------------------------
***          BPTM 0.13 micron technology
***--------------------------------------------------------------------

.LIB    BPTM130N_LIB .SUBCKT nmost  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +               L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0 .param  WOTn='9.35338e-08-(4.19715e-15/l)-(1.50197e-14/w)'

M1      Drain Gate Source Bulk   BPTM130N_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM130N_TYP NMOS +Level = 49

+Lint = 2.5e-08 Tox = 3.3e-09 +Vth0 = 0.332 Rdsw = 200

+lmin=1.3e-7 lmax=1.3e-7 wmin=1.3e-7 wmax=1.0e-4 Tref=27.0 version
=3.1 +Xj= 4.5000000E-08          Nch= 5.6000000E+17 +lln=
1.0000000            lwn= 0.00                 wln= 0.00 +wwn=
1.0000000            ll= 0.00 +lw= 0.00                  lwl= 0.00
wint= 0.00 +wl= 0.00                 ww= 0.00
wwl= 0.00 +Mobmod= 1                binunit= 2
xl= 0 +xw= 0                    binflag=  0 +Dwg= 0.00
Dwb= 0.00

+K1= 0.3661500              K2= 0.00 +K3= 0.00
Dvt0= 8.7500000            Dvt1= 0.7000000 +Dvt2= 5.0000000E-02
Dvt0w= 0.00                Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 3.5500000E-07         W0= 0.00 +K3b= 0.00
Ngate= 5.0000000E+20

+Vsat= 1.3500000E+05       Ua= -1.8000000E-09         Ub=
2.2000000E-18 +Uc= -2.9999999E-11        Prwb= 0.00 +Prwg= 0.00
Wr= 1.0000000              U0= 1.3400000E-02 +A0= 2.1199999
Keta= 4.0000000E-02        A1= 0.00 +A2= 0.9900000
Ags= -0.1000000            B0= 0.00 +B1= 0.00

+Voff= -7.9800000E-02      NFactor= 1.1000000         Cit= 0.00
+Cdsc= 0.00                Cdscb= 0.00                Cdscd= 0.00
+Eta0= 4.0000000E-02       Etab= 0.00                 Dsub=
0.5200000

+Pclm= 0.1000000           Pdiblc1= 1.2000000E-02     Pdiblc2=
7.5000000E-03 +Pdiblcb= -1.3500000E-02   Drout= 0.2800000
Pscbe1= 8.6600000E+08 +Pscbe2= 1.0000000E-20     Pvag= -0.2800000
Delta= 1.0100000E-02 +Alpha0= 0.00             Beta0= 30.0000000

+kt1= -0.3400000           kt2= -5.2700000E-02        At= 0.00
+Ute= -1.2300000           Ua1= -8.6300000E-10        Ub1=
2.0000001E-18 +Uc1= 0.00                 Kt1l= 4.0000000E-09
Prt= 0.00

+Cj= 0.0015                Mj= 0.7175511              Pb= 1.24859
+Cjsw= 2E-10               Mjsw= 0.3706993            Php=
0.7731149 +Cta= 9.290391E-04         Ctp= 7.456211E-04 Pta=
```

```
1.527748E-03 +Ptp= 1.56325E-03          JS=2.50E-08 JSW=4.00E-13
+N=1.0                     Xti=3.0 Cgdo=2.75E-10 +Cgso=2.75E-10
Cgbo=0.0E+00 Capmod= 2 +NQSMOD= 0                 Elm= 5 Xpart= 1
+Cgsl= 1.1155E-10          Cgdl= 1.1155E-10 Ckappa= 0.8912 +Cf=
1.113e-10             Clc= 5.475E-08 Cle= 6.46 +Dlc= 2E-08
Dwc= 0 Vfbcv= -1 .ENDS .ENDL

.LIB    BPTM130P_LIB .SUBCKT pmost Drain    Gate     Source   Bulk
+       M=1      W=10e-6 AD='(6e-07)*w'    PD='2*(6e-07)+w' NRD=0
+                L=10e-6 AS='(6e-07)*w'    PS='2*(6e-07)+w' NRS=0

M1      Drain Gate Source Bulk   BPTM130P_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM130P_TYP PMOS +Level = 49

+Lint = 2.e-08 Tox = 3.3e-09 +Vth0 = -0.3499 Rdsw = 400

+lmin=1.3e-7 lmax=1.3e-7 wmin=1.3e-7 wmax=1.0e-4 Tref=27.0 version
=3.1 +Xj= 4.5000000E-08         Nch= 6.8500000E+18 +lln= 0.00
lwn= 0.00                  wln= 0.00 +wwn= 0.00
ll= 0.00 +lw= 0.00                  lwl= 0.00
wint= 0.00 +wl= 0.00                 ww= 0.00
wwl= 0.00 +Mobmod=  1                binunit= 2
xl=  0 +xw=  0                    binflag=  0 +Dwg= 0.00
Dwb= 0.00

+K1= 0.4087000             K2= 0.00 +K3= 0.00
Dvt0= 5.0000000            Dvt1= 0.2600000 +Dvt2= -1.0000000E-02
Dvt0w= 0.00                Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 1.6500000E-07         W0= 0.00 +K3b= 0.00
Ngate= 5.0000000E+20

+Vsat= 1.0500000E+05       Ua= -1.4000000E-09         Ub=
1.9499999E-18 +Uc= -2.9999999E-11        Prwb= 0.00 +Prwg= 0.00
Wr= 1.0000000              U0= 5.2000000E-03 +A0= 2.1199999
Keta= 3.0300001E-02        A1= 0.00 +A2= 0.4000000
Ags= 0.1000000             B0= 0.00 +B1= 0.00

+Voff= -9.10000000E-02     NFactor= 0.1250000         Cit=
2.7999999E-03 +Cdsc= 0.00               Cdscb= 0.00
Cdscd= 0.00 +Eta0= 80.0000000         Etab= 0.00
Dsub= 1.8500000

+Pclm= 2.5000000           Pdiblc1= 4.8000000E-02     Pdiblc2=
5.0000000E-05 +Pdiblcb= 0.1432509        Drout= 9.0000000E-02
Pscbe1= 1.0000000E-20 +Pscbe2= 1.0000000E-20     Pvag=
-6.0000000E-02       Delta= 1.0100000E-02 +Alpha0= 0.00
Beta0= 30.0000000

+kt1= -0.3400000           kt2= -5.2700000E-02        At= 0.00
+Ute= -1.2300000           Ua1= -8.6300000E-10        Ub1=
2.0000001E-18 +Uc1= 0.00                Kt1l= 4.0000000E-09
Prt= 0.00

+Cj= 0.0015                Mj= 0.7175511              Pb= 1.24859
+Cjsw= 2E-10               Mjsw= 0.3706993            Php=
```

```
0.7731149 +Cta= 9.290391E-04         Ctp= 7.456211E-04
Pta= 1.527748E-03 +Ptp= 1.56325E-03          JS=2.50E-08
JSW=4.00E-13 +N=1.0                     Xti=3.0
Cgdo=2.75E-10 +Cgso=2.75E-10             Cgbo=0.0E+00
Capmod= 2 +NQSMOD= 0                 Elm= 5
Xpart= 1 +Cgsl= 1.1155E-10          Cgdl= 1.1155E-10
Ckappa= 0.8912 +Cf= 1.113e-10             Clc= 5.475E-08
Cle= 6.46 +Dlc= 2E-08                Dwc= 0
Vfbcv= -1 .ENDS .ENDL
```

C.4 100nm High-Speed BPTM Model Cards

```
***-----------------------------------------------------------------------
***          BPTM 0.10 micron technology
***-----------------------------------------------------------------------

.LIB    BPTM100N_LIB .SUBCKT nmost  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +               L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0 .param  WOTn='9.35338e-08-(4.19715e-15/l)-(1.50197e-14/w)'

M1      Drain Gate Source Bulk   BPTM100N_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM100N_TYP NMOS +Level = 49

