ARCHITECTURAL ASPECTS OF DESIGN FOR LOW STATIC POWER CONSUMPTION

Martin Hans

LYNGBY 2004
EKSAMENSPROJEKT
NR. 52

IMM

Trykt af IMM, DTU

Preface

This master's thesis was conducted at the *Computer Science and Engineering* division of *Informatics and Mathematical Modeling* at the *Technical University of Denmark* from February to August 2004. Peter Østergaard Nielsen from Vitesse Semiconductor Corporation had the original idea for this thesis and has provided input to it. Flemming Stassen has acted as my supervisor. The official project description is attached as appendix A.

I would like to thank Jacob Gregers Hansen and Michael Kristensen for excellent cooperation during the work on our three theses. I would also like to thank Alberto Nannarelli for his useful ideas for my project. Finally, I'd like to thank Juliana P. Zhou for cheering me up during the distressing last weeks of my project.

Martin Hans, Copenhagen

Abstract

In the presence of non-negligible leakage power, the way to design architectures for low power consumption may have changed. This master's thesis represents one step towards exploring low power design again. This thesis shows, that area is not a sufficient predictor of leakage power consumption when delay requirements are tight.

Architectural voltage scaling is re-evaluated and it is shown that it does not always reduce leakage power. Opportunities for reducing the leakage associated with repeaters used in long on-chip wires are explored.

Furthermore, a novel architecture level power estimation method is presented which allows the designer to explore design space early in the design process.

KEYWORDS: leakage power, static power, total power, architecture, high level power estimation, architectural voltage scaling, repeater leakage

Resumé

Efterhånden som lækstrømme får mere og mere betydning, kræves ændringer af måden hvorpå man designer chips med lavt effektforbug. Dette eksamensprojekt er et skridt på vejen mod at udforske dette område på ny. Projektet demonstrerer, at areal er et utilstrækkeligt mål til forudsigelse af lækstrømme når der gælder strenge krav til forsinkelsen.

Teknikken *architectural voltage scaling* tages op til fornyet evaluering og det fremgår, at teknikken ikke altid reducerer statisk effektforbrug. Muligheder for reduktion af lækstrømmene i forbindelse med *repeaters* på lange ledninger internt på chippen diskuteres.

Endelig præsenteres en ny metode til estimering af strømforbrug på arkitekturniveau. Denne gør det muligt for designeren at udforske løsningsrummet for lavt effektforbrug tidligt i designprocessen.

STIKORD: lækstrømme, statisk strømforbrug, totalt strømforbrug, arkitektur, højniveau estimering af strømforbrug, architectural voltage scaling, repeaterlækstrømme

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Four years ago, at the beginning of my first course in computer architecture, the professor opened class by telling us, that "this course is for those who believe, that a transistor is a switch". This is just the way we as computer architects like to think about the underlying technology. We like to think, that static CMOS is simple. This way we can concentrate on the more exciting issues of devising beautiful, fast and complex architectures.

Unfortunately, this is not so. While technology scaling has made it possible to put more and more transistors onto a simple chip while at the same time allowing them to run ever faster, less simple effects are starting to show. One of the current trends is, that power consumption is becoming a serious constraint on the designs. This is not only due to the recent popularity of battery powered products, but also because the power consumption of chips has been increasing to the point where heat removal has become a costly problem, [2]. Furthermore, for environmental reasons, we are required to burn no more power than necessary.

Even though the amount of switching energy dissipated per gate has decreased with geometric downsizing, the simultaneous increase of functionality per chip as well as the increase in clock frequency have resulted in a net increase of dynamic power. The latest challenge, however, that designers are facing, is static power.

Static CMOS has become the dominating technology because it provides simplicity, great reliability, zero static power consumption and, with geometric downsizing, also high density and good speed. While CMOS continues to be the technology of choice, the static power consumption is no longer zero. With the small feature sizes of 130 nm and below reached today, a transistor is no longer a device that can be turned completely off. We can no longer ignore, that a transistor is not a perfect switch. Figure 1.1 shows that static power consumption will reach the level of dynamic power consumption within a few years time. Leakage power is becoming a serious problem that needs handling at all levels of abstraction.

This thesis concentrates on the architecture level.

1.1 Levels of solutions

Figure 1.2 shows some levels of abstraction that exist in computer architecture. In this hierarchy of abstractions the lower levels create the conditions under which the upper levels

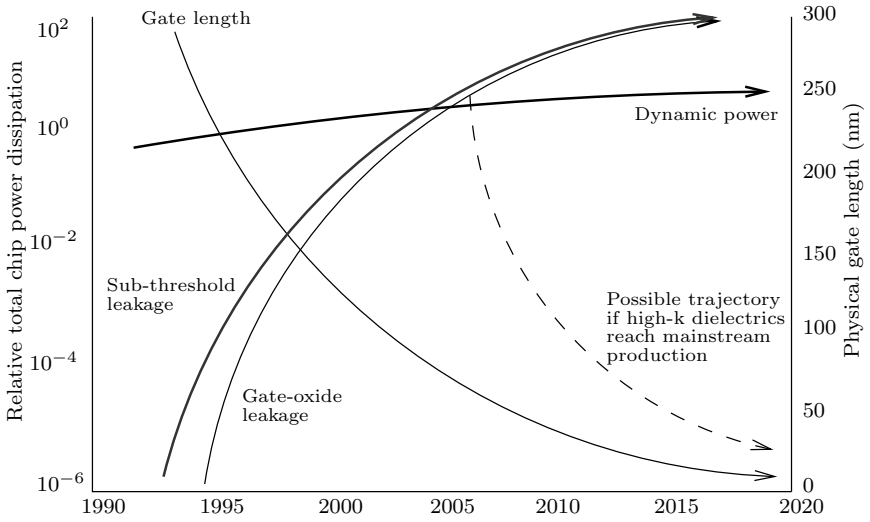


Figure 1.1: Projected development of power consumption over technology generations. Source: [1].

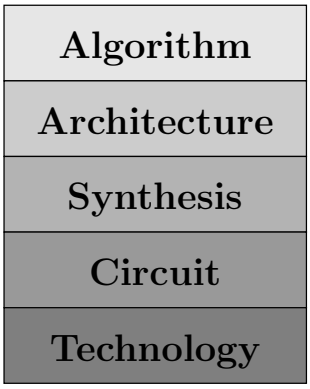


Figure 1.2: Levels of abstraction in computer architecture. Source:[2].

HS	low V_{th}	high speed	high leakage
LL	high V_{th}	low speed	low leakage

Table 1.1: High speed versus low leakage.

operate. But the details of the lower levels are abstracted away into a simple, less detailed model of the lower level.

For example, the circuit level deals with the details of the design of cells that implement logic gates. Here, transistors have to be sized to meet various requirements. The synthesizer uses a view of the circuit level that has much less detail. It views the cells as blocks with a certain function, propagation delay and area that can just be used. This way the synthesizer can concentrate on the task of manipulating circuits that meet the requirements that it has received from its own user, the architect. The circuit architect can in turn exploit the fact, that the synthesizer handles all the gory details of creating the actual netlist, analyzing timing and so forth. Thus he can concentrate on the things important at his level: Creating functionality and meeting architecture level constraints.

At the same time there is an information flow downwards. Algorithms become more complex and increase the requirements for all the underlying levels. The requirements propagate all the way down to the bottom levels and here they trigger efforts to solve the problems of area, timing and, as is the case in this thesis, power.

So while the source of the leakage problem is at the technology level, its effects range all the way up through the other levels. It is a widely accepted fact that the problem of power consumption must be handled at all levels of abstraction. At the technology level much research is being done in order to solve the problem. *Multiple Threshold CMOS* is one of the current approaches[1]. With Multiple Threshold CMOS, it is possible to have transistors with different threshold voltages (V_{th}) on the same die[2]. Typically, two versions are offered, high V_{th} and low V_{th}. As will be explained in section 2, both static power consumption and speed are highly dependent on V_{th}. As table 1.1 shows, this creates the choice between fast but leaky transistors (HS) and slower but less leaky transistors (LL).

For example, in the 70 nm process used in this thesis (see appendix B) a HS inverter has a leakage power that is 197 times the leakage of an LL inverter. At the same time, the LL inverter has a delay that is 49% higher than that of the HS inverter.

This thesis is one of three master theses performed at the same time, that deal with leakage current.

Jacob Gregers Hansen considers alternatives to static CMOS for low power design in his project *Design of CMOS cell libraries for minimal leakage currents* [1]. He re-evaluates a number of logic families under the new situation in order to decide whether static CMOS is still the best technology.

Michael Kristensen is concerned with logic synthesis in his thesis *Incorporating leakage current considerations in logic synthesis* [4]. Michael looks at the state of the art of logic synthesis and technology mapping and explores ways to reduce leakage power consumption.

This thesis deals with the problem at the architectural level of abstraction. Within this

[1] In this thesis, the common abbreviation MTCMOS will be avoided, since it has been used to designate a number of different things, ranging from the possibility to have transistors with different V_{th} on the same die to a specific circuit style using such transistors to implement fine-grained power supply gating.

[2] Some texts write the threshold voltage as V_{t}.

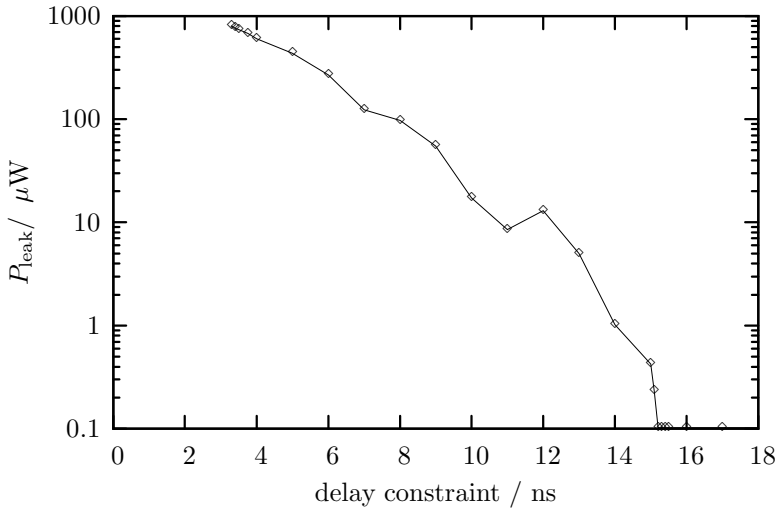


Figure 1.3: Leakage power consumption of a multiplier under varying timing constraints.

text, the term architecture refers to the register-transfer level (RTL) of abstraction, where the building blocks are components such as adders, memories and controllers.

Note, that the full architectural level also contains what is known as system level, which includes the context that the chip is used in, including things as printed circuit boards, software, power supply and connections between separate machines. In this thesis, only system level in the sense of *System on Chip* is considered. In other words: The discussion stays on chip.

The algorithm level is not dealt with at this time.

1.2 Scope of this project

The current situation is, that while leakage is becoming an increasingly urgent problem, architects are only beginning to think about the consequences it has for their work. Meanwhile, the necessary tools already exist. With cell libraries based on Multiple Threshold CMOS, circuits can be built, that only leak in the places where the extra speed of the HS cells is needed. The newest generation of synthesis tools automatically consider leakage power and choose between HS and LL cells.

This is illustrated for a multiplier synthesized under a number of timing constraints in figure 1.3. The figure shows, that a tighter timing constraint can result in considerably increased leakage power consumption. This example will be revisited later.

One thing that is still lacking is a strategy for the architectural level. Over the years, a wide range of architecture level techniques for low power design have been established in digital chip design. Power management, architectural voltage scaling, high level power estimation, caching, and bus encoding are examples of these. But all of these techniques were developed for minimizing dynamic power consumption. Now that total power consumption no longer equals dynamic power consumption, the way that low power design should be done might have changed.

The aim of this thesis is to re-evaluate a few existing techniques for low power design and examine how they should be applied in the presence of significant leakage. This thesis

is intended as one step towards the goal of understanding again how the choices made at the architectural level of the design process influence the power consumption of the resulting design. The aim is low total power consumption, not only low static power consumption, although the discussion will focus on the static contribution. The emphasis is on synchronous designs.

The subjects chosen for closer examination are architecture level power estimation, architectural voltage scaling and minimization of the wire associated with leakage. The motivation for choosing these will be explained in later chapters.

In this thesis, the existence of multiple threshold CMOS libraries is assumed, namely a library with two threshold voltages, HS and LL. Furthermore synthesis tools that are able to handle this are assumed to exist. In the work done here, the Synopsys Design Compiler is used, which can do this, although it is not very good at it. The discussion will be kept as independent from the specifics of the tool used as possible. The choice of a target process and cell library assumes to have been already made. This degree of freedom is not considered in the discussion.

1.3 Overview of the thesis

This thesis is structured into eight chapters and five appendices.

Chapter 2 takes the reader to the source of the problem at the technology level. Here we take a look at how power is consumed in static CMOS and which parameters influence the power consumption.

Chapter 3 discusses what architectural level techniques could be useful for minimizing leakage power consumption. Two of these have been chosen for closer examination and this choice is explained there.

Chapter 4 presents the design example, that will be used in chapters 5 to 7 for illustration of the techniques discussed.

Chapter 5 surveys a number of methods for power estimation at the architectural level. Power estimation is a necessary tool for the designer because it allows him to estimate the consequences of his choices for power before even implementing the design. A novel method of architectural power estimation is proposed.

Chapter 6 takes an in-depth discussion about the first of the two techniques that were chosen for closer examination in chapter 3, architectural voltage scaling. The findings are illustrated by application on the design example presented in chapter 4.

Chapter 7 is about on-chip communication. Here, part of the design space associated with long on-chip wires is explored.

Chapter 8 discusses the results obtained, points out directions for future work and concludes the thesis.

Finally, a number appendices document some of the details of the work done.

Appendix A contains the official project description for this thesis.

Appendix B describes the creation of a 70 nm cell library for use in this project.

Appendix C describes the framework created during this project used for characterization of library components with Synopsys

Appendix D describes the wire sizing tool created for evaluation of some of the methods discussed in chapter 7.

Appendix E is a collection of digital appendices contained on the CD-ROM attached. This contains the tools described in appendices B to D as well as the raw simulation data. It also contains an implementation of the power estimation tool proposed in chapter 5 in the form of a spreadsheet.

Chapter 2

Theory of power consumption

In static CMOS, power consumption can be divided into two contributions: *static* and *dynamic*. This chapter presents theory on both of these parts for use in later chapters.

The last section in this chapter summarizes the consequences for the architect.

2.1 Dynamic power consumption

Dynamic power consumption has been studied and handled as long as CMOS has existed. Its properties are very well known. Therefore, only a short introduction to dynamic power consumption will be given here.

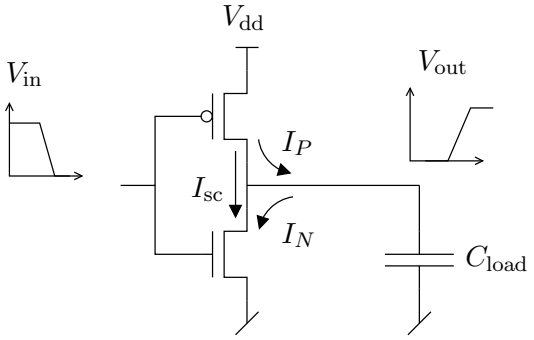


Figure 2.1: Dynamic power in a CMOS inverter. Source: [3, p. 6].

The inverter shown in figure 2.1 will be used for illustration. During the falling transition of the input voltage V_{in} there is a short period of time where both the NMOS and the PMOS transistors conduct current at the same time, resulting in the short circuit current I_{sc}. The power consumed by short circuit current will be referred to as P_{sc}.

Matching the rise and fall times of the gate will result in reduced P_{sc}. In practice, however, the times are not matched, since optimizing for propagation delay can result in unmatched times.

During and after the input transition, charge is moved from V_{dd} to the output of the inverter, hereby pulling V_{out} to V_{dd}. The lumped capacitance C_{load} results from parasitic wire capacitances and from gate capacitances of the logic gates driven by the inverter.

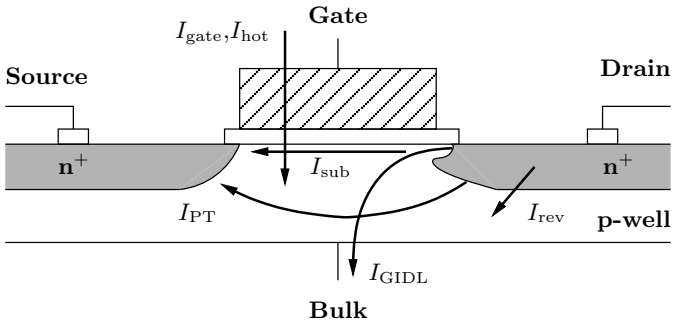


Figure 2.2: Leakage mechanisms in (NMOS) transistors.

Upon the opposite transition of the input, the PMOS transistor switches off and the NMOS transistor switches on. Now the charge stored on C_{load} is moved to ground. In summary, one rising and the following falling transition of the output consumes an energy of $C_{\text{load}}V_{\text{dd}}^2$.

If f is the clock frequency and the average number of low to high transitions (the switching activity) of the node is denoted by α then the power consumption due to capacitive switching is given by:

$$P_{\text{cap}} = \alpha C(V_{\text{dd}})^2 f \tag{2.1}$$

As a result, total dynamic power is

$$P_{\text{dyn}} = P_{\text{sc}} + P_{\text{cap}}$$

In practice, P_{cap} dominates P_{sc} as mentioned in [3, ch. 1]. It will be neglected in the following.

When changing the supply voltage from $V_{\text{dd,0}}$ to $V_{\text{dd,1}}$, the dynamic power is reduced by a factor K_{VD} as follows:

$$\begin{aligned} K_{\text{VD}} &= \frac{\alpha C(V_{\text{dd,1}})^2 f}{\alpha C(V_{\text{dd,0}})^2 f} \\ &= \frac{(V_{\text{dd,1}})^2}{(V_{\text{dd,0}})^2} \end{aligned} \tag{2.2}$$

A summary of dynamic power consumption is given at the end of this chapter.

2.2 Static power consumption

Traditionally, the static component of power consumption has been negligible in static CMOS. But as mentioned in the introduction, this is no longer the case. A number of leakage mechanisms begin to gain significance. Most of these mechanisms are directly or indirectly due to the small device geometries.

Figure 2.2 illustrates six different mechanisms in MOS transistor leakage. The following explanation of these is distilled from [5].

I_{rev} is called *reverse bias p-n junction leakage* and is caused by minority carriers drifting and diffusing across the edge of the depletion region and by electron-hole pair generation in the depletion region of the reverse bias junction.

I_{sub} is the *subthreshold leakage current* that is caused by the low V_{th} needed to maintain drive strength in processes with low V_{dd}. The result of this is, that I_{ds} can be considerable even when $V_g < V_{\text{th}}$.

I_{gate} is the *gate oxide tunneling* caused by thin gate oxides. Unlike the other effects, it occurs in both ON and OFF state of the transistor.

I_{hot} is the *gate current due to injection of hot carriers from substrate to gate oxide*. It is caused by electrons or holes gaining enough energy to enter the gate oxide layer. This current *can* occur in OFF state, but more typically it occurs during transitions of the gate voltage.

I_{GIDL} is *gate induced drain leakage*. The high field effects below the gate cause holes to accumulate at the silicon surface. This narrows the depletion edge at the drain and causes further increase in the electric field across the junction. Tunneling allows minority carriers to cross the gate and exit through the body terminal.

I_{PT}, *channel punch through leakage*, is caused by the small distance between source and drain. Due to the small geometries and due to doping profile, the depletion regions of source and drain can merge below the surface causing carriers to cross.

According to [5] and [6], I_{sub} dominates in processes down to 100 nm and I_{gate} is likely to be significant in the future. Only I_{sub} and I_{gate} will be described further in the following two sections.

2.2.1 Subthreshold leakage current

As mentioned in the introduction, subthreshold leakage occurs mainly in transistors with low V_{th}. The lowered V_{th} is dictated by the low supply voltage in small-geometry processes in order to preserve speed.

Subthreshold current flows between drain and source in an NMOS transistor when V_{GS} is below $V_{\text{th,n}}^{\text{sat}}$ (for PMOS when $V_{GS} > V_{\text{th,p}}^{\text{sat}}$). In figure 2.3, the subthreshold region is the linear part of the curves. The point of interest is at $V_{GS} = 0\,\text{V}$. The current that flows here is called I_{sub}, the power it consumes is P_{sub}. As seen in the figure, a lower V_{th} results in a higher I_{sub}.

The following expression for I_{DS} in the subthreshold region is from [5, p. 580]:

$$I_{DS} = \mu_0 C_{ox} \frac{W}{L} (m-1)(v_T)^2 e^{\frac{V_{GS}-V_{\text{th}}}{m v_T}} \left(1 - e^{\frac{-V_{ds}}{v_T}}\right) \tag{2.3}$$

V_{th} is the subthreshold voltage and v_T is the thermal voltage. μ_0 is the zero bias mobility. m is the body effect coefficient for the transistor. It is calculated by

$$m = 1 + \frac{C_{dm}}{C_{ox}}$$

where C_{dm} is the capacitance of the depletion layer and C_{ox} is the gate oxide capacitance.

The inverse of the slope of $\log(I_{DS})$ vs. V_{GS} is denoted S_t, the subthreshold slope:

$$S_t = \left(\frac{d(\log I_{DS})}{d\,\text{V}_{GS}}\right)^{-1} = 2.3\frac{mkT}{q} \tag{2.4}$$

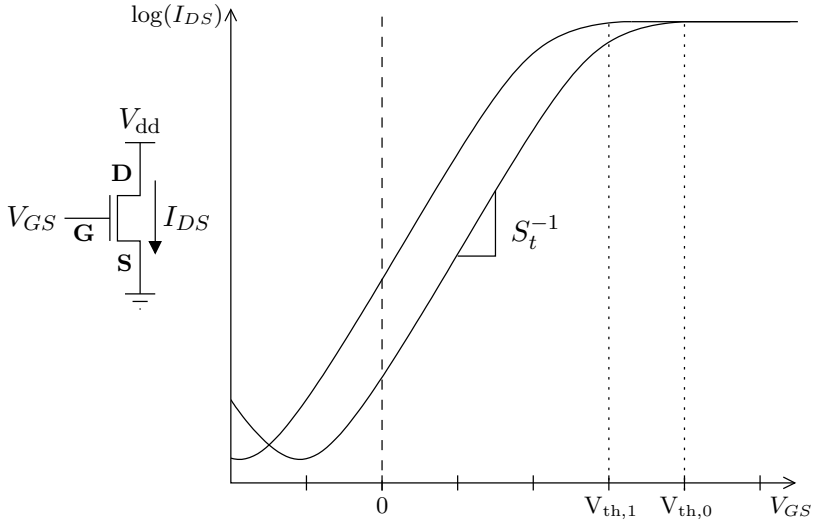


Figure 2.3: Drain-source current for two different values of V_{th}.

where q is the magnitude of electronic charge and k is Boltzmann's constant. A small value of S_t is desirable, since it means that I_{DS} can be cut off more effectively below V_{th}.

V_{th} itself is not a constant. Apart from being a complicated function of gate conductor and gate insulation materials, gate oxide thickness, impurities at the silicon-insulator interface, device geometries and the doping profile [7], it also depends on V_{DS}. This effect, which is called *drain induced barrier lowering* (DIBL), stems from the fact that the energy band diagrams from source and drain merge in short channel transistors as explained in [8], thereby lowering V_{th} by $\eta \cdot \mathrm{V}_{DS}$.

The *body effect* is another effect that influences the value of V_{th}. This effect happens when a biasing voltage is applied between well and source. The sensitivity of V_{th} on V_{BS} is as follows:

$$\frac{dV_{\mathrm{th}}}{dV_{BS}} = \frac{\sqrt{\epsilon_{st} q N_a / 2(2\psi_B + V_{SB})}}{C_{ox}} \tag{2.5}$$

This means, that V_{th} can be raised by raising the source-body voltage V_{SB}.

A model of the subthreshold current that includes both the body effect and DIBL is as follows:

$$I_{\mathrm{sub}} = A \times e^{\frac{1}{m v_T}(V_G - V_S - V_{\mathrm{th},0} - \gamma' \times V_S + \eta \times V_{DS})} \times \left(1 - e^{\frac{-\mathrm{V}_{DS}}{v_T}}\right) \tag{2.6}$$

where

$$A = \mu_0 C_{ox} \frac{W}{L_{\mathrm{eff}}} (v_T)^2 e^{1.8} e^{\frac{-\Delta V_{\mathrm{th}}}{\eta v_T}}$$

and $V_{\mathrm{th},0}$ is the zero bias threshold voltage. The body effect is presented by the term $\gamma' V_S$ where γ' is the linearized version of equation 2.5. As mentioned above, η is the DIBL coefficient. ΔV_{th} is a term that allows to take transistor-to-transistor leakage variations into account.

A consequence of this expression of the leakage is the so-called stacking effect. This arises when two or more transistors are connected in series as shown in figure 2.4. The two transistors in (b) act as a voltage divider, so both only see a V_{DS} of half the supply

voltage. Furthermore, the upper transistor has a raised V_D. As is obvious from equation 2.6, both effects reduce the leakage current.

$I_{\mathrm{leak}} = 5831\,\mathrm{pA}$ $I_{\mathrm{leak}} = 668\,\mathrm{pA}$

(a) A single transistor. (b) Two transistors in series.

Figure 2.4: The stacking effect with 70 nm transistors. Source: [1]

The stacking effect also means, that the leakage of a gate is input dependent. This is illustrated in figure 2.5.

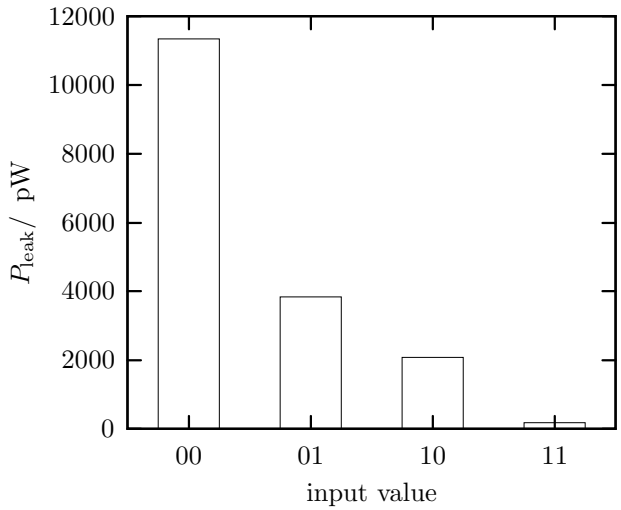


Figure 2.5: Input value dependence of subthreshold leakage in a 70 nm 2-input NOR-gate.

Subthreshold leakage is supply voltage dependent. In the following, the effect of changing the supply voltage from $V_{\mathrm{dd},0}$ to $V_{\mathrm{dd},1}$ while keeping all other factors constant will be calculated. This will change the subthreshold leakage power by a factor K_{VL} as follows:

$$\begin{aligned} K_{\mathrm{VL}} &= \frac{P_{\mathrm{sub},1}}{P_{\mathrm{sub},0}} \\ &= \frac{V_{\mathrm{dd},1} I_{\mathrm{sub},1}}{V_{\mathrm{dd},0} I_{\mathrm{sub},0}} \end{aligned}$$

$$
\begin{aligned}
&= \frac{V_{\mathrm{dd},1} \times A \times e^{\frac{1}{mv_T}(V_{\mathrm{th},0}+\eta\times V_{\mathrm{dd},1})} \times \left(1 - e^{\frac{-V_{\mathrm{dd},1}}{v_T}}\right)}{V_{\mathrm{dd},0} \times A \times e^{\frac{1}{mv_T}(V_{\mathrm{th},0}+\eta\times V_{\mathrm{dd},0})} \times \left(1 - e^{\frac{-V_{\mathrm{dd},0}}{v_T}}\right)} \\
&\approx \frac{V_{\mathrm{dd},1} \times e^{\frac{1}{mv_T}(V_{\mathrm{th},0}+\eta\times V_{\mathrm{dd},1})}}{V_{\mathrm{dd},0} \times e^{\frac{1}{mv_T}(V_{\mathrm{th},0}+\eta\times V_{\mathrm{dd},0})}} \\
&= \frac{V_{\mathrm{dd},1}}{V_{\mathrm{dd},0}} e^{\frac{\eta}{mv_T}(V_{\mathrm{dd},1}-V_{\mathrm{dd},0})} \qquad (2.7)
\end{aligned}
$$

Here, $V_{DS} = V_{\mathrm{dd}}$ and $V_{GS} = 0\,\mathrm{V}$ is assumed and the body effect is neglected. The factor in the parenthesis can be approximated away, because it will be very close to 1 for all realistic values of V_{dd}.

