Power Optimization and Prediction Techniques for FPGAs

by

Jason Helge Anderson

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Department of Electrical and Computer Engineering
University of Toronto

Power Optimization and Prediction Techniques for FPGAs

Jason Helge Anderson

Doctor of Philosophy, 2005

Graduate Department of Electrical and Computer Engineering

University of Toronto

Abstract

Field-programmable gate arrays (FPGAs) are a popular choice for digital circuit implementation because of their growing density and speed, short design cycle, and steadily decreasing cost. Power consumption, specifically leakage power, has become a major concern for the semiconductor industry and its customers. FPGAs are less power-efficient than custom ASICs, due to the overhead required to provide programmability. Despite this, power has been largely ignored by the FPGA research community, whose prime focus to date has centered on improving FPGA speed and area-efficiency. This dissertation presents new techniques for optimizing and predicting the power consumption of FPGAs.

First, two novel computer-aided design (CAD) techniques for FPGA leakage power reduction are presented. The proposed techniques are unique in that they substantially reduce leakage power, while imposing no cost, meaning that they have no impact on FPGA area-efficiency, speed, or fabrication cost.

Following this, the circuit-level design of low-power FPGA interconnect is considered. A family of new low-power FPGA routing switches is proposed. The switches significantly reduce

dynamic and leakage power in the interconnect, with varying amounts of area and/or performance cost. The proposed switches require only minor changes to traditional FPGA routing switches, allowing them to be easily incorporated into current FPGAs.

Next, a new power-aware technology mapping algorithm for look-up-table-based FPGAs is described. The algorithm takes an activity-conscious approach to logic replication, and allows trade-offs between circuit performance and power. The dynamic power of mapping solutions produced by the proposed algorithm is shown to be considerably less than competing techniques.

Finally, the topic of early dynamic power estimation for FPGAs is addressed. Empirical models are developed for the prediction of interconnect capacitance and switching activity in FPGA designs. The proposed models can be applied early in the design process, when detailed routing data is incomplete or unavailable, thereby reducing design effort and cost.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Field-Programmable Gate Arrays

Field-programmable gate arrays (FPGAs) are programmable logic devices (PLDs) that can be configured by the end-user to implement virtually any digital circuit. Since first introduced in the mid-80s, the popularity of FPGAs has grown steadily, and today, FPGAs account for more than half of the 3 billion dollar programmable logic industry. State-of-the-art FPGAs can implement circuits with millions of gates that operate at speeds in the hundreds of megahertz. The focus of this dissertation is the optimization and prediction of power consumption in FPGAs, through novel computer-aided design (CAD) algorithms, as well as circuit-level design techniques.

An FPGA is a VLSI chip consisting of a pre-fabricated two-dimensional array of programmable logic blocks that connect to one another through a configurable interconnection (routing) network. Static RAM (SRAM) cells, internal to the FPGA, define the logic function implemented by each logic block and the desired connectivity between logic blocks. An FPGA can be configured to implement a given circuit in a matter of seconds and can be reprogrammed any number of times. Custom ASICs are the primary competitor to FPGAs, and they require weeks or months for fabrication. Hence, a key advantage held by FPGAs over ASICs is that FPGAs reduce "time-to-market", which is crucial in the development of new electronic products.

The rapid expansion of the programmable logic market has been driven by a number of factors. Perhaps most important is that as technology scales, the costs associated with building a custom ASIC rise drastically. For example, in 90nm process technology, the cost of mask sets

alone is over a million US dollars [Lamm 03]. Such costs make design mistakes extremely costly, as they necessitate the creation of new mask sets and impose lengthy delays. Comprehensive and rigorous design verification is a mandatory part of custom ASIC design. In FPGAs, the requirement to "get it right the first time" is less critical, since mistakes, once identified, can be taken care of quickly and cheaply by re-programming the device.

Coupled with the high cost of ASIC fabrication, the CAD tools required to design an ASIC cost anywhere from hundreds of thousands to millions of dollars [Sant 03]. In contrast, FPGA vendor tools are typically provided free-of-charge by the FPGA vendors to their best customers, and third-party FPGA CAD tools, such as Synplicity, cost only tens of thousands of dollars. Initially, FPGAs were used only in low-volume production applications or for prototyping circuits that were eventually to be implemented as custom ASICs. However, the volume threshold at which FPGAs are cost-effective versus ASICs has advanced to a point such that modern FPGAs are cost-effective for all but high volume applications.

One of the drawbacks of FPGAs is that they are less area-efficient and also slower than custom ASICs. This characteristic has been the motivation for nearly two decades of academic and industrial research on FPGA CAD and architecture. The result has been a narrowing of the "gap" between ASICs and FPGAs from the area and speed viewpoints. Today, FPGAs are a viable alternative to custom ASICs and can be used in applications with speed and size requirements that previously, could only be met by ASICs.

1.2 Motivation

The ability to program and re-program an FPGA involves significant hardware overhead. More transistors are needed to implement a given logic circuit in an FPGA in comparison with a custom ASIC. This leads to a higher power consumption per logic gate in FPGAs [Geor 01, Zuch 02], and power-efficiency is undisputed as an area in which ASICs are superior to FPGAs [Full 04]. In fact, power has been cited as a limiting factor in the ability of FPGAs to continue to replace ASICs [Stok 03].

Despite the relative weakness of FPGAs from the power angle, their power consumption has, until recently, been largely ignored by the research community. Likewise, no commercial FPGA vendors offer hardware or software specifically targeted to low-power applications. The extent to which FPGA power can be optimized through either CAD, architecture, or circuit techniques has been an open question. The focus of commercial vendors, as well as the majority of published research on FPGA architecture and CAD, has concentrated on improving FPGA area-efficiency and performance. A thorough treatment of prior research on FPGA speed and area is outside the scope here; however, references [Rose 89, Rose 91, Betz 96, Betz 97a, Betz 99a] present the first and seminal work on FPGA logic and routing architecture, with important follow-up work appearing in [Sing 90, Marq 99, Ahme 02]. CAD algorithms that optimize the area and performance of FPGAs are well-studied, with some of the most important papers being [Brow 90, Fran 90, Fran 91a, Lemi 93, Cong 94a, Cong 94b, McMu 95, Betz 97b]. Though area and speed have been the main research focus to date, power is likely to be a key consideration in the design of future FPGAs, for the reasons outlined below.

A well-known consequence of technology scaling is the rapid increase in *static* (*leakage*) power relative to *dynamic* power. Dynamic power is consumed as a result of logic transitions that occur on a circuit's signals during normal operation. It increases in proportion to the rate of logic transitions (switching activity) on circuit signals and the amount of capacitance charged and discharged during logic transitions. Leakage power is consumed when a circuit is in a quiescent, idle state. Both dynamic and leakage power consumption have become major issues for semiconductor vendors and their customers [Inte 02]. Moreover, the considerable increase in leakage with each process generation has significant implications for FPGAs. Leakage current in a circuit is proportional to the circuit's total drawn transistor width [Jian 02]. Since FPGAs contain a huge number of transistors, as required to provide programmability, the need for effective leakage management techniques is amplified in FPGAs versus in other technologies. Given this, low leakage is certain to be a significant design objective in next-generation FPGAs.

Optimizing the power consumption of FPGAs has a number of benefits. First, reduced

power is mandatory if FPGAs are to break into the low-power ASIC market. Historically, applications such as portable or battery-powered electronics have been inaccessible to FPGA vendors, chiefly due to the tight power budgets they impose. For example, mobile applications have standby current limits of 10s to 100s of μA [Clar 02]. As few as 20 logic blocks in the Xilinx Spartan-3 FPGA would exceed this standby current limit [Tuan 03]. In addition to broadening the FPGA market, lowering power consumption would reduce packaging and cooling costs, which represent a sizable fraction of the cost of an IC. Packaging costs for a 90nm design are potentially on par with the cost of the silicon itself [Hawk 03]. Finally, it is worth noting that cooler chips have better reliability, leading to long lifespans.

In conjunction with reducing FPGA power, as power becomes a first-class design consideration for FPGAs, efficient power-aware design will require new estimation tools that gauge power dissipation at the early stages of the design process. Such tools would allow design trade-offs to be considered at a high level of abstraction, reducing design effort and cost.

1.3 Thesis Contributions

This dissertation focuses on two overarching themes:

1. Reducing FPGA power consumption, including dynamic and static power.

2. Early prediction of dynamic power consumption in FPGAs.

With respect to these themes, a number of different contributions are made, as summarized below.

Chapter 3 considers active leakage power dissipation in FPGAs and presents two "no cost" approaches for active leakage reduction[1]. It is well-known that the leakage power consumed by a digital CMOS circuit depends strongly on the state of its inputs. The first leakage reduction technique leverages a fundamental property of basic FPGA logic elements

[1] Active leakage is leakage in the used and operating part of an FPGA.

(look-up-tables) that allows a logic signal in an FPGA design to be interchanged with its complemented form without any area or delay penalty. This property is applied to select polarities for logic signals so that FPGA hardware structures spend the majority of time in low leakage states. The second approach to leakage optimization consists of altering the routing step of the FPGA CAD flow to encourage more frequent use of routing resources that have low leakage power consumptions. Such "leakage-aware routing" allows active leakage to be further reduced, without compromising design performance. In an experimental study, active leakage power is optimized in circuits mapped into a state-of-the-art 90nm commercial FPGA. Combined, the two approaches offer a total active leakage power reduction of 30%, on average. This work has been published in [Ande 04f] and [Ande 05a]. To the author's knowledge, this represents the first published work on active leakage optimization in FPGAs.

Chapter 4 presents circuit-level techniques for reducing power dissipation in FPGA interconnect. It proposes a family of new FPGA routing switch designs that are programmable to operate in three different modes: high-speed, low-power, or sleep. High-speed mode provides similar power and performance to traditional FPGA routing switches. In low-power mode, speed is curtailed in order to reduce power consumption. Leakage is reduced by 28-52% in low-power versus high-speed mode, depending on the particular switch design selected. Dynamic power is reduced by 28-31% in low-power mode. Leakage power in sleep mode, which is suitable for unused routing switches, is 61-79% lower than in high-speed mode. Each of the proposed switch designs has a different power/area/speed trade-off. All of the designs require only minor changes to a traditional routing switch, making them easy to incorporate into current FPGA interconnect. The applicability of the new switches is motivated through an analysis of timing slack in industrial FPGA designs. Specifically, it is observed that a considerable fraction of routing switches may be slowed down (operate in low-power mode), without impacting overall design performance. This work has been published in [Ande 04c], [Ande 04d], and [Ande 05b].

Chapter 5 presents a new power-aware technology mapping algorithm for look-up-table-based FPGAs. The algorithm aims to keep nets with high switching activity out of the FPGA routing network, and takes an activity-conscious approach to logic replication. Logic replication is known to be crucial for optimizing depth in technology mapping. An important contribution of this work is to recognize the effect of logic replication on circuit structure and to show its consequences on power. In an experimental study, the power characteristics of mapping solutions generated by several publicly available technology mappers are examined. Results show that for a specific depth of mapping solution, the power consumption can vary considerably, depending on the technology mapping approach used. Furthermore, results show that the proposed mapping algorithm leads to circuits with substantially less power dissipation than mappings produced by previous approaches. This work has been published in [Ande 02]. To the author's knowledge, this represents the first work on power/depth trade-offs in FPGA technology mapping.

Chapter 6 explores early power prediction for FPGAs. As mentioned above, the dynamic power consumed by a digital CMOS circuit is directly proportional to both switching activity and interconnect capacitance. Chapter 6 considers early prediction of activity and capacitance in FPGA designs. Empirical prediction models are developed for these parameters, suitable for use in power-aware layout synthesis, early power planning, and other applications. The models can be applied early in the design process, when detailed routing data is incomplete or unavailable. The impact of delay on switching activity in FPGAs is studied by examining how the switching activity of a signal changes when delays are zero (zero-delay activity) versus when logic delays are considered (logic-delay activity) versus when both logic and routing delays are considered (routed-delay activity). A novel approach for pre-layout activity prediction is proposed that estimates a signal's routed-delay activity using only zero-delay or logic-delay activity values, along with structural and functional circuit properties. For capacitance prediction, the model's prediction accuracy is improved by considering aspects of the FPGA interconnect architecture in addition

to generic parameters, such as signal fanout and bounding box perimeter length. We also demonstrate that there is an inherent variability (noise) in switching activity and capacitance that limits the accuracy attainable in prediction. Experimental results show that the proposed prediction models work well given the noise limitations. This work has been published in [Ande 03], [Ande 04b], and [Ande 04e].

1.4 Thesis Organization

The remainder of this dissertation is organized as follows: Chapter 2 reviews the background material relevant to the research, including a description of static and dynamic power consumption in CMOS circuits, the impact of technology scaling on leakage, an overview of FPGA technology, and coverage of recent research on FPGA power optimization.

The main research contributions, highlighted above, are presented in Chapters 3, 4, 5, and 6. For clarity, and owing to the range of topics considered, each chapter is self-contained. That is, the experimental results for each proposed power optimization or prediction technique are presented together with the technique's description in a single chapter.

Chapter 7 presents concluding remarks and suggestions for future work.

2 Background and Related Work

2.1 Introduction

This chapter presents the background material that forms the basis for the research presented in later chapters. Section 2.2 reviews power dissipation in CMOS circuits, covering both dynamic, as well as static power. Section 2.3 gives an overview of FPGA architecture and hardware structures, highlighting the features of two state-of-the-art commercial FPGAs. Section 2.4 examines power dissipation in the FPGA context and discusses the breakdown of dynamic and static power dissipation in FPGAs. Section 2.5 surveys the recent literature on FPGA power optimization.

2.2 Power Dissipation in CMOS Circuits

Power consumption in CMOS circuits can be classified as either *dynamic* or *static*. Dynamic power consumption is due to the logic transitions that occur on the signals of a logic circuit. Such transitions occur as a normal part of useful computation, and dynamic power scales in proportion to the rate of computation. Static power, on the other hand, also referred to as leakage power, is dissipated when a logic circuit is in a quiescent state.

2.2.1 Dynamic Power

Dynamic power is consumed through two mechanisms: short-circuit current and the charging and discharging of capacitance [Yeap 98]. Short-circuit current arises in a CMOS gate as its output transitions between logic states. During a transition, both the pull-up and pull-down

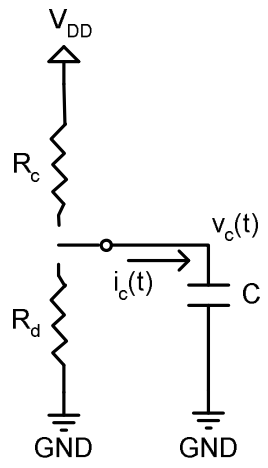


Figure 2.1: Equivalent circuit for a CMOS gate charging and discharging a capacitor [Yeap 98].

networks conduct concurrently for a short time window, resulting in a temporary short-circuit path from supply to ground within the gate. In well-designed circuits, short-circuit current typically represents only 5-10% of dynamic power [Chan 92]. By far, the majority of dynamic power dissipation is due to charging and discharging capacitance [Yeap 98].

Figure 2.1 shows an equivalent circuit for a CMOS gate charging or discharging a capacitance C, where V_{DD} represents the voltage supply and R_c (R_d) represents the resistance of the charging (discharging) circuitry[1]. The time dependent current/voltage characteristics of the capacitor are given by:

$$i_c(t) = C\frac{dv_c(t)}{dt} \tag{2.1}$$

Assume that the capacitor is initially uncharged at time t_0 and that it is fully charged at time t_1; that is, $v_c(t_0) = 0$ and $v_c(t_1) = V_{DD}$. The total energy drawn from the supply to charge the capacitor is given by:

$$E_s = \int_{t_0}^{t_1} V_{DD} \cdot i_c(t)\ dt \tag{2.2}$$

[1]Internal gate capacitances are ignored in the model of Figure 2.1.

Substituting (2.1) into (2.2) yields:

$$E_s = C \cdot V_{DD} \int_{t_0}^{t_1} \frac{dv_c(t)}{dt}\, dt = C \cdot V_{DD} \int_0^{V_{DD}} dv_c = C \cdot {V_{DD}}^2 \tag{2.3}$$

The energy drawn from the supply in a rising transition on the gate's output signal, E_{01}, is equal to E_s. No energy is drawn from the supply when discharging the capacitor in a falling transition, and therefore, $E_{10} = 0$. The average energy consumed per transition on the gate's output signal is:

$$E_{trans} = \frac{E_{01} + E_{10}}{2} = \frac{E_s + 0}{2} = \frac{C \cdot {V_{DD}}^2}{2} \tag{2.4}$$

The average dynamic power consumed by the gate, P_{gate}, depends on the average rate of logic transitions on gate's output signal:

$$P_{gate} = F_{trans} \cdot E_{trans} = \frac{F_{trans} \cdot C \cdot {V_{DD}}^2}{2} \tag{2.5}$$

where F_{trans} is referred to as the *switching activity* of the gate output signal and is expressed in units of transitions per second. In clocked circuits, it is convenient to normalize the switching activity by the clock period as follows:

$$F_{trans} = F_{clk} \cdot F \tag{2.6}$$

where F_{clk} represents the system clock frequency and F represents the average number of transitions on the gate output signal per clock cycle. F is referred to as the *normalized* switching activity.

Substituting (2.6) into (2.5) and summing over all signals yields the familiar equation for the average dynamic power consumed by a CMOS digital circuit:

$$P_{avg} = \frac{F_{clk}}{2} \sum_{i \,\in\, signals} C(i) \cdot F(i) \cdot {V_{DD}}^2 \tag{2.7}$$

where P_{avg} represents average power consumption, $C(i)$ is the load capacitance of a signal i,

and $F(i)$ represents the average number of transitions on signal i per clock cycle (signal i's normalized switching activity).

Various approaches to computing switching activity have been proposed in the literature, and they can generally be classified as either simulation-based approaches or as probabilistic approaches [Najm 94, Soel 00]. In a simulation-based approach, the circuit is simulated with representative vectors, and the simulation tool produces a profile of all signal activities during the simulation. Probabilistic approaches for activity estimation are well-studied (e.g., [Najm 93, Marc 98, Wrig 00, Chou 97, Juan 01, Meht 95]). These approaches require no simulation vectors. Rather, a user is simply required to specify the switching activity, and possibly other properties, of the circuit's primary inputs. An algorithmic approach is used to compute activities for the circuit's internal signals. The advantage of probabilistic approaches over simulation is primarily run-time; the disadvantage is that probabilistic approaches are generally less accurate.

When delays are considered, switching activity normally increases due to the introduction of *glitches*, which are spurious logic transitions on a signal caused by unequal path delays to the signal's driving gate. As transitions on gate inputs occur at different times, the signal experiences multiple transitions before settling to its final value. The extra activity due to glitching consumes dynamic power, and previous work has suggested that 20-70% of total power dissipation in ASICs can be due to glitches [Shen 92].

Chapter 6 studies switching activity, glitching severity, and capacitance in FPGAs and presents new techniques for early dynamic power estimation.

2.2.2 Static (Leakage) Power

The primary leakage mechanisms in an MOS transistor are illustrated in Figure 2.2, and consist of subthreshold leakage, gate oxide leakage, and junction leakage (also called band-to-band tunneling leakage) [Agar 04]. Junction leakage comprises a small fraction of total leakage [Doyl 02], and refers to the current flow across the reverse-biased p-n junctions at the interface between

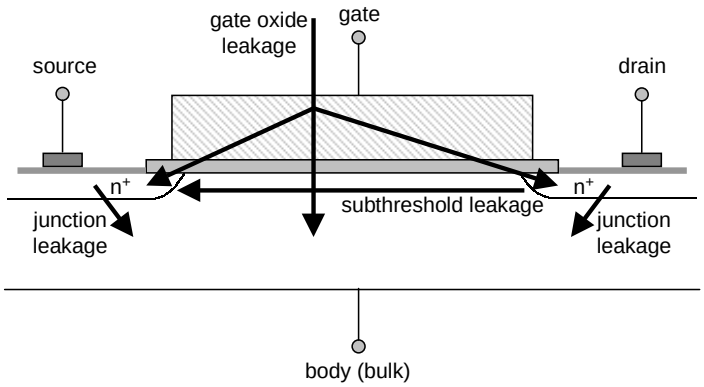


Figure 2.2: Leakage mechanisms in an NMOS transistor.

the source/drain and the substrate. The two dominant leakage mechanisms are subthreshold leakage and gate oxide leakage [Doyl 02, Nowa 02]. These two mechanisms, as well as the way in which they are impacted by technology scaling, are described below.