+Lint = 2.e-08 Tox = 2.5e-09 +Vth0 = 0.2607 Rdsw = 180

+lmin=1.0e-7 lmax=1.0e-7 wmin=1.0e-7 wmax=1.0e-4 +Tref=27.0
version =3.1 +Xj= 4.0000000E-08       Nch= 9.7000000E+17 +lln=
1.0000000          lwn= 1.0000000              wln= 0.00 +wwn=
0.00               ll= 0.00 +lw= 0.00                lwl= 0.00
wint= 0.00 +wl= 0.00                ww= 0.00
wwl= 0.00 +Mobmod=  1              binunit= 2                 xl=
0.00 +xw= 0.00               binflag=  0 +Dwg= 0.00
Dwb= 0.00

+ACM= 0                  ldif=0.00                   hdif=0.00
+rsh= 7                  rd= 0                       rs= 0 +rsc= 0
rdc= 0

+K1= 0.3950000           K2= 1.0000000E-02           K3= 0.00
+Dvt0= 1.0000000         Dvt1= 0.4000000             Dvt2=
0.1500000 +Dvt0w= 0.00             Dvt1w= 0.00
Dvt2w= 0.00 +Nlx= 4.8000000E-08      W0= 0.00
K3b= 0.00 +Ngate= 5.0000000E+20

+Vsat= 1.1000000E+05     Ua= -6.0000000E-10          Ub=
8.0000000E-19 +Uc= -2.9999999E-11 +Prwb= 0.00              Prwg=
0.00                 Wr= 1.0000000 +U0= 1.7999999E-02       A0=
1.1000000              Keta= 4.0000000E-02 +A1= 0.00
A2= 1.0000000              Ags= -1.0000000E-02 +B0= 0.00
B1= 0.00

+Voff= -2.9999999E-02    NFactor= 1.5000000          Cit= 0.00
+Cdsc= 0.00              Cdscb= 0.00                 Cdscd= 0.00
+Eta0= 0.1500000         Etab= 0.00                  Dsub=
0.6000000

+Pclm= 0.1000000         Pdiblc1= 1.2000000E-02      Pdiblc2=
7.5000000E-03 +Pdiblcb= -1.3500000E-02 Drout= 2.0000000
Pscbe1= 8.6600000E+08 +Pscbe2= 1.0000000E-20   Pvag= -0.2800000
Delta= 1.0000000E-02 +Alpha0= 0.00            Beta0= 30.0000000

+kt1= -0.3700000         kt2= -4.0000000E-02         At=
5.5000000E+04 +Ute= -1.4800000         Ua1= 9.5829000E-10
Ub1= -3.3473000E-19 +Uc1= 0.00               Kt1l= 4.0000000E-09
```

```
Prt= 0.00

+Cj= 0.0015             Mj= 0.72                     Pb= 1.25
+Cjsw= 2E-10            Mjsw= 0.37                   Php= 0.773
+Cjgate= 2E-14           Cta= 0                       Ctp= 0 +Pta=
0                  Ptp= 0                      JS=1.50E-08
+JSW=2.50E-13            N=1.0                        Xti=3.0
+Cgdo=3.493E-10          Cgso=3.493E-10               Cgbo=0.0E+00
+Capmod= 2               NQSMOD= 0                    Elm= 5
+Xpart= 1                cgsl= 0.582E-10              cgdl=
0.582E-10 +ckappa= 0.28           cf= 1.177e-10
clc= 1.0000000E-07 +cle= 0.6000000         Dlc= 2E-08
Dwc= 0 .ENDS .ENDL

.LIB    BPTM100P_LIB .SUBCKT pmost  Drain    Gate     Source Bulk
+       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w' NRD=0
+                L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w' NRS=0

M1      Drain Gate Source Bulk   BPTM100P_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS .model BPTM100P_TYP
PMOS +Level = 49

+Lint = 2.e-08 Tox = 2.5e-09 +Vth0 = -0.303 Rdsw = 300

+lmin=1.0e-7 lmax=1.0e-7 wmin=1.0e-7 wmax=1.0e-4 +Tref=27.0
version =3.1 +Xj= 4.0000000E-08            Nch= 1.0400000E+18
+lln= 1.0000000               lwn= 0.00
wln= 0.00 +wwn= 1.0000000              ll= 0.00
lw= 0.00 +lwl= 0.00                   wint= 0.00
wl= 0.00 +ww= 0.00                    wwl= 0.00
Mobmod=  1 +binunit= 2                  xl= 0.00
xw= 0.00 +binflag=  0                 Dwg= 0.00
Dwb= 0.00

+ACM= 0                       ldif=0.00
hdif=0.00 +rsh= 7                      rd= 0
rs= 0 +rsc= 0                       rdc= 0

+K1= 0.3910000                K2= 1.0000000E-02
K3= 0.00 +Dvt0= 2.6700001              Dvt1= 0.5300000
Dvt2= 5.0000000E-02 +Dvt0w= 0.00                 Dvt1w= 0.00
Dvt2w= 0.00 +Nlx= 7.5000000E-08           W0= 0.00
K3b= 0.00 +Ngate= 5.0000000E+20

+Vsat= 1.0500000E+05          Ua= -5.0000000E-10
Ub= 1.5000000E-18 +Uc= -2.9999999E-11 +Prwb= 0.00
Prwg= 0.00                        Wr= 1.0000000 +U0=
5.5000000E-03            A0= 2.0000000                  Keta=
4.0000000E-02 +A1= 0.00                     A2= 0.9900000
Ags= -0.1000000 +B0= 0.00                     B1= 0.00

+Voff= -7.0000000E-02         NFactor= 1.5000000
Cit= 0.00 +Cdsc= 0.00                    Cdscb= 0.00
Cdscd= 0.00 +Eta0= 0.2500000              Etab= 0.00
Dsub= 0.8000000

+Pclm= 0.1000000              Pdiblc1= 1.2000000E-02
```

```
Pdiblc2= 7.5000000E-03 +Pdiblcb= -1.3500000E-02       Drout=
0.9000000                  Pscbe1= 8.6600000E+08 +Pscbe2=
1.0000000E-20         Pvag= -0.2800000                   Delta=
1.0100000E-02 +Alpha0= 0.00                Beta0= 30.0000000

+kt1= -0.3400000               kt2= -5.2700000E-02
At= 0.00 +Ute= -1.2300000               Ua1= -8.6300000E-10
Ub1= 2.0000001E-18 +Uc1= 0.00                    Kt1l=
4.0000000E-09              Prt= 0.00

+Cj= 0.0015                    Mj= 0.7175511
Pb= 1.24859 +Cjsw= 2E-10                  Mjsw= 0.3706993
Php= 0.7731149 +Cjgate= 2E-14                Cta= 9.290391E-04
Ctp= 7.456211E-04 +Pta= 1.527748E-03            Ptp= 1.56325E-03
JS=2.50E-08 +JSW=4.00E-13                 N=1.0
Xti=3.0 +Cgdo=3.49E-10                Cgso=3.49E-10
Cgbo=0.0E+00 +Capmod= 2                    NQSMOD= 0
Elm= 5 +Xpart= 1                     cgsl= 0.582E-10
cgdl= 0.582E-10 +ckappa= 0.28                 cf= 1.177e-10
clc= 5.4750000E-08 +cle= 6.4600000               Dlc= 2E-08
Dwc= 0 .ENDS .ENDL
```

C.5 70nm High-Speed BPTM Model Cards

```
***--------------------------------------------------------------------
***          BPTM 0.07 micron high speed technology   Vtn=0.15 Vtp=-0.16
***--------------------------------------------------------------------

.LIB    BPTM70HSN_LIB .SUBCKT nmosths  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +                L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0 .param  WOTn='9.35338e-08-(4.19715e-15/l)-(1.50197e-14/w)'

M1      Drain Gate Source Bulk   BPTM70HSN_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM70HSN_TYP NMOS +Level = 49

+Lint = 1.6e-08 Tox = 1.6e-09 +Vth0 = 0.1902 Rdsw = 150

+lmin=7.0e-8 lmax=7.0e-8 wmin=0.7e-7 wmax=1.0e-4 +Tref=27.0
version =3.1  Xj= 2.9999999E-08 Nch= 1.2000000E+18 +lln= 1.0000000
lwn= 1.0000000   wln= 0.00  wwn= 0.00 +ll= 0.00    lw= 0.00 lwl=
0.00 wint= 0.00 wl= 0.00 +ww= 0.00 wwl= 0.00 Mobmod=1 binunit= 2
xl= 0.00 xw= 0.00 +Lmlt= 1 Wmlt= 1 binflag=  0 Dwg= 0.00 Dwb= 0.00