From this expression it can be seen, that P_{sub} is exponentially dependent on V_{dd}. A graph of K_{VL} against $V_{\mathrm{dd},1}$ is shown in figure 2.6.

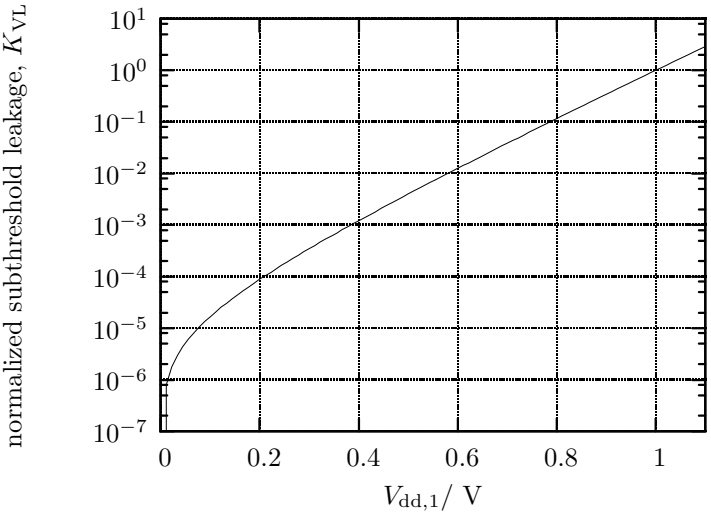


Figure 2.6: K_{VL} vs. $V_{\mathrm{dd},1}$ in a 70 nm process ($V_{\mathrm{dd},0} = 1.0\,\mathrm{V}$).

Subthreshold leakage is very temperature dependent. According to equation 2.4, S_t rises linearly with temperature. As can be seen from figure 2.3, I_{sub} rises exponentially when S_t falls. Furthermore, V_{th} decreases when the temperature rises, [8]. This gives the curve shown in figure 2.7.

2.2.2 Gate oxide tunneling

In small device geometries, the gate oxide becomes very thin, because the field strength has to be maintained when V_{dd} is reduced. In the presence of high electric fields across the gate oxide, tunneling effects begin to occur, allowing electrons and holes to cross the gate oxide. This destroys the infinite input impedance of the MOS transistors.

The gate oxide leakage is the result of several tunneling effects. They are described in [5]. For this discussion, the following simple expression given by [9] for the gate oxide

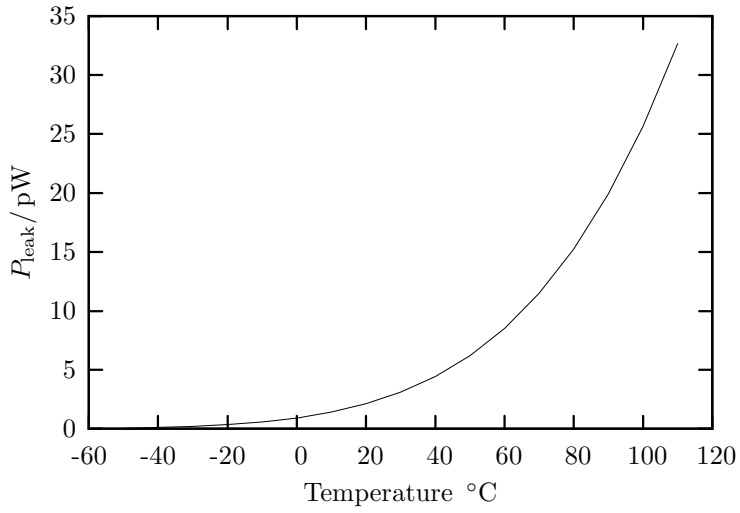


Figure 2.7: The impact of temperature on the leakage current for a 70 nm inverter.

tunneling current I_{gate} is sufficient:

$$I_{\text{gate}} = WL_{SDE}A_g \left(\frac{V_{\text{dd}}}{T_{\text{ox}}}\right)^2 \exp\left(\frac{-B_g\left[1-(1-V_{\text{dd}}/\Phi_{\text{ox}})^{\frac{3}{2}}\right]}{V_{\text{dd}}} T_{\text{ox}}\right) \tag{2.8}$$

Here L_{SDE}, the source-drain extension length, is the length of the overlap of the drain or source with the gate, so WL_{SDE} is the area causing the leakage. A_g and B_g are physical parameters determined by the process and T_{ox} is the oxide thickness. Φ_{ox} is the barrier height of tunneling electron or hole.

A sketch of I_{gate} versus V_{dd} and T_{ox} is shown in figure 2.8. I_{gate} rises quickly with decreasing T_{ox} and exponentially with rising V_{dd}, [6].

The stacking effect as described above influences gate leakage, so the leakage of a logic gate is input dependent.

This current can become quite significant. In [10], gate oxide leakage is reported to average 37% of static power consumption in a 100 nm process for a number of benchmark circuits.

For the circuit designer, not much freedom exists in controlling I_{gate} apart from gate area and supply voltage. In future processes, high-K dielectrics for gate oxide materials instead of SiO_2 may provide a solution to this problem.

Gate oxide leakage was not modeled during the work done on this thesis. The main reason is that no transistor models were available, that include this effect (see appendix B). Furthermore, the only ways to reduce gate leakage seem to be to reduce V_{dd}, reduce transistor width and exploit the input combination sensitivity, all of which also work for sub-threshold leakage.

2.3 Summary of power consumption theory

In this chapter, the theory necessary for an understanding of the power consumption problem was presented.

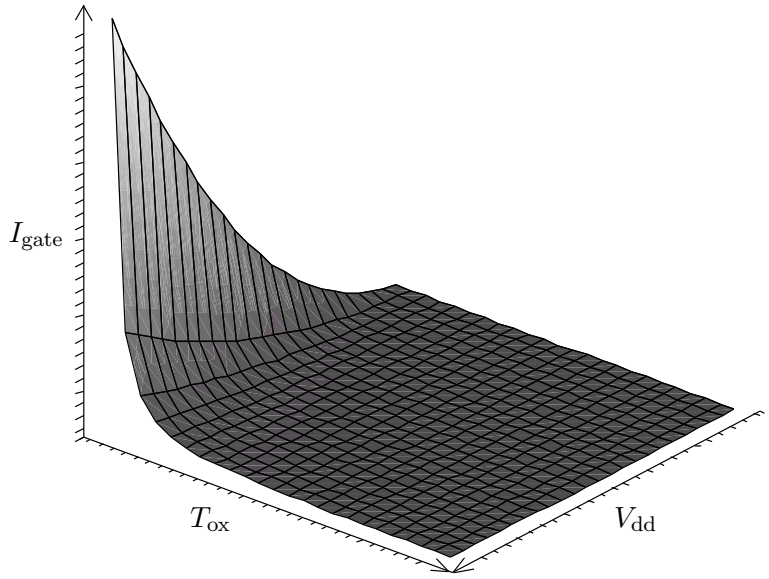


Figure 2.8: Oxide leakage dependence on T_{ox} and V_{dd}.

Static power consumption is clearly an increasing problem, especially with feature sizes below 100 nm. Increasing V_{th} and lowering V_{dd} as well as reducing the number and the size of transistors are the main tools the designer has to keep static power consumption under control. Further options such as circuit style are the subject of [1] and [4].

In practice, Multiple Threshold CMOS and cell libraries created for Multiple Threshold CMOS give the designer the choice between two types of cells, HS and LL, i.e. speed efficient cells and leakage efficient cells. Both subthreshold leakage and leakage due to gate oxide tunneling are present in LL cells, but due to the subthreshold component, these cells still exhibit a much better leakage performance than the HS cells.

In summary, the following are the possibilities the designer has to reduce static power consumption in static CMOS:

- increasing V_{th}
- reducing the total width of devices that leak
- increasing transistor stacking
- reducing operating temperature
- reducing V_{dd}
- applying less leaky inputs to gates

Similarly, the following is the list of ways to reduce the dynamic power consumption.

1. reduction of V_{dd}
2. reduction of the effective frequency of the nodal charging αf
3. reduction of the nodal capacitance C_{load}

Chapter 3

Techniques for power optimization

There is no single way to do low power design. Instead, there is a number of techniques and tricks that designers use. Each of these is more or less useful in a given situation. Design for low power is more a case to case approach than anything else.

The previous chapter lists the knobs available to the chip architect for controlling leakage. The complexity rises even further when moving from the level of single transistors and gates to the level of circuits. This introduces concepts such as critical paths and the mixture of HS and LL cells.

All in all, there is no lack of solution space. There are many possible directions the search for leakage reduction could take. This chapter discusses some of the possible options. Two of these have been chosen for closer examination in later chapters and this chapter explains why.

3.1 Arithmetic units

The many ways to make common arithmetic units such as adders and multipliers, leave the circuit architect some design freedom. For instance, a ripple carry adder provides low speed compared to a carry lookahead adder, but at the same time it consumes much less switching power, [3, sec. 7.3.1].

With the added component of leakage, this picture may have changed. Due to the smaller area of the ripple carry adder, it may also provide lower leakage. On the other hand, depending on the situation, the lower latency of the carry look-ahead adder may allow for the use of HS cells rather than LL cells, actually making a carry lookahead adder leak less than a ripple carry adder.

Today, the problem of choosing the right implementation for arithmetic units often is not the designer's task anymore. Synthesis tools are able to choose between alternative architectures given timing, area and power constraints, and although they may not do this very well for leakage constraints yet, they are likely to become better.

This problem is not investigated further during this project, because given the possibilities to let tools handle this the designer's time is probably spent better on other issues.

3.2 Dynamic power management

Power management is a technique well known from dynamic power reduction. The basic idea is to turn off units that are not in use. Turning off can be done completely by removing power supply or it can be done in a less complete way by turning off all or part of the clock tree. Of course, removing power supply is very effective, since no power consumption takes place any more. The disadvantage is, that all data stored in the circuit is lost, so they have to be stored before power down and the circuit must go through a reinitialization phase after powering up again. This may be unacceptable for timing.

Traditionally, clock gating has only been a technique used for dynamic power management. In itself it does not provide any leakage savings, as the leakage continues regardless of switching activity. But as mentioned in section 2.2.1, the leakage of CMOS gates depends on which inputs are applied to them. Therefore, some energy savings can be made by applying appropriate input vectors to combinational entities during standby. A method for finding such input vectors was proposed by [11].

A different approach is taken by a circuit style called MTCMOS, that uses *sleep devices* to turn off combinational parts of the circuit in standby mode. This can be applied to larger blocks of combinational circuitry, but as proposed by [12], also a fine grained approach at the gate level is feasible if some care is taken. However, the switches used to turn off the voltage also reduce the power supply available to the logic. This degrades performance.

Finally, leakage during standby can be reduced by lowering the supply voltage, which allows the circuit to keep it's state as proposed in [13] or by dynamically changing V_{th} which allows the circuit to continue operation at a lower speed. Adjusting V_{th} at run time can be done by controlling the body. In [14] appropriate circuitry for dynamic V_{th} scaling is presented.

Regardless of which approach is used, the management of the power takes some thought. Typically there is some wake up delay penalty that may degrade performance. A power penalty for wake up may also be incurred, so if the standby period is too short, going into standby mode may actually cost power. In order to achieve the right power management scheme, a thorough analysis should be done. Benini et al. provide a survey of the available techniques in [15, sec. 5.1].

This subject is not investigated any further due to the substantial research already done in this field.

3.3 Caches and memory

Currently, there is quite some research activity aiming at reducing leakage power consumption in memories, particularly in caches. This is quite natural as caches and SRAM blocks often take up a large fraction of the die area. At the same time, caches are typically in the critical path, so degrading their performance has a direct impact on the performance of the circuit.

Part of this work takes place at the circuit level in order to design less leaky SRAM cells, as in [16]. While this is interesting, it does not affect the work of the circuit architect much.

Some of the cache techniques that have been proposed are based on the observation

that only a small part of a cache is actually actively used during a given part of time. In [17], Powell et al. propose a method called V_{dd} *gating*. The idea is to predict, which lines in the cache will not be used any more. These lines are then simply turned off. Powell et al. achieve a 62% reduction in leakage power at a 4% penalty in execution time.

Flautner et al. use a different approach called *Drowsy caches* that achieves a similar reduction in leakage with only 1% performance degradation, [18]. Their approach switches lines that are not likely to be used again to a second, lower supply voltage. This supply voltage is high enough that the cache does not loose data, but not high enough for reading them. For reading, the lines have to be switched to the higher supply voltage, which takes one extra clock cycle. This is less than with V_{dd} *gating*, where the value has to be fetched from the next memory level. The approach has good leakage performance, because cache lines can be turned off more aggressively since the overhead for switching them on again is rather small.

A different approach is taken by Zhang et al. in [19]. Their *frequent-value data cache* uses a simple compression method that stores frequently cached values in a shorter representation. This means, that some bits in the cache lines that hold these encoded values are not used and they can thus be turned off. Because Zhang et al. find that 49.2% of the values are frequent values in their benchmarks they achieve 33% leakage power reduction at no performance penalty.

Due to the vast amount of work already done in this area, cache leakage is not examined any further in this thesis.

3.4 Architectural voltage scaling

Voltage scaling is one of the main classical ways to reduce dynamic power. The approach is to speed up the circuit by applying techniques such as parallelization and pipelining. Afterward, the supply voltage is lowered again until the performance requirements are just met. Since speed has a more or less linear relationship to V_{dd}, but dynamic power scales quadratically with V_{dd} as implied by equation 2.1 on page 15, this procedure results in a net power saving. Architectural voltage scaling is explained in [2, sec. 4.6].

As explained in sections 2.2.1 and 2.2.2, reducing the supply voltage also has a positive effect on leakage power. Speeding up the circuit by means of architecture typically increases the area and therefore also the amount of devices that leak. But as the dependence of leakage power on supply voltage is so strong, savings can still be expected.

Now that HS and LL cells give the circuit designer yet another degree of freedom it might be worth while to examine how architectural voltage scaling can be done best. This is done in chapter 6.

3.5 Retiming

Retiming as explained in [20] and [21] is a technique for speeding up a circuit. By balancing the amount of computation done between registers, the critical path can be shortened. This technique can be applied either by tools or by the designer. While retiming is closely related to pipelining, it is a technique in its own right as it can be used to move around registers that are there for other reasons than pipelining.

Outside the realm of architectural voltage scaling, retiming is used only for minimizing switching activity. But with cell libraries containing both HS and LL cells this may change. A combinational circuit that has very strict timing requirements will have to consist of more HS cells than the same circuit under more loose timing constraints. By using retiming to even out the time spent computing between registers, it may be possible to use more LL cells and thereby reducing leakage power.

This is not investigated in this thesis, but an opportunity for using retiming for leakage reduction is pointed out in section 6.4.2.

3.6 On-Chip communication

With the downscaling of devices, the capacitance that has to be driven is increasingly dominated by wire capacitance. The result is, that communication is consuming more and more power compared to computation. In large chips, long wires with high capacitance have to be driven at high speeds, requiring strong load drivers or repeaters. For dynamic power consumption this incurs only a cost per communication, but not a cost per wire. Leakage power, however is based on the amount of hardware present, and since the strong drive-inverters can be quite leaky, some amount of resource sharing may be in order. On the other hand, by using more hardware it can be possible to loosen the timing constraints. This may in turn allow the use of LL drivers instead of HS drivers.

This issue is discussed in chapter 7.

Chapter 4

Example design task

Before moving on to describing the actual work done on power estimation (chapter 5), voltage scaling (chapter 6) and the reduction of leakage associated with wires (chapter 7), one more thing is needed. For illustration of the techniques discussed in these chapters, an example design task will be presented here.

The example has two parts. While the first is a piece of computational hardware that will be used for illustration of the power estimation method and the architectural voltage scaling technique, the second is a bus that will be used for discussing the leakage issues of long wires.

4.1 A multiply-accumulate unit

For this purpose, a simple multiply-accumulate unit was chosen based on [22]. It is to be part of a hypothetic mobile phone application, handling some DSP during telephone calls and being idle the rest of the time. It computes the following function:

$$\text{Dout} = \sum_{i=1}^{225} \text{Din\_A}_i \cdot \text{Din\_B}_i$$

The unit has two 8-bit data inputs and one 24-bit data output. It communicates by handshaking at both ends. At the input it takes 225 number pairs, multiplies each of the pairs and outputs the sum of the multiplication results at the output.

The operation of the unit is documented by a waveform and a pseudo code description. These are only here for the sake of completeness and the reader should not bother too much about the details.

The handshaking protocol is a simple request-acknowledge based pull-protocol as shown in figure 4.1 on the following page. Figure 4.2 contains the pseudo-code description of the algorithm with handshaking.

4.1.1 Performance requirements

When choosing timing and power requirements for the example, there is a number of possibilities. Figure 4.3 shows an abstract representation of the solution space with power

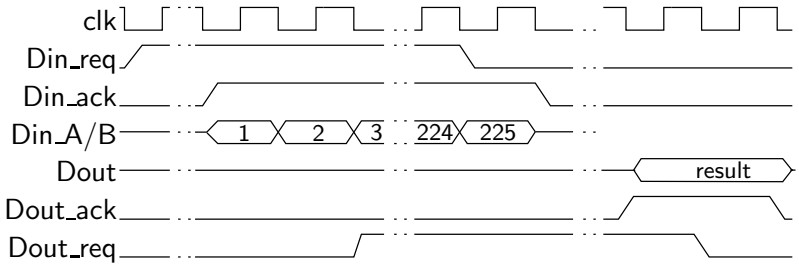


Figure 4.1: Handshaking protocol for the multiply accumulate unit.

```
1: loop
2:     Din_req ← 1
3:     wait until Din_ack = 1
4:
5:     sum ← 0
6:     for i ← 1 to 225 do
7:         a ← read
8:         b ← read
9:         sum ← sum + (a · b)
10:    end for
11:
12:    Din_req ← 0
13:    wait until Dout_req = 1
14:    output sum
15:    Dout_ack ← 1
16:    wait until Dout_req = 0
17:    Din_req ← 0
18:    wait until Din_ack = 0
19: end loop
```

Figure 4.2: Pseudo-code for the multiply-accumulate unit.

and speed requirements. Typically a minimum acceptable speed and a maximum acceptable power is dictated by the application, production cost, market etc. The curved line represents the limits to what is possible in terms of technology, cost, etc.

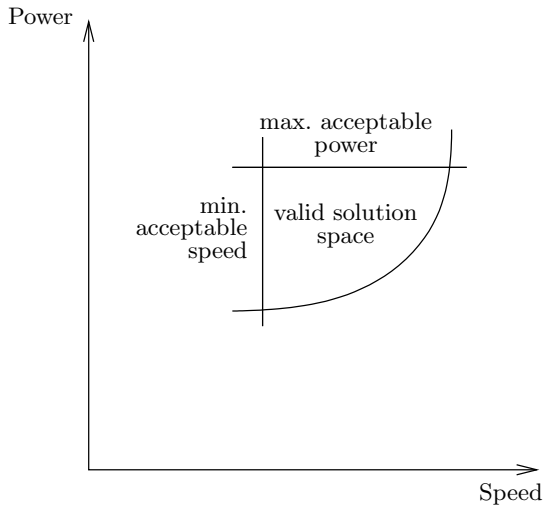


Figure 4.3: Solution space

There are different types of power constraints. Typical examples are

- average power consumption
- peak power consumption
- stand by power consumption

Timing constraints also come in various flavors such as

- latency
- throughput requirements
- local interface requirements
- global clock frequency requirements

Furthermore, timing and power requirements can be of two overall types:

exact requirements must be met, but doing better than required does not add any extra value.

elastic requirements must be met, and doing better than required is desirable.

Power requirements are typically elastic. Battery powered equipment may have some minimum battery lifetime requirement, but prolonging this can give an advantage. Exact power requirements may stem from facts such as cooling or power supply capacity.

Exact timing requirements are often seen in real time data processing, such as DSP applications, where data must be processed at a fixed sample rate. Computing results faster than needed makes no difference, since the application is I/O-bound. On the other hand, general purpose CPUs often have elastic timing requirements, as a few extra MHz will increase the market value.

That said, exact timing requirements may not be very exact at all. When designing circuits that strain the technology as much as possible in terms of speed, process variations cause some fraction of the otherwise fully functional chips to be too slow. This is typically handled by applying post-production sorting and discarding the slower chips or selling

them at a lower price. Having extra timing slack will thus increase the yield or the price at which the final product can be sold.

For the example in this text, a requirement to minimize power is of course needed. A non-trivial timing requirement is also needed, since having too much room in timing would make it too easy to eliminate leakage, as the design could simply consist of LL cells.

The following table lists the requirements chosen.

technology	the unit must be implemented in a 70 nm cell library.
supply voltage	V_{dd} must be 1.0 V or less.
throughput	once the data transfer at the input of the design has begun, one data pair must be taken every clock cycle until all 225 pairs have been read.
latency	when the last input pair has been read, a maximum of 6 clock cycles may pass before the signal Dout_ack goes high and the result is available.
clock period	the clock period is fixed at $t_{\mathrm{p}} = 8$ ns. This is an exact requirement.
average power	max. 15.5 μW during typical operation, but the less the better.

Typical operation is defined as processing data (handling phone calls) during 2% of the time it is switched on[1].

A 70 nm cell library was created as part of this project. It is described in appendix B.

Figure 4.4 on the next page shows a simple architecture implementing the algorithm. This architecture meets the requirements at $V_{\mathrm{dd}} = 1.0$ V. Apart from the multiplication and accumulation hardware, it contains a control unit to handle the handshaking and a timer unit that counts the 225 input value pairs. There is an input register to ensure that the computation hardware is given the full clock cycle. The enable signals to the registers gate the clock so that dynamic power can be assumed to be virtually zero when no computation is done (neglecting the state register in the control unit).

Given the reference implementation and these requirements, a situation has been created, where an adequate solution exists, but the solution could be optimized: The timing requirements are met. The power consumption is acceptable, but only just so. The goal is now to ameliorate the design in order to reduce the power consumption.

4.2 An on-chip bus

For use in the discussion about on-chip wires, an example including some long on-chip wires is needed. For this purpose, the setup in figure 4.5 is assumed. Here, one long on-chip 16 bit bus connects the data source with 16 multiply-accumulate units. The bus is time multiplexed, so during operation it has to supply two 8-bit numbers to each multiply-accumulate unit per 8 ns. This requires the wire to transport one sample per 0.5 ns.

[1] This roughly corresponds to 20 minutes of phone conversation on a 17-hour day. The average Danish user only uses a cell phone around 6 minutes a day, [23].

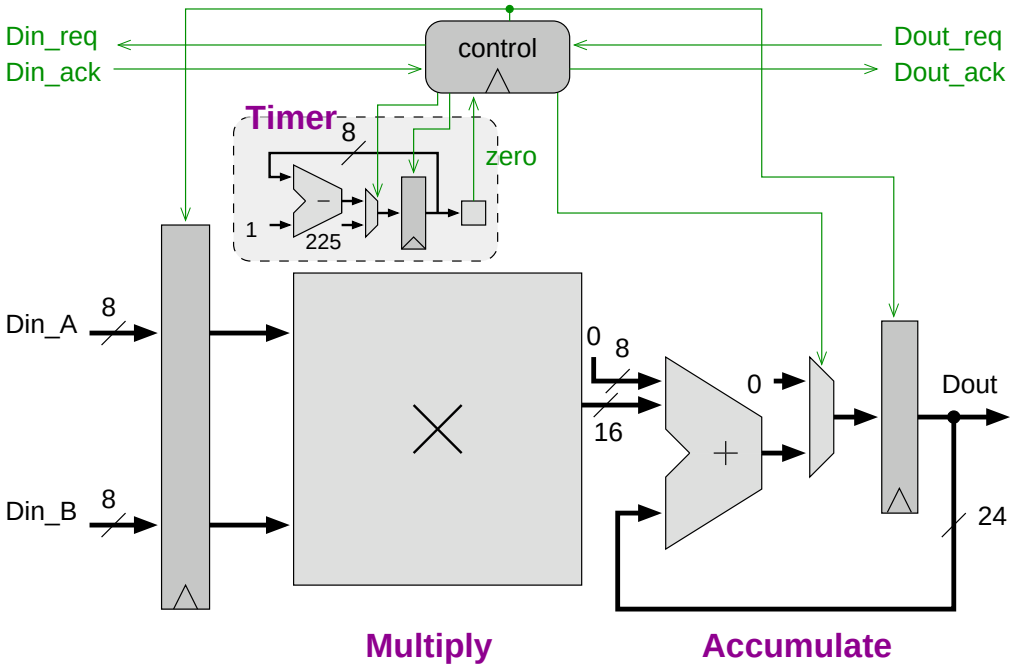


Figure 4.4: Baseline design for multiply-accumulate unit.

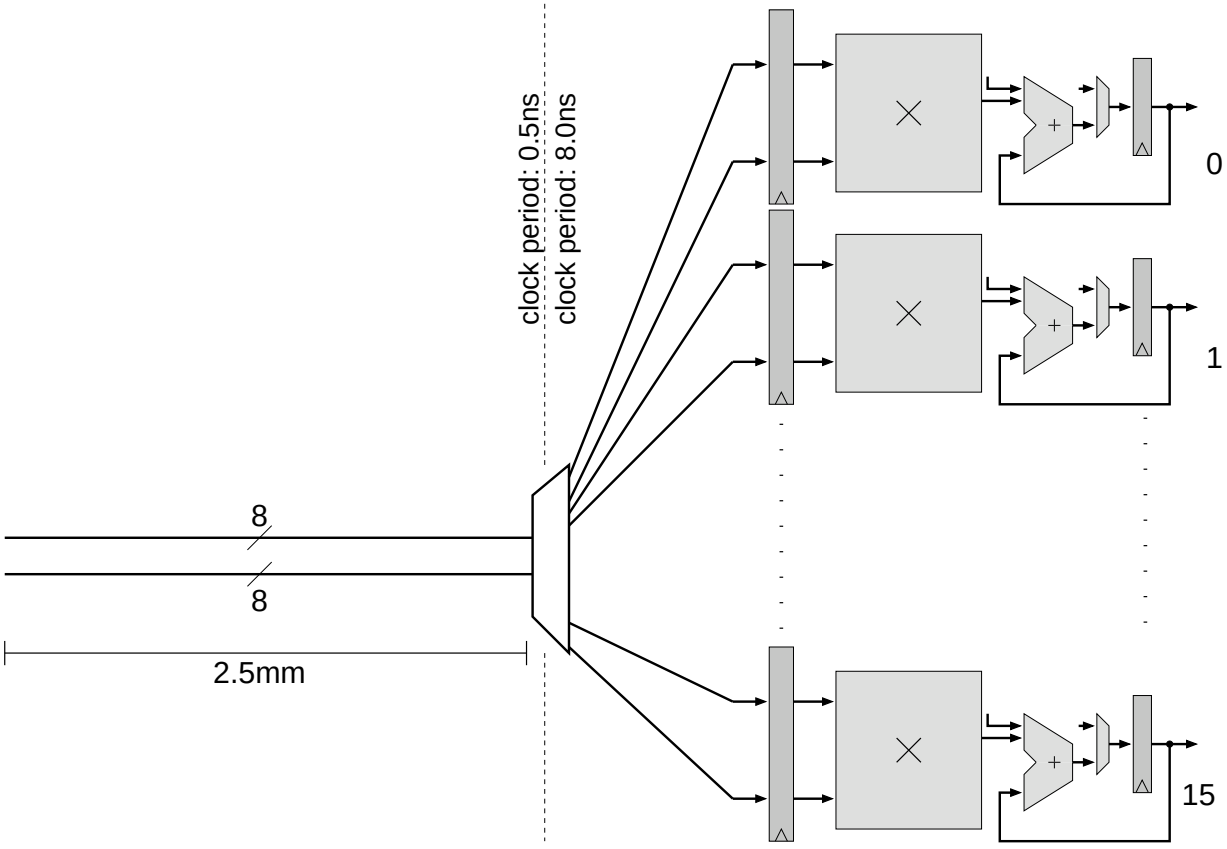


Figure 4.5: 16 multiply-accumulate units sharing one on-chip bus. The multiply-accumulate units run at 8 ns clocks which are skewed with respect to each other. The demultiplexer is a control unit that controls the data flow to the 16 units, so that one sample can be processed every 0.5 ns.