An "ideal" MOS transistor can be viewed as a perfect switch, with the gate terminal exhibiting perfect control over the drain-to-source current (I_{DS}). When the potential difference between the gate and source (V_{GS}) is less than the transistor's threshold voltage (V_{TH}), the transistor is said to be OFF and in the *cut-off* state. Ideally, $I_{DS} = 0$ in cut-off. In reality however, a non-zero *subthreshold* current may flow between the drain and source terminals in cut-off. Subthreshold leakage in a transistor depends on process parameters as well as bias conditions, as modeled by [Roy 03]:

$$I_{DS} = A \ \times \ e^{\frac{q}{mkT}(V_{GS}-V_{TH})} \ \times \ (1 - e^{\frac{-V_{DS}q}{kT}}) \tag{2.8}$$

where

$$V_{TH} = V_{th0} - \gamma\prime \cdot V_S + \eta \cdot V_{DS} \tag{2.9}$$

and

$$A = \mu_0 C_{ox} \frac{W}{L_{eff}} (\frac{kT}{q})^2 e^{1.8} \tag{2.10}$$

The parameters in (2.8), (2.9), and (2.10) are defined as follows: V_{th0} is the zero-bias threshold voltage, $\gamma\prime$ is the linearized body effect coefficient, η is called the drain-induced barrier lowering coefficient, C_{ox} is the gate oxide capacitance, μ_0 is the zero-bias mobility, m is the subthreshold swing, W and L_{eff} are the width and effective length of the transistor, k represents Boltzmann's constant, T is temperature in degrees Kelvin, and q is the electron charge.

From (2.8), (2.9), and (2.10), several interesting properties of subthreshold leakage can be inferred. First, subthreshold leakage *increases* exponentially as threshold voltage is reduced and *decreases* exponentially as gate/source bias (V_{GS}) is reduced. These properties arise from the first exponential term of (2.8). Second, threshold voltage depends on the drain/source bias (V_{DS}). This is referred to as drain-induced barrier lowering (DIBL), and is modeled by the third term on the right side of (2.9). Finally, subthreshold leakage increases exponentially with temperature, roughly doubling for every 10°C increase in temperature [Nowa 02].

Similar to how an MOS transistor is a non-ideal switch, the gate terminal of a transistor is an imperfect insulator. Gate oxide leakage is due to a non-zero tunneling current through the insulating gate oxide. It increases exponentially as oxide thickness is reduced. It also increases exponentially as the potential difference across the gate oxide is increased [Lee 04, Kris 02, Nowa 02]. Figure 2.3 (from [Thom 98]) illustrates the exponential dependence of gate oxide leakage on gate bias and oxide thickness. For an NMOS device, in the ON state, gate oxide leakage flows *from* the gate terminal *to* the channel, drain, source, and substrate [Agar 04], in a mechanism referred to as *direct tunneling* [Roy 03]. In the OFF state, the overlap between the gate and the source/drain regions permits leakage *from* the source/drain *to* the gate terminal. This is referred to as *edge-directed tunneling*, and its magnitude is much smaller than direct tunneling gate leakage [Lee 04].

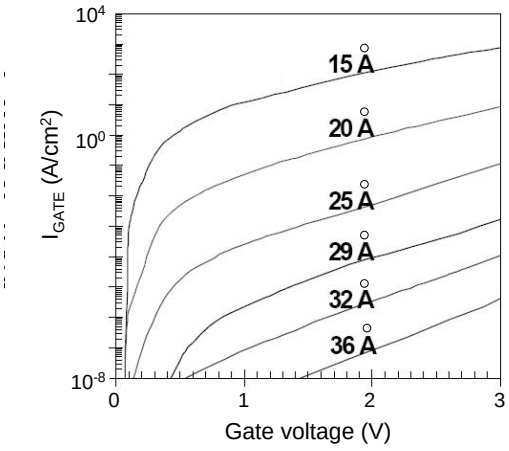


Figure 2.3: Gate oxide leakage dependence on oxide thickness and gate bias [Thom 98].

Impact of Technology Scaling on Leakage

"The number of transistors on an integrated circuit doubles every 18 months."

Moore's Law, first stated in the 1960s, has largely remained true for four decades, and is the basis for the incredible growth of the semiconductor industry throughout this period. Such drastic scaling has been made possible by the seemingly endless ability to shrink the size of a transistor, markedly increasing the density of transistors on a single IC.

As transistors are made smaller, there are two important consequences. First, the exponential growth in the number of devices on a single chip leads to a higher power consumption. Second, the electric fields internal to a transistor increase, which impacts transistor reliability[2]. To address these issues, supply voltage must be scaled in tandem with feature size. Figure 2.4 (from [Doyl 02]) shows the scaling of transistor gate length and supply voltage versus process generation. The supply voltage scales at approximately 0.85X per generation; the gate length scales at approximately 0.65X per generation.

The drive capability and associated speed performance of a transistor depends on the mag-

[2]Field strength in an MOS transistor influences failure due to gate oxide breakdown [Amer 98].

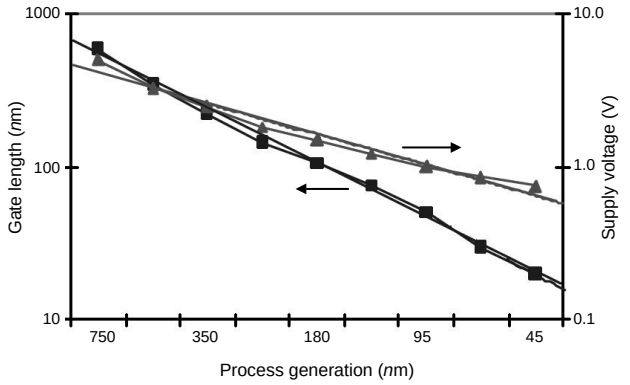


Figure 2.4: Scaling of gate length and supply voltage [Doyl 02].

nitude of $V_{DD} - V_{TH}$ [Sedr 97]. Consequently, as supply voltages are reduced, threshold voltages must also be reduced to mitigate performance degradations. As discussed in Section 2.2.2, reducing V_{TH} yields an exponential increase in subthreshold leakage. Thus, smaller feature sizes necessitate lower supply voltages, which in turn necessitate lower threshold voltages, which are associated with increased subthreshold leakage current. To be sure, subthreshold leakage is predicted to increase by roughly 5X per process generation [Bork 99].

Additionally, a transistor's drive current depends linearly on its gate oxide capacitance, C_{ox}, defined as:

$$C_{ox} = \frac{\varepsilon}{t_{ox}} \tag{2.11}$$

where ε is the permittivity of the gate insulator and t_{ox} is the oxide thickness. Current drive is improved by thinning the oxide (reducing t_{ox}), producing an exponential increase in gate oxide leakage[3]. Figure 2.5 (from [Taur 02]) illustrates the scaling of supply voltage, oxide thickness, and threshold voltage with process generation.

The scaling trends outlined above imply that leakage power will constitute an increasingly dominant component of total power in future process technologies. A survey conducted by

[3]Thinner oxides are also required to maintain adequate gate control over the drain current as technology scales [Fran 02].

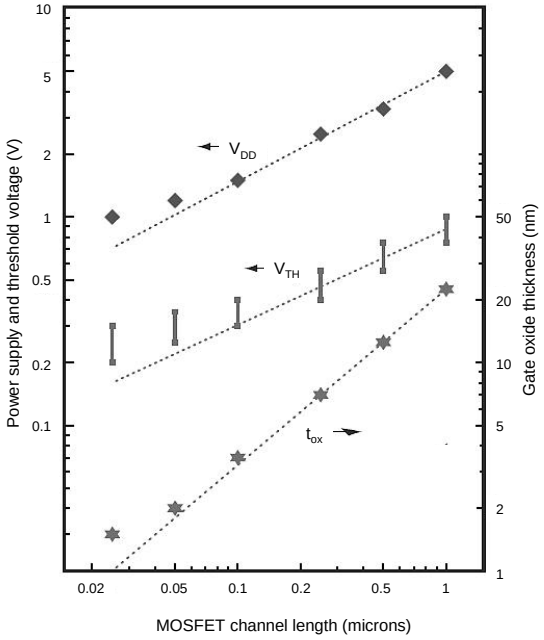


Figure 2.5: Scaling of V_{DD}, V_{TH}, and t_{ox} with process generation [Taur 02].

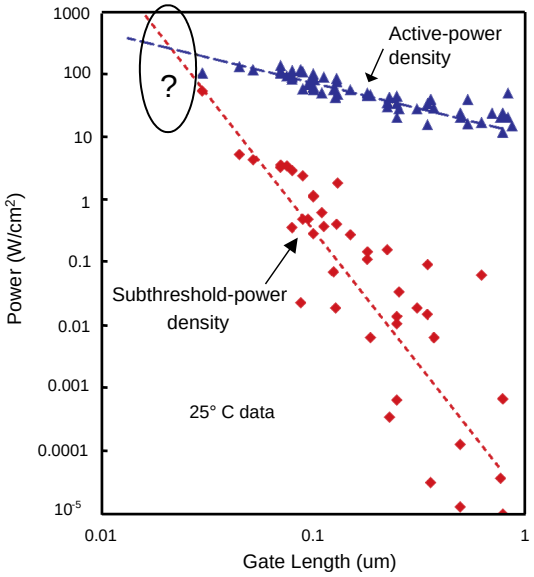


Figure 2.6: Scaling of subthreshold leakage power density and dynamic power density [Nowa 02].

Nowak produced the trend data shown in Figure 2.6 [Nowa 02]. The figure clearly illustrates that both static and dynamic power increase as technology scales; however, the rate of increase of static power is considerably faster. Recent work suggests that static power may constitute over 40% of total power at the 70nm technology node [Kao 02].

Leakage Reduction Techniques

Before going further, it is worthwhile to highlight a few of the most important leakage reduction techniques used in ASICs and microprocessors, including those that play a role in the research presented in subsequent chapters. A more detailed overview of leakage optimization techniques can be found in [Roy 03].

Prior work on leakage optimization in ASICs differentiates between *active* and *sleep* (or *standby*) leakage. Sleep leakage is that dissipated in circuit blocks that are temporarily inactive and that have been placed into a special "sleep state", in which leakage power is minimized.

Active leakage, on the other hand, is that dissipated in circuit blocks that are in use – blocks that are "awake".

Several recent papers have considered ASIC standby leakage power optimization. In [Anis 02, Saku 02], the authors introduce high threshold *sleep transistors* into the N-network and/or P-network of CMOS gates, as illustrated in Figure 2.7(a). Sleep transistors are ON when the circuit is active and are turned OFF when the circuit is in standby mode, effectively limiting the leakage current from supply to ground. A different approach to leakage reduction is based on the fact that a circuit's leakage depends on its input state. In [Halt 97, Abdo 02], a specific input vector is identified that minimizes leakage power in a circuit; the vector is then applied to circuit inputs when the circuit is in standby mode. This idea requires only minor circuit modifications and has been shown to reduce leakage by up to 70% in some circuits [Abdo 02].

Active leakage reduction has also been addressed in the literature. One approach performs dynamic V_{TH} adjustment based on system workload [Kim 02, Mart 02]. The body effect is used to raise transistor V_{TH} when high system throughput is not required, and the circuit can be slowed down. Figure 2.7(b) illustrates the concept. In the figure, VPB (VNB) would be set higher (lower) than V_{DD} (GND) in low leakage mode. Such body bias methods can also be used for standby leakage power reduction [Kesh 01]. Other circuit-level techniques include the use of dual or multi-threshold CMOS [Siri 02, Usam 02, Lee 03, Basu 04, Sriv 04b], wherein multiple transistor types with different threshold voltages are available. Low-V_{TH} transistors are selected for use in delay-critical paths and high-V_{TH} transistors are used in non-critical paths. Considerable leakage power reductions are possible, as there are usually few delay-critical paths. Similarly, dual-t_{ox} design techniques have been proposed recently for gate oxide leakage reduction [Sult 04].

Another popular active leakage optimization technique is to replace individual transistors in gates with "stacks" of transistors in series [Nare 01, John 02, Kao 02, Liu 02], as shown in Figure 2.7(c). Transistor stacks leak less than individual transistors when in the OFF state, a phenomenon widely referred to as the *stack effect* [Nare 01]. A related approach is to use

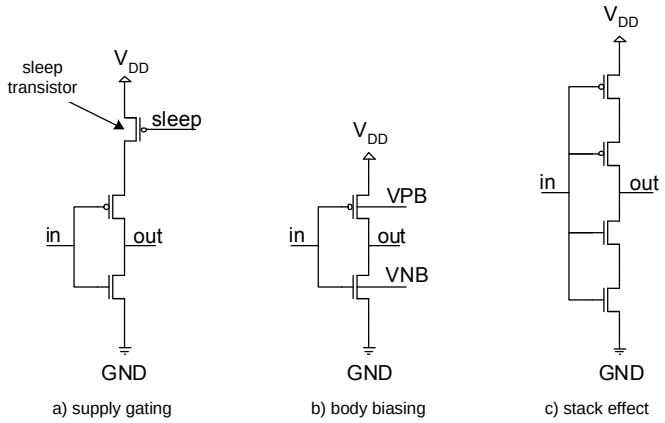


Figure 2.7: ASIC leakage optimization techniques.

transistors with longer channel lengths, which are known to have better leakage characteristics [Roy 03]. Note that the leakage improvements offered by all of the techniques mentioned here come with associated costs, impacting circuit area, delay, or fabrication cost.

In the future, in addition to the techniques noted above, leakage may be addressed through changes to the fabrication process, such as using metal gate electrodes, or through the adoption of alternate logic technologies, such as SOI or double-gate CMOS [Taur 02, Nowa 02, Thom 98, Doyl 02, Zeit 04].

2.3 FPGA Architecture and Hardware Structures

Having reviewed the basis of power dissipation in CMOS circuits, we now turn our attention to FPGAs. This section presents an overview of FPGA architecture and hardware structures using two recently-developed commercial FPGAs as example cases: the Xilinx Virtex-4 FPGA [Virt 04] and the Altera Stratix II FPGA [Stra 04].

FPGAs consist of a two-dimensional array of programmable logic blocks that are connected through a configurable interconnection fabric. Figure 2.8(a) provides an abstract view of an FPGA. As illustrated, pre-fabricated routing tracks are arranged in channels that are inter-

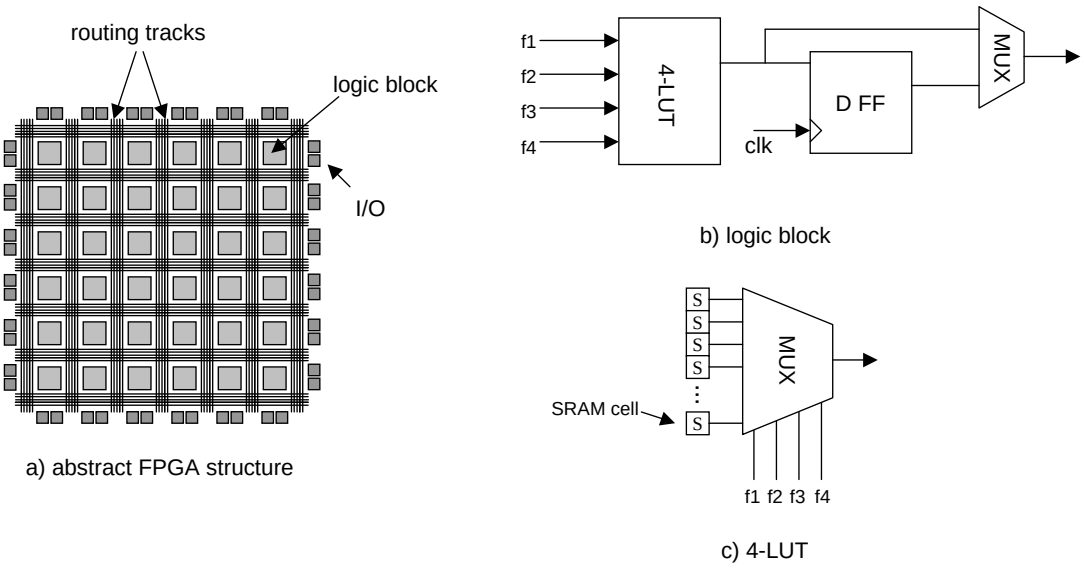


Figure 2.8: (a) Abstract FPGA architecture; (b) logic block; (c) LUT.

spersed between rows and columns of logic blocks. Today's commercial FPGAs use look-up-tables (LUTs) as the base element for implementing combinational logic functions, and contain flip-flops for implementing sequential logic. A K-input LUT (K-LUT) is a small memory capable of implementing any logic function that uses, at most, K inputs. A simplified FPGA logic block is shown in Figure 2.8(b), comprising a 4-LUT along with a flip-flop. A programmable multiplexer allows the flip-flop to be bypassed. Figure 2.8(c) shows the internal details of a 4-LUT. 16 SRAM cells hold the truth table for the logic function implemented by the LUT. The LUT inputs, labeled f1–f4, select a particular SRAM cell whose content is passed to the LUT output.

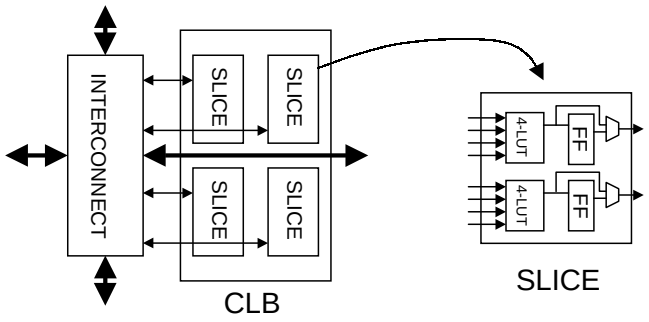
The logic blocks in commercial FPGAs are more complex than that of Figure 2.8(b) and contain *clusters* of LUTs and flip-flops. Figure 2.9 shows the logic blocks in Virtex-4 and Stratix II. A Virtex-4 logic block [Figure 2.9(a)] is referred to as a Configurable Logic Block (CLB) and it contains 4 SLICEs, where each SLICE has two 4-LUTs, two flip-flops, as well

as arithmetic and other dedicated circuitry. The two 4-LUTs in a SLICE can be combined to create a single 5-LUT; the LUTs in two SLICEs can be combined to create a single 6-LUT.

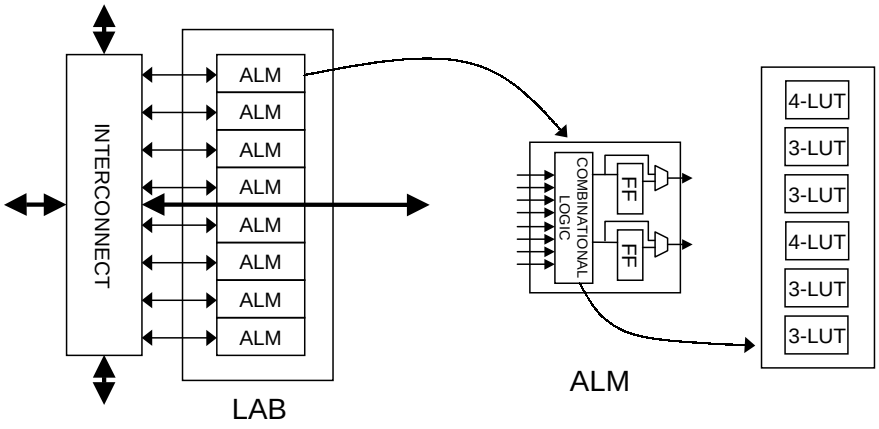
A Stratix II logic block, referred to as a Logic Array Block (LAB), is shown in Figure 2.9(b). A LAB contains eight sub-blocks, called Adaptive Logic Modules (ALMs). Each ALM contains two 4-LUTs and four 3-LUTs, two flip-flops, as well as arithmetic and other circuitry. Multiplexers and associated configuration circuitry make a single ALM quite flexible. Specifically, the smaller LUTs in an ALM may be combined to form larger LUTs. For example, all of the LUTs in an ALM can be combined to implement a 6-LUT. Alternately, the LUTs can be combined to create various combinations of two LUTs (e.g., a 5-LUT and a 4-LUT). Many other ALM configurations are also possible [Stra 04]. Comparing the two logic blocks, it is apparent that a LAB is more coarse-grained than a CLB. Since a LUT with K inputs requires 2^K SRAM configuration cells, a LAB contains $8 \times (2 \times 16 + 4 \times 8) = 512$ bits of LUT RAM memory; a CLB contains $4 \times (2 \times 16) = 128$ bits of LUT RAM memory.