+ACM= 0 ldif=0.00 hdif=0.00 rsh= 6 rd= 0 rs= 0 rsc= 0 rdc= 0

+K1= 0.3700000 K2= 1.0000000E-02 K3= 0.00 +Dvt0= 1.3000000 Dvt1=
0.5000000 Dvt2= 2.9999999E-02 Dvt0w= 0.00 +Dvt1w= 0.00 Dvt2w= 0.00
Nlx= 7.0000000E-08 W0= 0.00 +K3b= 0.00 Ngate= 5.0000000E+20

+Vsat= 1.1500000E+05 Ua= 5.0000000E-10 Ub= 1.0000000E-18 Uc=
-2.9999999E-11 +Prwb= 0.00 Prwg= 0.00 Wr= 1.0000000 U0=
2.5000000E-02 A0= 1.5000000 +Keta= 4.0000000E-02 A1= 0.00 A2=
1.0000000 Ags= -1.0000000E-02 +B0= 0.00 B1= 0.00

+Voff= -0.1500000 NFactor= 1.5000000 Cit= 0.00 Cdsc= 0.00 Cdscb=
0.00 +Cdscd= 1.0000000E-14 Eta0= 0.2000000 Etab= 0.00 Dsub=
1.0000000

+Pclm= 0.2500000 Pdiblc1= 1.2000000E-02 Pdiblc2= 7.5000000E-03
+Pdiblcb= -1.3500000E-02 Drout= 1.5000000 Pscbe1= 8.6600000E+08
+Pscbe2= 1.0000000E-20 Pvag= -0.2800000 Delta= 1.0000000E-02
+Alpha0= 0.00 Beta0= 30.0000000

+kt1= -0.3700000 kt2= -4.0000000E-02 At= 5.5000000E+04 +Ute=
-1.4800000 Ua1= 9.5829000E-10 Ub1= -3.3473000E-19 +Uc1= 0.00 Kt1l=
4.0000000E-09 Prt= 0.00

+Cj= 0.0015 Mj= 0.72 Pb= 1.25 Cjsw= 2E-10 Mjsw= 0.37 +Php= 0.77
Cjgate= 2E-14 Cta= 0 Ctp= 0 Pta= 0 Ptp= 0 +JS=1.50E-08
JSW=2.50E-13 N=1.0 Xti=3.0 +Cgdo=4.094E-10 Cgso=4.094E-10
Cgbo=0.0E+00 Capmod= 2 +NQSMOD= 0 Elm= 5 Xpart= 1 cgsl= 1E-10
cgdl= 1E-10 +ckappa= 0.08 cf= 1.266e-10 clc= 1.0000000E-07 cle=
0.6000000 +Dlc= 1.6E-08 Dwc= 0 .ENDS .ENDL

.LIB    BPTM70HSP_LIB .SUBCKT pmosths  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
```

```
NRD=0 +               L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0

M1      Drain Gate Source Bulk   BPTM70HSP_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM70HSP_TYP PMOS +Level = 49

+Lint = 1.5e-08 Tox = 1.7e-09 +Vth0 = -0.213 Rdsw = 280

+lmin=7.0e-8 lmax=7.0e-8 wmin=0.7e-7 wmax=1.0e-4 +Tref=27.0
version =3.1 Xj= 2.9999999E-08 Nch= 1.2000000E+18 +lln= 1.0000000
lwn= 0.00 wln= 0.00 wwn= 1.0000000 +ll= 0.00 lw= 0.00 lwl= 0.00
wint= 0.00 wl= 0.00 ww= 0.00 +wwl= 0.00 Mobmod=  1 binunit= 2 xl=
0.00 xw= 0.00 +Lmlt=  1 Wmlt=  1 binflag=  0 Dwg= 0.00 Dwb= 0.00

+ACM= 0 ldif=0.00 hdif=0.00 rsh= 7 rd= 0 rs= 0 rsc= 0 rdc= 0

+K1= 0.3800000 K2= 1.0000000E-02 K3= 0.00 Dvt0= 2.2000000 +Dvt1=
0.6500000 Dvt2= 5.0000000E-02 Dvt0w= 0.00 Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 8.0000000E-08 W0= 0.00 K3b= 0.00 Ngate= 5.0000000E+20

+Vsat= 8.5000000E+04 Ua= 1.8000000E-09 Ub= 3.0000000E-18 +Uc=
-2.9999999E-11 Prwb= 0.00 Prwg= 0.00 Wr= 1.0000000 +U0=
1.4500000E-02 A0= 1.2000000 Keta= 4.0000000E-02 +A1= 0.00 A2=
0.9900000 Ags= -0.1000000 B0= 0.00 B1= 0.00

+Voff= -0.1500000 NFactor= 1.2000000 Cit= 0.00 Cdsc= 0.00 +Cdscb=
0.00 Cdscd= 0.00 Eta0= 0.2700000 Etab= 0.00 Dsub= 0.9500000

+Pclm= 0.5500000 Pdiblc1= 1.2000000E-02 Pdiblc2= 7.5000000E-03
+Pdiblcb= -1.3500000E-02 Drout= 0.9000000 Pscbe1= 8.6600000E+08
+Pscbe2= 1.0000000E-20 Pvag= -0.2800000 Delta= 1.0100000E-02
+Alpha0= 0.00 Beta0= 30.0000000

+kt1= -0.3400000 kt2= -5.2700000E-02 At= 0.00 Ute= -1.2300000
+Ua1= -8.6300000E-10 Ub1= 2.0000001E-18 Uc1= 0.00 +Kt1l=
4.0000000E-09 Prt= 0.00

+Cj= 0.0015 Mj= 0.72 Pb= 1.25 Cjsw= 2E-10 Mjsw= 0.37 +Php= 0.77
Cjgate= 2E-14 Cta= 0 Ctp= 0 Pta= 0 Ptp= 0 +JS=1.50E-08
JSW=2.50E-13 N=1.0 Xti=3.0 +Cgdo=3.853E-10 Cgso=3.853E-10
Cgbo=0.0E+00 Capmod= 2 +NQSMOD= 0 Elm= 5 Xpart= 1 cgsl= 0.6422E-10
cgdl= 0.6422E-10 +ckappa= 0.08 cf= 1.266e-10 clc= 1.0000000E-07
cle= 0.6000000 +Dlc= 1.5E-08 Dwc= 0 .ENDS .ENDL
```

C.6 70nm Low-Leakage BPTM Model Cards

```
***-----------------------------------------------------------------------
***          BPTM 0.07 micron low leakage technology     Vtn = 0.35V Vtp = 0.30V
***-----------------------------------------------------------------------

.LIB    BPTM70LLN_LIB .SUBCKT nmostll  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +                L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0 .param  WOTn='9.35338e-08-(4.19715e-15/l)-(1.50197e-14/w)'

M1      Drain Gate Source Bulk   BPTM70LLN_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM70LLN_TYP NMOS +Level = 49

+Lint = 1.6e-08 Tox = 1.6e-09 +Vth0 = 0.3902 Rdsw = 150

+lmin=7.0e-8 lmax=7.0e-8 wmin=0.7e-7 wmax=1.0e-4 +Tref=27.0
version =3.1  Xj= 2.9999999E-08 Nch= 1.2000000E+18 +lln= 1.0000000
lwn= 1.0000000   wln= 0.00  wwn= 0.00 +ll= 0.00    lw= 0.00 lwl=
0.00 wint= 0.00 wl= 0.00 +ww= 0.00 wwl= 0.00 Mobmod=1 binunit= 2
xl= 0.00 xw= 0.00 +Lmlt= 1 Wmlt= 1 binflag=  0 Dwg= 0.00 Dwb= 0.00