The bus is 2.5 mm long, which is expected to be the side length of the average chip in 70 nm according to [24]. The capacitance per length is set to $600\,{}^{\mathrm{fF}}/_{\mathrm{mm}}$ and the resistance per length is set to $300\,{}^{\Omega}/_{\mathrm{mm}}$. This is a rather slow wire compared to e.g. the predictions of the UC Berkeley device group at [25], but this was chosen to bring out the problem more clearly.

For simplicity, the control hardware for the multiplexed bus is not considered in this example and neither are the necessary handshaking wires.

Chapter 5

Power estimation

In order to be able to make design decisions at an early stage in the design process, the hardware designer needs a way to estimate the consequences of his choices for power consumption.

The designer needs estimation tools for several levels of abstraction. The usual tool vendors provide such tools for the RTL level and below, but these tools require the HDL code for the design to be available. No tools seem to be publicly available for design time, before any code has been written. However, the designer needs a practical way of comparing two alternative architectures for power.

This chapter proposes an estimation method for this purpose. This method will also be used in the following chapter in order to evaluate the effect of architectural voltage scaling. The power estimation for communication infrastructure is not considered here.

This chapter first presents existing work in the area. However, these approaches are here shown to be unsuccessful in Multiple Threshold CMOS, especially since leakage power consumption depends strongly on delay constraints. Therefore, the leakage characteristics of various RTL level building blocks are examined and an attempt is made to create a mathematical model of leakage. This is shown to work well for uniform logic blocks with considerable logic depths such as multipliers. A simple model is also derived for blocks with only one level of logic such as registers. However, for some types of circuits, deriving a model fails.

Instead, a model based on precharacterized logic blocks is proposed based on the well known *Spreadsheet model*. A tool can then perform look up in this characterization data in order to evaluate the leakage of a design.

The multiply-accumulate unit design presented in the previous chapter is used to illustrate the method.

Estimation of high level dynamic power consumption has been treated thoroughly in the literature and will not be discussed further here. An overview of common techniques can be found in [15, 26, 27]. Dynamic power will, however, be part of the model proposed in section 5.3.

Since the main use of an architectural estimation method is comparison of alternatives, relative accuracy of the estimation will be more important than absolute accuracy. On the other hand, since the method should estimate both the static and the dynamic part of the power consumption, the easiest way to achieve good relative accuracy probably is to achieve good absolute accuracy.

5.1 Existing work

The most straight forward way to estimate leakage power is to base the estimate on design size, since every additional gate contributes to the total leakage. This approach has been taken by a number of authors. Butts et al. [28] propose the following analytically derived model.

$$P_{\text{leak}} = V_{\text{dd}} N k_{\text{design}} \hat{I}_{\text{leak}}$$

In this model, N is the number of transistors in the design, $\hat{I}_{\text{leak}}$ is a parameter dependent on technology, and k_{design} is a design dependent parameter. This model takes different circuit styles into account by means of the parameter k_{design}, which should be determined by simulation per circuit style (SRAM, muxes, adders, etc.).

A similar, but slightly simpler model is proposed by Kumar et al. in [29]:

$$P_{\text{leak}} = \chi M^S$$

Here the design size M is the cell count of the design. The parameters χ and S are estimated by characterizing a number of designs synthesized for the target cell library. The authors find that S in practice is close to one, so that the leakage power scales linearly with the number of cells.

These models are easy to use for designers because they are very intuitive. They are appropriate for estimation at the architectural level, since, as the authors state, an estimate of the circuit size usually will be available early in the design. Kumar et al. claim to reach an accuracy better than 12.5%. Butts et al. do not give any figures for the accuracy of their model.

The leakage power model for SRAM presented by Mamidipaka et al. in [30] represents a different approach to leakage power modeling. The authors derive simple analytical parameterized leakage power models for each part of an SRAM (memory core, address decoder, read column circuit etc.) based on technology parameters. State dependent leakage is incorporated and the estimation error is less than 24%.

In the article only one specific SRAM architecture is described, but the discussion is clear and comprehensive and allows the reader to adapt the model to other SRAM architectures.

5.2 Exploration of leakage behavior

One problem with the previously mentioned high-level models for leakage power estimation is, that they don't take into account, that a circuit can consist of a mix of both HS and LL cells. With the synthesis tools and cell libraries available today, this is not realistic. Given that the leakage of a HS cell is one to two orders of magnitude higher than the leakage of an LL cell, the mix is the dominating factor. It matters much more than area.

Three different circuits are examined in the following: A multiplier, a register and a 4 to 16 one-hot encoder.

Logic blocks of large logic depth

Figure 5.1 shows the leakage characteristic of a 16-bit multiplier. Using Synopsys Design Compiler, a 16-bit multiplier was synthesized and mapped to the 70 nm cell library for a number of delay constraints in the form of a maximum allowed propagation delay, $t_{\mathrm{p,max}}$. The dotted line shows the percentage of the cells that are LL cells.

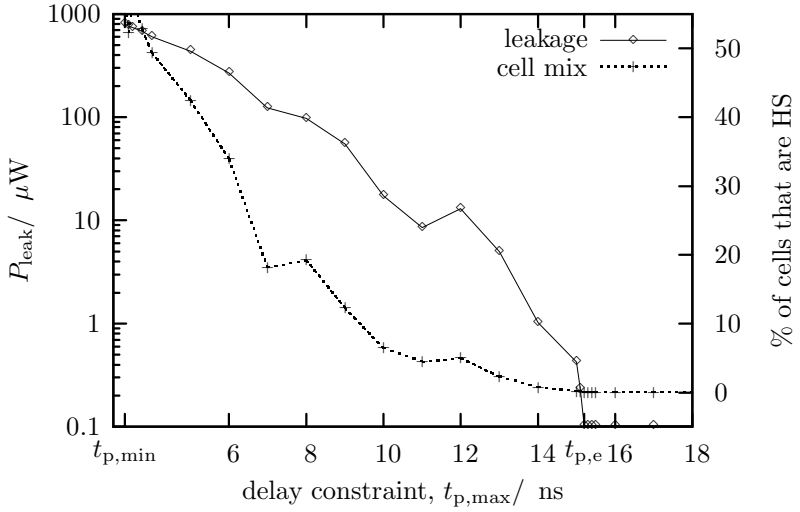


Figure 5.1: Leakage and cell mix of an 16-bit array multiplier vs. timing constraint.

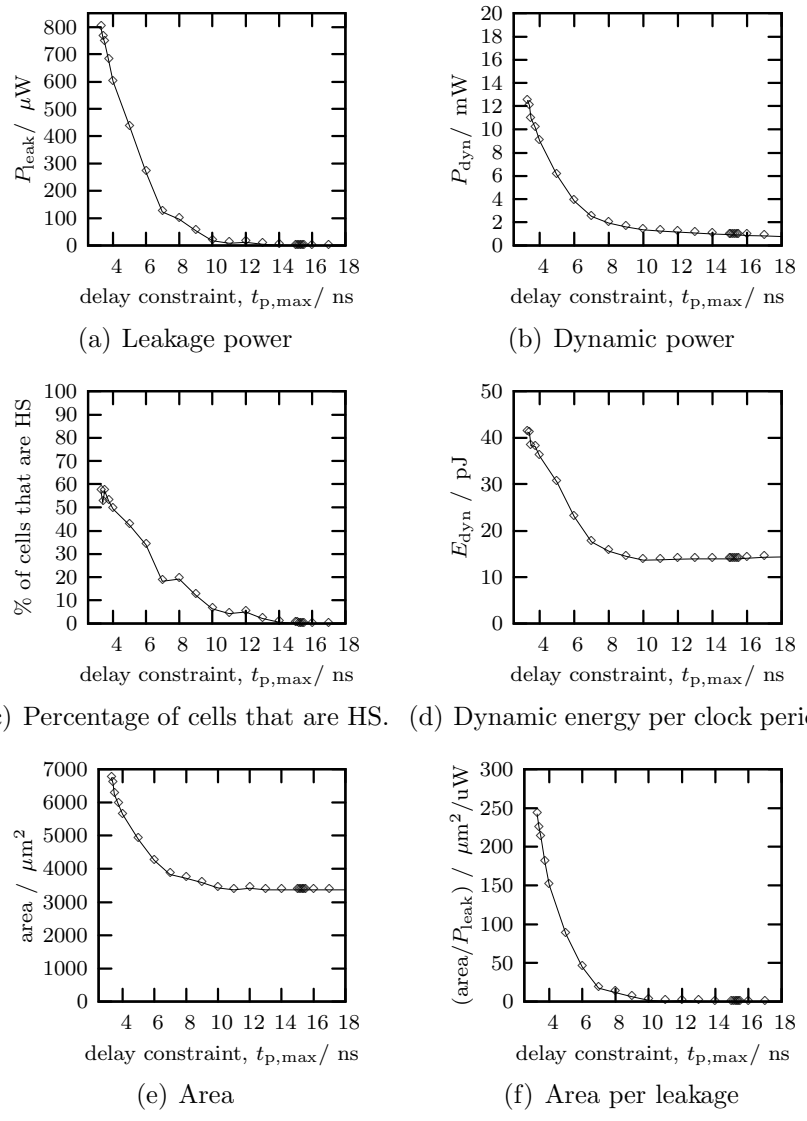
Obviously, there is a minimum delay constraint that it is possible to satisfy. This will be called $t_{\mathrm{p,min}}$. It is also clear that loosening the timing constraint past a certain point does not reduce the leakage power any further. The delay constraint at which this happens will be called $t_{\mathrm{p,e}}$. The minimum leakage power possible will be termed $P_{\mathrm{leak},\infty}$.

Figure 5.2 shows the dependence of a number of characteristics on the delay constraint. This time, the leakage power is displayed on a linear scale. Both dynamic power (fig. 5.2(b)) and area (fig. 5.2(e)) show some dependence on the delay constraint due to the synthesizer changing the structure of the circuit. This dependence is, however, much weaker than the dependency of P_{leak}. The tightening of the delay constraint merely results in a tripling of the power, while the area is less than doubled. The leakage power is increased more than 7800 times.

This enormous difference in leakage power consumption is due to a number of effects. First of all, the HS cells used at $t_{\mathrm{p,min}}$ leak about 200 times as much as the LL cells. Second, the doubled area also contributes further leaking devices. Third, the synthesizer uses cells with a higher drive strength when timing requirements are strict. These cells leak more. Finally, the synthesizer tends to use smaller cells when speed matters. Due to the lack of stacking effect, small cells leak more than the equivalent circuit built in larger cells.

In order to find a good and simple model for the leakage power dependence on $t_{\mathrm{p,max}}$, several candidates were evaluated and one candidate that provides a reasonable trade-off between simplicity and good model fit was identified:

$$P_{\mathrm{leak}} = \begin{cases} \frac{k_1}{t_{\mathrm{p,max}}+k_2} - k_3 & \text{for } t_{\mathrm{p,min}} \le t_{\mathrm{p,max}} \le t_{\mathrm{p,e}} \\ P_{\mathrm{leak},\infty} & \text{for } t_{\mathrm{p,max}} > t_{\mathrm{p,e}} \end{cases} \tag{5.1}$$



(a) Leakage power (b) Dynamic power

(c) Percentage of cells that are HS. (d) Dynamic energy per clock period

(e) Area (f) Area per leakage

Figure 5.2: Characterization data for a 16 by 16 bit multiplier.

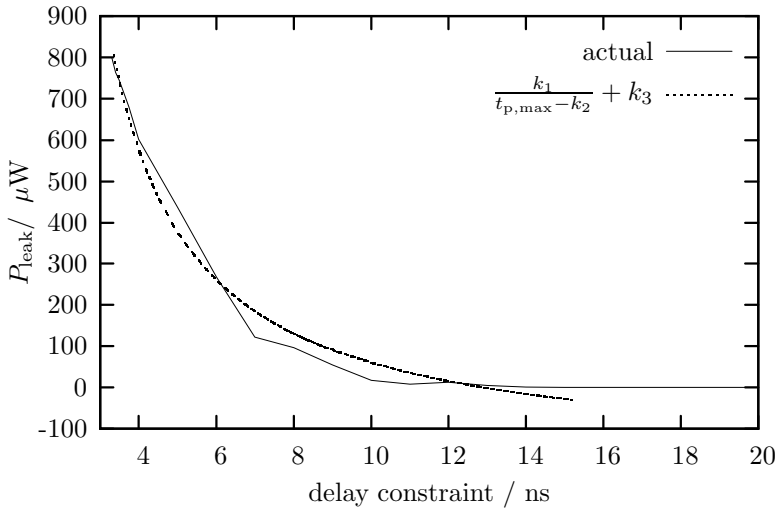


Figure 5.3: The fitted model.

Figure 5.3 shows this model fitted to the data for the multiplier. The values of the constants were found to be

$$\begin{aligned} k_1 &= 2144\,\mu\text{W}\cdot\text{ns} \\ k_2 &= 1.28\,\text{ns} \\ k_3 &= 184\,\mu\text{W} \end{aligned}$$

This model is relatively easy to use, as it provides the intuitive relationship that doubling $t_{\text{p,max}} - k_2$ will halve P_{leak} (neglecting k_3 and only below $t_{\text{p,e}}$).

The model fit is not very good. The reason for this is, that the shape of the curve is the result of several factors as mentioned above. The relative smoothness of the curve stems from the fact that a 16-bit multiplier has a relatively large logic depth.

Logic blocks with low depth

The leakage characteristic for a register is shown in figure 5.4. Here the logic depth is only one cell, so the change from all HS to all LL happens in one step. The plateau between the high and the low level, however, is due to the synthesizer choosing LL cells with a higher drive-strength in order to meet the timing requirements.

Registers have relatively high leakage per area compared to combinational circuits, so they cannot be neglected. For registers and similarly shallow circuit blocks a two-level model would be more appropriate:

$$P_{\text{leak}} = \begin{cases} k_4 & \text{for } t_{\text{p,min}} \le t_{\text{p,max}} \le t_{\text{p,e}} \\ P_{\text{leak},\infty} & \text{for } t_{\text{p,max}} > t_{\text{p,e}} \end{cases} \tag{5.2}$$

Other circuit types

Unfortunately, it is not always possible to deriving a simple leakage model for a circuit. Figure 5.5 shows the leakage characteristic for a 4 to 16 one-hot encoder. This circuit has

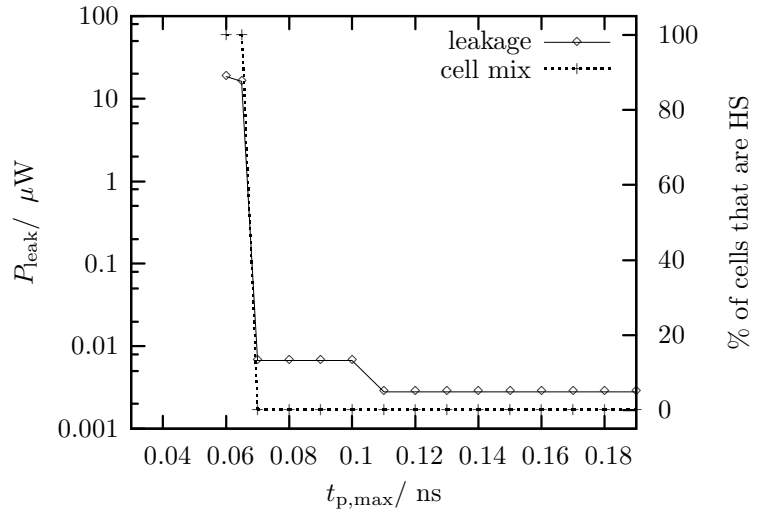


Figure 5.4: Leakage and cell mix of an 8-bit register vs. timing constraint.

only about four levels of logic. It is not clear why the synthesizer created this solution.

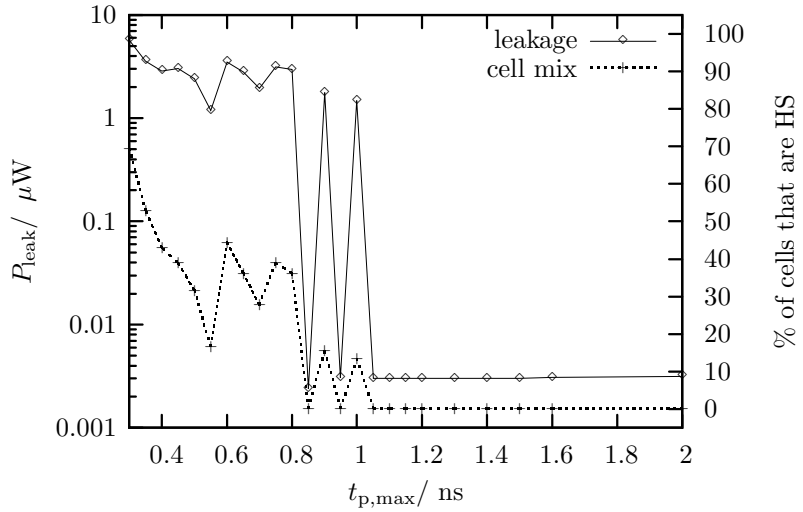


Figure 5.5: Leakage and cell mix of a 4 to 16 one hot encoder vs. timing constraint.

The synthesizer alternates between using LL and HS cells according to its internal cost function, which considers a number of factors. The lesson to be learned from this is that tools can be unpredictable.

Summary

All in all, leakage power is complex to predict. Mathematical models are only practical to derive analytically for very simple structures like ripple adders. Even for these, the solution that the synthesis tool arrives at depends on the details of the synthesis algorithm. Other factors that influence the outcome are

- the difference between the propagation delay of a HS cell and a LL cell

- the difference between the leakage power of a HS cell and a LL cell
- the leakage power, dynamic power, timing, and area constraints given to the synthesis tool
- the structure of the circuit

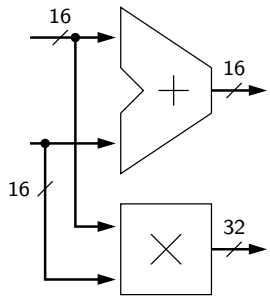
The observations made in this section are useful to the designer because they can give a feel for, how timing constraints affect the leakage power. This is useful for navigating in the solution space. It has been demonstrated, that area is an inappropriate predictor of leakage power in the presence of tight timing constraints. Even the models presented here are very inaccurate, and care must be taken when using them to estimate leakage power consumption. A proposal for practical power estimation at the architectural level is made in the following section.

5.3 Proposed model for leakage estimation

Failing the attempt to create a mathematical model leaves only the empirical approach. Using a library of precharacterized components for power estimation is an idea that has already been employed successfully for dynamic power estimation at the architectural level. It is the basis of the *spreadsheet model*.

This model, explained for instance in [15, p. 132-133], is a practical approach for early architectural level design-space explorations. Both static and dynamic power estimates are considered in this model. The estimation is based on models obtained by characterization of library components (as in [31, ch. 3]). The model is simple enough to be implemented in an ordinary spreadsheet. However, a specialized tool could provide better usability and better ease of reuse of the data. Unfortunately, since the original tool, PowerPlay (presented in [32]) is no longer available, no public tools seem to exist.

The spreadsheet model can best be explained by example. Figure 5.6 shows such an example. The two modules have been precharacterized for power and the data was entered into the table. Additionally the designer adds the number of components, the supply voltage and an idle time. The model is entered directly into the cells of the spreadsheet, so the resulting power is computed directly at each change of the numbers. This allows the designer to play around with them easily and see the results of possible circuits.



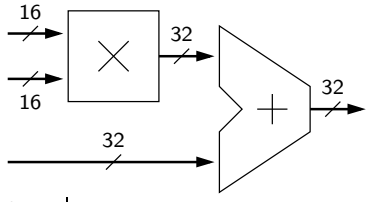
(a) Circuit

	#comp	V_{dd}	$P_{leak}/\,\mu$A	$P_{dyn}/\ \mu$A	%idle	$P_{tot}/\,\mu$A
add	1	1.0	5	15	40	11.0
mult	1	1.0	20	45	40	35.0

(b) Model

Figure 5.6: The basic spreadsheet model – an example.

With the knowledge presented in the previous section, this model needs to be extended.



multiplier	$t_{\mathrm{p,min}}$= 3.3 ns	$t_{\mathrm{p,e}}$= 15.2 ns
adder	$t_{\mathrm{p,e}}$= 0.76 ns	$t_{\mathrm{p,e}}$= 5.9 ns

Figure 5.7: Two components in series.

$t_{\mathrm{p,max}}$ should be one of the inputs and incorporated in the leakage estimation formula. One way to do this would be to fit the characterization data to models like equations 5.1 and 5.2. Given, however the macro capabilities of spreadsheet applications, the characterization data can be directly imported into the spreadsheet. An appropriate macro can then directly look up the leakage power, dynamic power and area values. It is even possible to automatically interpolate between the values in the table to obtain estimates in between the characterization points.

This has been done in the OpenOffice Calc spreadsheet included in appendix E. Appendix C presents a tool created for this project for characterizing building blocks.

The model will be extended to consider an operating voltage different from the characterization voltage in the next chapter, which is about voltage scaling.

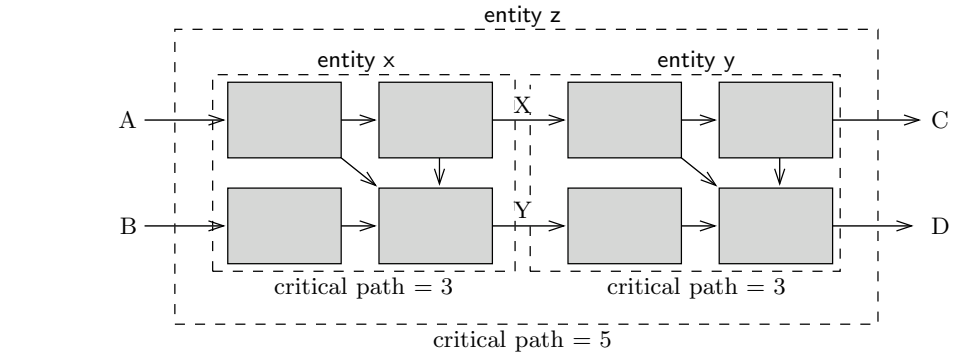
The spreadsheet model has a number of limitations, so care must be taken when using it.

First of all, it is input-invariant, that is: It predicts the same power consumption no matter which inputs are applied to the circuit. As explained in appendix C, characterization is done assuming completely random inputs. Often, this is not the case in actual applications. Real data inputs is often correlated in time or to other inputs. Disregarding this can lead to overestimation of dynamic power, downplaying the significance of leakage power.

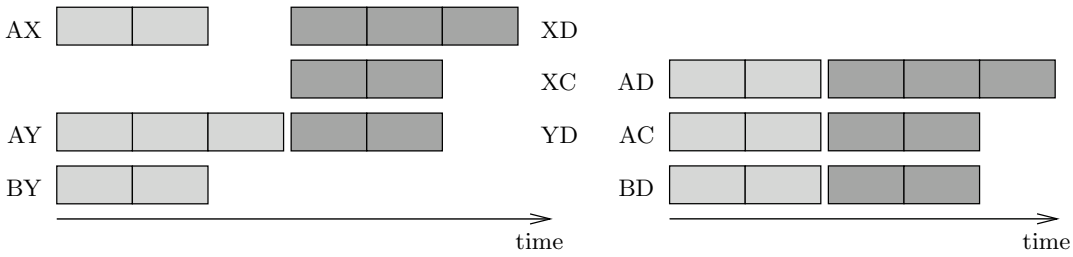
But most importantly, the estimation of power for library components in series contains serious pitfalls. Figure 5.7 shows such an example. From the figures for the individual components one could expect the combination of the two to have in the neighborhood of $t_{\mathrm{p,min}} = 3.3\,\mathrm{ns} + 0.76\,\mathrm{ns} = 4.06\,\mathrm{ns}$ and $t_{\mathrm{p,e}} = 15.2\,\mathrm{ns} + 5.9\,\mathrm{ns} = 21.1\,\mathrm{ns}$. However, this is not the case. For the series connection, the values are $t_{\mathrm{p,min}} = 3.8\,\mathrm{ns}$ and $t_{\mathrm{p,e}} = 16.2\,\mathrm{ns}$. What happens is, that the critical paths don't add up. The length of the critical path of two components in series can be smaller than the sum of the individual critical path lengths.

Figure 5.8 demonstrates this. The structure of entities `x` and `y` is similar to the structure of a multiplier. The critical paths don't add up because they are not connected to each other. This is a common situation for arithmetic units. As demonstrated in figure 5.9, the critical path of a unit generally goes from the least significant bit of the input to the most significant bit of the output. Because the most significant output bit of a unit is connected to the most significant output bit of the following unit, the two critical paths are not connected.

In the spread sheet model this means, that it is not possible to predict the length of the critical path of a series connection unless it is known which two paths combined paths will be the longest ones. Even if this was known, the leakage vs. $t_{\mathrm{p,max}}$ characteristic between



(a) Sum of critical paths



(b) Incorrect critical path estimation.

(c) Correct critical path estimation.

Figure 5.8: Estimation of critical path lengths.

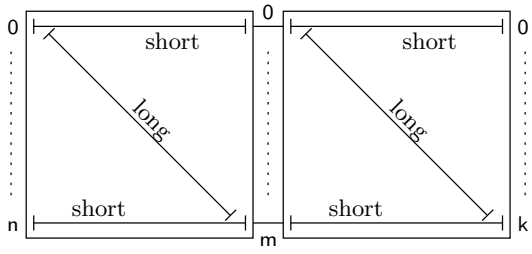


Figure 5.9: Two arithmetic units in series.

$t_{\mathrm{p,min}}$ and $t_{\mathrm{p,e}}$ might be totally different from the sum of the two individual characteristics. In summary, using the spread sheet model on a series connection of two individual components is not possible if both are on the critical path.

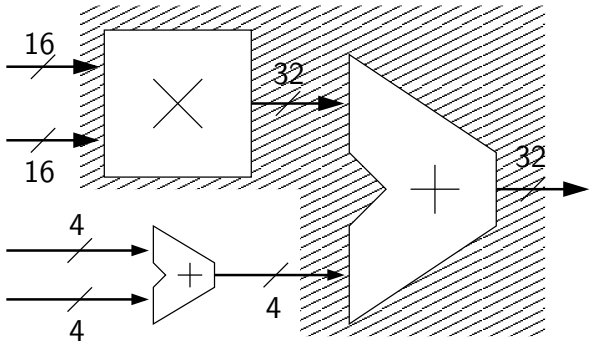


Figure 5.10: Series connection with one unit off the critical path.

In a setup like the one shown in figure 5.10 where only the multiplier and the big adder are on the critical path, these two units need to be characterized together as one library component as indicated by the shaded area. The small adder is not on the critical path and can be characterized separately. It is not clear, however, which $t_{\mathrm{p,max}}$ the small adder will see, but in the setup shown it is likely that it will be large enough so that the leakage of the adder is $P_{\mathrm{leak},\infty}$.

5.4 Estimating the example

For the application of the estimation model on the multiply-accumulate unit design from the previous chapter, appropriate blocks have to be chosen for characterization. The components along the critical path have to be characterized as one. This includes the components marked by shading in figure 5.11 on page 44.

The remaining components are characterized separately. No controller had been designed at this point, but it is estimated, that it would need about six states, three inputs and six outputs. From a previous project a state machine with five states, three inputs, and three outputs was available. It has been used instead. The zero detector is just an eight-input NOR gate. It is neglected in estimation.