Note that in addition to the LUT-based logic blocks described here, commercial FPGAs contain other hardware blocks, including block RAMs, multipliers, and DSP blocks [Virt 04, Stra 04]. Typically, such blocks are placed at regular locations throughout the two dimensional FPGA fabric. Furthermore, commercial FPGAs have programmable I/O blocks, capable of operating according to a variety of different signaling standards.

Connections between logic blocks in an FPGA are formed using a programmable interconnection network, having variable length wire segments and programmable routing switches. A typical FPGA routing switch is shown in Figure 2.10 [Lemi 02, Lemi 04, Lewi 03, Rahm 04]. It consists of a multiplexer, a buffer, and SRAM configuration bits. Within an FPGA, the switch's multiplexer inputs, labeled i1–in, connect to other routing conductors or to logic block outputs. The buffer's output connects to a routing conductor or to a logic block input. The programmability of an FPGA's interconnection fabric is realized through the SRAM cells in the configuration block, labeled "config" in Figure 2.10. The SRAM cell contents control which input signal is selected to be driven through the buffer. Combined, a routing switch and the



a) Xilinx Virtex-4 logic block



b) Altera Stratix-II logic block

Figure 2.9: Logic blocks in Xilinx and Altera commercial FPGAs.

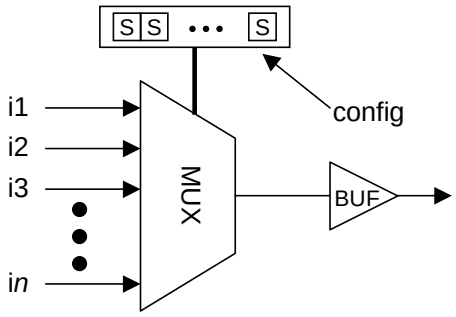


Figure 2.10: FPGA routing switch.

Table 2.1: Summary of commercial FPGA routing architectures (lengths given in CLB tiles or LAB tiles, as appropriate).

Resource type	**Virtex-4**	**Stratix II**
Local	Internal to CLB	Internal to LAB
Short range	DIRECT (8 neighbors) DOUBLE (length 2)	DIRECT (east/west neighbors)
Medium range	HEX (length 6)	C4, R4 (length 4)
Long range	LONG (length 24)	C16, R24 (length 16, 24)

conductor it drives are referred to as a *routing resource*. The connectivity pattern between logic blocks and routing, as well as the length and connectivity of routing conductors constitute the FPGA's *routing architecture*.

Virtex-4 and Stratix II have similar routing architectures. Both offer "local" routing resources for connections *within* a CLB or a LAB. Virtex-4 includes DIRECT resources that connect a CLB to its eight neighbors (including the diagonal neighbours). DOUBLE and HEX resources run horizontally and vertically and span two and six CLB tiles, respectively. LONG resources in Virtex-4 span 24 CLB tiles. In Stratix II, DIRECT resources provide connectivity between a LAB and its neighbours to the left and right. R4 and C4 resources span four LAB tiles and run horizontally and vertically, respectively. For long distance connections, C16 and R24 resources are available that span 16 and 24 LAB tiles, respectively. Table 2.1 summarizes the routing architectures of the two FPGAs.

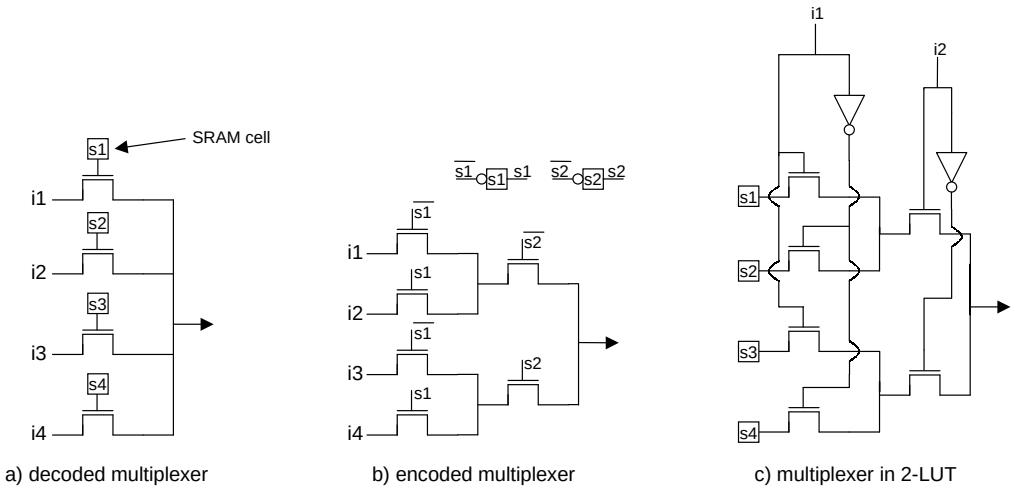


Figure 2.11: Multiplexers as deployed in FPGA routing switches and a LUT.

Given the discussion so far, the reader will appreciate that the multiplexer is perhaps the most important circuit element in an FPGA, since they are used extensively throughout the interconnect and are also used to build LUTs. It is therefore worthwhile to review this structure in some detail. The multiplexers in FPGAs are typically implemented using NMOS transistor trees [Lemi 02]. Figures 2.11(a) and (b) depict multiplexers, as they would be deployed in a routing switch. Full CMOS transmission gates are generally not used to implement multiplexers in FPGAs because of their larger area and capacitance [Lemi 03]. Figures 2.11(a) and (b) show two possible implementations of a 4-to-1 multiplexer. Figure 2.11(a) shows a "decoded" multiplexer, which requires four configuration SRAM cells if used in an FPGA routing switch. Input-to-output paths through this decoded multiplexer consist of a single NMOS transistor. Figure 2.11(b) shows an "encoded" multiplexer that requires only two configuration SRAM cells, though has larger delay as its input-to-output paths consist of two transistors in series. In larger multiplexers, a combination of the designs shown in Figure 2.11 is also possible, allowing one to trade-off area for delay or vice-versa. In a LUT, the LUT inputs drive multiplexer select signals; SRAM cells containing the truth table of the LUT's logic function attach to the

multiplexer's inputs. A multiplexer in a two-input LUT is shown in Figure 2.11(c).

2.4 Power Dissipation in FPGAs

2.4.1 Dynamic Power

A number of recent papers have considered the breakdown of dynamic power consumption in FPGAs [Poon 02b, Li 03, Shan 02]. [Shan 02] studied the breakdown of power consumption in the Xilinx Virtex-II commercial FPGA. The results are summarized in Figure 2.12. Interconnect, logic, clocking, and the I/Os were found to account for 60%, 16%, 14%, and 10% of Virtex-II dynamic power, respectively. A similar breakdown was observed in [Poon 02b]. The FPGA power breakdown differs from that of custom ASICs, in which the clock network is often a major source of power dissipation [Yeap 98].

The dominance of interconnect in FPGA dynamic power is chiefly due to the composition of FPGA interconnect structures, which consist of pre-fabricated wire segments, with used and unused switches attached to each wire segment. Such attached switches are not present in custom ASICs, and they contribute to the capacitance that must be charged/discharged in a logic transition. Furthermore, SRAM configuration cells and circuitry constitute a considerable fraction of an FPGA's total area. For example, [Ye 04] suggests that more than 40% of an FPGA's logic block area is SRAM configuration cells. Such area overhead makes wirelengths in FPGAs longer than wirelengths in ASICs. Interconnect thus presents a high capacitive load in FPGAs, making it the primary source of dynamic power dissipation.

2.4.2 Static (Leakage) Power

In comparison with dynamic power dissipation, relatively little has been published on FPGA leakage power. One of the few studies was published by Tuan and Lai in [Tuan 03], and examined leakage in the Xilinx Spartan-3 FPGA, a 90nm commercial FPGA [Spar 04]. Figure 2.13 shows the breakdown of leakage in a Spartan-3 CLB, which is similar to the Virtex-4 CLB described in Section 2.3. Leakage is dominated by that consumed in the interconnect, configu-

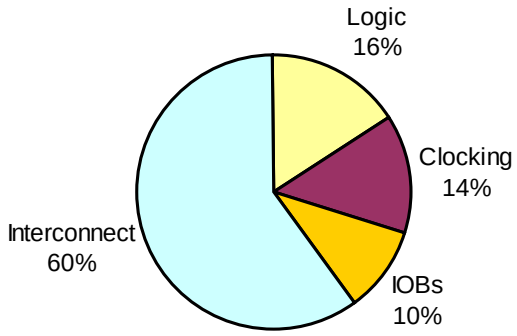


Figure 2.12: Dynamic power breakdown in Xilinx Virtex-II [Shan 02].

ration SRAM cells, and to a lesser extent, LUTs. Combined, these structures account for 88% of total leakage.

As pointed out in [Tuan 03], the contents of an FPGA's configuration SRAM cells change only during the FPGA's configuration phase. Configuration is normally done once – at power-up. Therefore, the speed performance of an FPGA's SRAM configuration cells is not critical, as it does not affect the operating speed of the circuit implemented in the FPGA. The SRAM cells can be slowed down and their leakage can be reduced or eliminated using previously-published low leakage memory techniques, such as those in [Kim 03], or by implementing the memory cells with high-V_{TH} or long channel transistors. Leakage was not a primary consideration in the design of Spartan-3. If SRAM configuration leakage were reduced to zero, the Spartan-3 interconnect and LUTs would account for 55% and 26% of total leakage, respectively.

Note that unlike ASICs, a design implemented in an FPGA uses only a portion of the underlying FPGA hardware. Leakage is dissipated in both the *used* and the *unused* parts of the FPGA. To be sure, [Tuan 03] suggests that up to 45% of leakage in Spartan-3 is "unused" leakage (assuming reasonable device utilization). Notably, today's commercial FPGAs do not yet offer support for a low leakage sleep mode for unused regions.

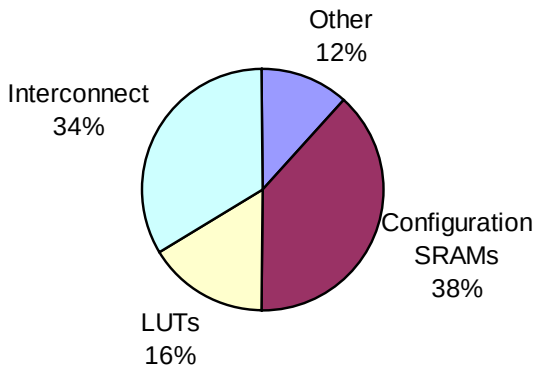


Figure 2.13: Leakage power breakdown in Xilinx Spartan-3 [Tuan 03].

2.5 FPGA Power Optimization

This section summarizes recent literature on FPGA power optimization, including techniques for leakage optimization, architecture/circuit-level techniques for dynamic power reduction, and CAD approaches for dynamic power reduction.

2.5.1 Leakage Power Optimization

Two recent papers focussed on optimizing leakage in the unused portion of an FPGA. Calhoun proposed the creation of fine-grained "sleep regions", making it possible for a logic block's unused LUTs and flip-flops to be put to sleep independently [Calh 03]. A more coarse-grained sleep strategy was proposed in [Gaya 04b], which partitioned an FPGA into entire *regions* of logic blocks, such that each region can be put to sleep independently. The authors restricted the placement of the implemented design to fall within a minimal number of the pre-specified regions, and studied the effect of the placement restrictions on design performance.

One of few papers to address leakage in FPGA interconnect is [Rahm 04], which applied well-known leakage reduction techniques to interconnect multiplexers. Four different techniques were studied. First, extra configuration SRAM cells were introduced to allow for multiple OFF transistors on unselected multiplexer paths. The intent is to take advantage of the "stack

effect", as illustrated in Figure 2.14(a). The left side of Figure 2.14(a) shows a typical routing switch multiplexer. Observe that there is a single OFF transistor on the unselected multiplexer path (highlighted). The right side of Figure 2.14(a) shows the redundant SRAM cell approach. The unselected path contains two OFF transistors, which limits subthreshold leakage along the path.

A second approach described in [Rahm 04] is to layout portions of the multiplexer in separate wells, allowing body-bias techniques to be used to raise the V_{TH} of multiplexer transistors that are not part of the selected signal path [see Figure 2.14(b)]. Third, [Rahm 04] proposes negatively biasing the gate terminals of OFF multiplexer transistors [Figure 2.14(c)]. The negative gate bias leads to a significant drop in subthreshold leakage [recall equation (2.8)]. Finally, [Rahm 04] proposes using dual-V_{TH} techniques, wherein a subset of multiplexer transistors are assigned high-V_{TH} (slow/low leakage), and the remainder of transistors are assigned low-V_{TH} (fast/leaky). The dual-V_{TH} idea, shown in Figure 2.14(d), impacts FPGA router complexity, as the router must assign delay-critical signals to low-V_{TH} multiplexer paths. A more recent paper by Ciccarelli applies dual-V_{TH} techniques to the routing switch buffers in addition to the multiplexers [Cicc 04].

Chapters 3 and 4 present novel leakage reduction techniques for FPGA logic and interconnect that are orthogonal to those mentioned here.

2.5.2 Dynamic Power Optimization (Architecture/Circuit Techniques)

The first comprehensive effort to develop a low-energy FPGA was by a group of researchers at UC Berkeley [Kuss 98, Geor 99, Geor 01]. A power-optimized variant of the Xilinx XC4000 FPGA [X4K 02] was proposed. Power reductions were achieved through significant changes in the logic and routing fabrics. First, larger, 5-input LUTs were used rather than 4-LUTs, allowing more connections to be captured within LUTs instead of being routed through the power-dominant interconnect. Second, a new routing architecture was deployed, combining ideas from a 2-dimensional mesh, nearest-neighbor interconnect, and an inverse clustering scheme. Third,

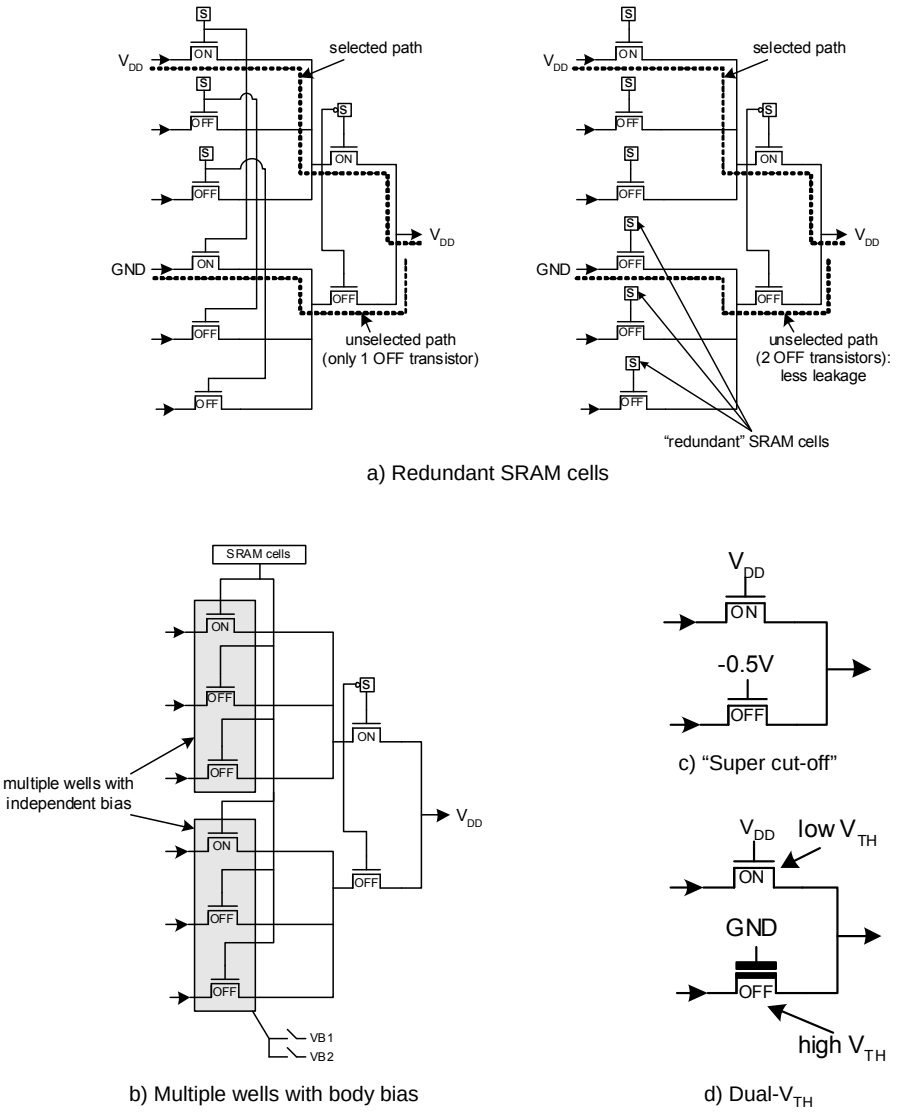


Figure 2.14: Multiplexer leakage reduction techniques [Rahm 04].

specialized transmitter and receiver circuitry were incorporated into each logic block, allowing low-swing signaling to be used. Last, double-edge-triggered flip-flops were used in the logic blocks, allowing the clock frequency to be halved, reducing clock power. The main limitations of the work are: 1) The proposed architecture represents a "point solution" in that the effect of the architectural changes on the area-efficiency, performance, and routability of real circuits was not considered; 2) The basis of the architecture is the Xilinx XC4000, which was introduced in the late 1980s and differs considerably from current FPGAs; 3) The focus was primarily on dynamic power – leakage was not a major consideration.

Power trade-offs at the architectural level were considered in [Li 03], which examined the effect of routing architecture, LUT size, and cluster size (the number of LUTs in a logic block) on FPGA power-efficiency. Using the metric of power-delay product, [Li 03] suggests that 4-input LUTs are the most power-efficient, and that logic blocks should contain 12 4-LUTs. A similar study by Poon and Wilton found that 3-LUTs are most energy-efficient, and that clusters containing 9 or 10 LUTs should used [Poon 02a]. In both studies, despite their focus on power, power-aware CAD tools were not used in the architectural evaluation experiments, possibly affecting the architectural conclusions. Also, as in the UC Berkeley work [Geor 01], the architectures evaluated are somewhat out-of-step with current commercial FPGAs. For example, [Li 03] suggests that a mix of buffered and unbuffered bidirectional routing switches should be used. Modern commercial FPGAs no longer use unbuffered routing switches; rather, they employ unidirectional buffered switches, like that in Figure 2.10.