+ACM= 0 ldif=0.00 hdif=0.00 rsh= 6 rd= 0 rs= 0 rsc= 0 rdc= 0

+K1= 0.3700000 K2= 1.0000000E-02 K3= 0.00 +Dvt0= 1.3000000 Dvt1=
0.5000000 Dvt2= 2.9999999E-02 Dvt0w= 0.00 +Dvt1w= 0.00 Dvt2w= 0.00
Nlx= 7.0000000E-08 W0= 0.00 +K3b= 0.00 Ngate= 5.0000000E+20

+Vsat= 1.1500000E+05 Ua= 5.0000000E-10 Ub= 1.0000000E-18 Uc=
-2.9999999E-11 +Prwb= 0.00 Prwg= 0.00 Wr= 1.0000000 U0=
2.5000000E-02 A0= 1.5000000 +Keta= 4.0000000E-02 A1= 0.00 A2=
1.0000000 Ags= -1.0000000E-02 +B0= 0.00 B1= 0.00

+Voff= -0.1500000 NFactor= 1.5000000 Cit= 0.00 Cdsc= 0.00 Cdscb=
0.00 +Cdscd= 1.0000000E-14 Eta0= 0.2000000 Etab= 0.00 Dsub=
1.0000000

+Pclm= 0.2500000 Pdiblc1= 1.2000000E-02 Pdiblc2= 7.5000000E-03
+Pdiblcb= -1.3500000E-02 Drout= 1.5000000 Pscbe1= 8.6600000E+08
+Pscbe2= 1.0000000E-20 Pvag= -0.2800000 Delta= 1.0000000E-02
+Alpha0= 0.00 Beta0= 30.0000000

+kt1= -0.3700000 kt2= -4.0000000E-02 At= 5.5000000E+04 +Ute=
-1.4800000 Ua1= 9.5829000E-10 Ub1= -3.3473000E-19 +Uc1= 0.00 Kt1l=
4.0000000E-09 Prt= 0.00

+Cj= 0.0015 Mj= 0.72 Pb= 1.25 Cjsw= 2E-10 Mjsw= 0.37 +Php= 0.77
Cjgate= 2E-14 Cta= 0 Ctp= 0 Pta= 0 Ptp= 0 +JS=1.50E-08
JSW=2.50E-13 N=1.0 Xti=3.0 +Cgdo=4.094E-10 Cgso=4.094E-10
Cgbo=0.0E+00 Capmod= 2 +NQSMOD= 0 Elm= 5 Xpart= 1 cgsl= 1E-10
cgdl= 1E-10 +ckappa= 0.08 cf= 1.266e-10 clc= 1.0000000E-07 cle=
0.6000000 +Dlc= 1.6E-08 Dwc= 0

.ENDS .ENDL
```

```
.LIB    BPTM70LLP_LIB .SUBCKT pmostll  Drain    Gate     Source
Bulk +       M=1      W=10e-6  AD='(6e-07)*w'    PD='2*(6e-07)+w'
NRD=0 +                L=10e-6  AS='(6e-07)*w'    PS='2*(6e-07)+w'
NRS=0

M1      Drain Gate Source Bulk   BPTM70LLP_TYP +       W=W L=L
AS=AS AD=AD PS=PS PD=PD M=M NRD=NRD NRS=NRS

.model BPTM70LLP_TYP PMOS +Level = 49

+Lint = 1.5e-08 Tox = 1.7e-09 +Vth0 = -0.353 Rdsw = 280

+lmin=7.0e-8 lmax=7.0e-8 wmin=0.7e-7 wmax=1.0e-4 +Tref=27.0
version =3.1 Xj= 2.9999999E-08 Nch= 1.2000000E+18 +lln= 1.0000000
lwn= 0.00 wln= 0.00 wwn= 1.0000000 +ll= 0.00 lw= 0.00 lwl= 0.00
wint= 0.00 wl= 0.00 ww= 0.00 +wwl= 0.00 Mobmod=  1 binunit= 2 xl=
0.00 xw= 0.00 +Lmlt=  1 Wmlt=  1 binflag=  0 Dwg= 0.00 Dwb= 0.00

+ACM= 0 ldif=0.00 hdif=0.00 rsh= 7 rd= 0 rs= 0 rsc= 0 rdc= 0

+K1= 0.3800000 K2= 1.0000000E-02 K3= 0.00 Dvt0= 2.2000000 +Dvt1=
0.6500000 Dvt2= 5.0000000E-02 Dvt0w= 0.00 Dvt1w= 0.00 +Dvt2w= 0.00
Nlx= 8.0000000E-08 W0= 0.00 K3b= 0.00 Ngate= 5.0000000E+20

+Vsat= 8.5000000E+04 Ua= 1.8000000E-09 Ub= 3.0000000E-18 +Uc=
-2.9999999E-11 Prwb= 0.00 Prwg= 0.00 Wr= 1.0000000 +U0=
1.4500000E-02 A0= 1.2000000 Keta= 4.0000000E-02 +A1= 0.00 A2=
0.9900000 Ags= -0.1000000 B0= 0.00 B1= 0.00

+Voff= -0.1500000 NFactor= 1.2000000 Cit= 0.00 Cdsc= 0.00 +Cdscb=
0.00 Cdscd= 0.00 Eta0= 0.2700000 Etab= 0.00 Dsub= 0.9500000

+Pclm= 0.5500000 Pdiblc1= 1.2000000E-02 Pdiblc2= 7.5000000E-03
+Pdiblcb= -1.3500000E-02 Drout= 0.9000000 Pscbe1= 8.6600000E+08
+Pscbe2= 1.0000000E-20 Pvag= -0.2800000 Delta= 1.0100000E-02
+Alpha0= 0.00 Beta0= 30.0000000

+kt1= -0.3400000 kt2= -5.2700000E-02 At= 0.00 Ute= -1.2300000
+Ua1= -8.6300000E-10 Ub1= 2.0000001E-18 Uc1= 0.00 +Kt1l=
4.0000000E-09 Prt= 0.00