Table 5.1 shows the figures for the design. The timer circuit and the data path have $t_{\mathrm{p,max}}$ set to the full clock period of 8 ns, while the three components of the timer see 2 ns each. Since $2\,\mathrm{ns} > t_{\mathrm{p,e}}$ for all of these three, the exact value of $t_{\mathrm{p,max}}$ does not matter.

The only component that has a $t_{\mathrm{p,max}}$ less than its $t_{\mathrm{p,e}}$ is the data path. Therefore it is responsible for almost all of the leakage of the circuit. Since it is also the component with the most switching activity, it also consumes most of the switching power.

Unfortunately a breakdown of the power consumption in this component is not available, although it would have been possible to extract these figures during characterization.

#comp	component	$t_{p,min}$/ ns	$t_{p,e}$/ ns	$t_{p,max}$/ ns	P_{leak}/ μW	P_{dyn}/ μW	% on	avg. P_{dyn}/ μW	P_{tot}/ μW
1	mult_mux_add	2.90	11.0	8.0	10.763	189.13	2%	3.78	14.55
1	add_rpl_8	0.85	1.6	6.0	0.0024	26.75	2%	0.54	0.54
1	mux_8	0.055	0.18	1.0	0.00129	4.00	2%	0.08	0.08
1	reg_8	0.06	0.07	1.0	0.00278	7.97	2%	0.16	0.16
1	controller	0.07	1.4	8.0	0.00258	6.08	2%	0.12	0.12
5					10.77	233.94		4.68	15.45

Table 5.1: Leakage estimation for the multiply-accumulate design.

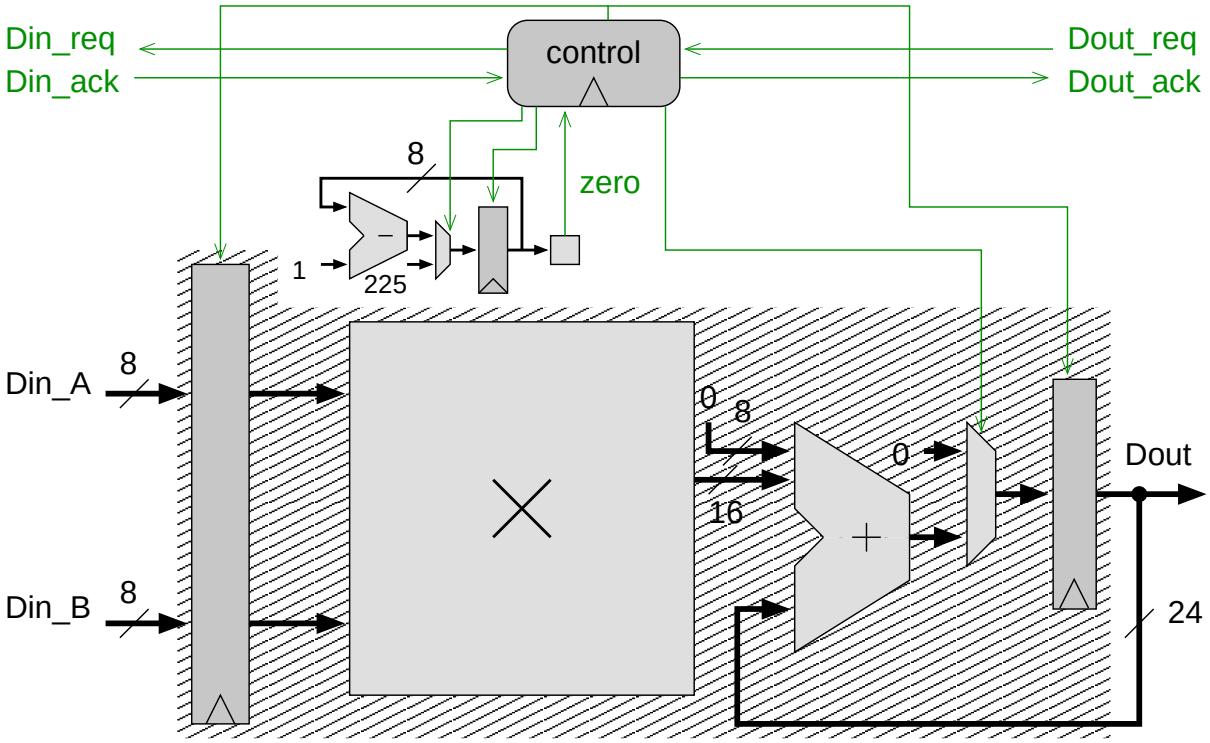


Figure 5.11: Partitioning of the design for characterization.

Chapter 6

Architectural voltage scaling

The motivation for architectural voltage scaling[1] comes from the fact that when V_{dd} is reduced, the circuit delay τ rises approximately linearly, while P_{leak} falls exponentially and P_{dyn} falls quadratically as already explained in chapter 2.

Therefore, if a circuit can be speeded up by a factor N and the supply voltage afterward be reduced by the same factor of N, the circuit will still have the same delay. However, due to the reduced supply voltage, dynamic power will have been reduced by a factor N^2 and P_{leak} will have been reduced even more, even though the area might have been increased.

It may seem that architectural voltage scaling still works, even in the presence of leakage power. This chapter attempts to answer the question, whether this is in fact so.

6.1 Does architectural voltage scaling still work?

Of course, the power supply cannot be made arbitrarily low. At some point, memory elements will start to loose their data. But even before that point, the delay of the circuit will become too high. The following first order relation from [26, sec. 2.2.1] expresses the influence of V_{dd} and V_{th} on circuit delay.

$$\tau \propto \frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^2} \tag{6.1}$$

This was graphed in figure 6.1 on the following page for constant V_{th}. As long as V_{dd} does not get too close to V_{th}, the delay increases linearly with falling V_{dd}. As a design rule, a minimum supply voltage of $V_{\mathrm{dd}} = 4|V_{\mathrm{th}}|$ is generally used when speed matters, [33]. In the low supply voltages used in the sub-100 nm processes, this leaves less freedom for voltage scaling than it used to.

As an example, in the cell library used in this project, the supply voltage normally used is 1.0 V. The HS cells have $|V_{\mathrm{th}}|$ around 0.15 V and the LL cells have 0.33 V. Thus for the HS cells $V_{\mathrm{dd}} = 6.8V_{\mathrm{th}}$ while for the LL cells, $V_{\mathrm{dd}} = 3V_{\mathrm{th}}$, which explains their lower speed. In order to stay in the linear area for the HS cells, the supply voltage can be reduced to no more than $4 \cdot 0.15\,\mathrm{V} = 0.6\,\mathrm{V}$. This voltage is only $2V_{\mathrm{th}}$ for the LL cells, so these cells would be several times slower.

[1] sometimes also called architecture driven voltage scaling.

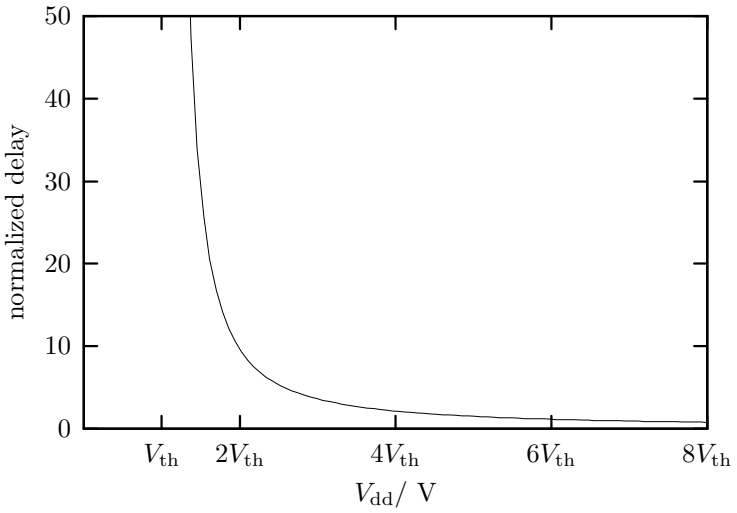


Figure 6.1: Delay vs. supply voltage.

These observations lead to a number of possible scenarios as shown in figure 6.2 on the next page. In each of these scenarios a design is first speeded up by means of hardware duplication. Pipelining or other ways of speeding it up would also be an option, but for the figures duplication is assumed. Next, the voltage is reduced. The point at which the design is found is indicated by the dot. Notice, that the leakage reduction in the last step is not drawn to scale, what looks like a zero leakage in the left column could still me more than what would look like a non-zero leakage in the right column.

The speeding up step increases the time the component has available to complete it's work ($t_{\mathrm{p,max}}$), so the dot moves to the right. The downscaling of the voltage increases the delay of the HS cells somewhat and increases the delay of the LL cells even more, so $t_{\mathrm{p,min}}$ and $t_{\mathrm{p,e}}$ also move to the right. At the same time, the amount that each cell leaks is reduced.

Scenario A is a case where the initial design consists of a mix of HS and LL cells and so does the final design.

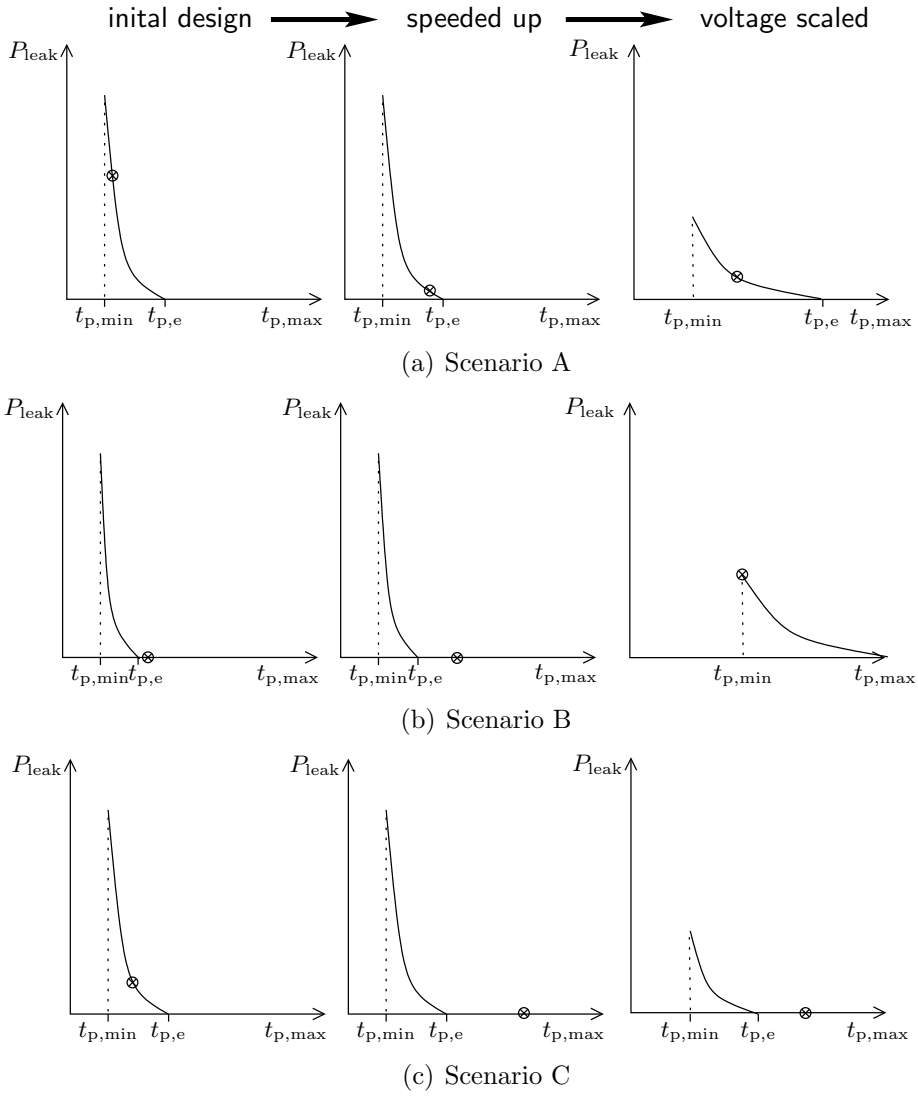
Scenario B has no HS cells in the initial design, but the delay of the circuit increases so much during downscaling that the final design has.

Scenario C is the opposite of scenario B. The initial design contains HS cells, but the speed-up is so large, that the following downscaling does not move the design back into the leaking area.

Of course, there could also be a scenario D, where the design both before and after the transformation contains no LL cells. This is a less interesting case. For the remaining scenarios, the following two questions should be answered:

1. Is the end result a reduction of the leakage power?
2. If it is, how large is this reduction?

These answers will be investigated in the following two sections. The first question in the form of an analysis of a worst case situation and the second question in the form of an extension to the power estimation method presented in chapter 5. Finally, architectural voltage scaling will be performed on the example design from chapter 4 in two different ways.



(a) Scenario A

(b) Scenario B

(c) Scenario C

Figure 6.2: Possible scenarios for voltage scaling. The leakage reduction in the right column has been downplayed.

6.2 Architectural voltage scaling in multiple threshold CMOS

In order to determine whether architectural voltage scaling will result in a leakage power reduction, scenario B, which can be considered the worst case situation, will be examined.

The leakage power in the speeded up version of scenario B (middle graph) is

$$P_{\text{leak},\infty} = \#\text{cells}_0 \, \overline{P_{\text{leak,LL}}}$$

where $\overline{P_{\text{leak,LL}}}$ is the average leakage of an LL cell. Correspondingly, $\overline{P_{\text{leak,HS}}}$ is the average leakage of an HS cell. $\#\text{cells}_0$ is the number of cells before the voltage reduction.

The leakage power in the voltage reduced version is

$$P_{\text{leak},1} = K_{\text{VL}}((1-\omega)\overline{P_{\text{leak,LL}}} + \omega\overline{P_{\text{leak,HS}}}) \, \#\text{cells}_1$$

where ω is the cell mix quotient, i.e. the number of HS cells divided by the number of cells in the design. $\#\text{cells}_1$ is the number after the voltage reduction. K_{VL} is defined in equation 2.7 on page 19.

In summary, a voltage reduction is seen if the following relation holds true.

$$\#\text{cells}_0 \, \overline{P_{\text{leak,LL}}} > K_{\text{VL}}((1-\omega)\overline{P_{\text{leak,LL}}} + \omega\overline{P_{\text{leak,HS}}}) \, \#\text{cells}_1$$

Generally, the area increase through the tightened delay constraint is only a few times as figure 5.2(e) on page 36 shows an example of. So $\#\text{cells}_1$ and $\#\text{cells}_0$ can be removed from the relation without affecting the argument much.

Moving things around a little gives the following result.

$$\frac{\overline{P_{\text{leak,LL}}}}{((1-\omega)\overline{P_{\text{leak,LL}}} + \omega\overline{P_{\text{leak,HS}}})} > K_{\text{VL}} \tag{6.2}$$

The variables of the left side of relation 6.2 can be considered known from characterization data and cell library data. ω is not constant with voltage, because LL become slower faster than HS cells do. This will decrease the left side of the equation.

K_{VL} depends on the amount of voltage reduction done, so the final voltage, $V_{\text{dd},1}$, has to be determined.

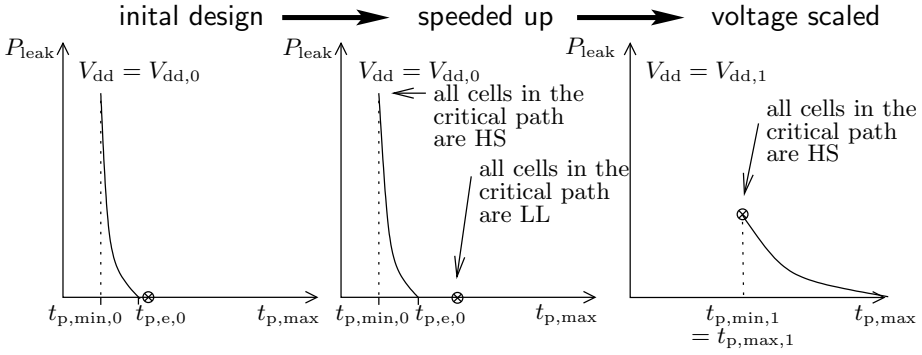


Figure 6.3: A closer view at scenario B.

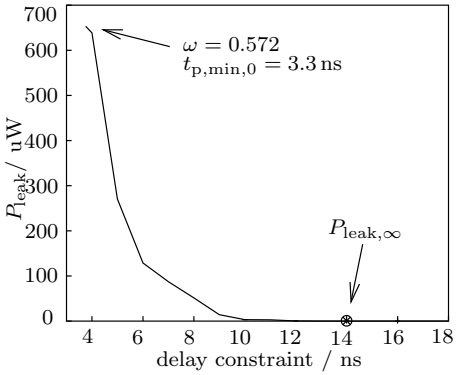


Figure 6.4: Voltage scaling a 16 by 16 multiplier – before the voltage reduction.

Before the voltage reduction in scenario B, all cells of the critical path are LL cells (see figure 6.3). After the voltage reduction (right graph), all cells of the critical path are HS cells. The shift of the leakage characteristic happens because the cells are getting slower. Specifically, $t_{\text{p,min}}$ is increased at the rate that the delay of HS cells increases when the voltage is reduced. This rate will be called $\Delta_{\text{d,HS}}$. The following expresses the relationship between $t_{\text{p,min,0}}$ and $t_{\text{p,min,1}}$ which equals $t_{\text{p,max,1}}$ in the final design in figure 6.3.

$$t_{\text{p,max,1}} = t_{\text{p,min,0}} \cdot \Delta_{\text{d,HS}} \tag{6.3}$$

The value of $\Delta_{\text{d,HS}}$ can also be expressed by the following expression derived from equation 6.1 as follows.

$$\Delta_{\text{d,HS}} = \frac{\tau_1}{\tau_0} = \frac{\frac{V_{\text{dd,1}}}{(V_{\text{dd,1}}-V_{\text{th,HS}})^2}}{\frac{V_{\text{dd,0}}}{(V_{\text{dd,0}}-V_{\text{th,HS}})^2}} = \frac{V_{\text{dd,1}}}{(V_{\text{dd,1}}-V_{\text{th,HS}})^2} \cdot \frac{(V_{\text{dd,0}}-V_{\text{th,HS}})^2}{V_{\text{dd,0}}} \tag{6.4}$$

Equation 6.4 can be solved for $V_{\text{dd,1}}$ in order to determine how much the voltage can be reduced to slow the critical path down by a factor $\Delta_{\text{d,HS}}$. The solution is the following expression.

$$V_{\text{dd,1}} = \frac{2\Delta_{\text{d,rel}}V_{\text{th,HS}} + 1 + \sqrt{4\Delta_{\text{d,rel}}V_{\text{th,HS}} + 1}}{2\Delta_{\text{d,rel}}} \tag{6.5}$$

where

$$\Delta_{\text{d,rel}} = \frac{\Delta_{\text{d,HS}}V_{\text{dd,0}}}{(V_{\text{dd,0}} - V_{\text{th,HS}})^2} \tag{6.6}$$

Now that $V_{\text{dd,1}}$ is known, it can be inserted in equation 2.7 on page 19 in order to calculate K_{VL} and through equation 6.2 determine if the voltage scaling procedure has resulted in a power reduction.

Unfortunately the complexity of the situation does not lend itself to an analytical analysis, but the following example shows that architectural voltage scaling does in fact not always result in a power saving.

Consider the 16 by 16 bit multiplier discussed in chapter 5. Assuming that the speed up of the circuit allows the component to complete its work in 14 ns (figure 6.4), the voltage is to be reduced until this timing requirement is just met. The initial voltage $V_{\text{dd,0}}$ is 1.0 V.

From the characterization data and cell library data the following parameters are known[2]:

$$\begin{aligned} P_{\text{leak},\infty} &= 0.1021\,\mu\text{W} \\ |V_{\text{th,HS}}| &= 0.15\,\text{V} \\ \overline{P_{\text{leak,LL}}} &= 282.5\,\text{pW} \\ \overline{P_{\text{leak,HS}}} &= 631,460.1\,\text{pW} \\ t_{\text{p,min},0} &= 3.3\,\text{ns} \\ \omega &= 0.572 \\ \eta &= 0.335 \\ m &= 1.350 \\ v_T &= 25.7\,\text{mV} \end{aligned}$$

Using the expressions in the above discussion, the resulting leakage power consumption can now be calculated.

From 6.3:

$$\Delta_{\text{d,HS}} = \frac{14\,\text{ns}}{3.3\,\text{ns}} = 4.2424$$

From 6.6:

$$\Delta_{\text{d,rel}} = \frac{4.2424 \cdot 1.0\,\text{V}}{(1.0\,\text{V} - V_{\text{th,HS}})^2} = 5.8719\,\text{V}^{-1}$$

Inserting this in equation 6.5 yields

$$V_{\text{dd},1} = 0.416\,\text{V}$$

With equation 2.7 on page 19 the leakage reduction factor K_{VL} can be determined to be

$$K_{\text{VL}} = 1.48 \cdot 10^{-3}$$

which is the right hand side of equation 6.2. The left hand side can be computed to

$$\frac{\overline{P_{\text{leak,LL}}}}{((1-\omega)\overline{P_{\text{leak,LL}}} + \omega\overline{P_{\text{leak,HS}}})} = 0.78186 \cdot 10^{-3}$$

Obviously, this is smaller than K_{VL}, so the reduction of the voltage did in fact increase the leakage power. It has, however, reduced the dynamic power.

This example shows, that the designer can no longer trust that reducing the voltage will always decrease the power consumption, because the synthesis tool will have to cope with the changed timing and may thereby destroy the improvement.

In other cases reducing the voltage also reduces the power consumption. Obviously, scenario C from figure 6.2 and the not displayed scenario D, where no HS cells are involved at any time during the design process, will benefit from voltage scaling. In scenario B, the voltage can be lowered at least to the point where the synthesizer starts to use HS cells. Scenario A is harder to predict.

[2] at 25 °C

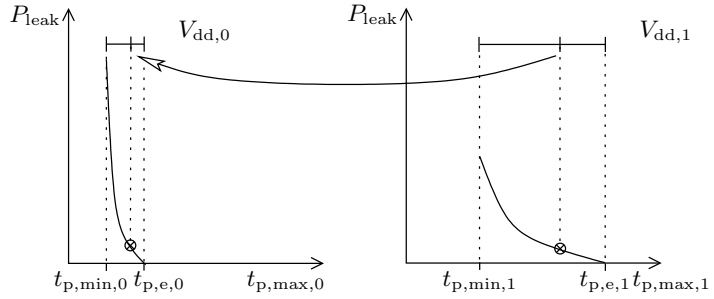


Figure 6.5: Using characterization data for $V_{\text{dd,0}}$ to look up a power estimate for $V_{\text{dd,1}}$.

Saying anything in general about how far the voltage can be reduced is not possible because it depends on a host of parameters, including complex ones such as the synthesis algorithm and the circuit structure.

Whether or not to reduce the voltage should be a decision made individually for each case after careful estimation of the effects. Since it is typically the complete chip and not just a single component that has its voltage reduced, reducing the supply voltage may reduce the leakage power consumption of one part of the circuit while it increases the leakage in another.

6.3 Estimation of power consumption with voltage scaling

The designer will need a convenient way to evaluate his decisions to determine the consequences of voltage scaling. Therefore the estimation technique presented in chapter 5 needs to be extended for this purpose.

This can be done in two ways. One way is to recharacterize the components for a number of voltages, but this can become very tedious. The other way is to re-use the data characterization data by means of the following method.

The procedure is essentially the same as the one outlined in the previous section. It needs, however, to be extended to estimate the leakage at any point of the characterization graph.

In estimation, the final voltage is decided by the designer as an input to the estimation tool, so $\Delta_{\text{d,HS}}$ (and $\Delta_{\text{d,LL}}$) can simply be calculated by equation 6.4.

As mentioned above, the critical path at $t_{\text{p,max}} = t_{\text{p,min}}$ consists of all HS cells. At $t_{\text{p,max}} = t_{\text{p,e}}$ it consists of all LL cells. Therefore the following dependencies between $t_{\text{p,e}}$ and $t_{\text{p,min}}$ at $V_{\text{dd,0}}$ and at $V_{\text{dd,1}}$ can be assumed.

$$t_{\text{p,min,1}} = \Delta_{\text{d,HS}}\, t_{\text{p,min,0}} \tag{6.7}$$

$$t_{\text{p,e,1}} = \Delta_{\text{d,LL}}\, t_{\text{p,e,0}} \tag{6.8}$$

This can be used to calculate the new position of the characteristic after the right shift during supply voltage reduction.

In order to look up the leakage value at $V_{\text{dd,1}}$, $t_{\text{p,max,1}}$ is mapped onto the leakage characteristic for $V_{\text{dd,0}}$ as shown in figure 6.5. By calculating where $t_{\text{p,max,1}}$ is placed with

respect to $t_{\mathrm{p,min,1}}$ and $t_{\mathrm{p,e,1}}$, it is possible to find the corresponding position with respect to $t_{\mathrm{p,min,0}}$ and $t_{\mathrm{p,e,0}}$.

$$\mathrm{pos} = \frac{t_{\mathrm{p,max,1}} - t_{\mathrm{p,min,1}}}{t_{\mathrm{p,e,1}} - t_{\mathrm{p,min,1}}}$$

pos is a fraction that specifies at which fraction of the distance between $t_{\mathrm{p,min,1}}$ and $t_{\mathrm{p,e,1}}$, $t_{\mathrm{p,max,1}}$ is. In the figure this is about 60%. Next, the same point is found in the characterized data:

$$t_{\mathrm{p,max,0}} = \mathrm{pos} \cdot (t_{\mathrm{p,e,0}} - t_{\mathrm{p,min,0}}) + t_{\mathrm{p,min,0}}$$

Finally, the $t_{\mathrm{p,max,0}}$ is used to look up the leakage in the characterized data and this leakage is multiplied by K_{VL} to obtain the estimate for the voltage scaled circuit.

This estimation method is very simple and can easily be implemented within the spreadsheet model. On the other hand, it assumes, that the shape of the characteristic does not change because of the voltage reduction, which may not be true.

In the following section two practical examples of architectural voltage scaling are evaluated using this method.

6.4 Application to the example

In the following two sections, architectural voltage scaling will be applied to the example design in two different ways. First, the data-path will be duplicated, so the design can run at a lower clock frequency. In the second example, the higher speed will instead be achieved through pipelining.

6.4.1 Duplication

As mentioned before, the critical path in the design in figure 4.4 on page 30 is the multiply-accumulate data-path. In the design shown in figure 6.6 on the following page, this data-path has been duplicated, so that each of the two data-paths will multiply and accumulate half of the incoming data pairs. A final adder combines the results of the two units. This extra stage and the alternated operation of the two data-paths adds two clock cycles to the latency but does not affect the throughput. According to the specification on page 29, this is acceptable.

The timing estimation for this architecture is shown in table 6.1 on page 55 both after the doubling and after the subsequent voltage reduction to be compared with figure 5.1 on page 43. The added complexity needed for the control unit has not been modeled. Area was increased 107%.

The estimation table now has two extra columns. $t_{\mathrm{p,max,0}}$ contains the value that is used to look up data in the characterization table. $t_{\mathrm{p,rel}}$ tells how many clock periods the multiply-accumulate unit can spend for its work. The duplicated data-path units each have two clock periods available, because they are clocked at half frequency. This is used for the calculation of the dynamic power consumption.

The doubling of the data-path has effectively removed the leakage. It was reduced from $10.77\,\mu$W to $0.144\,\mu$W. Dynamic power has increased slightly from $4.68\,\mu$W to $6.74\,\mu$W due to the extra adder and register. The interesting question is if reducing the voltage will improve the leakage power consumption further.