Dynamic power in CMOS circuits, computed through Equation (2.7), depends quadratically on supply voltage. The quadratic dependence can be leveraged for power optimization, and this property has led to the development of dual or multi-V_{DD} techniques, which have proved themselves effective at power reduction in the ASIC domain (e.g., [Nguy 03, Sriv 04a]). In a dual-V_{DD} IC, circuitry that is not delay-critical is powered by the lower supply voltage; delay-critical circuitry is powered by the higher supply. Level converters are generally needed when circuitry operating at the low supply drives circuitry operating at the high supply. In [Li 04c],

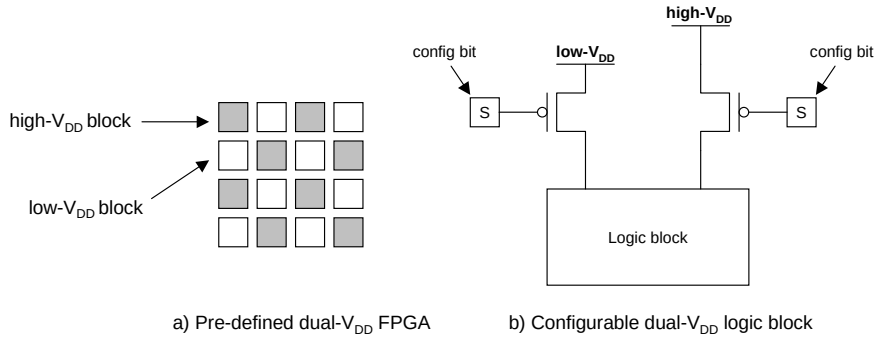


Figure 2.15: Dual-V_{DD} FPGA structures.

the dual-V_{DD} concept is applied to FPGAs. A heterogeneous architecture is proposed in which some logic blocks are fixed to operate at high-V_{DD} (high speed) and some are fixed to operate at low-V_{DD} (low-power, but slower). Figure 2.15(a) illustrates one of the pre-defined dual-V_{DD} fabrics studied in [Li 04c]. The power benefits of the heterogeneous fabric were found to be minimal, due chiefly to the rigidity of the fixed fabric and the performance penalty associated with mandatory use of low-V_{DD} in certain cases. In [Li 04b], the same authors extended their dual-V_{DD} FPGA work to allow logic blocks to operate at *either* high or low-V_{DD}, as shown in Figure 2.15(b). Using such a "configurable" dual-V_{DD} scheme, power reductions of 9-14% (versus single-V_{DD} FPGAs) were reported. A limitation of [Li 04c] and [Li 04b] is that the dual-V_{DD} concepts were applied only to logic, not interconnect. The interconnect, where most power is consumed, was assumed to always operate at high-V_{DD}. This limitation is overcome in [Gaya 04a] and [Li 04a], which apply dual-V_{DD} to *both* logic and interconnect.

Note that a dual-V_{DD} FPGA presents a more complex problem to FPGA CAD tools. CAD tools must select specific LUTs to operate at each supply voltage, and then assign these LUTs to logic blocks with the appropriate supply. To address these issues, algorithms for dual-V_{DD} mapping and clustering have been developed in conjunction with the architecture work mentioned above [Chen 04b, Chen 04a].

2.5.3 Dynamic Power Optimization (CAD Techniques)

Figure 2.16 shows the typical FPGA CAD flow, comprised of HDL synthesis, technology mapping, clustering, placement, and routing. In the HDL synthesis step, an input design is synthesized from a text description, typically VHDL or Verilog, into a circuit netlist, composed of generic primitive elements from a target library. The library may consist of standard logic gates (for example, AND, OR, NOT) or it may contain FPGA-specific elements (for example, LUTs). In technology mapping, the synthesized circuit is transformed into elements that resemble those available in the target FPGA device, primarily LUTs and flip-flops. As mentioned above, logic blocks in commercial FPGAs contain multiple LUTs, flip-flops, as well as arithmetic and other circuitry. A clustering or packing step is invoked after technology mapping to group LUTs and flip-flops into clusters corresponding to the logic blocks of the target FPGA. Placement assigns the logic blocks in the design to logic block sites on the FPGA. Routing forms the desired connections between logic blocks. Finally, the routed design is translated into a configuration bitstream for programming the device.

The potential for power optimization has been studied at each stage of the flow in Figure 2.16. We briefly describe some of the published approaches below.

Front-end Synthesis

A power-aware HDL synthesis system for FPGAs was recently described in [Chen 03]. The system leverages three observations that are unique to FPGAs: 1) datapath circuits often contain large multiplexers, and implementing multiplexers in FPGAs imposes a substantial demand on LUTs and interconnect; 2) FPGAs contain a large number of registers – typically one register per LUT; 3) interconnect accounts for the bulk of FPGA power consumption. The proposed synthesis algorithm aims to reduce interconnect usage by trading-off the number of multiplexer ports with register count. Through a 9% increase in register count, the number of multiplexer ports is reduced by 23%, significantly reducing demand on interconnect. Considerable power reductions of more than 30% are reported, in comparison with a commercial synthesis

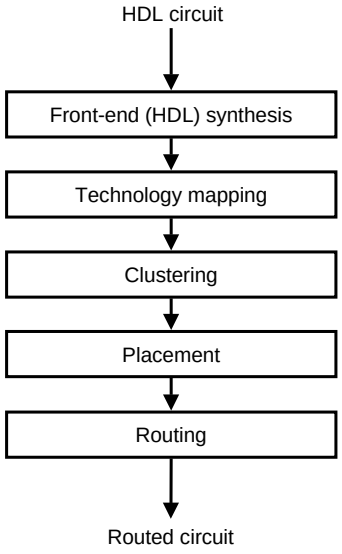


Figure 2.16: Typical FPGA CAD flow.

tool [Chen 03]. An extension of the work appears in [Chen 04a], which proposes a new approach to register binding and port assignment that further reduces the number of multiplexer inputs, realizing additional power reductions.

A recent paper by Wilton, Luk, and Ang examines the impact of pipelining on FPGA power consumption [Wilt 04]. The authors argue that pipelining is essentially "free" in FPGAs, due to the large number of available registers. Since pipelining shortens combinational paths, it reduces glitches and has been successfully applied for glitch reduction in the ASIC domain [Mont 93]. In [Wilt 04], the number of pipeline stages is increased gradually for each circuit in a set of benchmark circuits. Power reductions ranging from 40-90% are reported.

Technology Mapping

Several researchers have considered power optimization during the technology mapping step of Figure 2.16 [Farr 94, Wang 97, Li 01, Wang 01]. The key idea in power-aware technology mapping is to keep signals with high switching activity out of the power-hungry FPGA inter-

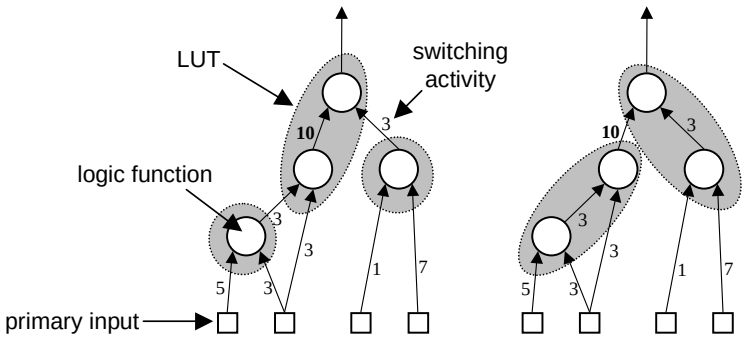


Figure 2.17: Power-aware technology mapping.

connect, as illustrated in Figure 2.17. The figure shows two mapping solutions for a circuit. The nodes represent logic functions; the shaded regions represent LUTs. Switching activity values are shown adjacent to each signal. In the left mapping solution, the high activity signal (with activity 10) is covered *within* a LUT, and therefore, this signal is not required to be routed through the interconnect. In the right mapping solution, the high activity signal is *between* LUTs and thus, must be routed through the interconnect. It is conceivable that the two mapping solutions have considerably different power characteristics.

In an early work, Farrahi and Sarrafzadeh proposed an algorithm that minimized power at the expense of both area and depth [Farr 94]. Their algorithm provides a 14% power improvement over an algorithm that solely optimizes area. Li, Mak, and Katkoori presented an algorithm that optimizes power in the portions of a circuit that are not depth-critical [Li 01]. Their approach reduces the mapping problem to a network flow formulation, similar to FlowMap [Cong 94a]. The authors use a novel approach to translate power objectives into edge capacities in the flow network. In [Wang 01], the authors focused on optimizing both area and power. Their method computes a set of candidate mapping solutions for each node in an input network, and applies a cost function to select the best solution. The approach yields mapping solutions that use 14% less power than those produced by [Farr 94], while at the same time requiring fewer LUTs.

Each published technology mapping algorithm has been shown to produce mapping solutions

requiring less power than solutions produced by a previously published approach. However, none of these works has studied the trade-offs between power and other design criteria, such as area or routability. In Chapter 5, a new power-aware technology mapping algorithm is described that permits trade-offs between power and depth.

Clustering

An interconnect-centric clustering algorithm was proposed in [Sing 02], and shown to offer significant power benefits. The algorithm uses Rent's rule [Dona 81] to create clusters with low interconnect requirements, leading to lower overall power consumption. Previous work on clustering focused on minimizing the total number of clusters [Betz 97a]. [Sing 02] showed that this objective does not necessarily equate with minimizing the number of signals external to clusters – signals that ultimately impose demand on the interconnect. The proposed algorithm uses Rent's rule to derive a preferred upper bound on the number of pins that should be used on a cluster. An attempt is made to honor this bound during clustering, resulting in a slight increase in the average number of clusters needed to implement circuits versus [Betz 97a]. However, the number of signals external to clusters is considerably reduced by this approach, leading to lower overall power.

Placement and Routing

Power optimization has also been considered at the placement and routing stages. [Roy 99] proposed a simulated annealing-based placer that simultaneously handles wirelength, delay, and power minimization. Power was taken into account through an extra term in the annealing cost function. The term tallies the product of estimated capacitance and switching activity for each signal, representing the dynamic power dissipation of the current placement. In a similar way, [Roy 99] realizes power-aware routing by augmenting the router's cost function with a power cost term. A different power-aware placement approach, based on recursive partitioning, was described in [Toga 98]. The method aims to avoid splitting high activity signals across

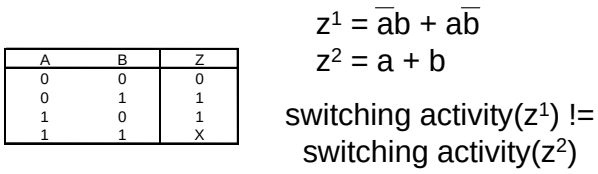


Figure 2.18: Basis of post-layout power optimization.

partitions, leading to shorter wirelengths, smaller capacitances, and lower dynamic power for such signals.

Post-layout Power Optimization

Post-layout power optimization is possible by way of techniques that re-program LUT functions to optimize the switching activity of signals routed through the interconnect [Kumt 00a, Hwan 98]. The aim is to exploit the concept of "don't cares" in Boolean functions. Specifically, don't cares give rise to alternatives in the choice of a function to be implemented by a LUT. Such alternatives may have different switching activities, and thus, offer different power characteristics. Figure 2.18 shows an example truth table with a "don't care", leading to two Boolean function alternatives, each having a different switching activity. In [Kumt 00a], a cluster of LUTs to re-program is selected, and a neighborhood of LUTs that connect to the cluster is identified. The cluster and its neighborhood represent a sub-circuit, H, of the entire circuit, having a specific set of inputs and outputs. Implementation alternatives for the cluster LUTs that meet the functional requirements on the inputs and outputs of H are enumerated, and the implementation having the best power characteristics is selected. An average power reduction of 11.5% was reported [Kumt 00a]. Since the circuit is optimized "in-place", without perturbing the placement or routing, the optimization is "free" from the area and performance perspectives.

Power-Aware CAD Flow

The research work noted above demonstrates that power optimization is possible at each step of the FPGA CAD flow. A question that arises, however, is whether the power reductions achieved at each stage are *additive*, or whether, for example, a power reduction achieved in technology mapping lessens the potential for power reduction at a later stage, such as placement. This question was addressed in an interesting paper by Lamoureux and Wilton, who constructed a complete power-aware FPGA CAD flow, comprised of technology mapping, clustering, placement, and routing [Lamo 03]. Applied in isolation, the power-aware algorithms produced power reductions of 7.6%, 12.6%, 3%, and 2.6%, respectively. When all of the algorithms were applied simultaneously, a total power reduction of 22.6% was observed – only slightly less than the sum of the individual power reductions (25.8%). The results suggest that the power reductions achieved at each stage are largely orthogonal.

2.6 Summary

Power dissipation in CMOS circuits can be classified as either dynamic or static. Dynamic power consumption is due to the logic transitions that occur on a circuit's signals during normal operation. Static power is dissipated even when a circuit is in an idle state. Historically, dynamic power has dominated power consumption in CMOS circuits; however, technology scaling trends have resulted in leakage becoming an increasing component of total power. The breakdown of power consumption in FPGAs is well-studied, and it has been observed that interconnect accounts for the bulk of an FPGA's static and dynamic power. In recent years, various approaches to reducing FPGA power have been proposed in the literature, including approaches for leakage reduction, as well as circuit, architecture, and CAD techniques for dynamic power reduction.

3 CAD Techniques for Leakage Optimization

3.1 Introduction

This chapter is focussed on optimizing the *active* leakage power dissipation of FPGAs. Recall that active leakage refers to leakage consumed in the *used and operating* part of an FPGA (see Section 2.2.2). Two novel CAD techniques for reducing active leakage are presented. Both of the proposed techniques are unique in that they are "no cost", meaning that they do not worsen area-efficiency, degrade circuit performance, or increase fabrication cost.

The leakage consumed by a digital CMOS circuit depends on its input state. Section 3.2 explores the extent to which this property holds for common FPGA hardware structures. Based on the results observed, Section 3.3 presents a first leakage reduction approach that leverages a specific property of the primary logic elements used in FPGAs, namely, LUTs. The property allows one to freely use either polarity of a logic signal, without incurring any area or delay penalty, and without requiring modifications to the underlying FPGA hardware. Polarities are chosen for signals in a way that places hardware structures into their low leakage states.

The second leakage optimization technique, described in Section 3.4, takes the leakage power consumptions of FPGA routing resources into account during the routing step of the CAD flow. The objective of such "leakage-aware routing" is to route design signals with a preference for using low leakage routing resources. Both of the proposed leakage reduction approaches are validated experimentally by applying them to optimize leakage in a 90nm Xilinx commercial FPGA. The chapter concludes with a summary in Section 3.5.

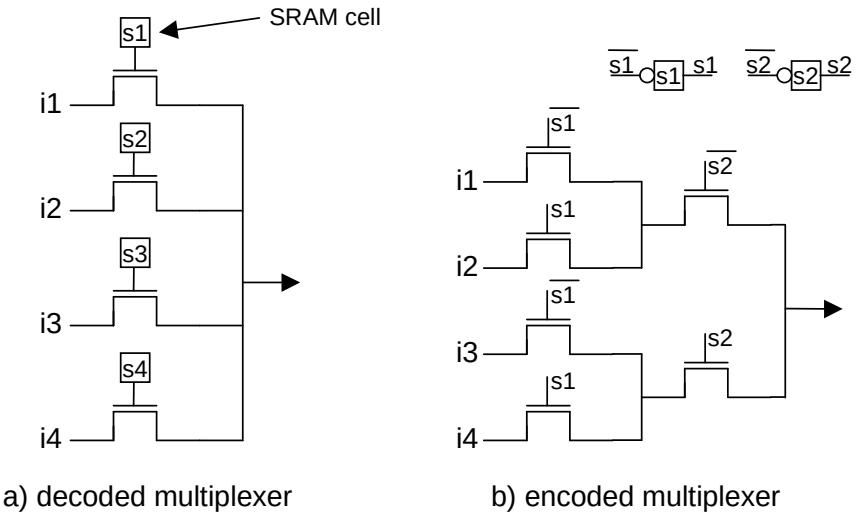


Figure 3.1: Two 4-to-1 multiplexer implementations.

3.2 FPGA Hardware Structures

Multiplexers and buffers comprise the basic building blocks of an FPGA, since they are deployed throughout FPGA logic and interconnect. Before describing the proposed leakage reduction methods, we study the leakage characteristics of such basic circuit blocks.

Figure 3.1, repeated here for convenience, shows two implementation alternatives for a 4-to-1 multiplexer, one "decoded" and the other "encoded". The trade-offs between these two designs were outlined in Section 2.3. To evaluate the dependence of multiplexer leakage on input state, SPICE simulations of both designs were performed. The simulations were conducted at 110°C using BSIM4 SPICE models for a 1.2V 90nm commercial CMOS process[1]. The select signals of the multiplexers were assigned values such that input $i1$ was selected and passed to the multiplexer output. All 16 possible input vectors were simulated.

Figure 3.2 shows the multiplexer leakage power results. A vertical bar illustrates the leakage for each input vector. Observe in Figure 3.2 that leakage power in the multiplexers is highly

[1]BSIM4 refers to the industry standard MOSFET SPICE simulation model, developed at UC Berkeley [BSIM4 04]. BSIM4 represents the state-of-the-art model. A significant enhancement of BSIM4 over BSIM3 models is the proper modeling of gate oxide leakage.

dependent on input state. For the decoded multiplexer, the highest leakage occurs when logic-0 appears on input $i1$ (the input whose signal is passed to the output) and logic-1 appears on all other inputs; the lowest leakage occurs when all inputs are logic-1. For the decoded multiplexer, there is a 13.7X difference in leakage power between the highest and lowest leakage states; for the encoded multiplexer, the leakage power difference is 14.2X. In addition to the leakage for each input vector, Figure 3.2 shows the average leakage power consumed when the output of the multiplexer is a logic-1 (solid horizontal line) and when the output of the multiplexer is a logic-0 (dashed horizontal line).

Observe that, for both multiplexers in Figure 3.1, the average leakage for passing a logic-1 to the multiplexer output is substantially smaller than the average leakage for passing logic-0. There are several reasons for this. First, when logic-1 (V_{DD}) is applied to the drain terminal of an ON NMOS device, a "weak-1" ($\approx V_{DD} - V_{TH}$) appears at the source terminal. The weak-1 leads to reduced subthreshold leakage power in other multiplexer transistors that are OFF, versus when the potential difference across an OFF transistor is V_{DD} [see Figure 3.3(a)]. This is partly due to the effect of drain-induced barrier lowering (DIBL) in short-channel transistors, which causes threshold voltage to increase (subthreshold current to decrease) when drain bias is decreased [Roy 03].

In addition to affecting subthreshold leakage, another reason for the input-dependence of the multiplexer leakage power is the reduction in gate oxide leakage when the multiplexer is passing logic-1 to its output. Gate leakage is a considerable fraction of total leakage in $90n$m technology. Gate leakage in an ON NMOS transistor depends significantly on the applied bias [Guin 03]. When an NMOS transistor is passing logic-0, the voltage difference between the gate and source is V_{DD} (that is, $V_{GS} = V_{DD}$) and the transistor is in the *strong inversion* state [see Figure 3.3(b)]. Conversely, when the transistor is passing logic-1, the transistor is in the *threshold* state ($V_{GS} \approx V_{TH}$) [see Figure 3.3(c)]. Gate oxide leakage in the threshold state is typically several orders of magnitude smaller than in the strong inversion state [Guin 03]. This property makes it preferable to pass logic-1 (versus logic-0) from the gate leakage perspective.