+Cj= 0.0015 Mj= 0.72 Pb= 1.25 Cjsw= 2E-10 Mjsw= 0.37 +Php= 0.77
Cjgate= 2E-14 Cta= 0 Ctp= 0 Pta= 0 Ptp= 0 +JS=1.50E-08
JSW=2.50E-13 N=1.0 Xti=3.0 +Cgdo=3.853E-10 Cgso=3.853E-10
Cgbo=0.0E+00 Capmod= 2 +NQSMOD= 0 Elm= 5 Xpart= 1 cgsl= 0.6422E-10
cgdl= 0.6422E-10 +ckappa= 0.08 cf= 1.266e-10 clc= 1.0000000E-07
cle= 0.6000000 +Dlc= 1.5E-08 Dwc= 0 .ENDS .ENDL
```

D Minimal Static CMOS Cell Library

This appendix contains the description of the minimized (CyHP[33]) Static CMOS Cell Library used for comparisons in this work. The 20 cells are presented here, followed by the simulation results from the HSPICE simulations, and finally the transistors netlists used for simulation.

D.1 CyHP - Compact yet High Performance

The Compact yet High Performance Library for Short Time-to-Market with New Technologies paper [33] introduces the work the authors have done to make a minimized static CMOS cell library. The CyHP library includes only 11 or 20 cells depending on how much delay and power overhead one would be willing to accept. The delay overhead is only 2% (5% of only 11 cells are being used) in comparison with a 400 cell library, and the power overhead is around 17% for the 20-cell library. The process in which these cells are built is not noted in the paper, but one must assume, that the power figures will be different with current processes. Yet, the work of finding a minimizes set of cells is still useful, as it helps defining a minimized set in this work also. Power and delay figures are measured in this appendix for comparison purposes.

The types, names, descriptions and equivalent names in the STM $180nm$ HCMOS8 cell library (LL for low-leakage, HS for high-speed) for the 20 cells are:

Type	Name	Description	STM name
Flip-flops	D-FF x1	Flip-flop 1x drive	FDM1HS
Flip-flops	D-FF x2	Flip-flop 2x drive	FDM1HSD
Flip-flops	D-FFN x1	Flip-flop 1x drive, negative edge triggered	n/a
Inverters	INV x1	Inverter, 1x drive	IVHS
Inverters	INV x2	Inverter, 2x drive	IVHSP
Inverters	INV x4	Inverter, 4x drive	IVHSx4
Inverters	INV x8	Inverter, 8x drive	IVHSx8
Inverters	INV x16	Inverter, 16x drive	IVHSx16
Primitive gates	2-NAND x1	2-input NAND gate, 1x drive	ND2HS
Primitive gates	2-NAND x2	2-input NAND gate, 2x drive	ND2HSP
Primitive gates	2-NOR x1	2-input NOR gate, 1x drive	NR2HS
Primitive gates	2-NOR x2	2-input NOR gate, 2x drive	NR2HSP
Primitive gates	3-NAND x1	3-input NAND gate, 1x drive	ND3HS
Primitive gates	3-NOR x1	3-input NOR gate, 1x drive	NR3HS
Primitive gates	2-XNOR x1	2-input XNOR gate, 1x drive	ENHS
Compound gates	2-InvNAND x2	2-input NAND gate, 1 inverted input, 2x drive	ND2AHSP
Compound gates	2-InvNOR x2	2-input NOR gate, 1 inverted input, 2x drive	NR2AHSP
Compound gates	2-AND-NOR x1	3-input gate: (A AND B) NOR C, 1x drive	AO6HS
Compound gates	2-OR-NAND x1	3-input gate: (A OR B) NAND C, 1x drive	AO7HS
Multiplexors	2-MUXInv x1	2-input multiplexor, 1 inverted input	MUX2INHS

Figure D.1: The CyHP 20 cell library simulated with $70nm$ BPTM modelcards. All minimum sized transistors.

The propagation delays in Table D.1 are measured as the worst case propagation delay from the input reaches 90% of its final value till the output has reached 90% of its final value. this is shown in Figure D.2. The input vectors contain the inputs (A, B, C, ...) in that order, where 'A' is the input connected to the transistor closest to the output in a transistor stack. For flip-flops the two bits given in this table represents the current output state of the register and the next state half-way propagated through the register.

Circuit	Model card	Average leak	Max. leak.	Input	Min.leak.	Input	Tpd fall	Tpd rise
and-nor3	70nm HS	10.15	16.23	(010)	4.32	(111)	70	190
or-nand3	70nm HS	5.32	11.6	(100)	3.98	(010)	58	90
inv-nand x2	70nm HS	19.01	27.48	(10)	9.2	(01)	52	26
inv-nor x2	70nm HS	18.7	31.2	(01)	12.02	(10)	20	80
nand3	70nm HS	3.51	11.82	(111)	0.346	(000)	78	58
mux2	70nm HS	16.4	19.6	(001)	13.9	(000)	70	61
nand2	70nm HS	4.6	7.9	(11)	0.668	(00)	28	29
nand2 x2	70nm HS	9.2	15.8	(11)	1.34	(00)	24	24
nor3	70nm HS	3.2	17.48	(000)	0.085	(111)	25	90
nor x1	70nm HS	4.48	11.65	(00)	0.182	(11)	17	44
nor x2	70nm HS	8.9	23.3	(00)	0.363	(11)	14	40
xnor	70nm HS	17.7	21.47	(00)	15.8	(11)	58	121
inv	70nm HS	4.85	5.8	(0)	3.95	(1)	20	30
inv x2	70nm HS	9.78	11.66	(0)	7.9	(1)	18	27
inv x4	70nm HS	19.55	23.3	(0)	15.8	(1)	18	25
inv x8	70nm HS	39.13	46.6	(0)	31.6	(1)	17	23
inv x16	70nm HS	78.43	93.6	(0)	63.26	(1)	16	20
dff +	70nm HS	24.45	25.4	(01)	23.5	(00)	70	68
dff + x2	70nm HS	29	27.5	(01)	31.2	(10)	61	70
dff -	70nm HS	24.45	25.4	(00)	23.5	(01)	70	68