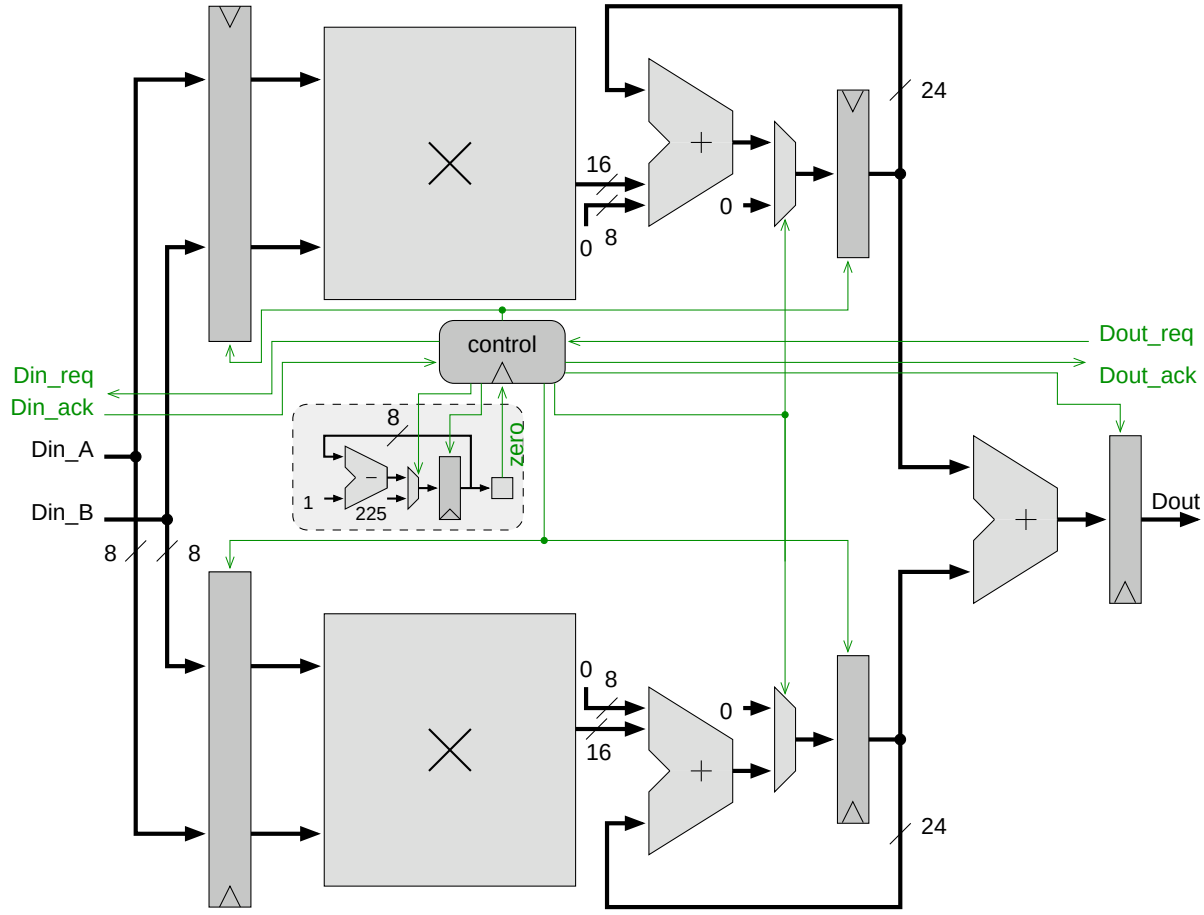


Figure 6.6: The multiply-accumulate unit with duplicated data-path.

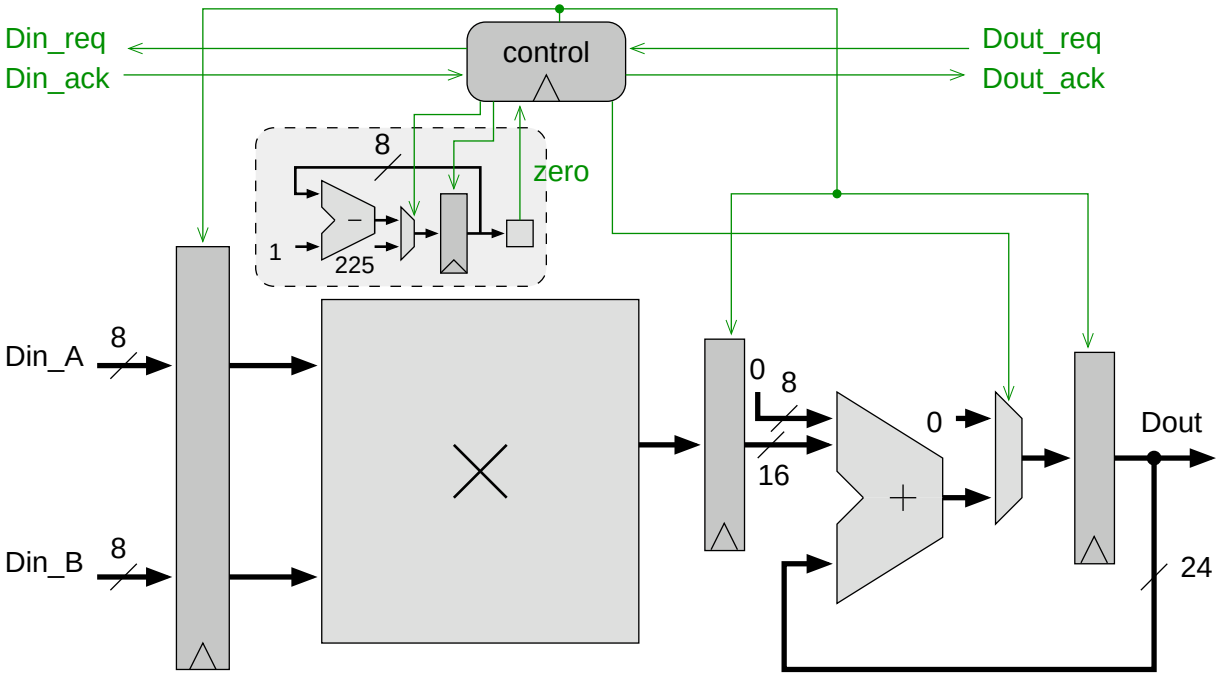


Figure 6.7: The multiply-accumulate unit with a pipelined data-path.

By experimentation, the minimal leakage power is found at $V_{\mathrm{dd}} = 0.904\,\mathrm{V}$. This is the point where the data-path has $t_{\mathrm{p,max}}$=$t_{\mathrm{p,min}}$. Any further reduction of the supply voltage past this point increases the leakage power consumption. The dynamic power has decreased a little to $5.51\,\mu\mathrm{W}$. Thus, leakage consumes only 1.0% of the total power.

Minimal total power is found at $V_{\mathrm{dd}} = 0.508\,\mathrm{V}$, where the design just meets it's timing requirement (data not listed, see 6.3 on page 58 for a summary). Here, leakage has increased to $2\,\mu\mathrm{W}$, but since the dynamic power has continued to fall to only $1.74\,\mu\mathrm{W}$, the total power consumption is now $3.75\,\mu\mathrm{W}$. Leakage now contributes 53.6% of the total power.

The estimation model has been implemented for this design in the spreadsheet to be found in appendix E. The reader is invited to explore it himself.

6.4.2 Pipelining

Figure 6.7 shows an alternative design. Through the insertion of an extra pipeline register in the data-path, the multiplier and the add-mux stage both have the full clock period of 8 ns to do their computation[3]. Surprisingly, area is decreased 8%, but this is an estimate and may not be entirely accurate.

Table 6.2 lists the estimation results for this design. Before the voltage reduction, the leakage was already reduced from the original design $10.77\,\mu\mathrm{W}$ to $0.068\,\mu\mathrm{W}$ because no HS cells are needed in the data-path any more. This is a little less than half the leakage of the parallelized design. Dynamic power has risen from $4.68\,\mu\mathrm{W}$ to $6.54\,\mu\mathrm{W}$, a little less than the parallelized design.

A minimal leakage of $0.051\,\mu\mathrm{W}$ was found at $V_{\mathrm{dd}} = 0.973\,\mathrm{V}$, where it consumes only

[3]The multiplier and the adder each have the full clock period minus the time needed for the registers and the mux to be LL.

#	comp.	$\frac{t_{\mathrm{p,min}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,e,0}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,max,1}}}{\mathrm{ns}}$	$t_{\mathrm{p,rel}}$	$\frac{t_{\mathrm{p,max,0}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{leak}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{dyn}}}{\mu\mathrm{W}}$	%on	$\frac{\overline{P_{\mathrm{dyn}}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{tot}}}{\mu\mathrm{W}}$	$\frac{P_{\mathrm{leak}}}{P_{\mathrm{tot}}}$
2	mult_add	2.800	13.00	16.0	200%	16.0	0.120	182.13	2%	3.64	3.76	82.9%
1	add_rpl_8	0.850	1.60	6.0	100%	6.0	0.00244	26.75	2%	0.54	0.54	1.7%
1	mux_8	0.055	0.18	1.0	100%	1.0	0.00129	4.00	2%	0.08	0.08	0.9%
1	reg_8	0.060	0.07	1.0	100%	1.0	0.00278	7.97	2%	0.16	0.16	1.9%
1	controller	0.070	1.40	8.0	100%	8.0	0.00258	6.08	2%	0.12	0.12	1.8%
1	add_rpl_24	2.600	4.50	7.5	100%	7.5	0.00732	86.41	2%	1.73	1.74	5.1%
3	reg_8	0.060	0.07	0.5	100%	0.5	0.00834	23.90	2%	0.48	0.49	5.8%
10							0.145	337.25		6.74	6.89	2.1%

(a) Without voltage reduction.

#	comp.	$\frac{t_{\mathrm{p,min}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,e,0}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,max,1}}}{\mathrm{ns}}$	$t_{\mathrm{p,rel}}$	$\frac{t_{\mathrm{p,max,0}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{leak}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{dyn}}}{\mu\mathrm{W}}$	%on	$\frac{\overline{P_{\mathrm{dyn}}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{tot}}}{\mu\mathrm{W}}$	$\frac{P_{\mathrm{leak}}}{P_{\mathrm{tot}}}$
2	mult_add	2.80	13.00	16.0	200%	12.99	0.0449	148.84	2%	2.98	3.02	83.5%
1	add_rpl_8	0.85	1.60	6.0	100%	4.64	0.000873	21.86	2%	0.44	0.44	1.6%
1	mux_8	0.055	0.18	1.0	100%	0.79	0.000461	3.27	2%	0.07	0.07	0.9%
1	reg_8	0.06	0.07	1.0	100%	0.60	0.000996	6.51	2%	0.13	0.13	1.9%
1	controller	0.07	1.40	8.0	100%	6.48	0.000923	4.97	2%	0.10	0.10	1.7%
1	add_rpl_24	2.60	4.50	7.5	100%	5.96	0.002619	70.61	2%	1.41	1.41	4.9%
3	reg_8	0.06	0.07	0.5	100%	0.31	0.002987	19.54	2%	0.39	0.39	5.6%
10							0.0538	275.60		5.51	5.57	1.0%

(b) With voltage reduced to 0.904 V

Table 6.1: Leakage estimation for the data-path duplicated design.

0.8% of total power. As before, the leakage power begins to rise past this point, while total power falls due to the reductions in dynamic power.

Minimal power is at the minimal voltage of $V_{\mathrm{dd}} = 0.507\,\mathrm{V}$ (not listed, see the summary in table 6.3.). Here, leakage consumes $1.5\,\mu\mathrm{W}$, which is 42.7% of the total power consumption which is now $3.18\,\mu\mathrm{W}$. This is almost a fifth of the original power consumption.

Again, the reader is encouraged to explore the design by means of the spreadsheet in appendix E.

Obviously, the placement of the pipelining register is sub-optimal. Since the adder and the multiplier have different delays, a pipeline register that was at midpoint in the datapath would balance the delays better and allow the synthesizer to use less HS cells. This pipeline register would have to be placed inside the multiplier.

In fact, retiming, that is, moving registers around in the design can be a good tool to balance computation between registers and thereby reduce the number of HS cells used. It can often be employed in situations where voltage scaling, pipelining or other parallelizations are not an option.

6.5 Conclusions for voltage scaling

Contrary to expectations, voltage scaling turns out not always to reduce leakage. Because the supply voltage is already very close to V_{th} for LL cells, the delay of these cells rises much faster than for HS cells, resulting in the synthesizer being forced to use more leaky HS cells.

Table 6.3 shows a summary of the power dissipation of the different versions of the design. It demonstrates clearly, that usage of more hardware can in fact reduce leakage power quite considerably.

In the examples explored, reductions of the voltage were only beneficial for the power consumption of a component as long as the timing constraint of the component stayed above $t_{\mathrm{p,e}}$. This was the optimal point. However, the power consumption is a result of many parameters including technology parameters, circuit and synthesis tool behavior. This may be different in a different setup.

The estimation method developed here was also implemented in the spreadsheet model attached in the digital appendices, appendix E. With this tool it is easy to explore design space, allowing the designer to quickly gain experience with the consequences of various modifications to the design for both leakage power and dynamic power.

#	comp.	$\frac{t_{\mathrm{p,min}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,e,0}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,max,1}}}{\mathrm{ns}}$	$t_{\mathrm{p,rel}}$	$\frac{t_{\mathrm{p,max,0}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{leak}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{dyn}}}{\mu\mathrm{W}}$	%on	$\frac{\overline{P_{\mathrm{dyn}}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{tot}}}{\mu\mathrm{W}}$	$\frac{P_{\mathrm{leak}}}{P_{\mathrm{tot}}}$
1	mult_8	1.80	7.00	7.93	100%	7.93	0.0277	258.50	2%	5.17	5.20	26.7%
7	reg_8	0.06	0.07	0.07	100%	0.07	0.0469	118.49	2%	2.37	2.42	45.2%
1	add_cla_24	0.80	7.00	7.75	100%	7.75	0.0159	75.61	2%	1.51	1.53	15.4%
1	mux_24	0.06	0.18	0.18	100%	0.18	0.0041	10.38	2%	0.21	0.21	3.9%
1	add_rpl_8	0.85	1.60	6.00	100%	6.00	0.0024	26.75	2%	0.54	0.54	2.4%
1	mux_8	0.055	0.18	1.00	100%	1.00	0.0013	4.00	2%	0.08	0.08	1.2%
1	reg_8	0.06	0.07	1.00	100%	1.00	0.0028	7.97	2%	0.16	0.16	2.7%
1	controller	0.07	1.40	8.00	100%	8.00	0.0026	6.08	2%	0.12	0.12	2.5%
14							0.104	507.78		10.16	10.26	1.0%

(a) Without voltage reduction.

#	comp.	$\frac{t_{\mathrm{p,min}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,e,0}}}{\mathrm{ns}}$	$\frac{t_{\mathrm{p,max,1}}}{\mathrm{ns}}$	$t_{\mathrm{p,rel}}$	$\frac{t_{\mathrm{p,max,0}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{leak}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{dyn}}}{\mu\mathrm{W}}$	%on	$\frac{\overline{P_{\mathrm{dyn}}}}{\mathrm{ns}}$	$\frac{P_{\mathrm{tot}}}{\mu\mathrm{W}}$	$\frac{P_{\mathrm{leak}}}{P_{\mathrm{tot}}}$
1	mult_8	1.800	7.000	7.40	200%	7.00	0.0207	121.88	2%	2.44	2.46	40.9%
7	reg_8	0.060	0.070	0.30	100%	0.26	0.0146	52.81	2%	1.06	1.07	28.8%
1	add_rpl_24	2.600	4.500	7.50	100%	7.23	0.00549	81.80	2%	1.64	1.64	10.8%
1	mux_24	0.060	0.180	0.20	100%	0.19	0.00306	10.25	2%	0.20	0.21	6.0%
1	add_rpl_8	0.850	1.600	6.00	100%	5.60	0.00183	25.33	2%	0.51	0.51	3.6%
1	mux_8	0.055	0.180	1.00	100%	0.94	0.000965	3.79	2%	0.08	0.08	1.9%
1	reg_8	0.060	0.070	1.00	100%	0.86	0.00209	7.54	2%	0.15	0.15	4.1%
1	controller	0.070	1.400	8.00	100%	7.57	0.00193	5.76	2%	0.12	0.12	3.8%
14							0.0507	309.15		6.18	6.23	0.8%

(b) With voltage reduced to 0.973 V

Table 6.2: Leakage estimation for the pipelined design.

	V_{dd}	P_{leak}	P_{dyn}	P_{tot}	% leak
initial	1.0 V	10.77 μW	4.68 μW	15.45 μW	69.7%
duplicated	1.0 V	0.144 μW	6.74 μW	6.89 μW	2.1%
	0.904 V	0.054 μW	5.51 μW	5.57 μW	1.0%
	0.508 V	2.0 μW	1.74 μW	3.75 μW	53.6%
pipelined	1.0 V	0.068 μW	6.54 μW	6.60 μW	1.0%
	0.973 V	0.051 μW	6.18 μW	6.23 μW	0.8%
	0.507 V	1.50 μW	1.68 μW	3.18 μW	47.2%

Table 6.3: Results for voltage scaling.

Chapter 7

Wire leakage minimization

Since the beginning of VLSI design, transistor sizes have continued to become smaller. This decrease in size has traditionally not been used to make smaller chips, but to put more devices onto the die. In fact, according to [24], the die size itself has been steadily increasing.

For interconnect, the good news is, that the delay of local wires more or less tracks the delay of gates, [34]. For wires that cross the chip the prospects are worse. If repeaters are inserted, the delay of these global wires remains about constant over technology generations. This means, that the disparity between the delay of global wires and the delay of gates has been growing and continues to grow. In fact, the number of gates that can be fitted onto a die is now so large that these gates can no longer all communicate within a single clock cycle. This problem has become one of the main bottlenecks to performance.

The repeaters inserted into wires in order to keep the delay low are typically very wide. These repeaters are very leaky, which is likely to become a problem in the future.

In this section, some of the design space available to the architect with respect to long wires will be explored.

The discussion is kept at an architectural level, so most of the lower level issues such as crosstalk and noise margins are abstracted away.

Only simple communication infrastructures, namely point-to-point links and shared buses, are considered. Comparing more elaborate structures such as as on-chip networks is not done here. However, such structures typically also include long wires (although often not quite as long) and may benefit from this discussion.

7.1 The cost of a wire

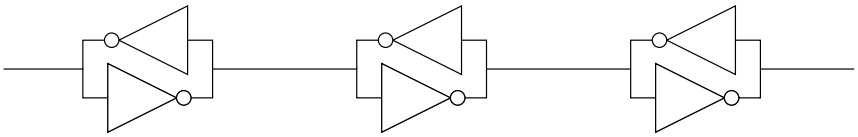
Historically, wires were free. In the ideal model, a wire is just an equipotential area with zero delay. This model is still more or less usable for local wires, but for global wires it is not. Global wires have considerable capacitance, resistance and inductance which delay the signal propagation through them.

Both the capacitance C_w and the resistance R_w of a wire are proportional to wire length. The delay of a wire is roughly

$$\tau_w = R_w C_w$$

thus the delay of an uninterrupted wire scales quadratically with wire length. The solution to this problem is to insert repeaters in the wire at strategic positions. Repeaters are typically inverters as shown in figure 7.1, but they can also be buffers, which makes the insertion of them logically easier. The delay of each stage of the wire is the same, so by using repeaters the delay of the wire becomes proportional to the wire length.

(a) Unidirectional wire.



(b) Bidirectional wire.

Figure 7.1: Wires with repeaters.

A number of authors present algorithms for optimal insertion, dimensioning and placing of repeaters according to different optimality criteria such as delay, bandwidth, area or power, among these [34, 35, 36]. This issue is best solved by tools and will not be discussed any further here.

To help evaluate the options discussed here, a simple repeater insertion tool was created. It is described in appendix D.

The number of wires that need buffering is increasing exponentially over technology generations, [37]. According to [35], the number of buffers per chip is about $25,000$ at 130 nm and will reach $797,000$ at 70 nm. According to [38], buffers and inverters currently already contribute up 50% of total transistor width. These repeaters typically need a high drive strength to meet timing requirements[1] ([35] talks about strengths of 2x to 100x). Since repeaters consist of inverters, that have no stacking effect and since the delay requirements typically cause them to be HS, they consume a large amount of leakage power.

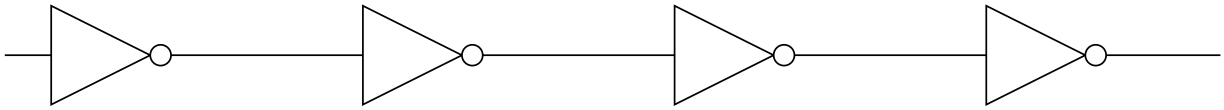
In summary, wires are no longer free. Before leakage power became a problem, long wires only cost area and design effort. A wire that was not used did not cost any power. This situation has now changed. Every wire with repeaters causes a constant power consumption. In the following section, the architect's possibilities to reduce this contribution will be discussed.

7.2 Design space for wire leakage

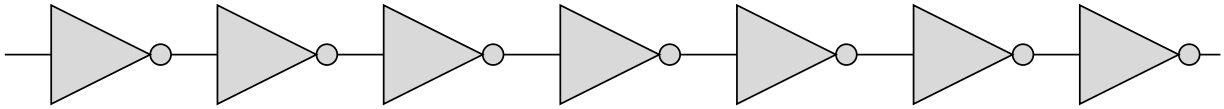
The discussion in this section will evolve around the list of leakage reduction opportunities first given on page 21, namely

- increasing V_{th}

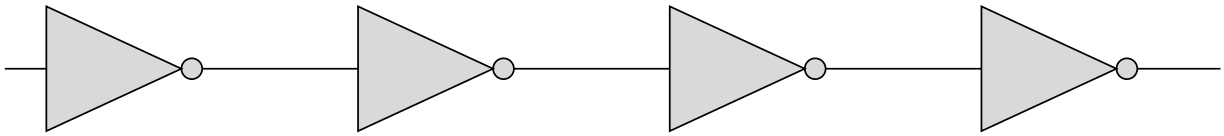
[1] Drive strength refers to the width of the transistors in an inverter compared to the width of a unit size inverter. So the transistors in a x2 inverter are twice as wide as the transistors in a minimum size inverter. However, large drive strengths are typically implemented by using several smaller transistors in parallel.



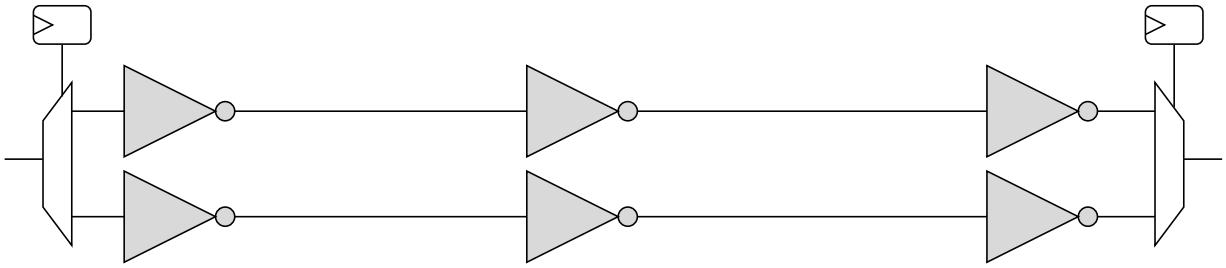
(a) Wire with HS repeaters.



(b) Wire with LL repeaters and shortened distance between repeaters.



(c) Wire with LL repeaters with drive strength matched to the HS repeaters.



(d) Two time multiplexed wires with LL repeaters.

Figure 7.2: Using LL repeaters instead of HS repeaters.

- reducing the total width of devices that leak
- increasing transistor stacking
- reducing operating temperature
- reducing V_{dd}
- applying less leaky inputs to gates

Reducing the temperature does not provide any good opportunities for the circuit designer, and since repeaters have to be inverters, neither does increasing the transistor stacking. The remaining ones will be discussed in the order they are listed. The last two will be treated together.

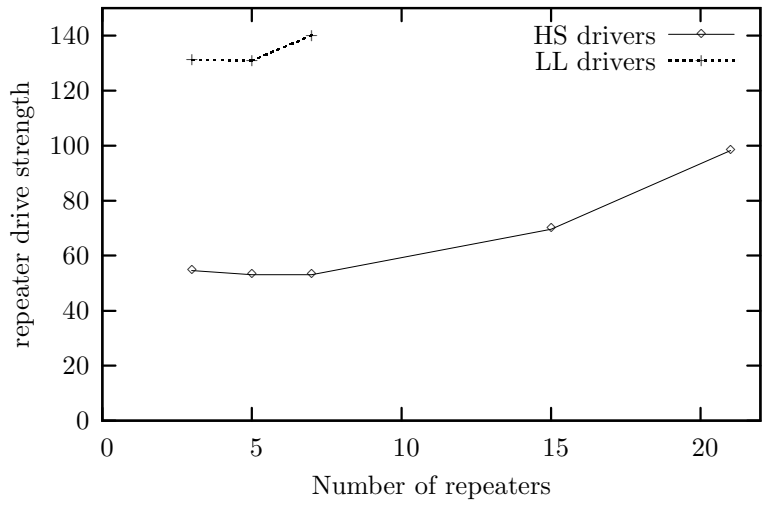
7.2.1 Increasing the threshold voltage

As seen in voltage scaling, using LL cells instead of HS cells, can reduce the leakage power of a circuit very much, bringing it close to zero. But since LL inverters are somewhat slower than HS cells, they may not be an option for global wire drivers since these typically have very strict timing requirements.

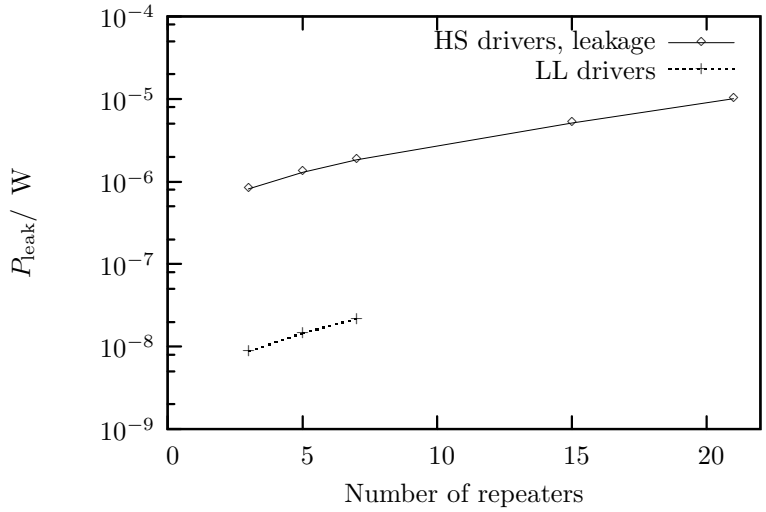
In order to compensate for this deficiency in speed, a number of options exist as summarized in figure 7.2. Options (b) and (c) will be treated together and option (d) will be treated separately.

Repeater count versus repeater drive strength

The first two options (b) and (c) are simply to use more repeaters or to make the repeaters stronger in order to match the speed of HS-repeaters. Since choosing the number and size of repeaters is a tool task and the result depends on which optimality criteria the tool uses, this option will not be discussed deeply here.



(a) Drive strength



(b) Leakage power

Figure 7.3: Drive strength (a) and leakage power consumption (b) of a long wire vs. number of repeaters. Total delay: 0.5 ns.

To investigate this, the long 16 bit time multiplexed bus from the example design was examined. Each of the wires in the bus must have a propagation delay of no more than 0.5 ns. A repeater insertion tool (appendix D) was given the task to meet this requirement for a certain number of repeaters N. Figure 7.3(a) shows the drive strengths required by of each of the N repeaters to meet the 0.5 ns total delay requirement.

The graph for the HS driver cells show, that for a low number of segments, the wire delay

	no. of repeaters	total rep. width	P_{leak}	P_{dyn}	P_{tot}
HS	3	164	12.98 μW	471.9 μW	484.91 μW
LL	3	393	0.140 μW	502.2 μW	502.35 μW
doubled LL	3	228	0.143 μW	454.1 μW	454.21 μW
tripled LL	3	209	0.077 μW	401.8 μW	401.87 μW

Table 7.1: Minimum leakage and dynamic power for various bus architectures. The dynamic power is with white noise input and activity 2% of the time.

dominates, so stronger repeaters are needed to drive it. For a high number of segments, the repeater delay dominates, so the faster repeaters are needed. Thus there is a minimum drive strength.