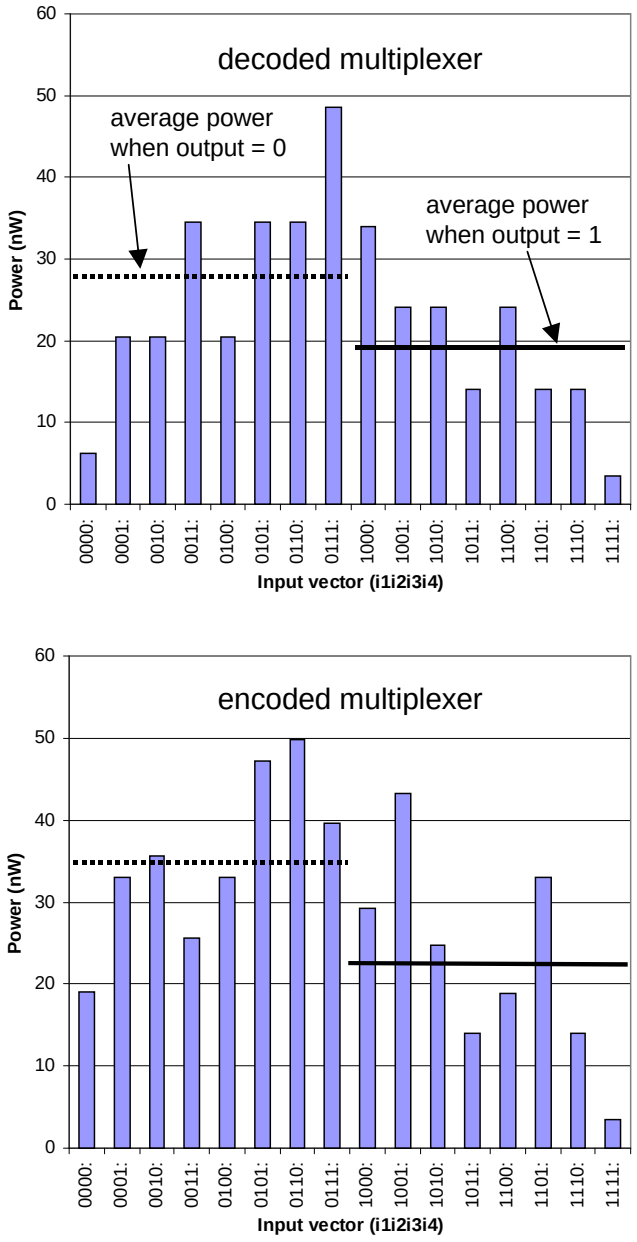
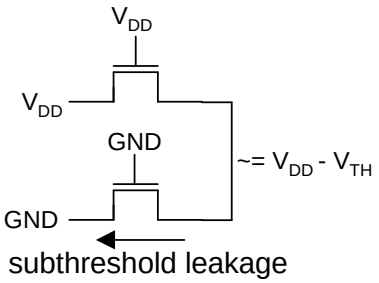
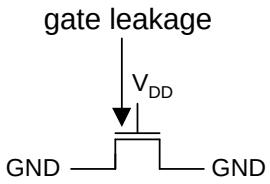


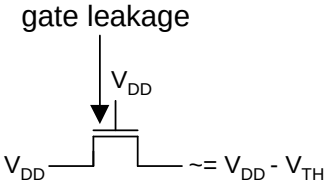
Figure 3.2: Leakage power for multiplexers.



a) reduced subthreshold leakage

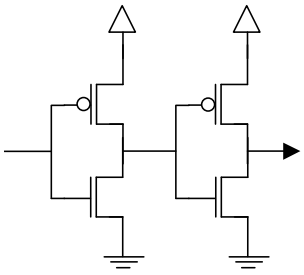


b) high gate leakage



c) low gate leakage

Figure 3.3: Examples of transistor leakage states.



Input	Power (nW)
0	56.1
1	46.6

Figure 3.4: Buffer implementation and leakage power.

Another important circuit element in FPGAs is a buffer, since they are present throughout the routing fabric and also within logic blocks. The two stage buffer shown in Figure 3.4 was simulated and its leakage power in both input states was evaluated. The buffer's transistors were sized to achieve equal rise and fall times; the second stage was chosen to be three times larger than the first stage. Leakage power results for the buffer are shown on the right side of Figure 3.4. Although the difference in power between the two input states is not as pronounced as the differences observed for the multiplexers, one can see that about 20% more power is consumed when the buffer's input is a logic-0 versus when its input is a logic-1. The dependence of the buffer's leakage on input state is a result of NMOS and PMOS devices having considerably different subthreshold and gate oxide leakage characteristics, and also due to the dependence of leakage on transistor size. For example, gate oxide leakage is considerably higher in NMOS versus PMOS transistors [Yu 00] and is also directly proportional to transistor size. Therefore, overall gate leakage is minimized when the large NMOS transistor in the buffer's second inverter stage is OFF, which occurs when the buffer's output state is logic-1.

Subthreshold leakage increases exponentially with temperature, and consequently, leakage is primarily a problem at high temperature. This work concerns *active* leakage power in the operating (hot) part of the FPGA and therefore, in this chapter, the proposed leakage reduction techniques are evaluated at high temperature (110°C). Unlike subthreshold leakage, gate oxide leakage is almost insensitive to temperature [Agar 04], making it a larger fraction of total leakage

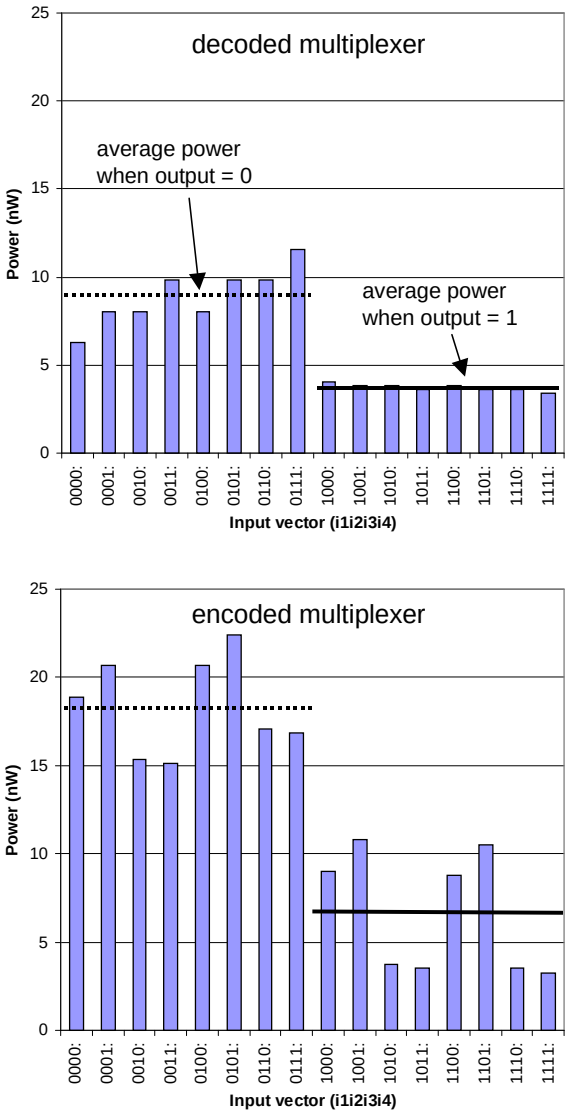
at low temperature. For completeness, the leakage characteristics of the basic FPGA hardware structures were examined at low temperature (40°C). The results are shown in Figure 3.5. Observe that similar leakage bias trends are apparent at low temperature; namely, less leakage in consumed when logic-1 is passed through the multiplexers and buffer versus when logic-0 is passed through these structures. In fact, in the multiplexers, the bias is more pronounced at low temperature. For example, in the decoded multiplexer, the average leakage power when the output is logic-0 is 140% higher than when the output is logic-1. At high temperature, the leakage difference between the two states is only 44%. In the buffer, the same bias is present at low temperature, but it is less pronounced; buffer leakage in the logic-0 state is about 7% higher than in the logic-1 state (versus 20% higher at high temperature).

3.3 Active Leakage Power Optimization via Polarity Selection

In Section 3.2, we observed that in a modern commercial CMOS process, the leakage power dissipated by elementary FPGA hardware structures, namely buffers and multiplexers, is typically smaller when the output and input of these structures is logic-1 versus logic-0. The first approach to active leakage power optimization approach works by choosing a polarity for each signal in an FPGA design, in a manner that enables signals to spend the majority of their time in the logic-1 state (the logic state associated with low leakage power). A fundamental property of a digital signal is its *static probability*, which is the fraction of time a signal spends in the logic-1 state. A signal with static probability greater than 0.5 spends more than 50% of its time at logic-1. The leakage reduction approach alters signal polarity to achieve high static probability for most signals. Unlike in ASICs, signal polarity inversion in FPGAs can be achieved without any area or delay penalty, by leveraging a unique property of the basic FPGA logic element[2].

Figure 3.6 illustrates how a signal's polarity can be reversed in an FPGA. Part (a) of

[2]Changing a signal's polarity implies a change in the *direction* of the signal's logic transitions. The "no cost" property of the polarity selection optimization assumes that FPGA performance is not tied to the direction of logic transitions. This assumption holds for today's commercial FPGAs from Altera and Xilinx.



a) Leakage power for multiplexers

Input	Power (nW)
0	19.3
1	18

b) Leakage power for buffer

Figure 3.5: Low temperature leakage power results for multiplexers and buffer (40°C).

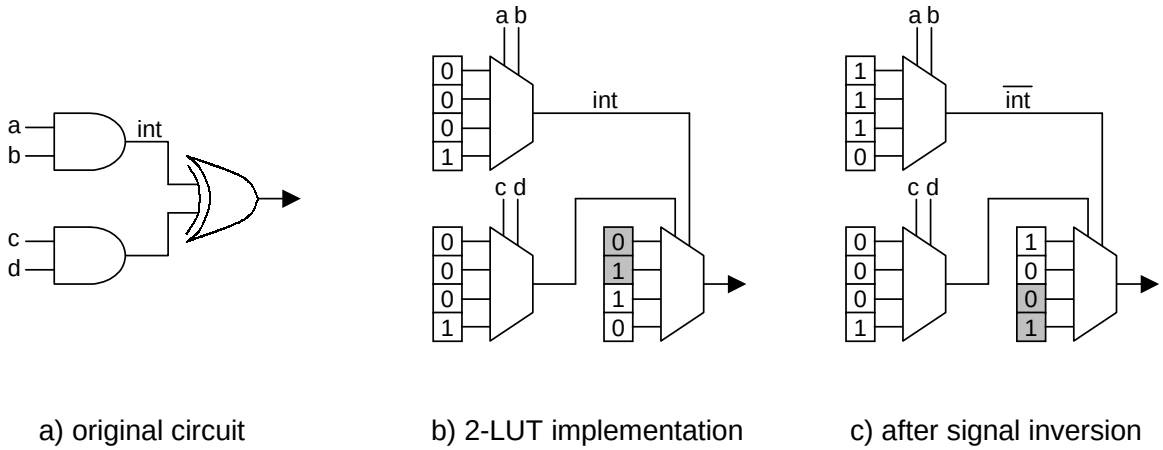


Figure 3.6: LUT circuit implementation; illustration of signal inversion.

the figure shows a logic circuit having two AND gates and an exclusive-OR gate. Part (b) of the figure shows the circuit mapped into 2-input LUTs. The memory contents are shown for each LUT and represent the truth table of the logic function implemented by the LUT's corresponding gate. In this example, the aim is to invert the signal *int*, so that its complemented rather than its true form is produced by a LUT and routed through the FPGA interconnection network. There are two steps to inverting a signal. First, the programming of the LUT producing the signal must be changed. Specifically, to invert the signal, all of the 0s in its driving LUT must be changed to 1s and the 1s must be changed to 0s. Second, the programming of LUTs that are fanouts of the inverted signal must be altered to "expect" the inverted form. This is achieved by permuting the bits in the SRAM cells of such "downstream" LUTs. Part (c) of Figure 3.6 shows the circuit after the signal *int* is inverted. The permutation of bits in the inverted signal's fanout LUT is shown through shading; the contents of the top two SRAM cells in the downstream LUT are interchanged with the contents of the bottom two SRAM cells in the LUT. Through this method, signal inversion in FPGAs can be achieved by simply re-programming LUTs.

```
function OptimizeLeakage(design, signal static probabilities)
    for each signal n in the design do
        if static_probability(n) < 0.5 then
            if signal n can be inverted then
                invert(n)
                // FPGA is re-programmed; n replaced with n̄
    return new design
```

Figure 3.7: Leakage optimization algorithm.

The approach to leakage power optimization is shown in Figure 3.7. The input to the algorithm is an FPGA circuit, as well as static probability values for each signal in the circuit. A loop iterates through the signals and selects those signals having static probability less than 0.5. Such signals spend most of their time in the logic-0 state, and thus, they are candidates for inversion. For each candidate signal, a check is made to determine if it can be inverted (discussed below). If a candidate signal is invertible, it is inverted by re-programming the FPGA configuration memory accordingly. After processing all signals, the output of the algorithm is a modified design, having signals that spend the majority of their time in the logic state favourable to low leakage power.

The majority of signals in FPGA designs are produced by LUTs and drive LUTs, and all such signals can be inverted using the approach shown in Figure 3.6. In a commercial FPGA however, in addition to LUTs, other types of hardware structures are usually present. Some signals driven by or driving non-LUT structures may also be invertible, since FPGA vendors frequently include extra circuitry for programmable inversion. However, some signals may not be invertible, such as those driving special control circuitry, entering the FPGA device from off-chip, or driving certain pins on non-LUT structures. As a concrete example, consider that the Xilinx Virtex-II PRO FPGA contains 18-by-18 block multipliers [Virt 03]. The inputs to

the multipliers do not have programmable inversion. Therefore, any signal feeding a multiplier input *should not* be inverted by the proposed polarity selection approach, as doing so would be functionally incorrect (it would change the multiplication results). Similarly, Virtex-II contains large blocks of static RAM memory. Inverting a signal that drives a block RAM address input is not straightforward, as it implies a shuffling of memory contents, and block RAM memory contents is frequently pre-loaded during an FPGA's initial configuration phase. A two-pass approach would be needed to invert block RAM address signals: First, the polarity selection optimization would be executed, permitting block RAM address signal inversion. Then, the polarity selection results would be used to determine the appropriate rearrangement of block RAM memory contents. The memory contents would be shuffled appropriately, prior to FPGA configuration.

Altering the polarity of a signal n with static probability $P(n)$, changes the signal's probability to $1 - P(n)$. Therefore, for signals having static probability close to 0.5, the benefits of inversion on leakage optimization are minimal, since the static probability of such signals remains close to 0.5 after inversion. Low leakage power can be achieved when signals have static probability close to 0 or 1. A question that arises, then, is whether the signals in real circuits exhibit this property. Below, we show that it is unlikely that the majority of signals in circuits will have probabilities close to 0.5, which bodes well for the proposed leakage optimization approach.

The average number of logic transitions per clock cycle on a (non-clock) signal n, $F(n)$, can be expressed as a function of the signal's static probability [Ciri 87, Yeap 98]:

$$F(n) = 2 \cdot P(n) \cdot [1 - P(n)] \tag{3.1}$$

Recall that $F(n)$ is referred to as signal n's *normalized switching activity*. Since $P(n)$ ranges from 0 to 1, $F(n)$ in (3.1) ranges from 0 to 0.5. Note that (3.1) is a frequently used approximation that becomes exact in the absence of temporal correlations in signal n's switching activity. That is, (3.1) assumes that n's values in two consecutive clock cycles are independent.

Solving (3.1) for $P(n)$, yields:

$$P(n) = \frac{1 \pm \sqrt{1 - 2 \cdot F(n)}}{2} \tag{3.2}$$

which is plotted in Figure 3.8. Observe that $P(n)$ is 0.5 only when $F(n)$ is 0.5 and that for a fixed decrease in $F(n)$, there is a change in $P(n)$ towards either 0 or 1. From Figure 3.8, one can infer that if the switching activities of the majority of signals in circuits are not clustered close to 0.5, then the static probabilities of signals will also not be clustered close to 0.5. Switching activity in combinational circuits is well-studied. Prior work by Nemani and Najm found that switching activities are generally not clustered around a single value and that on average, activity decreases quadratically with combinational depth in circuits [Nema 99]. One can therefore expect there to be a range of different static probabilities amongst the signals of a circuit and that "deeper" signals in circuits will have static probabilities approaching either 0 or 1. This analysis suggests that for many signals, changing polarity will have a significant impact on leakage power.

Often, design verification is done at the post-routing stage, through HDL simulation with extracted routing delays, or by using logic analyzers, such as Xilinx ChipScope [XilinxCh 04]. ChipScope is an IP core that is inserted within a user's FPGA design. It allows one to view the states of the design's internal signals during execution on the FPGA hardware. The presence of the polarity selection optimization within the CAD flow may complicate such verification. Specifically, the inversion of some signals may make verification results unintelligible to users. This is analogous to the effect that compiler optimizations have on one's ability to run a debugger on optimized code in software development. In this case, however, a simple work-around is available: the verification tools can be made "aware" of which signals have been inverted, and can present the "true" form of signals to users, as appropriate.

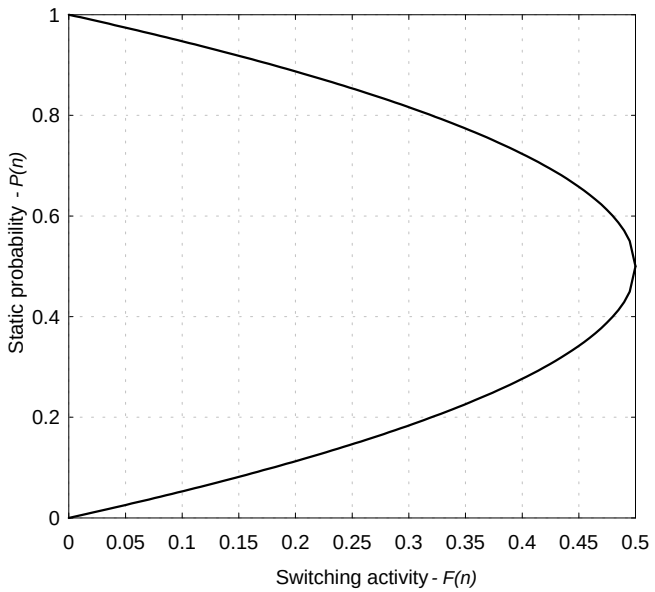


Figure 3.8: Static probability versus switching activity.

3.3.1 Experimental Study and Results

The effectiveness of the proposed leakage power reduction approach was gauged by applying it to optimize active leakage in the Xilinx Spartan-3 1.2V 90nm commercial FPGA. The breakdown of leakage in Spartan-3 has recently been studied by other authors [Tuan 03]. This section describes the experimental methodology and subsequently, provides results.

Methodology

The target FPGA is composed of an array of configurable logic block (CLB) tiles, I/Os, and other special-purpose blocks, such as multipliers and block RAMs. Smaller versions of the FPGA contain only the CLB array and I/Os. An embedded version of the FPGA, containing the CLB array only, is also available for incorporation into custom ASICs. In this chapter, the focus is on leakage optimization within the FPGA's CLB array, which represents the bulk of the FPGA's silicon area, especially in smaller devices and the embedded version. The non-CLB blocks (e.g., block RAMs) are not unique to FPGAs; leakage optimization in these blocks has

Table 3.1: Major circuit blocks in target FPGA.

Circuit block	**Details**
IMUX	30-to-1 multiplexer, buffer
DIRECT	24-to-1 multiplexer, buffer
DOUBLE	16-to-1 multiplexer, buffer
HEX	12-to-1 multiplexer, buffer
LONG	n-to-1 multiplexer, buffer (n device/orientation dependent)
LUT	16-to-1 multiplexer, in/out buffers
FLIP-FLOP	programmable set/reset

been studied in other contexts.

A Spartan-3 CLB tile contains both logic and routing resources and is similar to a Virtex-4 CLB, shown in Figure 2.9. The CLB's logic resources consist of four logic sub-blocks, called SLICEs. Each SLICE contains two 4-LUTs, two flip-flops, as well as arithmetic and other circuitry. Like Virtex-4, the Spartan-3 interconnect consists of variable length wire segments that connect to one another through programmable, buffered switches similar to that shown in Figure 2.10. Table 3.1 provides further detail on the major circuit blocks in the Spartan-3 CLB tile, including the widths of the interconnect multiplexers. The input multiplexer (IMUX) selects and routes a signal to a SLICE input pin. The DIRECT interconnect block selects and routes a signal from a SLICE output pin to a neighboring logic block. DOUBLE blocks drive wire segments that span 2 CLB tiles. HEX blocks drive wires that span 6 CLB tiles. LONG resources span the entire width or height of the FPGA. Note that a single CLB tile contains multiple instances of each of the blocks listed in Table 3.1.

Figure 3.9 shows the leakage optimization and analysis flow. As mentioned in Section 3.3, the input to the optimization algorithm is an FPGA circuit, as well as the static probability value for each of the circuit's signals. The benchmark circuits consist of 10 large combinational MCNC circuits and 6 industrial circuits collected from Xilinx customers; the circuits are listed

in Table 3.2[3]. The MCNC circuits were first synthesized from VHDL using Synplicity's Synplify Pro tool (version 7.0). Then, the circuits were technology mapped, placed, and routed in the target FPGA using the Xilinx software tools (version M6.2i). The industrial circuits were already available in technology mapped form so only the placement and routing steps were required for these circuits.