Table D.1: *Minimum, maximum and average leakage current and propagation delays for 20 static CMOS cells simulated with 70 nm High Speed BPTM model cards. All currents are in nano-Amps and all times in pico-seconds.*

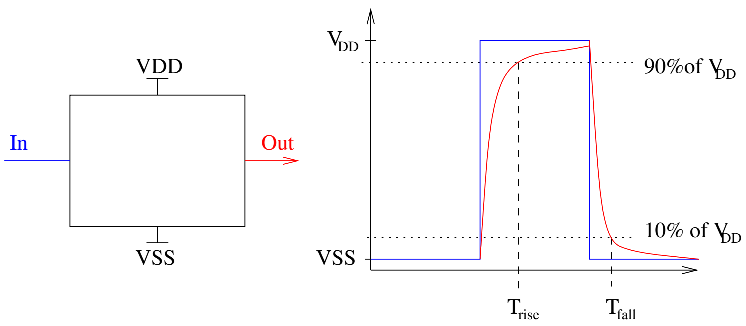


Figure D.2: *Measuring rising and falling output propagation delay*

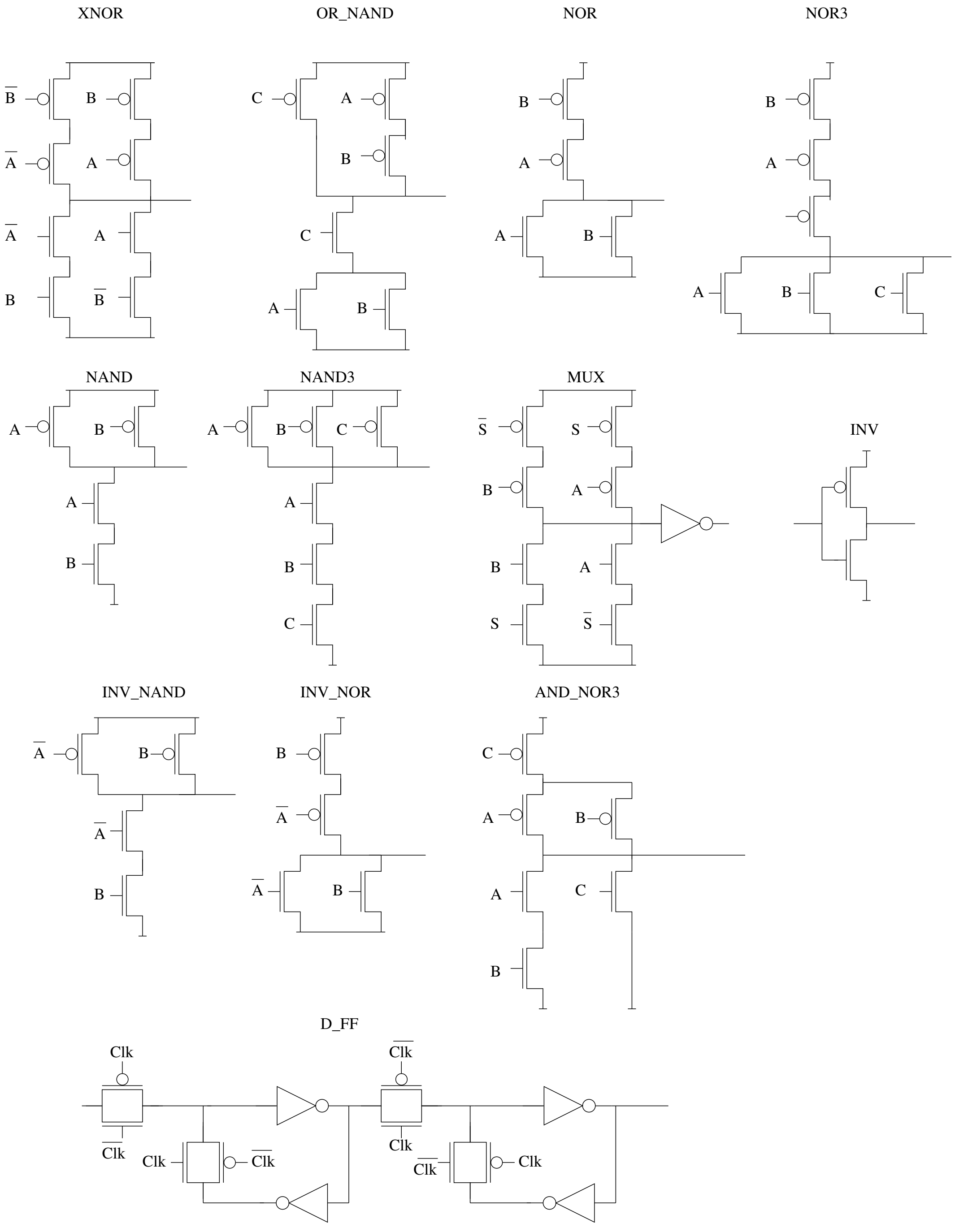


Figure D.3: *Transistor netlists for the CyHP 20 cell library. Different drive strength are modelled as multiplying the width of the output driving transistors with the drive strength factor. Negative edge triggered flip-flops were built by replacing* Clk *with* $\overline{Clk}$

E A MacroCMOS Cell

E.1 An example MacroCMOS cell

For the purpose of demonstrating the optimizations possible by building MacroCMOS cells, this example has been constructed. A logic function was derived by arbitrarily selecting 2- and 3-input logic gates using as many different logic functions as possible. Inputs to the gates were assigned pseudo-randomly also.

The logic expression is:

Z =(((A AND B)OR(C NAND D))AND(C NAND E)AND(D NOR F NOR G))XOR (H OR I)

which in logic gates is presented like this:

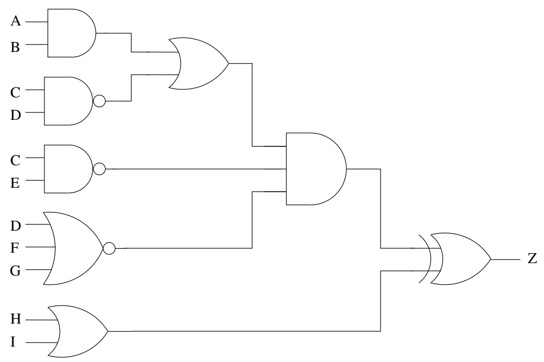


Figure E.1: *The basic design*

E.1.1 Synopsys optimizations

To explore what optimizations are possible with current synthesis tools and cell libraries, Synopsys Design Compiler was used to optimize the basic design logically.

The result was a design, where two large inverting gates, were used combined with four smaller gates. This design, depicted in Figure E.2, is the best solution under relaxed timing constraints. If the timing would be set much tighter, Synopsys comes up with the design in Figure E.3. Here, almost only NAND-gates have been used due to them being the fastest multi-input logic gate in the cell library.

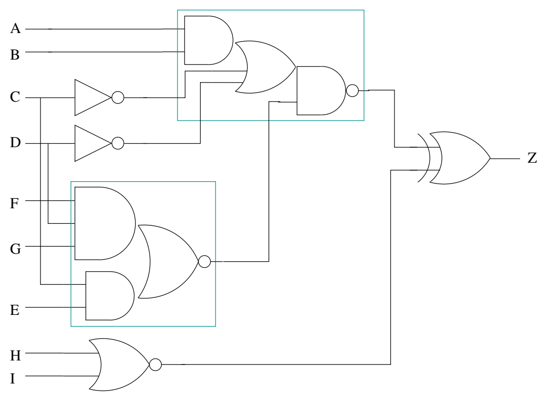


Figure E.2: *The basic design optimized under loose timing bounds*

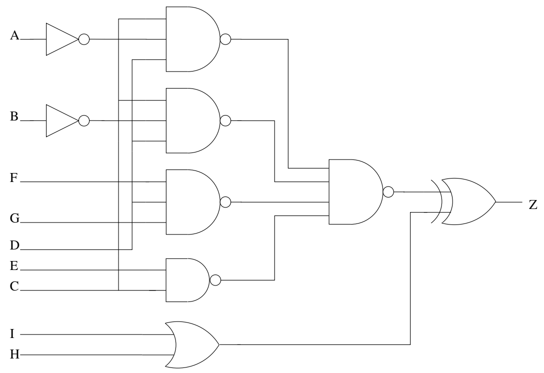


Figure E.3: *The basic design optimized for speed*

E.1.2 Leakage evaluation of the three designs

As no $70nm$ cell library is available to Synopsys, the CyHP cell library from Appendix D is used to evaluate the leakage of the designs. Whenever a cell is not present in the CyHP cell library the leakage of the cell is approximated from reconfiguration of the CyHP cells. For example can an AND-gate be approximated by a NAND-gate with inverted output. The leakage will then be the leakage of a NAND-gate and a inverter together.

Further, a NAND4-gate was simulated for fairness, as this gate most probably would be in any cell library. In the 'Synopsys optimization' the two larger cells were also designed to show the difference between the full small-cell implementation and the implementation with the two larger cells.

Design	Average leakage	Minimum leakage
Basic gate	66.53 nA	45.73 nA
Synopsys optimization	44.27 nA	28.3 nA
Synopsys opt. for speed	56.83 nA	38.58 nA

Where the designs were not simulated (but built from the CyHP table) the minimum leakage was evaluated by hand, by assessing the best low-leakage input vector. It was confirmed, that larger cells quite reduce the work load of finding such a vector.

E.1.3 The MacroCMOS implementation

The cell was now implemented in a MacroCell fashion. The first attempt was to draw the XOR gate and then replace the respective pull-up and pull-down paths with corresponding paths from the other gates. That is, if the output α from logic gate β is connected to a pMOS transistors gate, this pMOS transistor is replaced by the dual of β´s pull-down network.

Iteratively repeating this procedure yields one large gate that implements the entire logic block. The transistor netlist is presented in Figure E.4.

This block will definitely leak less than the original gate due to the decreased number of paths and the increased number of transistor on each path. But as the XOR gate was the skeleton for construction of the gate, every input is required in both inverted and non-inverted form. Inverting all inputs will cost as much leakage current as the basic gate.

The logic block was therefore built in a more inverter-regarding way. This was done by dividing the logic into larger blocks, that can be built as inverting blocks, see Figure E.5. The big logic block is then optimized by inverting it on the output and then pushing the inverters back towards the inputs by inverting functions and inputs. This implementation is called the MacroCMOS implementation. The netlist of it is depicted in Figure E.6. With this dividing of the logic block, the design was simulated for leakage. The block was a bit slower than the original design, so it was sped up a bit by transistor sizing.