The leakage graph in figure 7.3(b) shows that the minimum drive strength is not the point with the minimum leakage. The fewer the number of repeaters, the lower the leakage. This happens because the drive strength of the repeaters is less than doubled when the number of repeater stages is. As a conclusion, option 7.2(c) is more leakage efficient than option 7.2(b).

Obviously, the total leakage of the wire driven by LL repeaters is much less than the HS version. However, the LL drivers have to be very strong in order to meet the timing requirements. In fact, the LL graph contains only a few points because this was not practically possible to create inverters that were fast enough for less than two or more than seven repeater stages.

The LL repeaters are in general more than twice as wide as the HS repeaters.

Both the HS and LL version of the bus have minimum power at the minimum number of segments. This is true for both dynamic and static power. The figures are summarized in table 7.1.

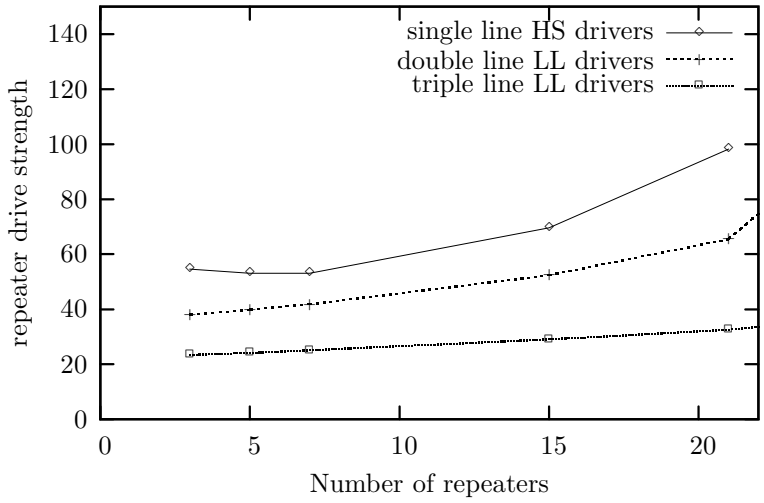
Clearly, dynamic power dominates the static power consumption, but this may be different in a different situation. In this case, the leakage power is reduced to slightly more than 1% of the original value. Dynamic power is, however, increased due to the capacitances of the larger repeaters, so the total result is an increase in power consumption. With this method, the decrease of static power comes at a cost of an increase in dynamic power.

Duplicating wires

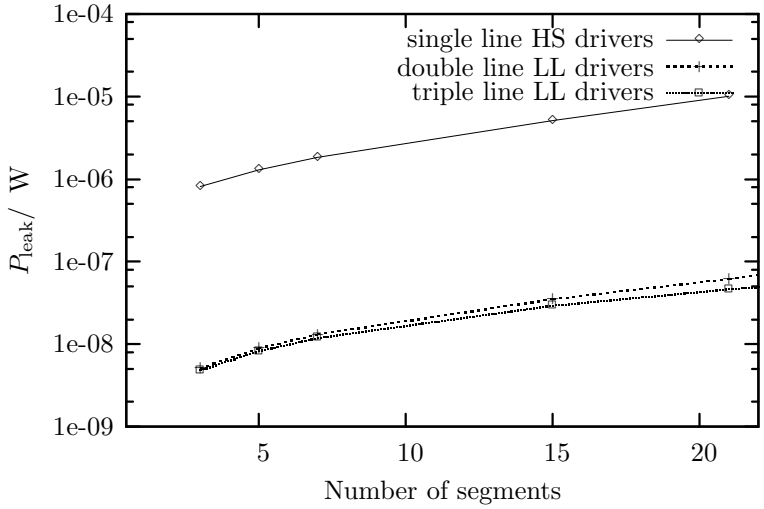
The last option in figure 7.2 shows a rather radical approach to minimizing the leakage, duplicating the link to reduce its timing requirement. Given the large difference between HS and LL repeater leakage, this is likely to still result in a leakage power reduction. However, the area penalty may be quite large.

Furthermore, this method requires some logic to control the time multiplexing, which will either require that both ends are synchronized by a global clock as shown or that the link is self-timed.

This situation was simulated and the result is in figure 7.4. The doubling of the delay requirement has meant a considerable reduction in LL repeater strength. Taking this approach one step further and tripling the bus reduces the drive strength and hereby the leakage further.



(a) Drive strength



(b) Leakage power

Figure 7.4: Drive strength and leakage power for the single HS line (delay: 0.5 ns) compared to the duplicated LL line (delay: 1.0 ns) and tripled LL line (delay: 1.5 ns) topology.

Table 7.1 on page 63 shows the results of these two experiments. This time dynamic power is also reduced, because the driver capacitances are decreased. Consequently, both possibilities decrease the total power consumption.

This solution does decrease the total active area used by repeaters, but it also increases the total metal usage. This limits the total on-chip bandwidth.

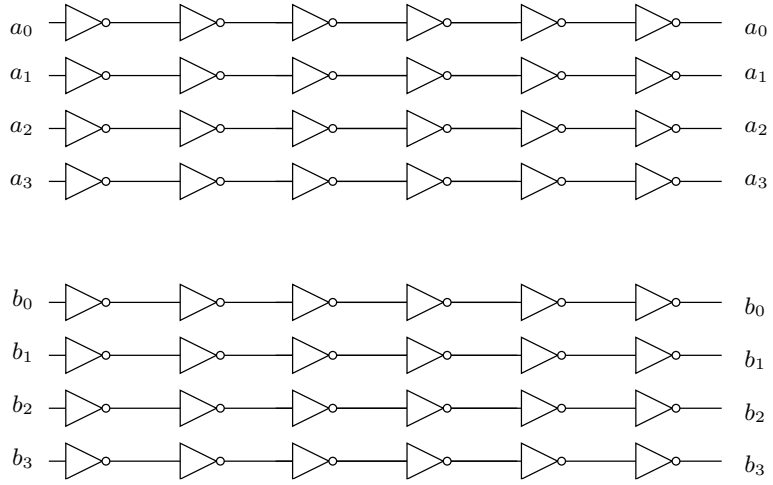
7.2.2 Reducing the total width of leaking devices

Reducing the number of repeaters means to reduce the number of wires between two points. If two communication links are never used at the same time they are good candidates for time multiplexing. This may also be possible if there is enough bandwidth on one link to support the transfer and a little extra latency does not matter.

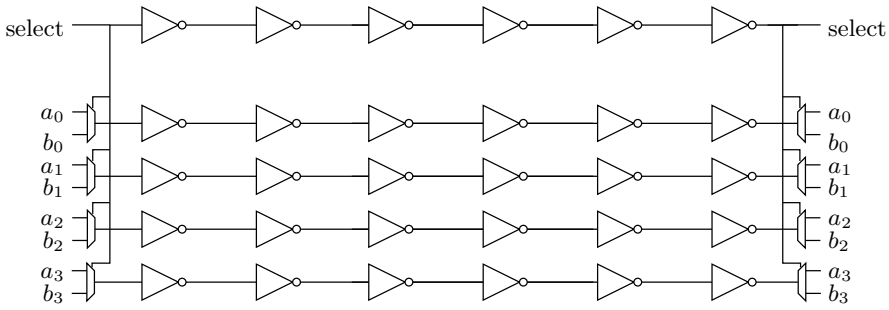
Figure 7.5 illustrates how two buses from the same location to the same destination can be replaced by a single time multiplexed line. If M buses of N bit each are replaced by one bus of N bit, then the leakage from drivers is reduced by a factor of

$$\frac{N + \lceil \log_2 M \rceil}{N \cdot M}$$

since $\lceil \log_2 M \rceil$ is the number of bits needed to communicate to the other end, which data the bus is carrying. These savings can become quite substantial for wide buses or many buses.



(a) Two 4-bit links.



(b) Time multiplexed 4-bit link.

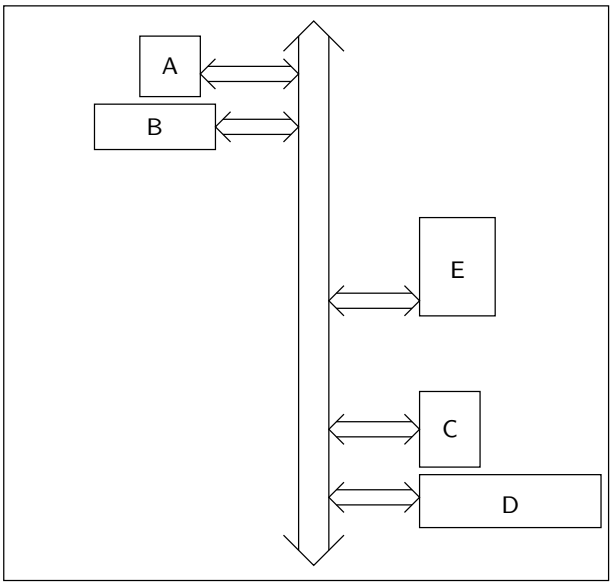
Figure 7.5: Time multiplexing a link.

In the example used here, no other bus is available which could be used for this purpose, but if there had been, say, two of them, their overall reduction would have been 53%.

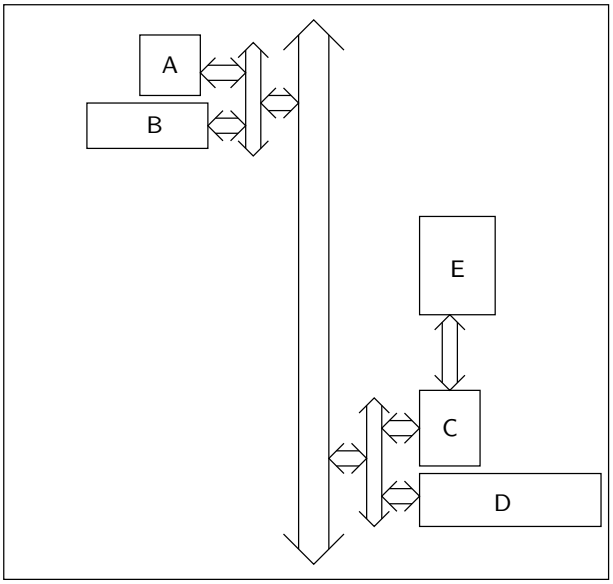
Note, that this is different from the duplication strategy used above. Here, two buses with a surplus of communication capacity are combined to save hardware. Above, a bus

that was heavily stressed in terms of timing was doubled to lighten the stress.

This sharing strategy is just a special case of the traditional shared on-chip bus principle, where multiple components share a single communication channel. Figure 7.6(a) shows such an example.



(a) Shared bus.



(b) Shared bus and local buses.

Figure 7.6: Exploiting locality for driver minimization.

Even in this scheme, reducing the amount of driver hardware might be possible. Since the units A and B as well as C and D are physically close, they can share the access to the bus. By adding another bus hierarchy, only one large driver onto the global bus is needed for these two units. If the two units never communicate, the local bus can be as simple as a number of OR-gates and a little arbitration hardware.

If two units that are nearby only communicate with each other, they need not use the bus. As in the case of unit C and E, a specialized point to point link can be less leaky if the distance is short enough.

The savings achieved by sharing bus access or by replacing bus access by local infrastructure will typically be much smaller than the ones achieved by sharing the bus in the first place, as above. Since the bulk driving of the bus is done by repeaters which are existent in the bus anyway, replacing two bus accesses by one saves just that: a single driver. This saving may or may not be worth the overhead. However, exploiting locality in communication does also reduce dynamic power consumption.

7.2.3 Reducing the supply voltage

One of the classic ways to reduce the dynamic power consumption of a bus is to use a low swing bus. The repeaters of such a bus would also use the low voltage and thus consume less dynamic and static power. This is, however, a circuit level problem and will not be discussed further here. A good overview of these techniques can be found in [39] where a number of such schemes for dynamic energy, delay and signal to noise ratio are evaluated.

Architecturally more interesting is the idea to entirely turn off the supply voltage in periods of inactivity on the wire. This could almost entirely eliminate leakage power during these periods, but on the other hand, it would also reduce the speed. According to [1], a 99.95% reduction in idle current comes at a cost of a 75% to 88% increased speed. Furthermore, this technique requires a certain wake-up period. This problem has all the classical issues of prediction of optimal idle and wake-up times. As already mentioned earlier, Benini et al. provide a survey of techniques involved in [15, sec. 5.1]. Calhoun et al. present circuitry to implement supply voltage gating in [12].

In the present example design, the bus is used only 2% of the time, so ideally, 98% of the leakage energy can be saved, if the bus is switched off the rest of the time.

On the other hand, this technique is only an option if delay constraints are not very strict. In this case, a single, small LL driver driving the whole line may be just as leakage efficient at a much lower design effort. The technique does, however, offer reductions for dynamic power.

7.2.4 Input vector control

As an alternative to turning off the supply voltage, the fact that the leakage power consumption of CMOS gates is input dependent can be exploited for reducing the standby leakage power. For instance, the 55x HS inverter used in the original minimum leakage solution dissipates 1.47 times as much static power when applied a 0 as input as it does when applied a 1.

This can not be exploited for reducing the static power consumption of wires repeated by means of inverters, because a 1 at the beginning of the line would turn in to a 0 at every other stage, causing every other inverter to enter its leaky state.

It does, however, make sense in wires repeated by buffers as shown in figure 7.7. Since one of the inverters is smaller than the other, it will leak much less. By placing the large inverters in its less leaky state, static energy can be reduced. This is easily accomplished as it can be done from the driving end of the wire.

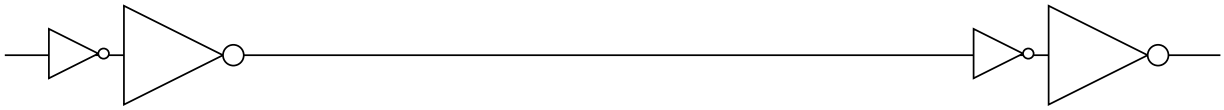


Figure 7.7: Wire with buffers as repeaters.

If the leakage of the smaller inverter is negligible compared to the bigger inverter then 67% of the leakage can be reduced by this procedure. But normally, this will not be the case and the saving would be smaller. In any case, using buffers instead of inverters may be a good idea.

A similar approach is presented by Deogun et al. in [38]. The authors present a method based on asymmetric inverters consisting of one HS and one LL transistor. This creates inverters that have a particularly non-leaky state. Inverters with LL pull-up and pull down are interleaved on the wire, giving each wire an LL state.

Deogun et al. use statistical information of the data transfered over the bus to create a bus encoding technique that results in increased probability that each wire is in a less leaky state. Hereby, run-time power is reduced. The encoding is a simple look-up table based encoding. Obviously, this only works if the distribution of the data to be transferred is known at design time.

7.3 Conclusions for on-chip communication

This chapter has explored part of the design space for long wires on chip. A number of techniques offer opportunities for leakage reduction.

The first technique is to replace HS repeaters by LL repeaters. This requires a compensation for the lower speed of the LL repeaters, which can be achieved either by using repeaters with a higher drive strength or by duplicating communication links.

Duplicating communication links offers advantages as it reduces both static and dynamic power as well as repeater area, but it is only possible if throughput and not latency is the limiting factor.

Another technique for reducing leakage associated with wires is to share them. If the bandwidth of two buses is higher than necessary, then the communication infrastructure can be shared. This is a well known infrastructure for connecting modules on and off chip. However, the reasons for using shared buses have been others than leakage. A bus that is not used is not free power-wise anymore.

Input vector control offers opportunities for some reductions in standby power consumption. With buffers as repeaters or specially designed inverters, this solution is simple to apply.

Finally, while low swing buses don't offer any distinct advantages over the other techniques for leakage power reduction, the total power consumption may benefit due to the simultaneous dynamic power reduction.

Chapter 8

Discussion and conclusions

8.1 Discussion

This thesis attempts to answer part of the question asked in the introduction: How should the leakage problem be handled by the chip architect? In order to do so, two main subjects were approached: Estimation and design.

From a number of promising candidate leakage reduction subjects, two design subjects were chosen for closer examination: architectural voltage scaling and the reduction of the leakage associated with wires. Furthermore, high level power estimation was examined.

Architectural voltage scaling is one of the traditionally very effective ways of reducing power consumption. Since leakage power scales even faster with supply voltage than dynamic power consumption does, this was expected to work well. As shown in chapter 6, this is not necessarily the case.

First of all, speeding up the circuit alone can bring considerable leakage reduction, since it reduces the amount of HS cells needed. The subsequent re-tightening of the delay constraints may in fact increase the leakage power consumption if it means that more HS cells are being used. This happens because the effect of the cell mix on leakage dominates the effect of the supply voltage.

A second effect is due to the low supply voltage used in leaky processes. This means that the speed of LL cells degrades faster with supply voltage than the speed of the HS cells. This was not modeled in the experiments done here, but it can be expected to reduce the usefulness of voltage reductions further.

In summary, since the cell mix dominates the result of voltage reductions, voltage scaling may not work if it has a negative effect on the cell mix. In total, however, the simultaneous reduction of dynamic power consumption may dominate.

Wire leakage was treated in chapter 7. The findings were similar to the ones for voltage scaling. If bandwidth is abundantly available then two units can share a communication link and thereby save power. On the other hand, if the delay requirements are strict then duplicating a link may yield power savings.

Low swing buses were found to offer no clear leakage reducing advantages over the other techniques examined. Input vector control offers moderate savings at a small design effort.

Architecture level power estimation was treated in chapter 5. In acknowledging the fact that area is an inappropriate predictor of leakage, given the fact that synthesis tools can mix HS and LL cells, a novel leakage estimation method was proposed. This method

is an extension of the well known spreadsheet model. By using precharacterized building blocks, the estimation method can model the behavior of the synthesis tool. This only requires simple interpolation between a values determined beforehand. A general simulation framework for collecting this data by precharacterizing such building blocks was created and presented in appendix C.

Including the effect of the synthesis tool in estimation is essential, since its effect on leakage power is large. As a side-effect, this also predicts area and dynamic power better than the traditional spreadsheet model does. The spreadsheet model is simple enough to be implemented with macros in a regular spreadsheet, and this was done for the multiply-accumulate example of this thesis.

With the extensions derived in chapter 6, the estimation method also handles voltage scaling. This method is rather approximate as it neglects that LL cells slow down more than HS cells when the voltage is reduced.

The estimation method has some limitations. First of all, its dependence on synthesizer behavior, which is its strength, can also become its weakness when the synthesizer behaves in a non-consistent way. Unfortunately, this is not uncommon. Next, the method requires building blocks to be precharacterized. This may not be practical for a new design unless it consists of standard building blocks or unless a similar design has been done before.

Furthermore, the requirement to have precharacterized data is worsened by the fact that the result of connecting two combinational blocks is not always predictable. This requires groups of building blocks between registers in the critical path to be precharacterized together, which makes building a comprehensive library of precharacterization data much less viable.

Finally, the method suffers from the disadvantage of all input-independent estimation methods. Its accuracy is limited by the fact that it does not consider switching activity and input probabilities.

However, all in all the proposed estimation method is more accurate than both the area based estimators and the original spreadsheet model.

As a side product of this thesis, a 70 nm cell library was created. The creation method, scaling down an existing 180 nm cell library, only makes this cell library a guess at what a real 70 nm cell library will look like. However, the care taken when doing the scaling, makes it a guess that can be expected to be accurate enough for the conclusions based upon it to be trusted.

8.2 Future work

Not all questions have been answered by this thesis. As the problem of leakage power consumption is relatively new, not much work has been done at the architectural level to evaluate the consequences.

In chapter 3, a number of possible research areas were presented. Some of these already see a lot of research, but others still need some work. On the subject of the leakage efficiency of arithmetic units, some work could be done. Current synthesis tools don't consider the delay constraint dependency of leakage when choosing architectures for arithmetic units. From experience it is known that this may lead to sub-optimal decisions. More research in this area could help building better tools.

Retiming is another subject which has not been fully explored for its leakage reduction potential. Retiming, originally a designer task, is now beginning to become a tool task, [40]. Currently such tools only use retiming for satisfying timing and area constraints. More research in this area could help extend them to also minimize leakage.

While the proposed power estimation method provides an acceptable trade-off between ease of use and accuracy, further work could improve it. First of all, a specialized tool could make it much more usable. Integrating logic block characterization with the tool would make the effort of using it much smaller. The tool PowerPlay [32], which unfortunately is no longer available demonstrated how characterization data could be handled in a structured way and made available to several designers at the same time.

More research on the effect of series connecting logic blocks might help reduce the number of logic blocks that have to be characterized in order to estimate the power consumption of a circuit.

Finally, there is hope that synthesis tools will continue to improve their synthesis for low leakage, thereby creating much more predictable results than today.

Only part of the design space for leakage power associated with on chip communication was only explored in this thesis. Other aspects exist, especially in the context of the current trend towards on-chip networks. The leakage overhead associated with the communication infrastructure may change the optimal topology for a given situation, since networks have a relatively large amount of wires per communication than do for instance shared buses.

On the subject of repeater sizing, developing a wire sizing algorithm that minimizes for total power instead of only dynamic power, as is currently the case, still needs to be done.

8.3 Conclusions

Now that a transistor is no longer just a switch, the way low power design should be done has changed. The changes are not radical, but neither can they be ignored. With leakage power now being part of the equation, both power estimation and low power design have become more complicated. There are two main reasons for this.

First of all, reducing total power is no longer equal to reducing dynamic power. Since static power is now non-negligible, handling the power problem means to balance dynamic and static power consumption, since what reduces one of them might at the same time increase the other.

The second reason is, that the now-significant leakage power contribution is more complicated than dynamic power was. Since with Multiple Threshold CMOS, the leakage depends highly on the cell mix, any part of a circuit can be in one of two regions of the design space as depicted in figure 8.1. In the delay un-constrained region, leakage power is low and the area of the circuit is a good predictor of the leakage power. In the delay constrained region, the leakage power consumption is considerably higher and depends strongly on the delay requirements.

Furthermore, in this leakage critical area, the leakage power is highly dependent on the performance of the synthesis tool. The synthesis tool will now more than before have to be part of the simplified model that architects use for the lower levels in the design hierarchy.

With the proposed estimation method it is possible to explore the design space both for static and total power consumption – given an appropriate set of precharacterized building

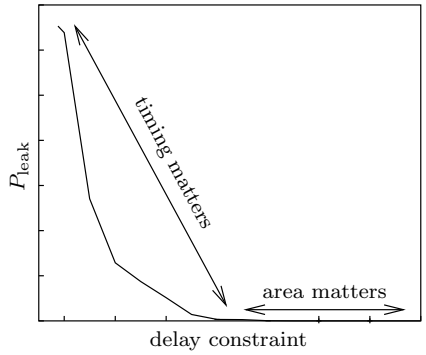


Figure 8.1: The two regions of leakage.

blocks.

With multiple-V_{th} cell libraries, designing for low leakage power to a large extent becomes a matter of navigating the various parts of the circuit out of the leaky area. This may be done by speedups such as parallelization or by moving slack around in the circuit by means of for instance retiming. Other approaches, such as reducing area or voltage only have a secondary effect on leakage power consumption unless the circuit is in the non-leakage critical region. However, dynamic power consumption may counteract the moves in the leakage department. This should not be forgotten.

It is, after all, still *total* power consumption that counts.

Appendix A

Official project description

NR.:	52
Title:	Architectural aspects of design for low static power consumption
Student:	Martin Hans
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-V_{th} and high-V_{th} versions. This offers the option of fast, low-V_{th} cells with high static power losses where timing is critical, and a slower, high-V_{th} design for other parts. Traditional synthesis tools do not offer the means to optimize for multiple-V_{th} libraries to reduce static power consumption. The solution, using such known synthesis tools, consists of synthesizing a design using a low-V_{th} 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-V_{th} by high-

V_{th} library cells whereever possible. The update process does not involve any re-synthesis steps.

This thesis work addresses the consequences of the availability of multiple-V_{th} libraries at architectural level. This includes.a re-evaluation of traditional, architectural level design choices with respect to static power losses.

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 design of libraries offering various speed to power alternatives, while the other work concentrates on the incorporation of static power consumption metrics in the synthesis process.

Supervisor: Flemming Stassen

Appendix B

Creating a cell library

Before the beginning of the work on this thesis, only a 180 nm cell library from ST Microelectronics (STM), [41, 42] was available at this department. Although it is available in both a HS version and a LL version, this is not quite suitable for examining the leakage problem of the future. At 180 nm leakage power still is very low, so illustrating the tradeoffs between leakage power and dynamic power would have required a very large circuit with very low switching activity.

To be able to illustrate the problem in a more realistic setup, a cell library in a smaller technology was needed. In this appendix, the creation of the 70 nm cell library used in the project is described.

At the time of writing, 90 nm was being introduced in foundries around the world. Introduction of 65 nm is scheduled to start in 2005 according to [43].

The UC Berkeley Device group has made predictive SPICE transistor models publicly available at [25] for 180 nm, 100 nm and 70 nm processes. Currently this provides one of the best ways to predict the behavior of future transistors.

A device size of 70 nm was chosen, since it allows results obtained in this project to be valid for at least one more technology generation. Also, the Berkeley 70 nm model does not reach too far into the future, so its predictions can be considered to have an acceptable accuracy.

To arrive at a complete 70 nm cell library, the following paths were considered:

1. creation and characterization of a custom cell library by means of SPICE
2. downscaling of the existing 180 nm cell library to a smaller technology

The first option includes modeling a number of cells in SPICE and simulating these netlists without doing any layout of the cells. Without layout, the effect of local routing inside the cell is neglected, but this can be considered small compared to the routing between the cells.

As shown in [44], the number of cells would not have to be large. According to the article, reducing the number of cells from 400 to 11 only increases the delay of a circuit by 5%. Using 20 cells, the increase is only 2%. However, area and power are increased by 35% and 58% for the 11 cell library and by 5% and 17% for the 20 cell library.

Characterizing a cell library is, however, not at all a simple task. Apart from characterizing the individual cells for delay, capacitances, dynamic and static power, creating a cell library from scratch would also involve synthesizing, layouting and routing a number of typical circuits and collecting statistical information of wire lengths and loads in these

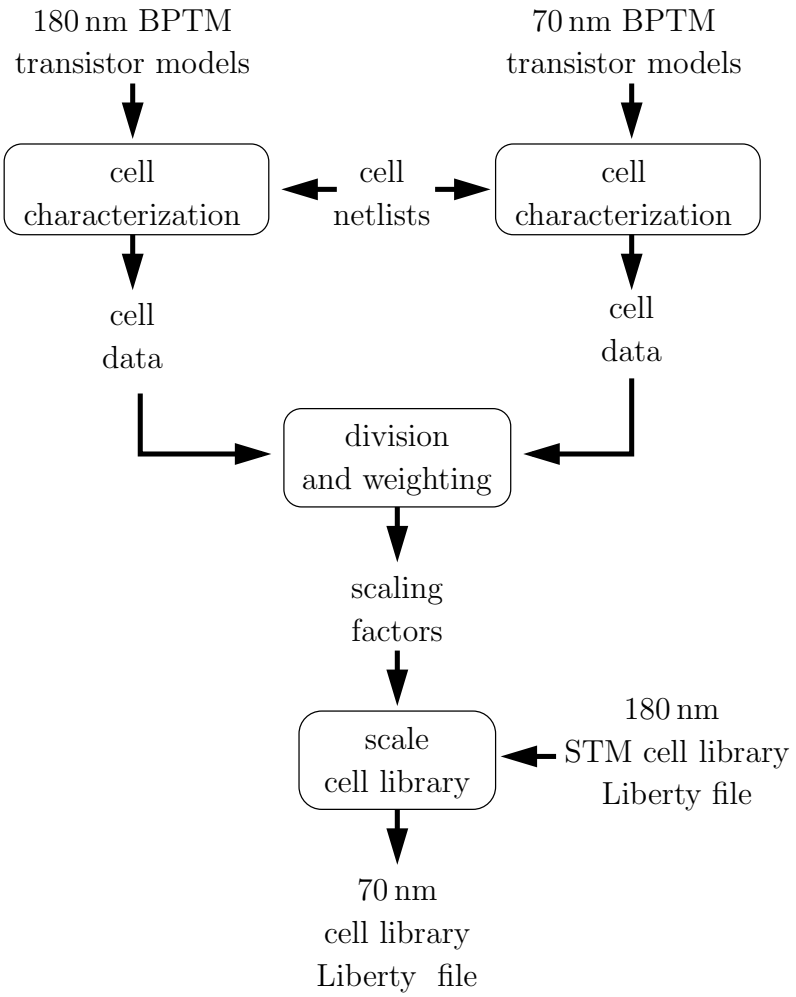


Figure B.1: The scaling process.

circuits.