An earlier version of this work presented preliminary results for circuits that were not optimized for speed performance [Ande 04f]. In practice, however, most FPGA users seek a high performance design implementation. Consequently, the Xilinx place and route tools were used to generate a performance-optimized layout for each design as follows: First, each design was placed and routed with an easy-to-meet timing (critical path delay) constraint; then, based on the performance achieved, a more aggressive constraint was generated, and the place and route tools were re-executed using the new constraint. The entire process was repeated until a constraint that could not be met by the layout tools was encountered. The proposed leakage reduction technique is evaluated for the layout solution corresponding to the most aggressive, but achievable, constraint observed throughout the entire iterative process.

To gather static probability data, the routed circuits were simulated using either the Synopsys VHDL System Simulator (VSS) (version 2000.06) or Mentor Graphics' ModelSIM (version 5.7a). The simulators have built-in capabilities for capturing the fraction of time a signal spends at logic-1 (static probability). Representative simulation vectors for the circuits were not available, and therefore, the circuits were simulated using 10,000 randomly chosen input vectors. Clock and control inputs in the industrial circuits were identified by examining load pin types and by inferring functionality based on signal names. Such clock/control inputs were presented with appropriate, non-random signals in the simulations. For the control inputs, a trial-and-error approach was used to select between applying an "active high" or an "active low" input waveform. Note that the 6 industrial circuits were selected from an initial, larger pool of circuits, from which the balance had to be eliminated due to a lack of knowledge re-

[3]The industrial circuits are single clock designs. Multi-clock designs were not used, since, for such designs, the relationship between clock signals is difficult to determine without access to user-provided simulation vectors.

Table 3.2: Characteristics of benchmark circuits.

Circuit	LUTs	FFs
alu4	500	0
apex4	1078	0
cps	524	0
dalu	323	0
ex1010	1112	0
ex5p	557	0
misex3	257	0
pdc	609	0
seq	1193	0
spla	229	0
industry1	1511	2128
industry2	1654	1278
industry3	2818	368
industry4	2942	1262
industry5	8676	5507
industry6	4895	318

garding their internal functionality. For such eliminated circuits, it was not possible to identify a simulation vector set that realized toggling on a significant fraction of each circuit's internal signals.

In the randomly generated input vectors, the probability of each primary input toggling between successive vectors was set to 50%. Note that, given the static probabilities of a circuit's primary input signals, the static probabilities of the circuit's internal signals can be computed using well-known probabilistic techniques [Yeap 98]. Thus, simulation is not a requirement for the use of the optimization approach, and it is expected that the approach could be incorporated into EDA tools that automatically perform the proposed leakage optimization. Certainly, it seems possible that the static probabilities initially known and supplied to the proposed algorithm may differ from those probabilities actually experienced by the circuit in the field. This potential issue can be addressed easily since the polarity selection optimization does not disrupt placement, routing, or design performance. One can freely re-apply the algorithm when an updated set of static probabilities is available, and the resulting updated bitstream can then be used to re-program an already-deployed FPGA design in the field.

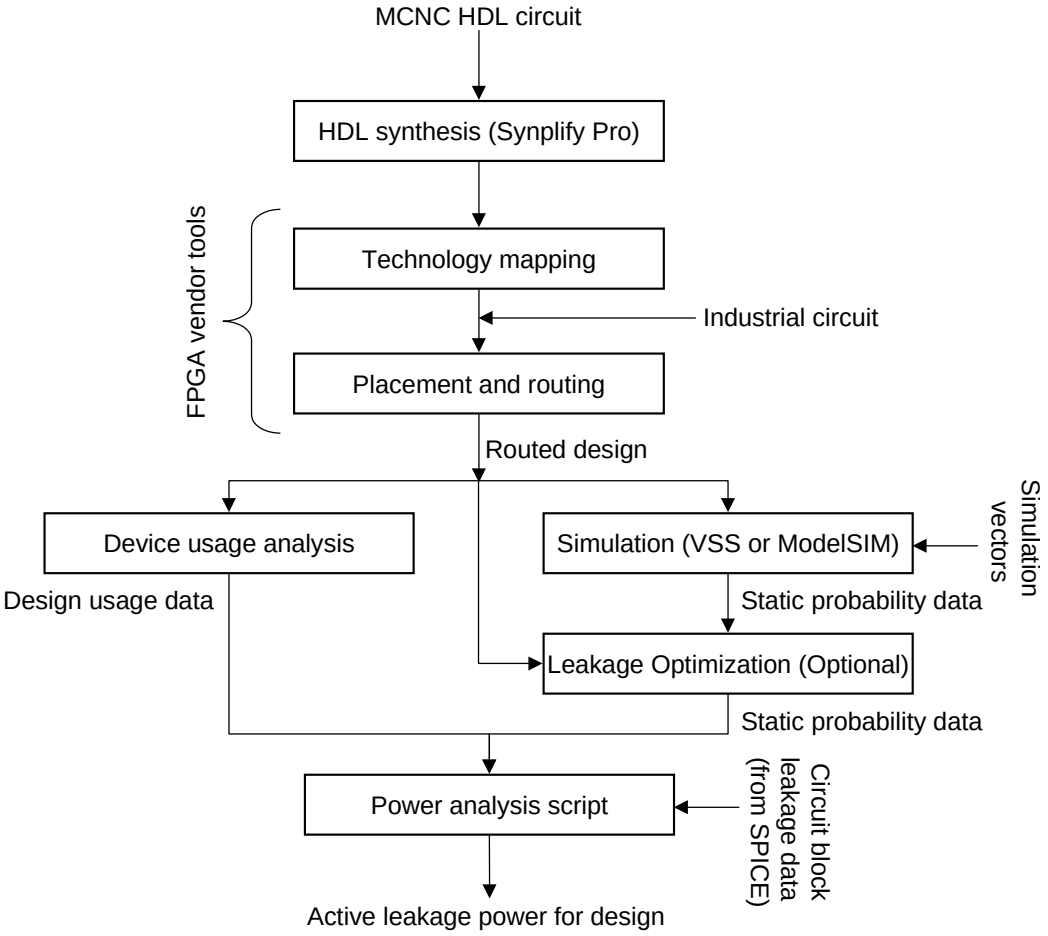


Figure 3.9: Leakage analysis flow.

SPICE simulations were performed for each type of circuit block in the FPGA's CLB tile. The leakage power consumed by each block for each of its possible input vectors was captured. Circuit regularity permitted the blocks with many inputs to be partitioned into sub-blocks, which were then simulated independently. To illustrate, consider a 16-to-1 multiplexer, constructed using four 4-to-1 multiplexer in a "first stage", and a fifth 4-to-1 multiplexer in a "second stage". One need not simulate all 2^{16} input combinations of the 16-to-1 multiplexer to gather accurate leakage data for each of these input combinations. One can simulate the individual 4-to-1 multiplexers and combine their leakage results to produce leakage data for the large 16-to-1 multiplexer. This was the approach taken to gather leakage data for the commercial blocks with many inputs. Notably, we observed the leakage characteristics of the commercial FPGA's circuit blocks to be similar to those of the generic structures studied in Section 3.2.

The total active leakage power, L_{active}, was computed twice for each benchmark circuit, both with and without the proposed optimization technique. L_{active} is defined as the sum of the leakage power in each *used* circuit block. By analyzing the routed FPGA implementation for a benchmark, one can determine its circuit block usage, including the signals on the inputs and outputs of each used circuit block.

Computing the leakage for a *used instance* of a circuit block in a benchmark involves combining the power data extracted from the block's SPICE simulation with usage data from the benchmark circuit's FPGA implementation and static probability data from the benchmark's HDL simulation. It is worth reinforcing that we do not use the power data presented in Section 3.2 in the experimental study; rather, the results here are based on power data extracted from SPICE simulations of the commercial FPGA's circuit blocks.

Consider a used instance B of a circuit block in a benchmark and let $\vec{v}_B$ represent an input vector that may be presented to block B. Each bit i in vector $\vec{v}_B$, $\vec{v}_B(i)$, corresponds to an input i on block B. Let $S_B(i)$ represent the signal on input i of block B in the benchmark's FPGA implementation. The static probability of signal $S_B(i)$, $P(S_B(i))$, is a known quantity,

extracted from the benchmark's HDL simulation. If bit $\vec{v}_B(i)$ is logic-1 in vector $\vec{v}_B$, then we define the static probability of bit $\vec{v}_B(i)$, $P(\vec{v}_B(i))$, to be equal to $P(S_B(i))$. On the other hand, if $\vec{v}_B(i)$ is logic-0 in $\vec{v}_B$, then $P(\vec{v}_B(i))$ is defined to be $1 - P(S_B(i))$. One can compute the probability of vector $\vec{v}_B$ appearing on the inputs of block B, $P(\vec{v}_B)$, as the product of its constituent bit probabilities:

$$P(\vec{v}_B) = \prod_{\vec{v}_B(i) \ \epsilon \ \vec{v}_B} P(\vec{v}_B(i)) \tag{3.3}$$

Note that it is entirely possible that some inputs to a used circuit block may have no signal on them. For example, some inputs to a routing switch (see Figure 2.10) may attach to conductors that are not used in the FPGA implementation of a benchmark circuit. In a commercial FPGA, unused routing conductors are not allowed to "float" to an indeterminate voltage state. In the target Xilinx FPGA, unused routing conductors are pulled up to logic-1. Pulling unused routing conductors into the low leakage, logic-1 state benefits overall leakage in the FPGA, since an FPGA implementation of a benchmark circuit requires only a fraction of the FPGA's routing resources. To demonstrate this, a detailed analysis of a portion of the routing in the *industry4* benchmark was performed. In *industry4*'s routing, we found that there were 15,235 used DOUBLE resources, and 5,918 used HEX resources. On average, 10.4 (of 16) inputs on each DOUBLE resource in *industry4*'s routing attached to unused routing conductors. The remaining 5.6 inputs (on average) attached to routing conductors with an active logic signal on them; that is, 5.6 inputs attached to routing conductors that were used in the routing of *industry4*. Likewise, the HEX resources in *industry4* had 7.6 (of 12) inputs attached to unused routing conductors, on average, with the remaining 4.4 inputs attached to used routing conductors. In other words, considering all HEX and DOUBLE resources used in *industry4*, nearly 2/3 of the inputs to these resources attach to unused routing conductors, and are therefore pulled to logic-1. This amplifies the need for the "prefer logic-1" approach taken in the polarity selection optimization.

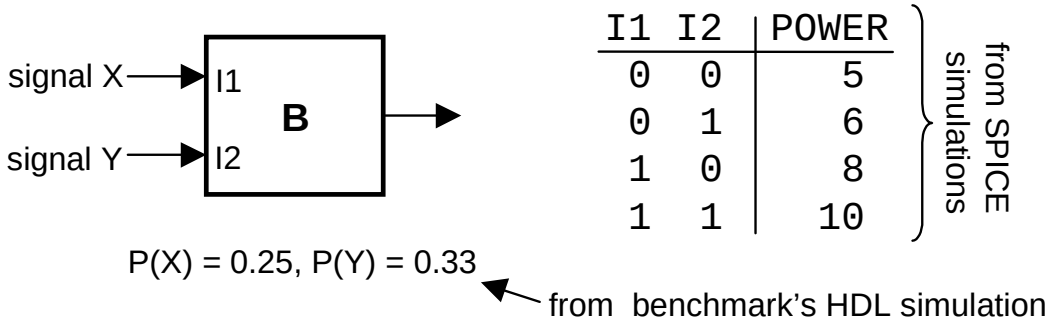


Figure 3.10: Example active leakage power computation.

The average active leakage power for a used circuit block B, $L_{active}(B)$, is computed as a weighted sum of the leakage power consumed by B for each of its input vectors:

$$L_{active}(B) = \sum_{\vec{v}_B \ \epsilon \ V_B} P(\vec{v}_B) \cdot L_{active}(\vec{v}_B) \tag{3.4}$$

where V_B represents the set of all possible input vectors for circuit block B, and $L_{active}(\vec{v}_B)$ represents the leakage power consumed by block B when its input state is vector $\vec{v}_B$, obtained from SPICE simulations.

An example of the leakage power computation approach for a block with two inputs is shown in Figure 3.10. In the example, the signal, X, on block input $I1$ has a static probability of 0.25 and the signal, Y, on input $I2$ has a static probability of 0.33. A table gives the power consumed by the block for each possible input vector. Consider, for example, the vector in which $I1 = 1$ and $I2 = 0$. The leakage power consumed by the block for this vector is 8. The probability of the vector appearing on the inputs of the block is: $P(X) \cdot [1 - P(Y)] = 0.25 \cdot (1 - 0.33) = 0.1675$. Thus, the contribution of this vector to the block's active leakage is $0.1675 \cdot 8 = 1.34$, which is the third term in the equation shown in Figure 3.10.

Leakage power was not a primary design consideration in the target commercial FPGA.

We envision that the proposed active leakage reduction approach will be used in conjunction with a future, leakage-optimized FPGA architecture. Consequently, the experimental results presented here consider only the active leakage power and ignore leakage in the unused parts of the FPGA. Reducing unused leakage can be viewed as a separate optimization problem that can be addressed by either powering down the unused circuit blocks, by applying the standby leakage optimization techniques mentioned in Section 2.2.2, or through circuit techniques that will be introduced in the next chapter. Moreover, the results do not include the leakage in the FPGA's SRAM configuration cells. As observed in Section 2.4.2 (page 27), the contents of such cells changes only during the initial FPGA configuration phase, and thus, their speed performance is not critical. In a future low leakage FPGA, the SRAM configuration cells can be slowed down, and their leakage greatly reduced.

Results

We begin by comparing the active leakage power consumed in the unoptimized circuits with that consumed in the optimized circuits. Figure 3.11 shows the percentage reduction in active leakage power for each circuit. The improvement ranges from 15% to 38%, with the average being 25%. The power benefits observed are quite substantial, considering that the proposed optimization has no impact on circuit area or delay and requires no hardware changes.

Table 3.3 gives the detailed power results for each circuit. Columns 2-4 give power data for the unoptimized circuits. Columns 2 and 3 present the power dissipated in the interconnect and non-interconnect (labeled "other") circuit blocks, respectively. Column 4 presents the total active leakage power for each circuit. Columns 5-7 present analogous data for the optimized circuits. In these columns, percentage improvement values, relative to the unoptimized circuits, are shown in parentheses. From Table 3.3, one can see that the proposed optimization is more effective at reducing leakage in the interconnect versus the non-interconnect circuit blocks. The non-interconnect blocks include LUTs, flip-flops, and other circuitry. A more in-depth analysis revealed that flip-flop leakage power was only slightly dependent on whether the flip-flop was

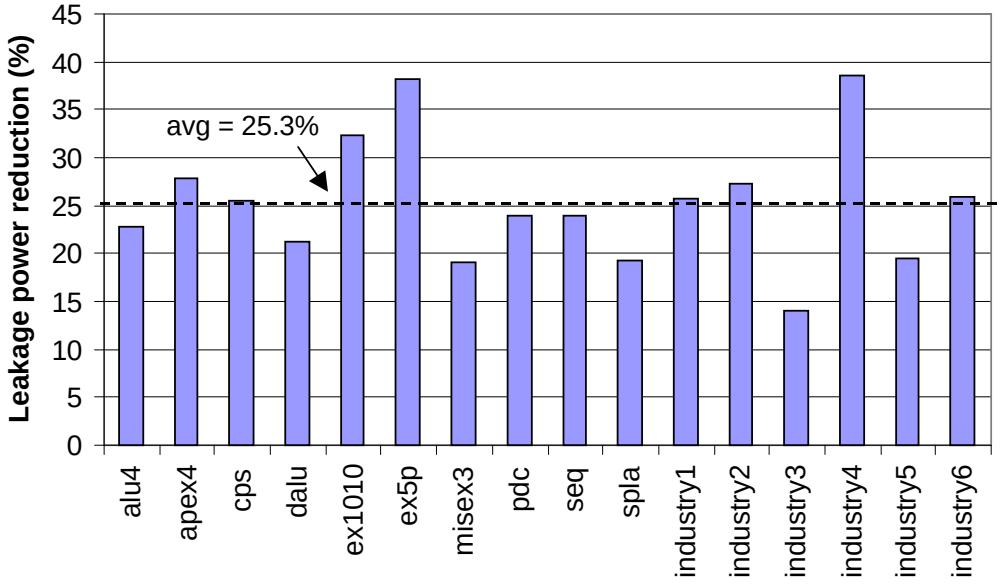


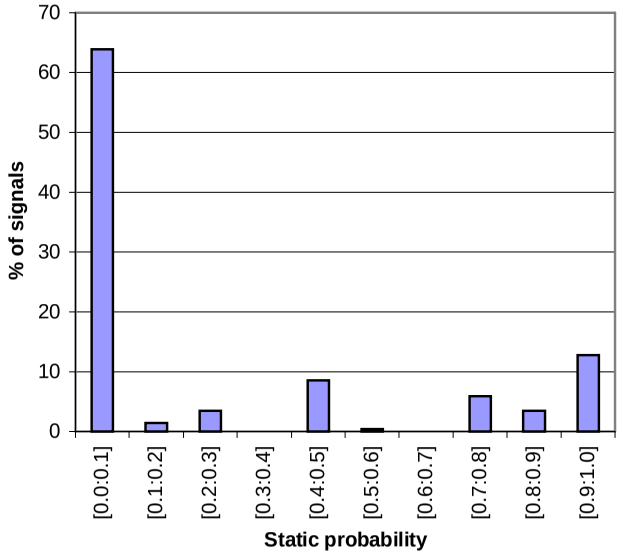
Figure 3.11: Leakage power reduction results.

Table 3.3: Detailed active leakage power results.

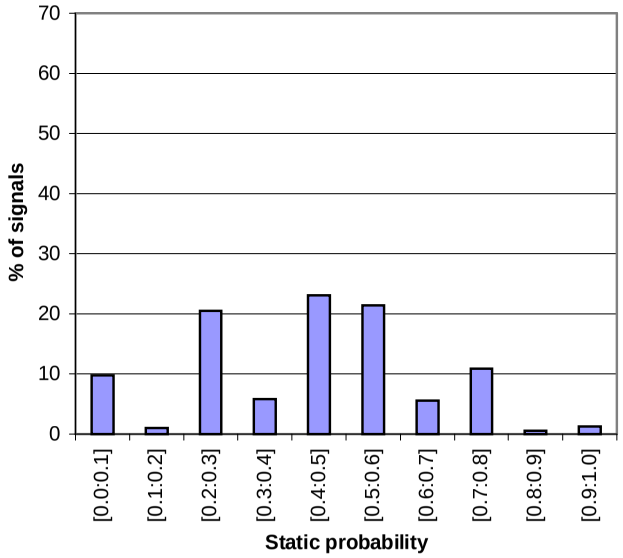
	Unoptimized			**Optimized**		
Circuit	**Interconnect (μW)**	**Other (μW)**	**Total (μW)**	**Interconnect (μW) (%)**	**Other (μW) (%)**	**Total (μW) (%)**
alu4	690	193	883	494 (28.4)	187 (3.1)	681 (22.8)
apex4	1625	415	2040	1060 (34.8)	410 (1.2)	1470 (27.9)
cps	698	183	881	476 (31.8)	180 (1.6)	656 (25.6)
dalu	465	126	591	341 (26.7)	125 (0.8)	466 (21.1)
ex1010	1747	427	2174	1045 (40.2)	424 (0.7)	1469 (32.4)
ex5p	829	210	1039	432 (47.8)	210 (0.0)	642 (38.2)
misex3	295	99	394	222 (24.8)	97 (2.0)	319 (19.1)
pdc	854	235	1089	598 (30.0)	230 (2.1)	828 (24.0)
seq	1895	453	2348	1335 (29.6)	451 (0.4)	1786 (23.9)
spla	315	89	404	239 (24.1)	87 (2.2)	326 (19.3)
industry1	3415	1557	4972	2164 (36.7)	1530 (1.7)	3693 (25.7)
industry2	2392	1340	3732	1408 (41.1)	1306 (2.6)	2713 (27.3)
industry3	4987	1573	6560	4086 (18.1)	1558 (0.9)	5644 (14.0)
industry4	6856	1927	8782	3531 (48.5)	1856 (3.7)	5386 (38.7)
industry5	14696	6486	21183	10657 (27.5)	6412 (1.1)	17069 (19.4)
industry6	6429	3524	9953	3919 (39.0)	3464 (1.7)	7382 (25.8)
			Average:	**33.1%**	**1.6%**	**25.3%**

storing a 0 or a 1. Consequently, flip-flop leakage is not affected substantially by the proposed method. Similarly, it was observed that the LUTs in the target FPGA contain additional input buffers and other circuitry that make their leakage less sensitive to their input state. In the unoptimized circuits, 24% of active leakage power is dissipated in the non-interconnect circuit blocks and 76% in the interconnect blocks, on average. In the optimized circuits, 32% of leakage is attributable to non-interconnect blocks.