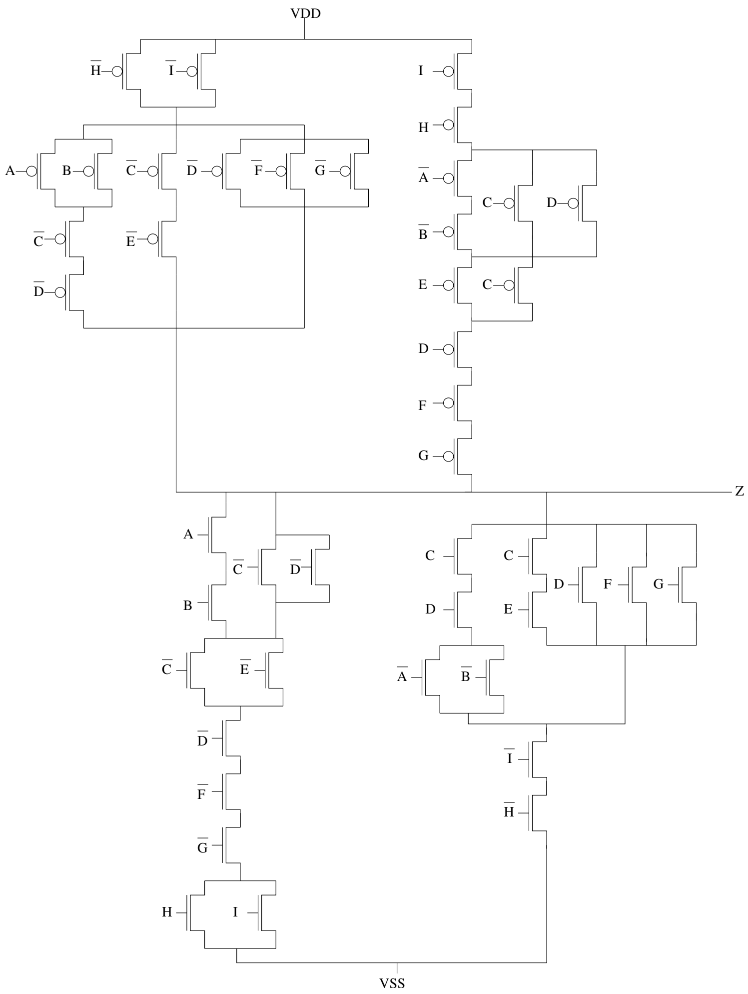


Figure E.4: *The entire logic block as one MacroCMOS transistor netlist*

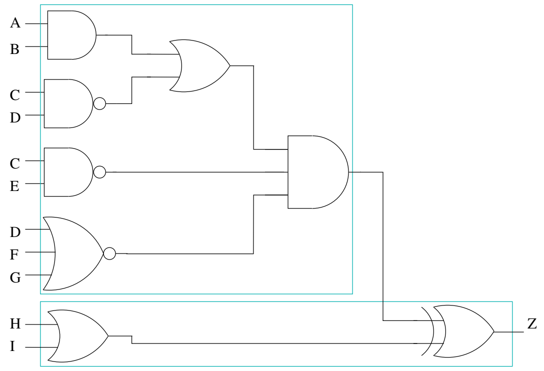


Figure E.5: *Dividing the logic into larger areas in a beneficial way.*

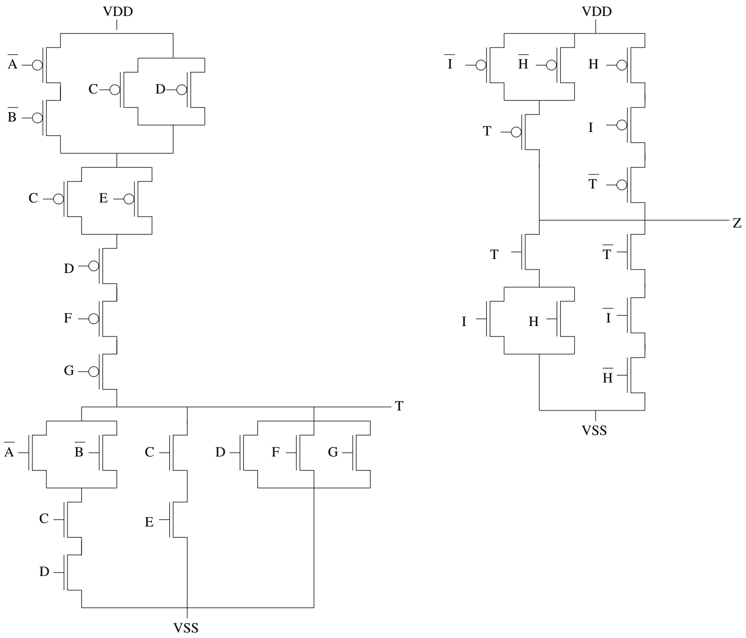


Figure E.6: *The transistor netlist of the logic divided into two large blocks.*

Figure E.7 shows simulation results. The top line (the dotted) represents the distribution of leakage currents at all 512 input states for the Synopsys logic optimized version (not the speed-optimized version). The much darker curve represents the results from the MacroCMOS version optimized to match the same speed as the Synopsys optimized version.

A variety of leakage optimizations can be done to the MacroCMOS block. By inspection of Figure E.6 transistors are located that can be sized for low leakage. The three *D*, *F* and *G* transistors in parallel have relative fast pull-down in comparison with the serialized transistor. By increasing the length of these three transistors the total leakage is reduced considerably without affecting the delay of the cell beyond the equivalent delay of the Synopsys optimized version.

By iteratively adjusting the sizings of the transistors to match the timing of the Synopsys optimized version, optimizations were graduately achieved. Only the most obvious optimization was done since the workload of manually simulating and sizing transistors is very high. Clearly this has to be automated in a synthesis tool. The last and lowest curve represents results from simulating the MacroCMOS gate after leakage optimizations.

Table E.8 summarizes the results from these simulations.

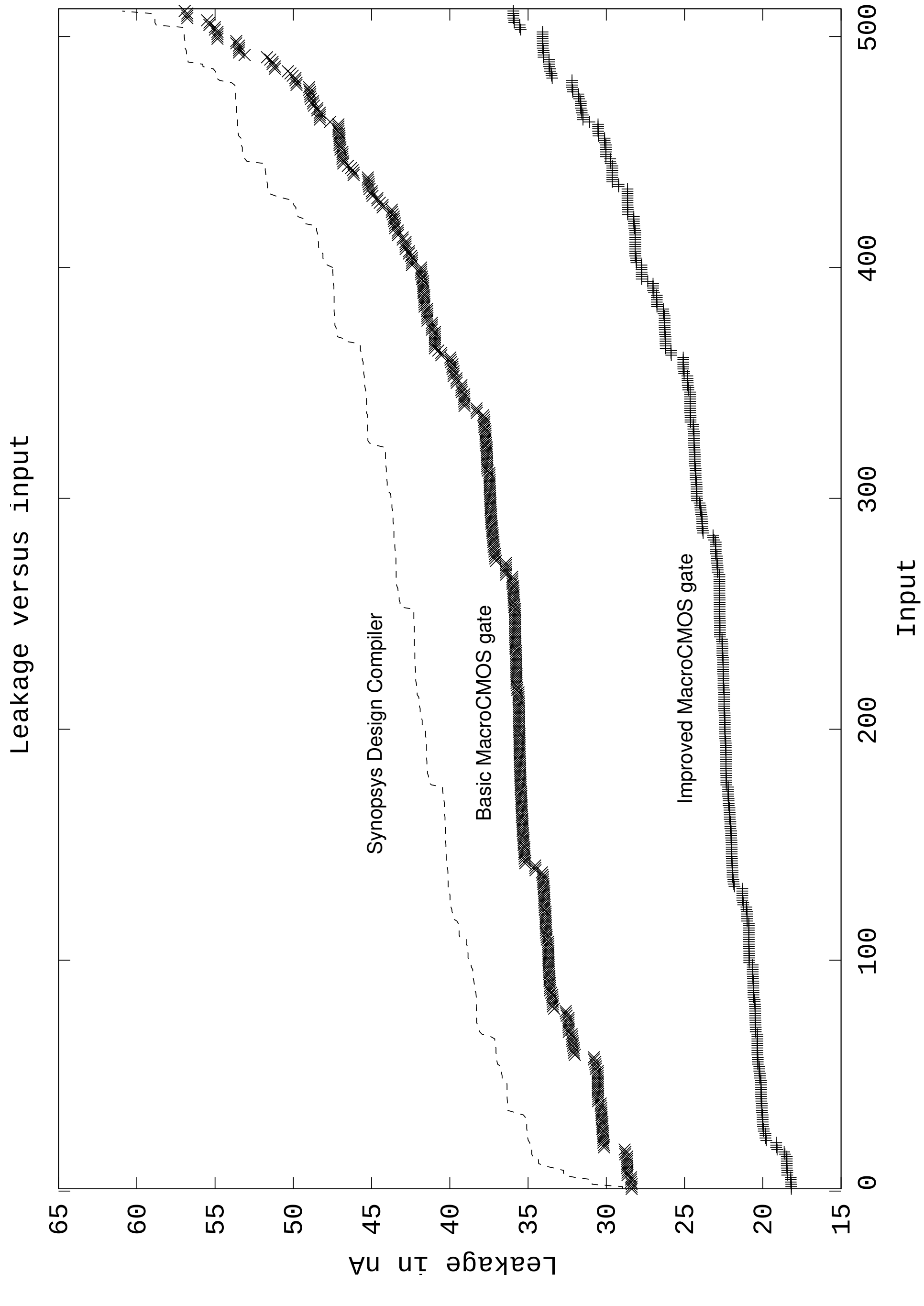


Figure E.7: *Leakage current distribution with all input 512 (9 inputs) sorted by leakage value.*

Design	Average leakage	Minimum leakage
Synopsys version	44.27 nA	29.1 nA
MacroCMOS basic	38.12 nA	28.2 nA
MacroCMOS opt. for low leakage	24.33 nA	17.2 nA

Figure E.8: *Synopsys Design Compiler versus MacroCMOS*

F Contents of Included Disk

F.1 The Contents of the Included Disk

The included disk provides the used SPICE simulation files, transistor model cards etc. together with a digital version of this report.

All files are compressed into the file *disk.tar.gz*. Here a list of the contents is given:

/Report - all files used to write this report

../FIG - all figures in this report

../PoC - not-included Proof-of-Concept of MacroCMOS

../Report.pdf - digital version of the report

/netlists - folder for all netlists/SPICE files

/netlists/allTransistors.lib - all modelcards in a library

../20cells - netlists for the 20 CyHP cells

../simulationCases - folder for the simulation work

../../20cellsSimulation - the simulation of the 20 cells

../../../LeakageEvaluation - evaluation of leakage for the 20 cells

../../../TpdEvaluation - evaluation of propagation delay of the 20 cells

../../CPL - the simulation of CPL

../../MacroCMOS - the simulation of MacroCMOS

../../Domino - the simulation of Domino logic

../../LeakageSurvey - the simulation of the introductory leaking device survey

../../MTCMOS - the simulation of MTCMOS

If problems should be encountered reading the disk, the files will also be available on the following URL: http://www.izaq.dk/pep/disk.tar.gz

Bibliography

[1] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Addison-Wesley Pubslishing Company, Second Edition ed., 1993.

[2] Martin Hans, “Architectural Aspects of Design for Low Static Power Consumption,” 2004.

[3] Michael Kristensen, “Incorporating Leakage Current Considerations in Logic Synthesis,” 2004.

[4] STMicroelectronics, “CORELIB8DHS / CORELIB8DLL HCMOS8D 3.1 Users Manual,” 2001.

[5] Synopsys Corporation, *Library Compiler: Modelling Timing and Power*. 2003.

[6] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage Current In Deep-Submicron CMOS Circuits,” *Journal of Circuits, Systems and Computers*, vol. 11, no. 6, pp. 575–600, 2002.

[7] N. sung koim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanon, “Leakage Current: Moore´s Law Meets Static Power,” *IEEE Computer Society*, 2003.

[8] C. Svensson and A. Alvandpour, “Low Power and Low Voltage CMOS Digital Circuit Techniques,” *ISPLED*, 1998.

[9] G. Sery, S. Borkar, and V. De, “Life Is CMOS: Why Chase the Life After?,” *Intel Corporation*, 2001.

[10] V. I. Authors, “International Technology Roadmap for Semiconductors 2003 Editions,” 2003.