The second option provides a complete cell library with the same number of cells as the existing 180 nm cell library (777 cells). The approach is to take a Synopsys Liberty file [45] for the 180 nm cell library, which contains characterization data for each cell (such as delay, leakage power, switching power, and wire load figures) and multiply all of these figures with a factor to simulate the downscaling to a smaller technology.

While the first approach would have yielded the most accurate results, it would also have been the most time consuming and circuits using the resulting library would suffer from the power penalty of small cell libraries mentioned above. The second approach gives a full size cell library, but the actual results are less accurate.

For the purpose of this project, accurate predictions of the delay and power figures in future technologies are not necessary. Here, the purpose of the cell library is to allow for illustration of the relative magnitudes of these figures. Therefore the second option was chosen, but the scaling factors were estimated using SPICE simulations on a few cells as in the first approach. The scaling flow used is illustrated in figure B.1.

First, the same cell netlists were simulated with both 180 nm and 70 nm transistors,

then the characterization data obtained hereby was used to find the scaling factors, one for each type of information contained in a Synopsys Liberty file (leakage power, dynamic power, timing, capacitance etc.). Since different cells scaled differently from 180 nm to 70 nm, a simple weighting procedure was used, to find a single best-guess value for each of the factors.

Unfortunately, transistor models were only available as BSIM3.3 and not in a BSIM4 version. BSIM3.3 does not model gate tunneling leakage[1].

These scaling factors were applied to the existing cell library, hereby creating the 70 nm cell library, which could be used by the synthesis tool.

The details of this process will be documented in the following chapters. First the information present in the Liberty file is described along with the method used for scaling it. The following section describes the details of the cell characterizations and the weighting procedure.

The tools created or modified for this work flow are included on the CD-ROM attached as appendix E.

The cell library itself is not attached, as it is a derived work from a STM cell library, which is confidential. However, with the tools included and the three step procedure given in appendix E for producing the 70 nm cell library, the interested reader can easily reproduce it, given the original STM file.

B.1 Scaling cell library contents

This section describes the models for timing, power and wire loading used in the Liberty file as well as the method used to scale it. The models described here are from [46, 47].

One section for each of these parameters describes

1. how the parameter is modeled
2. how it is specified in a Liberty file
3. how it is scaled

For explanation of the contents of the Liberty file, a sample file can be found in section B.3 on page 88. References to line numbers relate to this file. It is a shaved down version, that only contains the parts needed for the discussion here. As such, it is not a complete Liberty file.

Timing, power etc. can be modeled in several different ways in a Liberty file. Here, only the models actually used in the STM file are used.

Scaling will be done by means of a script presented in section B.2.3. It takes the following inputs, each of which will be explained in the following section.

B.1.1 Timing information

Two types of timing information are contained in the Liberty file, propagation delay and transition times.

[1] BSIM3 and BSIM4 are developed by the Device Research Group of the Department of of Electrical Engineering and Computer Science, University of California, Berkeley and copyrighted by the University of California.

`l`	leakage factor
`rd`	rise cell delay factor
`fd`	fall cell delay factor
`rt`	rise transition time factor
`ft`	fall transition time factor
`rp`	rise dynamic power factor
`fp`	fall dynamic power factor
`c`	gate capacitance factor
`a`	area factor
`ru`	resistance/unit length factor
`cu`	resistance/unit length factor
`lib`	set name of new library to this
`opc`	set name of operating conditions to this
`voltage`	set voltage of new library to this
`temp`	set temperature of new library to this

Table B.1: Inputs to the `scale_cellib` script.

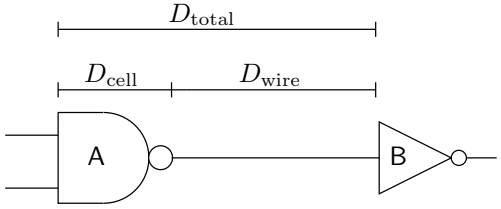


Figure B.2: The delay model used.

Modeling

The delay model used is called the *CMOS nonlinear delay model* [46, pp. 2-19 – 2-30]. It can be expressed as

$$D_{\text{total}} = D_{\text{cell}} + D_{\text{wire}}$$

as shown in figure B.2. The total delay from the input of one cell A to the input of the next cell B, D_{total}, is the sum of the propagation delay through A, D_{cell} and the wire delay from A to B, D_{wire}.

As shown in figure B.3, D_{cell} is the delay between the change of the input and output voltages measured at 50% of V_{dd}. The rising and falling delays are specified separately.

The delay D_{cell} is a function of the output load C_{load} and the input transition time $t_{\text{transition,in}}$:

$$D_{\text{cell}} = f(C_{\text{load}}, t_{\text{transition,in}})$$

The transition time is measured at the output of a cell as shown in figure B.4. It is the time needed for the output to change from 10% of V_{dd} to 90% of V_{dd} for a rising transition and from 90% to 10% for a falling transition.

The transition time at the output of a cell is also a function of C_{load} and $t_{\text{transition,in}}$, i.e.

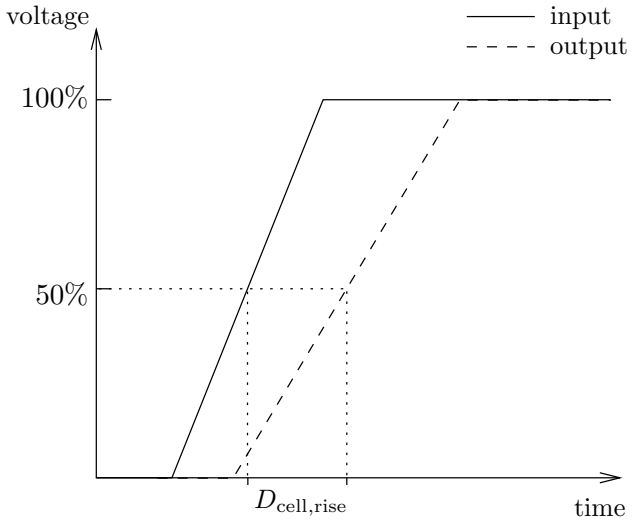


Figure B.3: Cell delay

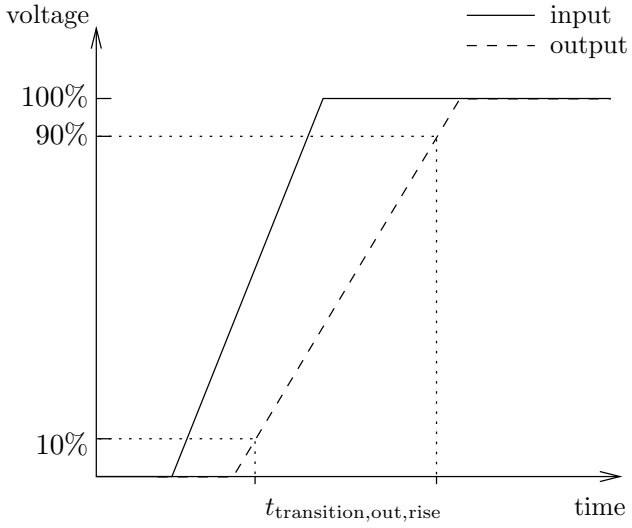


Figure B.4: Transition time

$$t_{\text{transition,out}} = g(C_{\text{load}}, t_{\text{transition,in}})$$

D_{wire} is modeled by a balanced tree model. The load is assumed to be distributed evenly on the N fanout branches as follows

$$D_{\text{wire}} = \frac{R}{N}(\frac{C}{N} + C_{\text{pin}})$$

where N is the fanout and C_{pin} is the input capacitance of the pin driven.

Representation

Timing information is specified in the Liberty file for each cell. For each cell timing information is specified for each of the output pins. Furthermore, for each of the output pins timing is specified separately for a rising and a falling transition resulting from a transition of each of the input pins. Timing resulting from the simultaneous change of several input pins is not specified. Figure B.7 illustrates this.

In the Liberty file, D_{cell} is specified as a lookup table for various values of C_{load} and $t_{\text{transition,in}}$. For example as in lines 112 to 118. This is a table of $D_{\text{cell,rise}}$ for various values of $t_{\text{transition,in}}$ horizontally and C_{load} vertically as defined in the look up table definition in lines 27 to 32. The unit is ns. Figure B.5 shows a 3D plot of this table. For timing analysis, the Synopsys tools perform a look up in these tables and interpolate the values to find an estimate of the delay.

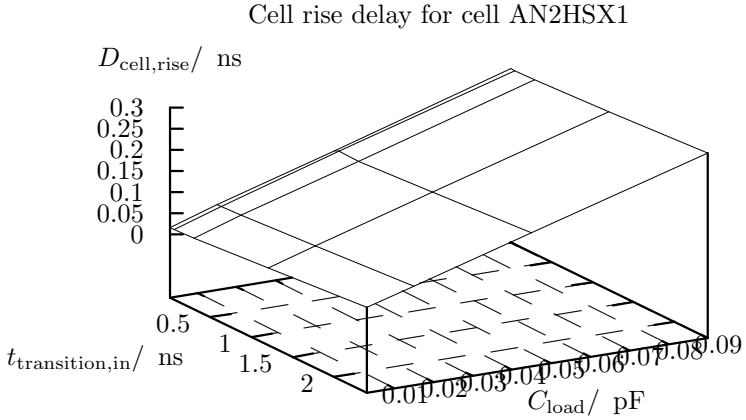


Figure B.5: Cell delay model

The transition time is specified in a table just like D_{cell}. In the sample file, this happens in the lines 127 to 139. As above, the Synopsys tools estimate $t_{\text{transition,out}}$ by performing interpolation of the values looked up. The resulting values of $t_{\text{transition,out}}$ are used as $t_{\text{transition,in}}$ in subsequent levels of cells.

Every wire is associated with a wire load model (unless back-annotation from layout and routing is used). Lines 42 to 55 shows two wire load model specifications. Of these, only the capacitance and the resistance are used. The fanout N is taken from the netlist.

The synthesizer chooses a wire load model by the area of the module it is in. The lines 58 to 64 specify, that the wires in modules with an area between 0 and 900 μm should use the wire load model `maxarea_000990`.

Scaling

For downscaling, the values of $D_{\text{cell,rise}}$ are multiplied with the rise cell delay factor `rd` and the values of $D_{\text{cell,fall}}$ are multiplied with the fall cell delay factor `fd`.

The values of $t_{\text{transition,out,rise}}$ are multiplied with the rise transition time factor `rt` and the values of $t_{\text{transition,out,fall}}$ are multiplied with the fall transition time factor `ft`.

The load capacitance values in the table index (line 31) are scaled by the wire capacitance factor `cu`, since wire capacitance is expected to dominate gate capacitance. The input transition times were not modified, since they were also an input to the simulations, so they were not known beforehand. However, it turned out, that they changed very little during scaling, so this should not introduce a significant error.

Using constant factors for all of these assumes, that all cells will become faster or slower by the same amount. This may lead to inaccuracies.

The resistance and capacitance values from the wire load model are simply multiplied with the resistance per unit length and capacitance per unit length factors `ru` and `cu` respectively.

Wire load selection models are scaled by multiplying the area figures in the wire load selection rules by the area factor `a`.

B.1.2 Leakage power

Modeling

Since the leakage of a cell depends on the input values applied to it, the leakage model reflects this.

$$P_{\text{leak},j,t} = h(v_{j,0,t}, v_{j,1,t}, \ldots)$$

where h is some function and $v_{j,i,t}$ is the value of input j to cell i at time t.

Representation

In the Liberty file, leakage power is given in several places. Lines 16 and 17 list the default values for cells where nothing else is given. For each cell leakage values can be specified separately for each of the possible input combinations as well as a default value for any input combination not listed. Lines 71 to 83 illustrate this. The leakage power unit is pW.

Scaling

For scaling, all leakage values in the file are just multiplied by the leakage factor, `l`. This introduces some error since the leakage power does not change at the same rate for all cells during technology downscaling. For instance, the leakage power has been found to increase faster for cells with a high drive strength than it does for single drive strength cells. See the details in table B.5(a) on page 87.

This is likely to make the synthesizer use more high-drive strength cells, than it otherwise would have.

B.1.3 Dynamic power

Modeling

The dynamic power model used in the Liberty format is

$$P_{\text{dyn}} = P_{\text{dyn,int}} + P_{\text{dyn,cap}}$$

where $P_{\text{dyn,int}}$ is the dynamic power dissipated inside the cell due to the charging and discharging of capacitances internally to the cell and due to short circuit current. $P_{\text{dyn,cap}}$ is power dissipated by the charging and discharging of the capacitance output of the cell i.e. wire capacitances and gate capacitances of cells driven by the output of the cell.

$P_{\text{dyn,int}}$ can be expressed as

$$P_{\text{dyn,int}} = E_{\text{switch}} \cdot \text{TR}$$

where TR is the toggle rate activity and E_{switch} is the energy dissipated during one transition of the output. TR is defined as the number of transitions per time period.

The capacitive switching power is modeled as follows

$$P_{\text{dyn,cap}} = \frac{1}{2} C V_{\text{dd}}^2 \text{TR}$$

where the half factor stems from the fact that capacitive switching power is dissipated once per rising and consecutive falling transition of the output. So TR is half the effective frequency of the nodal charging αf introduced in equation 2.1 on page 15.

Representation

Like with timing, E_{switch} is some function of C_{load} and $t_{\text{transition,in}}$. In the Liberty file, E_{switch} is specified in a look up table in the `internal_power` section, lines 90 to 108. As with timing, each output pin has such a section for each of the input pins. Rise and fall power values are specified independently. In the sample Liberty file, the unit of E_{switch} is pJ.

The capacity of a wire is determined by the wire load model as mentioned above.

Scaling

The scaling is done simply by multiplying the switch energies by the rising power factor `rp` or the falling power factor `fp`.

The scaling of the load capacitance has been taken care of through the scaling of the wire load model and the table definition as mentioned above.

	$V_{\text{th,n}}^{\text{sat}}$	$V_{\text{th,p}}^{\text{sat}}$	$T_{\text{ox,n}}$	$T_{\text{ox,p}}$	$R_{\text{dsw,n}}$	$R_{\text{dsw,p}}$	L_{eff}
BPTM 180 nm							
HS	0.25 V	−0.25 V	40Å	42Å	250 $\Omega/_{\text{sq}}$	450 $\Omega/_{\text{sq}}$	100 nm
LL	0.40 V	−0.40 V					
BPTM 70 nm							
HS	0.15 V	−0.16 V	16Å	17Å	150 $\Omega/_{\text{sq}}$	280 $\Omega/_{\text{sq}}$	38 nm
LL	0.35 V	−0.30 V					

Table B.2: Properties of the transistors used.

B.1.4 Area and capacitance

An area is associated with each cell in the library, as in line 70. The area unit is μm^2. This is scaled simply by multiplying it with the area factor `a`.

In the same way, the maximum load capacitances and gate capacitances (lines 88, 89, 156 and 161) are scaled by multiplying them with the gate capacitance factor `c`. The gate capacitance unit is pF.

B.2 Determining the factors

B.2.1 Simulation setup

As mentioned above, the available 180 nm cell library [41, 42], contains a large number of cells, each of them in both a HS and a LL version. In order to scale this down to a 70 nm cell library, some realistic estimates of how the different numbers scale, had to be obtained. SPICE simulations were used for this.

Unfortunately, SPICE models of the transistors in the 180 nm cell library were not available, and neither were models of the transistors used in these cells. To create an approximate model of the transistors in the STM cell library, SPICE models from the UC Berkeley Device Group's Predictive Technology Model at [25] were used. This group, which also created the BSIM transistor models, has created SPICE models for current and future (predicted) technologies. These models represent a good guess about future technology generations. Both the 180 nm and 70 nm transistor models were taken from here.

On the BPTM website at [25], it is possible to specify parameters for the transistor models wanted. Among these parameters are feature size, V_{th}, T_{ox}, etc. This way, both HS and LL transistor models for the two technologies used were created. The default values proposed by the website were used. For reference, these are given in table B.2, for an explanation of the parameters, see the BPTM manual at [48].

It is of course sub-optimal to use transistor models from BPTM that are not the same models that were used by STM when creating the cell library. Furthermore, the 180 nm STM cell library is only characterized at a best case corner of 1.6 V and −40 °C. As is clear from figure 2.7 on page 20, the leakage current at −40 °C is very small. This is not acceptable for the present purpose, so an operating temperature of 25 °C was chosen for the downscaled cell library. The 70 nm BPTM transistors run at a nominal voltage of 1.0 V.

In all simulations, minimum length transistors are used. The NMOS transistor has $W_\mathrm{n} = \frac{3}{2}L$. Furthermore, $W_\mathrm{p} = \frac{3}{2} \cdot W_\mathrm{n}$ to optimize speed. This was taken from [1].

These facts and choices give the simulation parameters listed in table B.3. In interpreting the simulation results, it should be kept in mind, that it represents a scaling across both technology generation, temperature and supply voltage.

technology	T	V_dd	L	W_n	W_p
180 nm BPTM	−40 °C	1.6 V	180 nm	$\frac{3}{2} \cdot 180\,\mathrm{nm}$	$\frac{3}{2} \cdot \frac{3}{2} \cdot 180\,\mathrm{nm}$
70 nm BPTM	25 °C	1.0 V	70 nm	$\frac{3}{2} \cdot 70\,\mathrm{nm}$	$\frac{3}{2} \cdot \frac{3}{2} \cdot 70\,\mathrm{nm}$

Table B.3: Transistor parameters used for simulation.

For simulation a small selection of commonly used cells was selected. The cells are shown in table B.4. Inverters of drive strengths 4 and 8 are included to represent cells with a drive strength higher than 1. The 3-AND-NOR gate represents more complex gates. Jacob Gregers Hansen has kindly made the SPICE netlists for the cells available.

name	**function**
IVHSX1	inverter, drive strength 1, HS
IVLLX1	inverter, drive strength 1, LL
IVHSX4	inverter, drive strength 4, HS
IVLLX4	inverter, drive strength 4, LL
IVHSX8	inverter, drive strength 8, HS
IVLLX8	inverter, drive strength 8, LL
ND2HSX1	2-input NAND, drive strength 1, HS
ND2LLX1	2-input NAND, drive strength 1, LL
NO2HSX1	2-input NOR, drive strength 1, HS
NO2LLX1	2-input NOR, drive strength 1, LL
A3NOHSX1	3-AND-NOR, drive strength 1, HS
A3NOLLX1	3-AND-NOR, drive strength 1, LL

Table B.4: Cells simulated for scaling factor estimation.

The characterization of the cells was performed using GSPICE [49], a special tool for this purpose. GSPICE sets up input files for SPICE, runs SPICE and analyzes the output. Unfortunately, GSPICE only handles characterization of cells for timing and input capacitance. For the use in this project, GSPICE has been extended to handle also leakage power. Extending GSPICE for the characterization of switching power failed due to a bug in GSPICE. Instead, a custom Perl script, that operates in a similar manner to GSPICE does this. The version of SPICE used was the Synopsys version, HSPICE.

For timing and power simulation, 180 nm and 70 nm cells were simulated for a number of load capacitances and input transition times. As mentioned on page 82, the load capacitances were scaled by the wire load factor `cu` for the 70 nmversion.

The cell netlists, the modified version of GSPICE, and the custom script for switching power characterization are included on the CD-ROM, appendix E, along with a description of the usage of the tools.

The raw simulation data is also found here.

B.2.2 Calculation of the scaling factors

During simulation, each cell was simulated for a number of input signals and the following information was extracted.

leakage power was found for each possible value of the inputs.

timing information was determined for each combination of a number of capacitive output loads and a number of input transition times as explained in section B.1.1.

switching energy was also determined for a number of capacitive output loads and a number of input transition times.

Furthermore, the input pin capacitance was determined for each input.

The results are listed in figure B.5(a) for the HS cells and in table B.5(b) for the LL cells. The figures in the tables are factors as defined in table B.1 on page 79, not absolute values. For example, the leakage power of a x1 HS inverter in 70 nm is 496 times as high as it was in 180 nm.

Since the factors found varied from cell to cell, they had to be weighted in order to combine them into a single, reasonable figure. This was done by using a synthesized 32-bit MIPS processor and collecting statistical information about the number of each cell actually used. The weights are listed in the second column of the tables.

The weights were found by grouping the MIPS processor as well as in the simulation set by area and then using the percentage of the cells in the processor that belonged into a group as the weight in that same group in the simulation setup. For example, about 22% of the cells in the processor had an area close to that of a 2-input NAND or NOR gate. Thus the weight for these two is set to 11%.

Since no cells of a drive strength higher than 4 were used in the design, the values for the x8 inverter were not used (zero weight).

Interestingly, the delay of both HS and LL cells was increased during downscaling. This is due to the simultaneous increase in operating temperature by 65 °C. Dynamic power was decreased considerably, while leakage power was increased.

The area factor `a` was calculated as follows:

$$\mathtt{a} = \frac{(70\,\mathrm{nm})^2}{(180\,\mathrm{nm})^2} = 0.1512$$

The wire unit length capacitance and resistance values were calculated by means of the BPTM predictive wire calculator [25], which gives prospected values for wire capacitance, resistance and inductance for wires in future technology generations. The values for local wires were used, since only small designs are evaluated in this project. The resistance per length and capacitance per length for wires in 180 nm and 70 nm found here were simply divided by each other to obtain `ru` and `cu`.

All factor calculations are contained in the `factors.sxw` OpenOffice Calc spreadsheet on the CD-ROM, appendix E.

B.2.3 Summary for cell library scaling

The values of the inputs to the scaling script are as listed in figure B.6.

The scaling script itself, `scale_cellib`, is found on the CD-ROM. It is a script written in Perl that parses the liberty file and applies the scaling as explained. As mentioned,

cell	weight	l	rd	fd	rt	ft	rp	fp	c
IVHSX1	0.0995	496.0900	1.1606	1.0000	1.2532	1.1221	0.3463	0.3475	0.3861
IVHSX4	0.0001	827.1400	1.1697	0.9010	1.1501	1.0513	0.3293	0.3341	0.3860
IVHSX8	0	930.4600	1.2030	0.8498	1.0770	1.0156	0.3141	0.3200	0.3860
ND2HSX1	0.1102	392.1500	1.1231	1.0200	1.2227	1.1476	0.3239	0.3263	0.3885
NO2HSX1	0.1102	373.2000	1.2525	0.9712	1.3404	1.0953	0.2672	0.2757	0.3892
A3NOHSX1	0.6800	343.2400	1.2148	0.9467	1.2320	1.0749	0.2086	0.2184	0.3787
		367.3296	1.2036	0.9629	1.2452	1.0900	0.2415	0.2495	0.3817

(a) For HS cells.

cell	weight	l	rd	fd	rt	ft	rp	fp	c
IVLLX1	0.0995	4.7101	1.3870	1.2807	1.5040	1.3334	0.3471	0.3477	0.3875
IVLLX4	0.0001	17.4730	1.3922	1.2061	1.3173	1.2192	0.3301	0.3330	0.3914
IVLLX8	0	34.1020	1.4018	1.1836	1.2242	1.1490	0.3094	0.3145	0.3919
ND2LLX1	0.1102	3.4545	1.3489	1.3689	1.4592	1.4460	0.3240	0.3258	0.3843
NO2LLX1	0.1102	2.9291	1.5731	1.2501	1.6211	1.2973	0.2680	0.2734	0.3886
A3NOLLX1	0.6800	2.8178	1.5483	1.2799	1.5076	1.3545	0.0794	0.0802	0.3796
		3.0905	1.5130	1.2865	1.5144	1.3562	0.1538	0.1552	0.3819

(b) For LL cells.

Table B.5: Calculation of scaling factors for cells.

	HS	LL
l	367.3296	3.0905
rd	1.2036	1.5130
fd	0.9629	1.2865
rt	1.2452	1.5144
ft	1.0900	1.3562
rp	0.2415	0.1538
fp	0.2495	0.1552
c	0.3817	0.3819
a	0.1512	
ru	1.8371	
cu	0.7603	
voltage	1.0 V	
temp	25 °C	

Table B.6: Input values to the `scale_cellib` script.

the cell library itself is not contained there, but the CD-ROM contains a simple, detailed procedure for reproducing it from the original STM Liberty file.

The cell library produced was used for entity characterizations as explained in appendix C.

The procedure used is a very rough one, as it assumes that the characteristics of all cells in the library scale by the same amount, which is only approximately true. Furthermore, it assumes that the STM cells can by modeled by the 180 nm BPTM transistors which is also inaccurate. The scaling over both temperature and supply voltage introduces further sources of error.

However, the approach has given a complete cell library, which can be expected to give a reasonable preview into the 70 nm cell library generation. At least the relative size of e.g. dynamic and static power consumption can be expected to be correct.

Interconnect models were scaled assuming that all wires are local wires which should work as long as the designs are small. The example entities characterized during the work on this thesis are small enough to contain only short wires, so this should not cause problems.

Correspondingly, this absence of long wires eliminates the need for high-fanout drivers, so the fact that real high-fanout cells leak more than specified in the constructed cell library, should not matter.

B.3 Sample Liberty file

This section contains the sample Liberty file used in the discussion earlier in this appendix. Figure B.6 on the next page shows an overview of the structure of the file and figure B.7 on page 90 the structure of a single cell declaration.

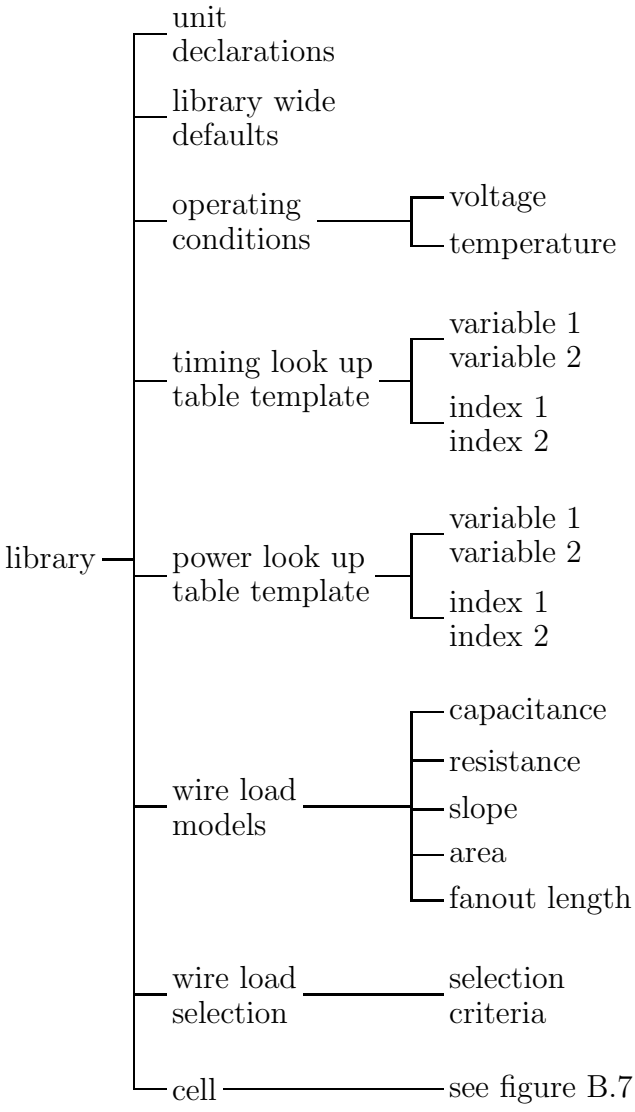


Figure B.6: Overall structure of the sample Liberty file.