The results in Table 3.3 show a wide variation in improvement across the circuits. This can be partially explained by considering the distribution of static probabilities amongst a circuit's signals. The proposed technique offers the greatest benefit in circuits having many signals with low static probability, and the least benefit in circuits having many signals with static probability ≥ 0.5, as these signals are already in the low leakage state. Note that the static probability of a signal in a circuit is a function of both the simulation vector set, as well as the circuit's logic functionality. According to the data in Table 3.3, the best results were achieved for the circuit *industry4*, with leakage reduced by 38%. Figure 3.12(a) shows a histogram of static probabilities in this circuit, extracted from the ModelSIM simulation. The horizontal axis represents static probability; the vertical axis represents the fraction the circuit's signals having static probability in a specific range. Observe that, for this circuit, the majority of signals have low static probability, with more than 60% of signals having probability less than 0.1. We verified that the skewed distribution was not a result of the simulation vector set failing to adequately exercise the circuit. In fact, more than 90% of the signals in circuit *industry*4 experienced toggling during its simulation. Figure 3.12(b) shows the histogram for the circuit *industry3*, for which the worst leakage reduction results were observed. Here, it is apparent that many signals have static probability close to 0.5. For such signals, the static probability remains close to 0.5 after inversion, limiting the benefit of the leakage reduction approach. Further characterization and control of static probability in FPGA circuits is a direction for future work.



a) circuit *industry4*



b) circuit *industry3*

Figure 3.12: Histograms of static probability.

3.4 Active Leakage Power Optimization via Leakage-Aware Routing

This section introduces a second approach to active leakage optimization, referred to as "leakage-aware FPGA routing". The idea is based on two observations:

1. Different routing switch types in an FPGA have different leakage power consumptions. For example, as illustrated in Table 3.1, some switch types have wider input multiplexers or larger buffers than other switch types, leading to higher average leakage.

2. Between any two logic block pins in an FPGA, there exist a variety of different routing paths, comprised of different routing switch types. The routing step of the CAD flow is tasked with selecting a path between the driver and load pin(s) for each of a design's signals.

FPGA routers employ a cost function and aim to find low-cost paths through the routing fabric from each signal's source pin to its load pin(s) [McMu 95, Swar 98]. The cost of a complete routing path is defined to be the sum of the costs of the path's constituent routing resources (switches). A cost function associates a particular cost value with each routing resource in the FPGA. Cost values can be chosen based on any number of criteria, for example, delay, scarcity, capacitance, or congestion. The main idea in leakage-aware routing is to select the cost for each routing resource in proportion to the resource's leakage power consumption, and then to use such costs during routing. The intent is to associate higher costs with more "leaky" switch types, making them less likely to be selected during routing, ultimately producing routing solutions having lower active leakage power consumptions.

The router in the Xilinx CAD flow classifies a design's driver/load connections as either critical or non-critical, based on their timing slack relative to the design's performance constraints. Critical and non-critical connections are then routed in timing-driven or cost-driven mode, respectively [Ande 04a]. In timing-driven mode, detailed RC delay calculations are used

during routing to minimize driver/load connection delay. In cost-driven mode, each routing resource is given a specific cost, as mentioned above, and the router attempts to minimize the total path cost for a given driver/load connection. The specific resource cost assignment used within the Xilinx router is proprietary; however, it reflects a combination of delay, wirelength, and scarcity. The original, unmodified Xilinx router is referred to as the *baseline* router.

The proposed leakage-aware routing approach can be applied in tandem with the polarity selection optimization described in Section 3.3. Consequently, the optimized circuits (optimized through polarity selection) were used to derive a set of new, leakage-aware routing resource costs. The leakage of each used routing resource in the optimized circuits was analyzed, and from this, the average leakage of each routing resource type was computed. The results are shown in Figure 3.13, normalized to the leakage consumed by a DOUBLE resource. Observe that the average leakage of a HEX resource, which spans 6 CLB tiles, is slightly lower than that of a DOUBLE resource, which spans 2 CLB tiles, implying that on a leakage basis, using a HEX should be "cheaper" than using a DOUBLE[4]. This relative costing is counter to other traditional costing criteria, such as wirelength, in which the cost of a HEX would be set considerably higher than the cost of a DOUBLE.

The Xilinx router was modified by altering the cost values used in cost-driven mode. Specifically, the cost of each routing resource was set to be proportional to the average leakage of its routing resource type. Since the aim here is to reduce leakage without compromising performance, we continue to allow the router to route timing-critical connections in timing-driven mode. Only non-critical connections are routed using the new leakage-derived costs. The modified router is referred to as the *leakage-aware* router.

[4] A routing resource that drives a long wire segment may consume *less* leakage than some other resource that drives a short wire segment. This is possible since switch leakage does not depend on the metal segment length. Rather, leakage depends on the switch multiplexer size and structure, and transistor sizings in the multiplexer and buffer.

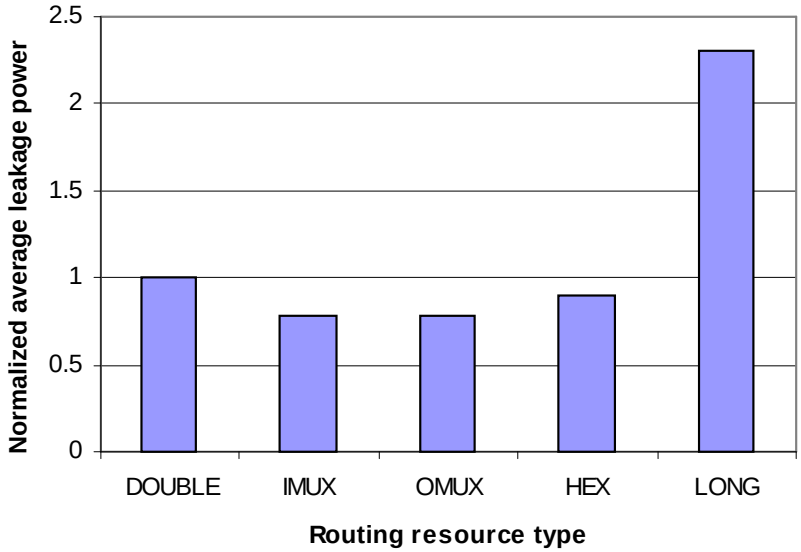


Figure 3.13: Average leakage of routing resource types.

3.4.1 Experimental Study and Results

Using the leakage-aware router, we again target the 90nm commercial FPGA described in Section 3.3.1 with the same set of 16 benchmark circuit designs. To begin with, the procedure described in Section 3.3.1 was repeated that computes an aggressive, but feasible, timing constraint for each design. The constraints achieved using the leakage-aware router were compared with those achieved using the baseline router. The results are shown in Table 3.4. Columns 2 and 3 show the critical path delay constraint for each circuit, routed using the baseline and leakage-aware routers, respectively. Note that the same placer was used in both cases. The parentheses in column 3 show the percentage degradation in performance when the leakage-aware router is used versus the baseline router. Ten of the 16 circuits experienced a slight performance degradation, though no degradation was larger than 4%. The performance of the remaining 6 circuits actually improved slightly (negative values in the table). Changes to the router's cost function lead to variability in the routing solutions produced, resulting in perfor-

Table 3.4: Effect of leakage-aware routing on critical path delay.

Circuit	Baseline routing performance (ns)	Leakage-aware routing performance (ns) (% degradation)
alu4	11.05	11.12 (0.6)
apex4	14.31	14.79 (3.4)
cps	11.59	11.90 (2.7)
dalu	11.43	11.32 (-0.9)
ex1010	21.84	22.08 (1.1)
ex5p	12.92	12.32 (-4.6)
misex3	11.89	11.53 (-2.9)
pdc	13.93	13.63 (-2.0)
seq	13.30	13.55 (1.9)
spla	10.76	10.91 (1.4)
industry1	4.63	4.51 (-2.6)
industry2	10.83	10.78 (-0.5)
industry3	16.79	17.19 (2.3)
industry4	4.83	4.94 (2.1)
industry5	5.13	5.23 (2.0)
industry6	21.86	22.07 (1.0)
	Average Degradation:	**0.3%**

mance improvements in some cases. On average, the degradation across all circuits was 0.3%, which we consider to be noise. One can therefore conclude that any reductions in leakage power offered by leakage-aware routing do not come at the expense of speed performance. As with the polarity selection optimization presented in Section 3.3, leakage-aware routing is a "no cost" leakage reduction technique.

The polarity selection optimization was applied in conjunction with leakage-aware routing, and the leakage in the resultant circuits was computed. Leakage was computed using the same approach described in Section 3.3.1. Figure 3.14 summarizes the results observed and illustrates the reduction in leakage in the optimized versus unoptimized circuits. Each bar in the figure represents the percentage reduction in leakage for a given circuit; the bars are partitioned to show the portion of the total reduction due to the polarity selection and leakage-aware routing optimizations, respectively. The average reduction across all circuits is 30.2%. Though the bulk

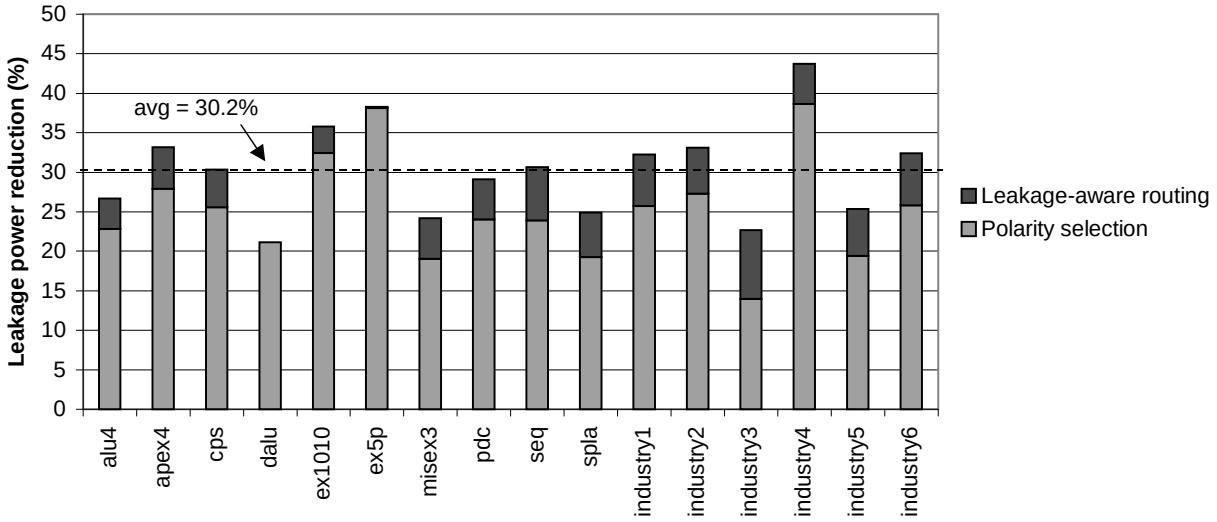


Figure 3.14: Leakage power reduction results for combined polarity selection and leakage-aware routing.

of the power reduction is due to the polarity selection optimization, the benefits of leakage-aware routing are nonetheless substantial, especially in the industrial benchmark circuits.

Detailed leakage power results for each circuit are shown in Table 3.5. Columns 2 and 3 give data for the interconnect and non-interconnect (labeled "other") circuit blocks, respectively. Column 4 gives the total active leakage power. The numbers in parentheses are percentage improvement values that show the reduction in leakage power relative to the *unoptimized* circuits; they compare the data in Table 3.5 with the data in columns 2 through 4 of Table 3.3. Notice that, as expected, only leakage in the interconnect circuit blocks is affected by leakage-aware routing; leakage in the "other" circuit blocks is unchanged versus using the polarity selection optimization alone (see column 6 of Table 3.3). For the MCNC circuits, the average reduction in total active leakage was 29.4%. In the industrial circuits, larger leakage reductions were observed, with the average reduction being 31.6%, due primarily to larger reductions in interconnect leakage for these circuits. The circuit *industry4* experienced the largest leakage reduction of nearly 44%. Column 5 of Table 3.5 shows the percentage improvement in leakage

Table 3.5: Detailed active leakage power results for leakage-aware routing combined with polarity selection.

Circuit	Interconnect (μW) (%)	Other (μW) (%)	Total (μW) (%)	% improvement versus polarity selection alone
alu4	460 (33.3)	187 (3.1)	647 (26.7)	5.0
apex4	953 (41.4)	410 (1.2)	1363 (33.2)	7.3
cps	434 (37.9)	180 (1.6)	614 (30.3)	6.4
dalu	341 (26.7)	125 (0.8)	466 (21.2)	0.7
ex1010	972 (44.4)	424 (0.7)	1396 (35.8)	5.0
ex5p	431 (48.0)	210 (0.0)	641 (38.3)	0.2
misex3	201 (31.6)	97 (2.0)	298 (24.2)	6.3
pdc	542 (36.5)	230 (2.1)	772 (29.1)	6.7
seq	1177 (37.9)	451 (0.4)	1628 (30.7)	8.8
spla	217 (31.3)	87 (2.2)	304 (24.9)	7.0
industry1	1840 (46.1)	1530 (1.7)	3369 (32.2)	8.8
industry2	1191 (50.2)	1306 (2.6)	2497 (33.1)	8.0
industry3	3515 (29.5)	1558 (0.9)	5073 (22.7)	10.1
industry4	3085 (55.0)	1856 (3.6)	4941 (43.7)	8.3
industry5	9404 (36.0)	6412 (1.1)	15816 (25.3)	7.3
industry6	3268 (49.2)	3464 (1.7)	6731 (32.4)	8.8
Average (MCNC):	**36.9%**	**1.4%**	**29.4%**	**5.3%**
Average (Industrial):	**44.3%**	**2.0%**	**31.6%**	**8.5%**

relative to applying the polarity selection optimization alone. That is, the data in column 5 compares the leakage numbers in column 4 with the leakage numbers in column 7 of Table 3.3. On average, leakage-aware routing provides a 5.3% leakage reduction in the MCNC circuits, and an 8.5% reduction in the industrial circuits. In summary, the results show that the additional leakage power reductions offered by leakage-aware routing are considerable, especially given that the approach involves software changes only, and imposes no hardware, fabrication, or performance cost.

As mentioned above, the cost of a HEX resource in the leakage-aware router is similar to that of a DOUBLE resource. Whereas, in the baseline router, the cost of HEX is higher than that of a DOUBLE. Certainly, leakage-aware routing leads to higher HEX utilization, and, since the capacitance of a HEX is larger than that of a DOUBLE, it is conceivable that leakage-aware

routing may increase dynamic power consumption. A future research direction is to investigate this possibility, and, if deemed a problem, to enhance leakage-aware routing to account for it, perhaps by taking signal switching activity into account when deciding how a signal should be routed. That being said, we anticipate that the proposed techniques will be applied in a future low leakage FPGA, perhaps implemented in 65 or 45nm process technology. At such technology nodes, we expect that leakage power, not dynamic power, will be the overriding power consideration.

3.5 Summary

This chapter presented two "no cost" approaches to active leakage power reduction in FPGAs. The leakage power characteristics of common FPGA hardware structures were studied, and it was observed that their leakage depends strongly on the state of their inputs. A novel approach to leakage power reduction was proposed, in which polarities are selected for logic signals to place hardware structures into low leakage states, as much as possible. The technique is based on a unique property of FPGA logic elements (LUTs) that permits either the true or complemented form of a signal to be generated, without any area or delay penalty. Experimental results for a 90nm state-of-the-art commercial FPGA show that the proposed approach reduces active leakage by 25%, on average. Subsequently, the concept of leakage-aware routing was introduced, in which the cost function used during the routing step of the CAD flow is altered to consider the leakage power consumptions of routing resources. Leakage-aware routing incurs no significant performance penalty, and offers additional leakage reductions. Combining the two techniques produces a total active leakage reduction of up to 44%, with the average reduction being 30%.

4 Circuit Techniques for Low-Power Interconnect

4.1 Introduction

Interconnect plays a dominant role in the dynamic and static (leakage) power dissipation of FPGAs. In comparison with custom ASICs, FPGA interconnect presents a high capacitive load, due to the presence of lengthy pre-fabricated wire segments and the programmable routing switches attached to each wire. Dynamic power scales in direct proportion to amount of capacitance switched in a logic transition. Leakage power, on the other hand, is proportional to total transistor width and interconnect comprises roughly 2/3 of an FPGA's total silicon area [Rahm 04]. The influence of interconnect on overall FPGA power implies that any future low-power FPGA must include a low-power interconnection fabric. This chapter presents a family of novel FPGA routing switch designs that offer reduced leakage and dynamic power dissipation.

A property common to all of the proposed switch designs is the concept of "programmable mode". Specifically, the routing switches can be programmed to operate in one of three modes: high-speed, low-power, or sleep mode. In high-speed mode, power and performance characteristics of the proposed switches are similar to those of current FPGA routing switches. Low-power mode offers reduced leakage and dynamic power, albeit at the expense of speed performance. As noted in Section 2.4.2, an FPGA implementation of a design uses only a portion of the underlying FPGA hardware. Leakage is dissipated in both the used and the unused parts of the FPGA. The sleep mode of the proposed switch designs is suitable for unused routing switches, and it offers leakage reductions significantly beyond those available in low-power mode. The

remainder of the chapter is organized as follows: Section 4.2 presents related work and relevant background material. The proposed switch designs are described in Section 4.3. Section 4.4 analyzes the timing slack present in industrial FPGA designs implemented in a $90n$m commercial FPGA, and demonstrates that a large fraction of routing switches may operate in low-power mode, without compromising overall circuit performance. Experimental results are given in Section 4.5. A summary is provided in Section 4.6.

4.2 Preliminaries

4.2.1 Related Work

Section 2.2.2 reviewed techniques for leakage optimization in ASICs and microprocessors. The proposed switch designs draw upon ideas from two previously published techniques for sleep leakage reduction, briefly reviewed here. The first technique introduces sleep transistors into the N-network (and/or P-network) of CMOS gates [Anis 02], as shown in Figure 4.1(a). Sleep transistors ($MPSLEEP$ and $MNSLEEP$) are ON when the circuit is active and are turned OFF when the circuit is in sleep mode, gating the leakage current from supply to ground. A limitation of this approach is that in sleep mode, internal voltages in sleeping gates are not well-defined and therefore, the technique cannot be directly applied to data storage elements.