[11] S. Thompson, P. Packan, and Mark Bohr, Intel Corporation, “MOS Scaling Transistor Challenges for the 21st Century,” *http://www.intel.com/technology/itj/q31998/articles/art_3.htm*, 2003.

[12] C. K. And, “Dynamic VTH Scaling Scheme for Active Leakage Power Reduction,” vol. citeseer.ist.psu.edu/572435.html, 1998.

[13] Z. Chen, L. Wei, M. Johnson, and K. Roy, “Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks,” *International Symposium on Low Power Electronics and Design*, vol. Proceedings of the 1998 international symposium on Low power electronics and design, 1998.

[14] G. McFarland and M. Flynn, “Limits of Scaling MOSFETs,” 1995.

[15] Y. Zhang, D. Parikh, and K. S. et.al., “HotLeakage: A Temperature-Aware Model of Subthreshold and Gatge Leakage for Architechts,” 2003.

[16] D. Lee and W. K. et.al., “Simultaneous Subthreshold and Gate-Oxide Tunnelling Leakage Current Analysios in Nanometer CMOS Desing,” 2003.

[17] X. Xiand, J. He, M. Dunga, and B. Heydari, “The Berkeley Predictive Technology Model3 ver. 3.0 Homepage,” *http://www-device.eecs.berkeley.edu/ bsim3.html*, 1998.

[18] C. Hu, "BSIM Model for Circuit Design Using Advanced Technologies," *2001 Symposium on VLSI Circuits Digest of Technical Papers*, 2001.

[19] D. Sylvester and K. Keutzer, "Rethinking Deep-Submicron Circuit Design," *Computer*, vol. 32, no. 11, pp. 25–33, 1999.

[20] W. Liu, X. Jin, and J. C. et.al., "BSIM 3v3.2.2 MOSFET Model," *Department of Electrical Engineering and Computer Sciences*, vol. University of California, Berkeley, 1999.

[21] F. Stassen, "Design Rules and Electrical Parameters for a 0.18 micron CMOS Process," *Informatics and Mathematical Modelling, Computer Science and Engineering, Technical University of Denmark*, 2004.

[22] J. M. Rabaey and M. Pedram, *Low Power Design Methodologies*. 1996.

[23] J. Kao and A. Chandrakasan, "Mtcmos sequential circuits," *Proceedings of the 27th European Solid-State Circuits Conference*, pp. 332–335, 2001.

[24] N. Hanchate and N. Ranganathan, "LECTOR: A Technique for Leakage Reduction in CMOS Circuits," 2004.

[25] A. Ghani, "High-speed low-power design in cmos," *Master's Thesis at Department of Informatics and Mathematical Modelling, Technical University of Denmark*, 2002.

[26] F. Stassen, *Practical Aspects of CMOS Layout*. Technical University of Denmark/Department of Information Technology, 1996.

[27] Q. Wang and S. B. K. Vrudhula, "Static Power Optimization of Deep Submicron CMOS Circuits for Dual Vt Technology," 1998.

[28] J. P. Halter and F. N. Najm, "A Gate-Level Power Reduction Method for Ultra-Low-Power CMOS Circuits," 1997.

[29] D. V. Campenhout, T. Mudge, and K. A. Sakallah, "Timing Verification of Sequential Dynamic Circuits," 1999.

[30] S. K. Karandikar and S. S. Spatnekar, "Technology Mapping for SOI Domino Logic Incorporating Solutions for the Parasitic Bipolar Effect," 2001.

[31] A. P. Chandrakasan, S. Sheng, and F. I. (Robert W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[32] R. Zimmermann and W. Fichtner, "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1–12, 1997.

[33] N. M. Duc and T. Sakurai, "Compact yet high performance (CyHP) library for short time-to-market with new technologies," in *Proceedings of the 2000 conference on Asia South Pacific design automation*, pp. 475–480, ACM Press, 2000.

[34] J. Kao, Chandrakasan, and D. Antoniandis, "Transistor Sizing Issues and Tool For Multi-Threshold CMOS Technology," *Proc. of DAC'97*, vol. June 1997, 1997.

[35] K. Y. et. al., "A 3.8 ns CMOS 16 x 16 multiplier using complementary pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 388–395, 1990.

[36] F. Hamzaoglu and M. R. Stan, "Circuit-level techniques to control gate leakage for sub-100nm CMOS," in *Proceedings of the 2002 international symposium on Low power electronics and design*, pp. 60–63, ACM Press, 2002.

[37] Y. Xu, Z. Luo, and X. Li, "A Maximum Total Leakage Current Estimation Method," 2004.

[38] S. K. Karandikar and S. S. Sapatnekar, "Technology Mapping for SOI Domino Logic - Incorporating Solutions for the Parasitic Bipolar Effect," *DAC*, vol. June 18-22, 2001.

[39] M. Lefebvre, D. Marple, and C. Sechen, "The Future of Custom Cell Generation in Physical Synthesis," *34th Design Automation Conference*, 1997.

[40] T. Ekebrand and N. Funke, "A Parameterizable Standard Cell Generator," 2003.

[41] D. Bhattacharya and V. Boppana, "Design Optimization with Automated Flex-Cell Creation," 2002.

[42] Various Authors from Intel Corporation, "Library Architecture Challenges for Cell-Based Design," *Intel Technology Journal*, vol. 08, no. 01, pp. 61–67, 2004.