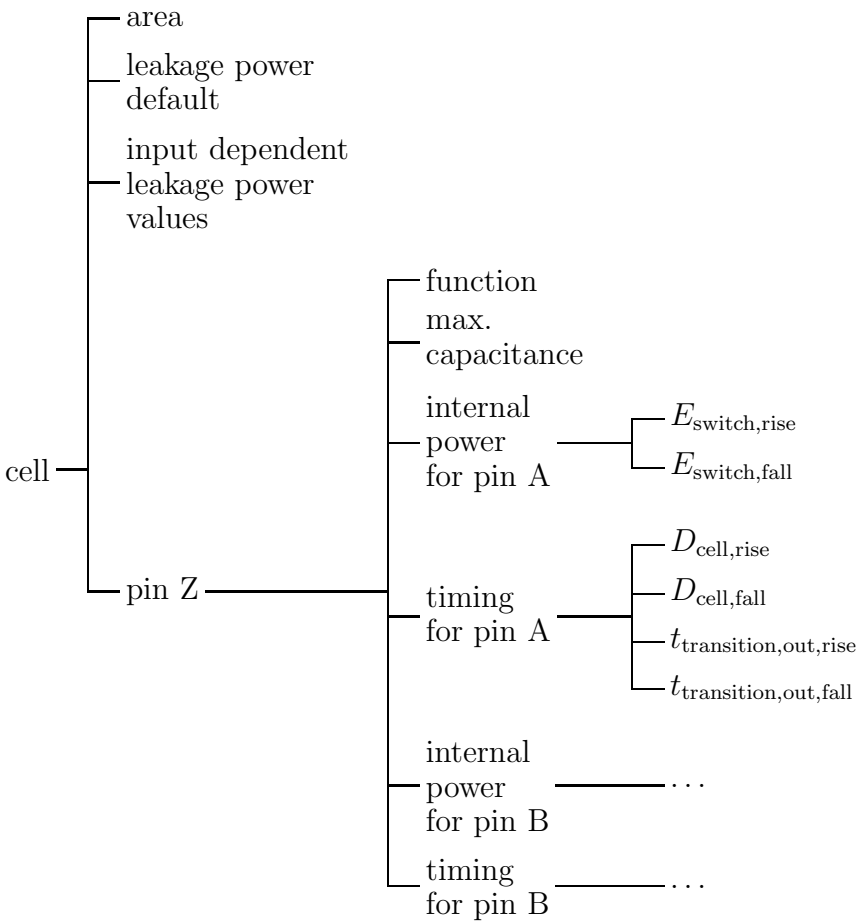


Figure B.7: Overall structure of the cell section of the Liberty file.

```
1  library( some_library ) {
2    /* -------- units -------- */
3    simulation                : true;
4    time_unit                 : "1ns";
5    voltage_unit              : "1V";
6    current_unit              : "1mA";
7    pulling_resistance_unit   : "1kohm";
8    leakage_power_unit        : "1pW";
9    capacitive_load_unit(1,pf);
10
11   /* -------- defaults -------- */
12   default_inout_pin_cap          : 0.01;
13   default_input_pin_cap          : 0.01;
14   default_output_pin_cap         : 0.0;
15   default_max_transition         : 2.4 ;
16   default_cell_leakage_power     : 268.73 ;
17   default_leakage_power_density  : 32.804 ;
18
19   /* -------- operating conditions -------- */
20   operating_conditions(bc_1.60V_m40C){
21      voltage     :   1.60;
22      temperature : -40.0;
23   }
24
25   /* -------- look up tables -------- */
26   delay_model : table_lookup;
27   lu_table_template( table_1 ) {
28     variable_1 : input_net_transition ;
29     variable_2 : total_output_net_capacitance ;
30     index_1 (" 0.01, 0.06, 0.3, 1.2, 2.4 ");
31     index_2 (" 0.003, 0.015, 0.045, 0.09 ");
32   }
33
34   power_lut_template( power_table_1 ) {
35     variable_1 : input_transition_time ;
36     variable_2 : total_output_net_capacitance ;
37     index_1 (" 0.01, 0.06, 0.3, 1.2, 2.4 ");
38     index_2 (" 0.003, 0.03, 0.09, 0.18 ");
39   }
40
41   /* -------- wire load models -------- */
42   wire_load(maxarea_000990) {
43     resistance  : 0.00021 ;
44     capacitance : 0.00017 ;
45     slope       : 9.28 ;
46     area        : 0 ;
```

```
47      fanout_length( 1 , 9.28);
48    }
49
50    wire_load(maxarea_003500) {
51      resistance  : 0.00021 ;
52      capacitance : 0.00017 ;
53      slope       : 14.60 ;
54      area        : 0 ;
55      fanout_length( 1 , 14.60);
56    }
57
58    wire_load_selection(default_by_area){
59      wire_load_from_area( 0 , 990 , maxarea_000990)
60      wire_load_from_area( 990 , 3500 , maxarea_003500)
61    }
62    default_wire_load           : maxarea_003500
63    default_wire_load_selection : "default_by_area"
64    default_wire_load_mode      : "enclosed"
65
66    /* -------- cells -------- */
67
68    /* 2 input AND, 1x drive */
69    cell(AN2HSX1) {
70      area : 21.02 ;
71      cell_leakage_power : 512.533 ;
72      leakage_power() {
73        value : 590.10000 ;
74        when  : "A*B" ;
75      }
76      leakage_power() {
77        value : 537.60000 ;
78        when  : "!A*B" ;
79      }
80      leakage_power() {
81        value : 409.90000 ;
82        when  : "!B*A" ;
83      }
84
85      pin(Z) {
86        direction : output ;
87        function : "A*B";
88        max_capacitance : 0.15000 ;
89        capacitance : 0.00000 ;
90        internal_power() {
91          related_pin : "A" ;
92          rise_power(power_table_1) {
```

```
93
94            values( "0.0153, 0.0531, 0.1355, 0.2616", \
95                    "0.0150, 0.0531, 0.1387, 0.2614", \
96                    "0.0174, 0.0542, 0.1376, 0.2627", \
97                    "0.0315, 0.0659, 0.1499, 0.2719", \
98                    "0.0517, 0.0846, 0.1648, 0.2870" );
99          }
100
101         fall_power(power_table_1) {
102           values( "0.0147, 0.0191, 0.0957, 0.2108", \
103                   "0.0142, 0.0197, 0.0961, 0.2113", \
104                   "0.0167, 0.0179, 0.0944, 0.2095", \
105                   "0.0305, 0.0059, 0.0829, 0.1981", \
106                   "0.0505, 0.0122, 0.0654, 0.1808" );
107         }
108       }
109       timing() {
110         related_pin : "A" ;
111         timing_sense : positive_unate ;
112         cell_rise(table_1) {
113           values( "0.0271, 0.1965, 0.5735, 1.1356", \
114                   "0.0365, 0.1965, 0.5717, 1.1372", \
115                   "0.0616, 0.2248, 0.5742, 1.1376", \
116                   "0.0805, 0.3826, 0.7578, 1.2286", \
117                   "0.1198, 0.5050, 0.9574, 1.5157" );
118         }
119         cell_fall(table_1) {
120         values ( "0.0381, 0.2694, 0.7840, 1.5533", \
121                  "0.0510, 0.2777, 0.7916, 1.5621", \
122                  "0.1043, 0.3228, 0.8312, 1.6024", \
123                  "0.2424, 0.5857, 1.0785, 1.7611", \
124                  "0.3717, 0.8482, 1.4152, 2.1436" );
125         }
126
127         rise_transition(table_1) {
128         values ( "0.0730, 0.5906, 1.7328, 3.4373", \
129                  "0.0788, 0.5814, 1.7270, 3.4726", \
130                  "0.1453, 0.6158, 1.7329, 3.4260", \
131                  "0.3576, 0.8556, 1.8534, 3.4699");
132         }
133         fall_transition(table_1) {
134         values ( "0.0739, 0.5818, 1.7068, 3.4450", \
135                  "0.0807, 0.5834, 1.7211, 3.3830", \
136                  "0.1446, 0.6112, 1.7211, 3.3841", \
137                  "0.3484, 0.8601, 1.8423, 3.4479", \
138                  "0.6147, 1.1392, 2.2538, 3.7682" );
```

```
139         }
140         timing_label : "A_Z" ;
141       }
142       internal_power() {
143         related_pin : "B" ;
144
145          /*  power values for pin B go here */
146       }
147       timing() {
148         related_pin : "B" ;
149
150         /*  timing values for pin B go here */
151       }
152     }
153
154     pin(A) {
155       direction : input ;
156       capacitance : 0.00338 ;
157     }
158
159     pin(B) {
160       direction : input ;
161       capacitance : 0.00325 ;
162     }
163   } /* end of cell */
164 } /* end of library */
```

Appendix C

Building block characterizations

The estimation method presented in chapter 5 requires the building blocks of the design to be precharacterized. A tool was created for this task: Multisim.

Multisim is a generic tool for running Synopsys DC_SHELL simulations systematically and extracting information about the synthesized designs. It is a script written in Perl made specifically for this project.

Given a working Synopsys synthesis setup, Multisim only needs the following things to be able to characterize a component:

- a VHDL or Verilog description of the building block
- a simulation setup file
- a DC_SHELL script template

As output, the script produces an XML file which contains information about dynamic power, leakage power, area, timing, cell mix and module implementations.

In the following, the setup used in the characterization of the building blocks used in this thesis is described. However, Multisim is general enough to be useful for other kinds of Synopsys systematic DC_SHELL runs. Synopsys version 2003.12-SP1 was used for the characterizations done for this thesis.

Multisim can be found on the attached CD-ROM in appendix E along with the simulation setup and raw simulation data for the building blocks of the multiply accumulate unit. A number of small useful tools for manipulating the output of Multisim are also included.

C.1 The characterization setup

The following explanation of the characterization setup will use the characterization of an 8bit ripple carry adder as an example. Characterizing other kinds of building blocks is very similar.

The VHDL file for the adder is shown in figure C.1 on the next page. It simply uses the general DesignWare adder building block. This allows the same source VHDL file to be used for both ripple carry adders, carry lookahead adders and the other adder architectures available through DesignWare. The width of the adder is parameterizable by a generic parameter.

The configuration setup file for the adder is given in figure C.2. Apart from the circuit name and name of the output result file, the name of the DC_SHELL script template is given.

```
1  library IEEE, DWARE, DW01;
2  use IEEE.std_logic_1164.all;
3  use DWARE.DWpackages.all;
4  use DW01.DW01_components.all;
5  entity adder is
6    generic ( W :      integer := 32 );
7    port ( a    : in  std_logic_vector(W-1 downto 0);
8           b    : in  std_logic_vector(W-1 downto 0);
9           ci   : in  std_logic;
10          sum  : out std_logic_vector(W-1 downto 0);
11          co   : out std_logic
12          );
13 end adder;
14
15 architecture func of adder is
16
17 begin  -- Instance of DW01_add
18
19   U1 : DW01_add
20     generic map ( width => W )
21     port map ( A        => a,
22                B        => b,
23                CI       => ci,
24                SUM      => sum,
25                CO       => co );
26
27 end func;
```

Figure C.1: VHDL file for a generic adder.

```
1  circuit adder
2  xmlfile results_adder_rpl_8bit.xml
3  dc_script adder.dc
4  architecture func
5  implementation rpl
6  bits 8
7  period 0.7 0.75 0.8 0.9 1.0 2 3 4 6 7 9 11 15 20
```

Figure C.2: Multisim control file for a 8-bit ripple carry adder.

The remaining lines list inputs to the simulation. Here, one architecture name (func), one adder implementation name (ripple adder), one bit width (8 bits) and 14 delay constraints are given. This will result in $1 \cdot 1 \cdot 1 \cdot 14$ syntheses, since one synthesis is performed for every possible combination of each of these parameters' values. The names of the parameters is not predefined, but can be chosen at will. They will be passed on to the DC_SHELL script template.

```
1  /* target_library = { IMM_HS, IOLIB } */
2  /* target_library = { IMM_LL, IOLIB } */
3  target_library = { IMM_LL, IMM_HS, IOLIB }
4  link_library= target_library + { dw01.sldb, dw02.sldb, \
5                dw03.sldb, dw04.sldb, dw05.sldb, dw06.sldb }
6
7  remove_design -all
8
9  analyze -format vhdl -lib WORK { [ $circuit ].vhd }
10
11
12 [ if(defined $bits){
13    $OUT .= "elaborate $circuit -arch \"$architecture\" -lib WORK ".
14            "-parameters W=$bits";
15 } else {
16    $OUT .= "elaborate $circuit -arch \"$architecture\" -lib WORK";
17 }
18
19    $OUT .= "set_implementation DW01_add/$implementation U1"
20                                if defined $implementation;
21
22 ]
23
24 set_max_delay [ $period ] -from all_inputs() -to all_outputs()
25
26 set_switching_activity  -toggle_rate [ $period/2 ] -period [ $period ]  \
27                         -static_probability 0.5 -select inputs
28
29 link
30
31 uniquify
32
33 [ if(defined $implementation) {
34     $OUT .= "write -f db -hier -output ${circuit}_$implementation.db";
35   } else {
36     $OUT .= "write -f db -hier -output ${circuit}.db";
37 } ]
38
39 set_max_leakage_power 0 mW
40
41 compile  -map_effort medium
42
43
44 [ if(defined $implementation) {
45     $OUT .= "write -f db -hier -output ${circuit}_$period_$implementation.db";
46   } else {
47     $OUT .= "write -f db -hier -output ${circuit}_$period.db";
48 } ]
```

Figure C.3: Generic DC_SHELL script adders.

Figure C.3 shows the DC_SHELL template. This file is not read directly by DC_SHELL, but by Multisim. After modifying it, Multisim passes it on to DC_SHELL. This is done once for each of the simulations defined in the setup file.

Most of the template is simply an ordinary synthesis script that instructs the synthesizer

to use a specific cell library and to analyze, elaborate and compile the design. Any text contained in brackets is interpreted as Perl code. The parameters from the simulation setup file are defined as variables and contain their value for the current run. For instance, `[ $period ]` will iterate over the 14 values for the delay constraint defined in the setup file. Inside such blocks of Perl code, the construct `$OUT .= "something"` will insert the string `something` into the script.

The `elaborate` statement in lines 13 and 14 sets the bit width of the adder according to the bit width parameter in the file. The `set_implementation` command sets the implementation of the DesignWare adder. The `set_max_delay` in line 24 sets the delay constraint and the following `set_switching_activity` command sets the input switching activity to one switch per cycle and the probability of a input signal to have the value 1 at any time to 50%.

Only a timing constraint and a maximum leakage constraint is given to the synthesizer.

After performing the synthesis, Multisim extracts a number of characteristic from the design and store them in an XML file as shown in figure C.4. Here, only two of the 14 synthesis runs are shown. Most of this file should be self-explanatory. The parameters passed to Multisim are contained in this file. In the first run, the timing requirement was not met (VIOLATED), in the second run it was (MET). The power unit is μW and the timing unit is ns.

On the CD-ROM a tool, `extract_all.sh`, is included that converts the XML file to a tabulator separated file suitable for plotting with gnuplot or importing in a spreadsheet.

```
<?xml version='1.0' standalone='yes'?>
<simulations>
  <simulation circuit="adder" dc_script="adder.dc">
    <run machine="sunfire" runtime="00:00:22"
         starttime="2004-07-06␣15:30:12">
      <constraint type="period">0.8</constraint>
      <constraint type="implementation">rpl</constraint>
      <constraint type="bits">8</constraint>
      <implementation count="1" module="DW01_add">rpl</implementation>
      <result type="cell_internal_power">226.1538</result>
      <result type="cell_leakage_power">7.6425</result>
      <result type="net_switching_power">52.9429</result>
      <result type="total_dynamic_power">279.0966</result>
      <result type="HS_area">91.658647</result>
      <result type="HS_cells">8</result>
      <result type="LL_area">8.670413</result>
      <result type="LL_cells">1</result>
      <result type="other_area">0</result>
      <result type="other_cells">0</result>
      <result type="slack">-0.02</result>
      <result type="timing_status">VIOLATED</result>
      <result type="cell_area">100.329063</result>
      <result type="total_area">undefined</result>
    </run>
    <run machine="sunfire" runtime="00:00:20"
         starttime="2004-07-06␣15:30:34">
      <constraint type="period">0.85</constraint>
      <constraint type="implementation">rpl</constraint>
      <constraint type="bits">8</constraint>
      <implementation count="1" module="DW01_add">rpl</implementation>
      <result type="cell_internal_power">218.4532</result>
      <result type="cell_leakage_power">7.1565</result>
      <result type="net_switching_power">50.2809</result>
      <result type="total_dynamic_power">268.7341</result>
      <result type="HS_area">92.897277</result>
      <result type="HS_cells">8</result>
      <result type="LL_area">0</result>
      <result type="LL_cells">0</result>
      <result type="other_area">0</result>
      <result type="other_cells">0</result>
      <result type="slack">0.00</result>
      <result type="timing_status">MET</result>
      <result type="cell_area">92.897278</result>
      <result type="total_area">undefined</result>
    </run>
  </simulation>
</simulations>
```

Figure C.4: Resulting XML file for the Multisim run.

Appendix D

Repeater sizing

To enable experiments with long global wires, a repeater sizing tool was created, simply called `wiresim.pl`. It takes the following arguments:

- the number of repeaters to be inserted
- the delay requirement to meet
- the length of the wire
- the capacitance per length of the wire
- the resistance per length of the wire
- the repeater type (HS or LL)

Using HSPICE simulations, `wiresim.pl` then determines the drive strength necessary to reach the delay specified. The leakage power of the line and the dynamic energy consumed per switch is also determined.

The wire model sent to HSPICE is the Elmore wire model. As shown in figure D.1, a wire is modeled as consisting of N segments each with the same resistance R and capacitance C.

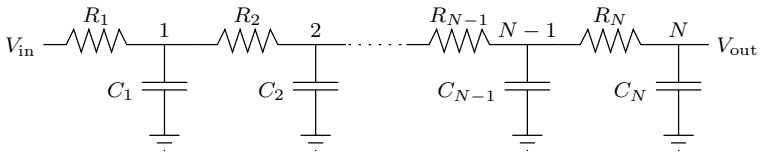


Figure D.1: The Elmore wire model.

`wiresim.pl` uses a fixed number of wire segments $N = 210$, with the capacitance and resistance of the wire distributed evenly on all segments. The repeaters to be inserted are then inserted at equidistant places. These are the two repeaters drawn in black in figure D.2. The two white inverters at the end are unit size inverters whose task it is to present a realistic load at the wire end.

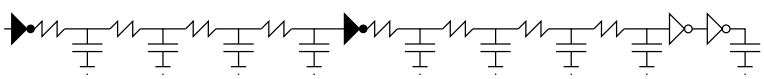


Figure D.2: Repeater insertion.

To create graphs like the ones in chapter 7, a series of such simulations was run on the same wire and delay requirement but with a different number of repeaters.

The tool was implemented in Perl and uses a similar templating technique as the one used for multisim (appendix C) to create the HSPICE input files. It automatically iterates the simulations varying the drive strength until the total wire delay reaches the target delay with some maximum error specified by the user. In the simulations used in chapter 7, this maximum error was set to 1%. `wiresim.pl` uses an internal feedback controller to direct the simulation.

The repeater sizing tool can be found on the CD-ROM in appendix E along with the raw simulation output. All arguments given to it are persistent, so the tool remembers its state from the last run. This makes it possible to use the tool interactively from the command line, since it can be interrupted and restarted again from the point it left off but with one or more options altered.

Appendix E

Digital appendices - CD-ROM

The CD-ROM attached to this thesis contains the following things:

- the power estimation tool in the form of a spreadsheet with the examples of this thesis included
- the logic building block characterization tool Multisim
- the repeater sizing tool `wiresim.pl`
- the cell library scaling tool `scale_cellib`
- the cell library characterization tool chain that determines the scaling factors
- the modified version of GSPICE

A guide to the files on disc can be found in the root of the CD-ROM in the form of an HTML-file called `index.html`.

Bibliography

[1] Jacob Gregers Hansen. Design of CMOS cell libraries for minimal leakage currents. Master's thesis, Informatics and Mathematical Modeling, Technical University of Denmark, 2004.

[2] Anantha P. Chanandrakasan and Robert W. Brodersen. *Low power digital CMOS design*. Kluwer Academic Publishers, 1995.

[3] Jan M. Rabaey and Massoud Pedram, editors. *Low power design methodologies*. Kluwer Academic Publishers, 1996.

[4] Michael Kristensen. Incorporating leakage current considerations in logic synthesis. Master's thesis, Informatics and Mathematical Modeling, Technical University of Denmark, 2004.

[5] Kaushuik Roy, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. Leakage current in deep-submicron CMOS circuits. *Journal of Circuits, Systems, and Computers*, 11(6):575–599, 2002.

[6] George Sery, Shekhar Borkar, and Vivek De. Life is CMOS: Why chase the life after? In *Proceedings of the 39th annual conference on Design automation conference*, pages 78–83. ACM, june 2002.

[7] Neil N. E. Weste and Kamram Eshraghian. *Principles of CMOS VLSI design – A systems perspective*. Addison-Wesley, second edition, 1993.

[8] Ali Keshavarzi, Kaushik Roy, and Charles F. Hawkins. Intrinsic leakage in low-power deep submicron CMOS ICs. In *Proceedings of the IEEE International Test Conference*, pages 146–155. IEEE Computer Society, 1997.

[9] Yongjun Xu, Zuying Luo, and Xiaowei Li. A maximum total leakage current estimation method. In *Proceedings of IEEE International Symposium of Circuits and Systems*, pages II – 757 – 760, 2004.

[10] Dongwoo Lee, Wesley Kwong, David Blaauw, and Dennis Sylvester. Simultaneous subthreshold and gate-oxide tunneling leakage current analysis in nanometer CMOS design. In *Proceedings of the Fourth International Symposium on Quality Electronic Design*, page 287. IEEE, 2003.

[11] S. Naidu and E. Jacobs. Minimizing stand-by leakage power in static CMOS circuits. In *Proceedings of the conference on Design, automation and test in Europe*, pages 370–376. IEEE Press, 2001.

[12] Benton H. Calhoun, Frank A. Honore, and Anantha Chandrakasan. Design methodology for fine-grained leakage control in MTCMOS. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 104–109. ACM Press, 2003.

[13] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey. A voltage reduction technique for digital systems. In *37th International Solid-State Circuits Conference, Digest*

of Technical Papers, pages 238 –239. IEEE International, IEEE, 1990.

[14] C. Kim and K. Roy. Dynamic VTH scaling scheme for active leakage power reduction. In *Proceedings of the conference on Design, automation and test in Europe*, page 163. IEEE Computer Society, 2002.

[15] Luca Benini and Giovanni de Micheli. System-level power optimization: techniques and tools. *ACM Trans. Des. Autom. Electron. Syst.*, 5(2):115–192, 2000.

[16] Navid Azizi, Andreas Moshovos, and Farid N. Najm. Low-leakage asymmetric-cell sram. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 48–51. ACM Press, 2002.

[17] Michael Powell, Se-Hyun Yang, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar. Gated-Vdd:a circuit technique to reduce leakage in deep-submicron cache memories. In *Proceedings of the 2000 international symposium on Low power electronics and design*, pages 90–95. ACM Press, 2000.

[18] Krisztin Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proceedings of the 29th annual international symposium on Computer architecture*, pages 148–157. IEEE Computer Society, 2002.

[19] Chuanjun Zhang, Jun Yang, and Frank Vahid. Low static-power frequent-value data caches. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Volume I (DATE'04)*. IEEE Computer Society, 2004.

[20] Narendra Shenoy. Retiming: Theory and practice. *Integration, the VLSI Journal*, 22(1-2):1–21, 1997.

[21] Soha Hassoun and Carl Ebeling. Architectural retiming: An overview. In *Proceedings of the 1995 ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 1995.

[22] Steen Pedersen. N3: Course project specification. Course 49363, Design of Integrated Circuits 2, 2001.

[23] Niels Seidler Andersen, Jesper Dencker, Martin Hans, Ole Fink Hansen, Michael Kristensen, and Henrik Slotholt. Projekt 3. Project report in course 02401 Data Analysis and Introductory Statistics.

[24] The national technology roadmap for semiconductors. Technical report, SIA – Semiconductor Industry Association, 1997.

[25] UC Berkeley Device Group. Predictive Technology Model. `http://www-device.eecs.berkeley.edu/~ptm/`.

[26] Paul Landman. High-level power estimation. In *Proceedings of the 1996 international symposium on Low power electronics and design*, pages 29–35. IEEE Press, 1996.

[27] E. Macii, M. Pedram, and F. Somenzi. High-level power modeling, estimation and optimization, 1997.

[28] J. Adam Butts and Gurindar S. Sohi. A static power model for architects. In *International Symposium on Microarchitecture*, pages 191–201, 2000.

[29] Rahul Kumar and C.P. Ravikumar. Leakage power estimation for deep submicron circuits in an ASIC design environment. In *Proceedings of the 15th International Conference on VLSI Design*, page p.45. IEEE Press, 2002.

[30] Mahesh Mamidipaka, Kamal Khouri, Nikil Dutt, and Magdy Abadir. Leakage power estimation in SRAMs. Technical Report 03-32, Center for Embedded Computer Systems, 2003.

[31] Paul Landman. *Low power architectural design methodologies*. PhD thesis, University of California at Berkeley, 1994.

[32] David Lidsky and Jan M. Rabaey. Early power exploration – a world wide web application. In *Proceedings of the 33rd annual conference on Design automation conference*, pages 27–32. ACM Press, 1996.

[33] Dennis Sylvester and Kurt Keutzer. Rethinking deep-submicron circuit design. *Computer*, 32(11):25–33, 1999.

[34] M. Horowitz, R. Ho, and K. Mai. The future of wires. In *Proceedings of the IEEE*, volume 89-4, pages 490–504, 2001.

[35] Jason Cong. An interconnect-centric design flow for nanometer technologies. *Proceedings of the IEEE*, 89(4):505–528, April 2001.

[36] Pawan Kapur, Gaurav Chandra, and Krishna C. Saraswat. Power estimation in global interconnects and its reduction using a novel repeater optimization methodology. In *Proceedings of the 39th conference on Design automation*, pages 461–466. ACM Press, 2002.

[37] Prashant Saxena, Noel Menezes, Pasquale Cocchini, and Desmond A. Kirkpatrick. The scaling challenge: Can correct-by-construction design help? In *Proceedings of the 2003 international symposium on Physical design*, pages 51–58. ACM Press, 2003.

[38] Harmander S. Deogun, Rajeev R. Rao, Dennis Sylvester, and David Blaauw. Leakage- and crosstalk-aware bus encoding for total power reduction. In *Proceedings of the 41st annual conference on Design automation*, pages 779–782. ACM Press, 2004.

[39] Hui Zhang, Varghese George, and Jan M. Rabaey. Low-swing on-chip signaling techniques: effectiveness and robustness. *IEEE Trans. Very Large Scale Integr. Syst.*, 8(3):264–272, 2000.

[40] Synopsys Inc. Miron Bar-am. Galaxy 2004 technical overview. Presentation at Nokia Denmark, 2004.

[41] STMicroelectronics. *CORELIB8DHS HCMOS8D 3.1 User's Manuals*, september 2001.

[42] STMicroelectronics. *CORELIB8DLL HCMOS8D 3.1 User's Manuals*, september 2001.

[43] Heise Online. Überall Verzögerungen bei der 90-Nanometer-Chipfertigung. `http://www.heise.de/newsticker/meldung/46834`, April 2004.

[44] Nguyen Minh Duc and Takayasu 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*, pages 475–480. ACM Press, 2000.

[45] Synopsys Inc. Synopsys products – liberty. `http://www.synopsys.com/partners/tapin/lib_info.html`.

[46] Synopsys Inc. *Library Compiler – User Guide: Modeling Timing and Power Technology Libraries*, version u-2003.03 edition, March 2003.

[47] Synopsys Inc. *Library Compiler – Reference Manual: Technology and Symbol Libraries*, version u-2003.03 edition, March 2003.

[48] UC Berkeley Device Group. Predictive Technology Model Manual. `http://www-device.eecs.berkeley.edu/~ptm/manual.html`.

[49] Duane Galbi. GSpice. `http://www.veripool.com/gspice.html`.