A way of dealing with the data retention issue was proposed in [Kuma 98] and is shown in Figure 4.1(b). Two diodes, DP and DN, are introduced in parallel with the sleep transistors. In active mode, the virtual V_{DD} voltage (V_{VD}) and the virtual ground voltage (V_{VGND}) are equal to rail V_{DD} and GND, respectively. In sleep mode, the sleep transistors are turned OFF and $V_{VD} \approx V_{DD} - V_{DP}$, where V_{DP} is the built-in potential of diode DP. Likewise, $V_{VGND} \approx GND + V_{DN}$ in sleep mode. The potential difference across the latch in sleep mode is well-defined and equal to $V_{DD} - V_{DP} - V_{DN}$, making data retention possible. In sleep mode, both subthreshold and gate oxide leakage are reduced as follows: 1) The reduced potential difference across the drain/source (V_{DS}) of an OFF transistor yields an exponential decrease in subthreshold leakage, due to the drain-induced barrier lowering (DIBL) effect (see Section 2.2.2),

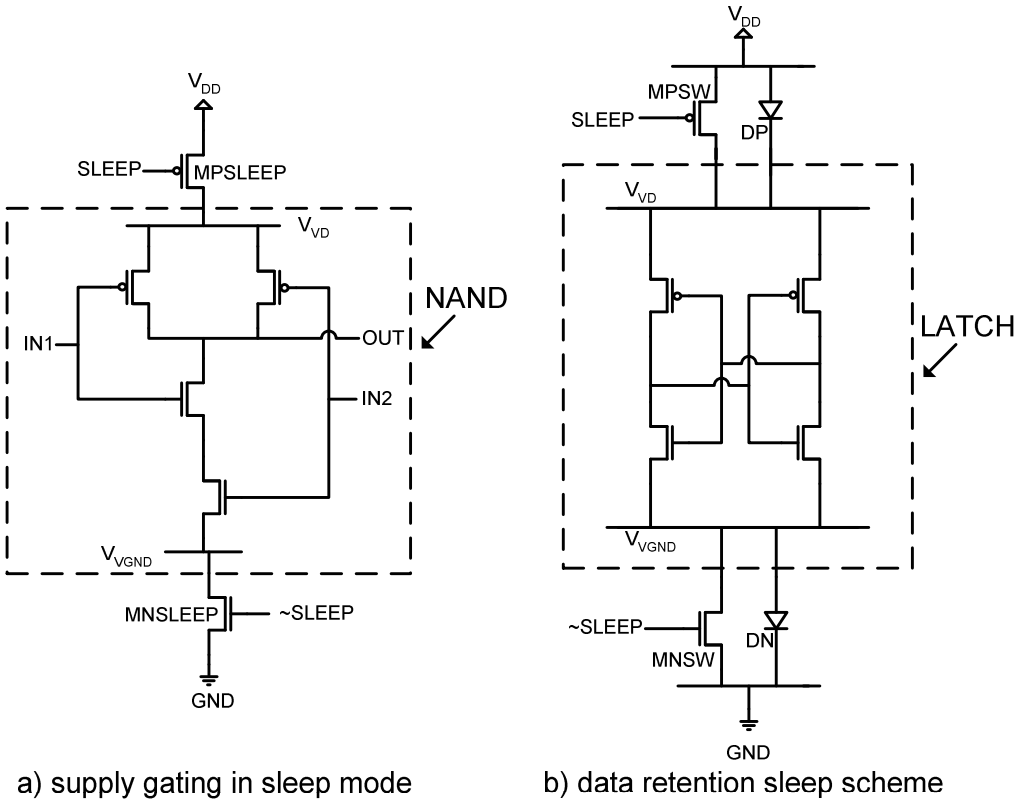


Figure 4.1: Sleep leakage reduction techniques [Anis 02, Kuma 98].

and, 2) Gate oxide leakage decreases superlinearly with a reduction in gate/source potential difference (V_{GS}).

4.2.2 FPGA Interconnect Structures

Section 2.3 gave an overview of FPGA interconnect structures. Figure 4.2(a), repeated here for convenience, shows a typical buffered FPGA routing switch, similar to those in modern Xilinx and Altera commercial FPGAs [Lewi 03]. A transistor-level view of a switch with 4 inputs is shown in Figure 4.2(b) [Rahm 04, Gaya 04a]. Observe that the buffer in Figure 4.2(b) is "level-restoring" – transistor $MP1$ serves to pull the buffer's input to rail V_{DD} when logic-1 is passed through the switch [Rahm 04]. Without $MP1$, if a logic-1 (V_{DD}) were passed through

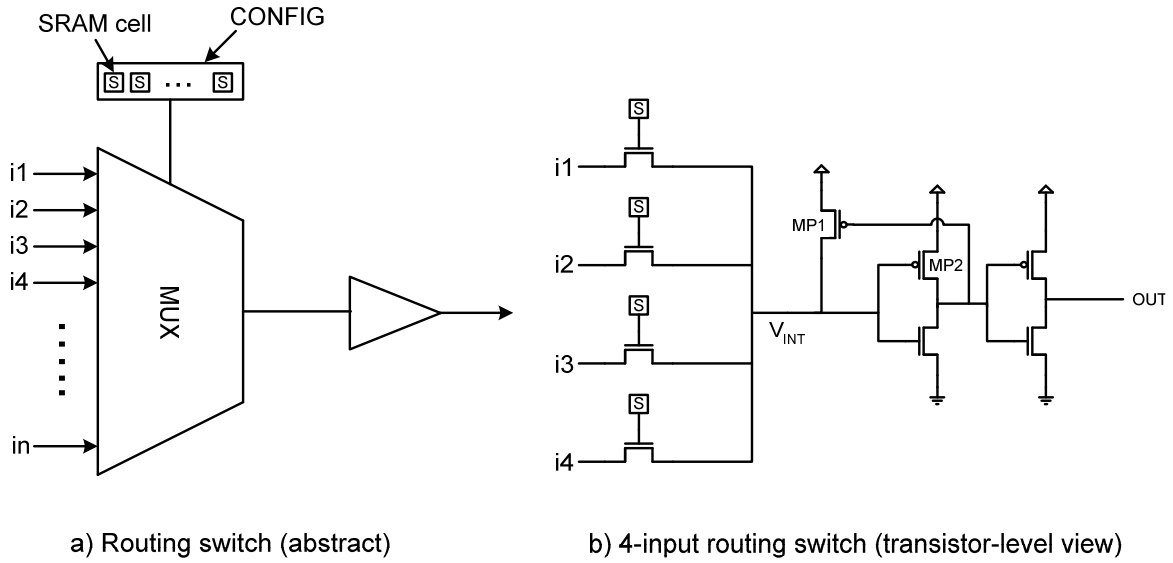


Figure 4.2: Traditional routing switch: abstract and transistor-level views [Rahm 04, Gaya 04a].

the multiplexer, a "weak-1" would appear on the multiplexer's output ($V_{INT} \approx V_{DD} - V_{TH}$), causing $MP2$ to turn partially ON, leading to excessive buffer leakage.

4.3 Low-Power Routing Switch Design

The proposed switch designs are based on three key observations that are specific to FPGA interconnect:

1. Routing switch inputs are tolerant to "weak-1" signals. That is, logic-1 input signals need not be rail V_{DD} – it is acceptable if they are lower than this. This is due to the level-restoring buffers that are *already* deployed in FPGA routing switches [see Figure 4.2(b)].

2. There exists sufficient timing slack in typical FPGA designs to allow a sizable fraction of routing switches to be slowed down, without impacting overall design performance. This assertion will be demonstrated in the next section.

3. Most routing switches simply feed other routing switches, via metal wire segments. This

observation holds for the majority of switches in commercial FPGAs, such as the Xilinx Spartan-3 FPGA [Spar 04]. Observation #1, above, permits such switches to produce "weak-1" signals. The main exceptions to this observation are switches that drive inputs on logic blocks.

Based on these three observations, we propose the new switch design shown in Figure 4.3. The switch includes NMOS and PMOS sleep transistors in parallel (MNX and MPX). The sleep structure is similar to that in Figure 4.1(b), with diode DP being replaced by an NMOS transistor, MNX. The new switch can operate in three different modes as follows: In high-speed mode, MPX is turned ON and therefore, the virtual V_{DD} (V_{VD}) is equal to V_{DD} and output swings are full rail-to-rail. The gate terminal of MNX is left at V_{DD} in high-speed mode, though this transistor generally operates in the cut-off region, with its $V_{GS} < V_{TH}$. During a 0-1 logic transition however, V_{VD} may temporarily drop below $V_{DD} - V_{TH}$, causing MNX to leave cut-off and assist with charging the switch's output load.

In low-power mode, MPX is turned OFF and MNX is turned ON. The buffer is powered by the reduced voltage, $V_{VD} \approx V_{DD} - V_{TH}$. Since $V_{VD} < V_{DD}$, speed is reduced versus high-speed mode. However, output swings are reduced by V_{TH}, reducing switching energy, and leakage is reduced for the same reasons mentioned above in conjunction with Figure 4.1(b). Lastly, in sleep mode, both MPX and MNX are turned OFF, similar to the supply gating notion in Figure 4.1(a).

In addition to the switch in Figure 4.3, a second buffer design is proposed in Figure 4.4, and it offers a different power/area trade-off. In the alternate design, the bodies of the PMOS transistors are tied to V_{VD}, rather than the typical V_{DD}. This lowers the threshold voltage of the PMOS transistors in low-power mode, via the "body effect", thus increasing their drive strength. In high-speed mode, as mentioned above, V_{VD} drops temporarily below V_{DD} during a 0-1 logic transition, and therefore, improved PMOS drive capability may also be exhibited in this mode. The benefit of enhanced drive strength is that the sleep transistors can be made smaller, reducing the area overhead of the proposed switch versus a traditional switch. The

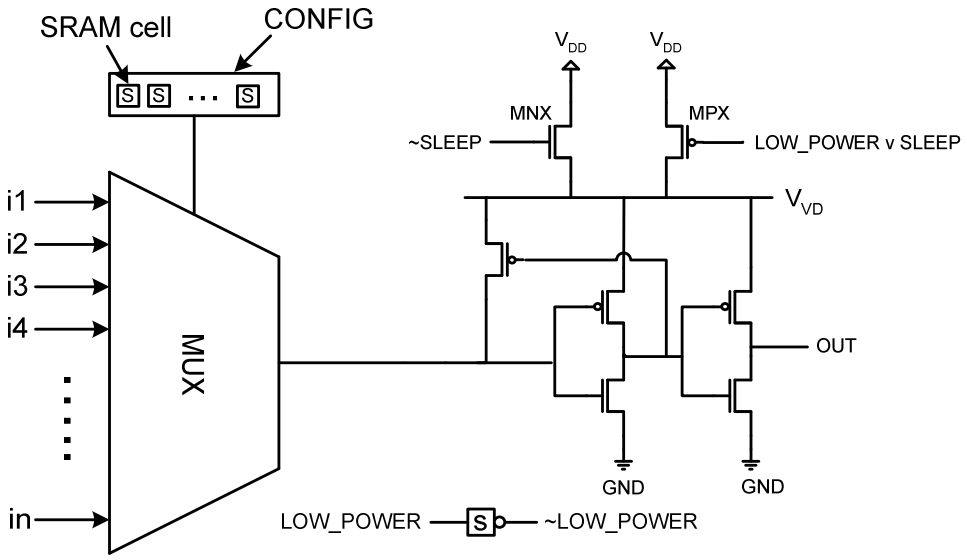


Figure 4.3: Programmable low-power routing switch.

downside is that the reduced threshold voltage of the PMOS transistors will likely lead to greater subthreshold leakage in these transistors versus leakage in the initial design of Figure 4.3. For the remainder of the chapter, the switch design in Figure 4.3 is referred to as the *basic* design, and the one in Figure 4.4 as the *alternate* design.

The *alternate* design offers a different *leakage/area* trade-off versus the *basic* design; that is, the *alternate* design requires less area, but is likely more "leaky". For both designs, a straightforward extension can be made to realize a different *leakage/speed* trade-off. Specifically, one can apply the sleep structure discussed above to the multiplexer that precedes the buffer, as shown in Figure 4.5. Two additional sleep transistors, MNX_M and MPX_M, are introduced into the pull-up network of the multiplexer and its configuration circuitry. The programmable multiplexer concept can be combined with both the *basic* buffer design, as well as the *alternate* design. These switch variants are referred to as *basic+MUX* and *alternate+MUX*, respectively.

In high-speed mode, the multiplexer is powered by V_{DD}, similar to a standard routing switch. In low-power mode, the multiplexer is powered by $V_{DD} - V_{TH}$. Recall that the multiplexer select

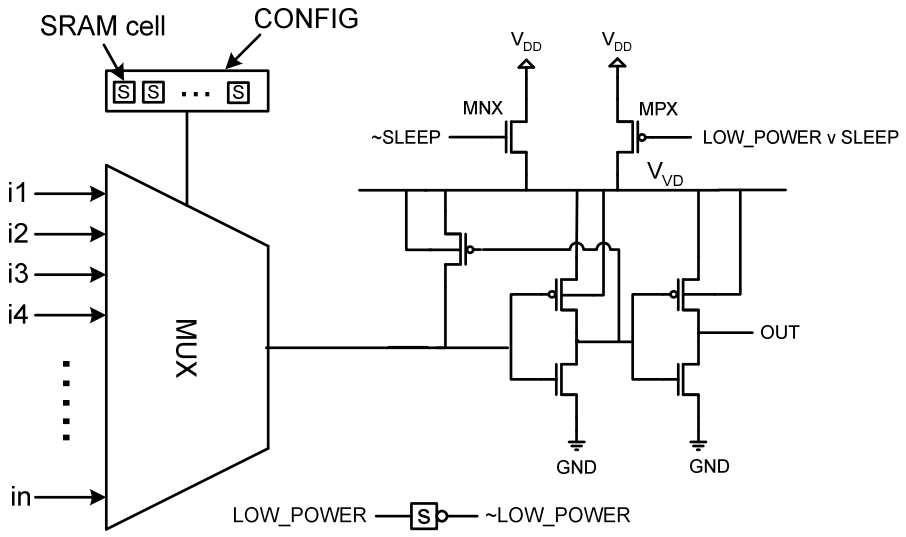


Figure 4.4: Routing switch buffer alternate design.

lines attach to the gate terminals of NMOS transistors (see Figure 4.2). The reduced voltage on the select lines in low-power mode will lower gate oxide leakage in the multiplexer. Leakage in the SRAM configuration cells will also be reduced in low-power mode. Of course, signal propagation delay through the multiplexer will increase in low-power mode. Note that, because the contents of the SRAM configuration cells do not change during normal operation, the SRAM cell performance is not critical. Consequently, transistors MNX_M and MPX_M can be made very small. While the sizes of MNX and MPX strongly influence the FPGA's performance, the sizes of MNX_M and MPX_M do not.

There are several reasons for introducing two additional sleep transistors (MNX_M and MPX_M) instead of simply using the existing sleep transistors (MNX and MPX) to control *both* the buffer and the multiplexer. First, as mentioned above, the V_{VD} signal powering the buffer may swing below $V_{DD} - V_{TH}$ during a 0-1 logic transition. If the same sleep transistors were shared between the multiplexer and buffer, such a voltage drop, depending on its magnitude, could destabilize the contents of the SRAM configuration cells – a catastrophic

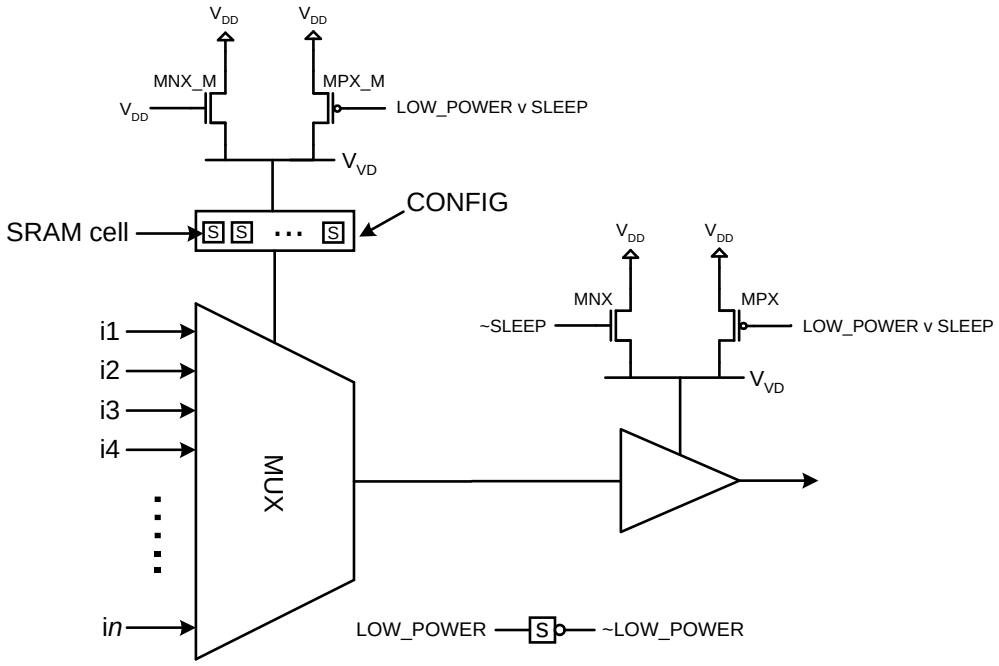


Figure 4.5: Switch multiplexer with programmable mode.

device failure. Second, sleep mode works differently in the buffer versus the multiplexer. In the buffer, both MNX and MPX are turned OFF in sleep mode. In the multiplexer, sleep and low-power mode are identical. If MNX_M and MPX_M were turned OFF in sleep mode, the SRAM configuration cells would lose their state. Moreover, the voltages on the multiplexer select lines would not be well-defined, potentially turning ON one or more multiplexer paths, and introducing additional capacitive loading on upstream routing switches.

Finally, for all of the switch designs, we also consider a variant for sleep mode, shown in Figure 4.6. Transistor $MSLEEP$ is added to pull node V_{INT} to ground in sleep mode. The intent of $MSLEEP$ is to set the buffer's internal node voltages (V_{INT}, V_{INTB}, OUT) to a known state in sleep mode, thus improving buffer leakage. This differs from the switch designs described above, wherein the internal node voltages are allowed to "float" in sleep mode, possibly leading to a scenario in which both transistors in an inverter stage are (partially) ON.

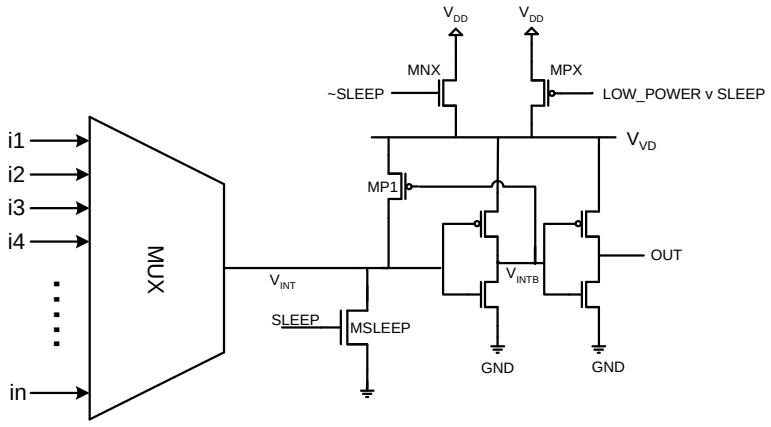


Figure 4.6: Sleep mode variant.

When $MSLEEP$ is ON, V_{INT} is pulled to logic-0, V_{INTB} is pulled to V_{VD}, and, provided V_{VD} is sufficiently high, OUT is pulled to logic-0. Note that since $MSLEEP$ is loading the multiplexer output, its size should be kept very small. Observe that V_{INT} cannot be pulled high (instead of low) in sleep mode. Doing so would cause $MP1$ to turn ON, pulling V_{VD} high to V_{DD}, thereby negating the benefit of MNX and MPX being OFF in sleep mode.

Figure 4.7 summarizes all of the switch designs considered in this study. As shown, the switch buffer can be of either the *basic* design (Figure 4.3) or the *alternate* design (Figure 4.4). Two different switch multiplexers are possible: one with the sleep structure in its pull-up network, and one without the sleep structure. This yields a total of four different switch designs. The NMOS pull-down transistor on the buffer input (for use in sleep mode) can be introduced into any of the four designs, and therefore, eight different sleep modes will be evaluated.

In essence, the new switch designs mimic the programmable dual-V_{DD} concepts proposed in [Li 04c, Li 04b, Gaya 04a, Li 04a], while avoiding the costs associated with true dual-V_{DD}, such as distributing multiple power grids and providing multiple supply voltages at the chip level. In traditional dual-V_{DD} design, level converters are required to avoid excessive leakage when circuitry operating at low supply drives circuitry operating at high supply. However, in

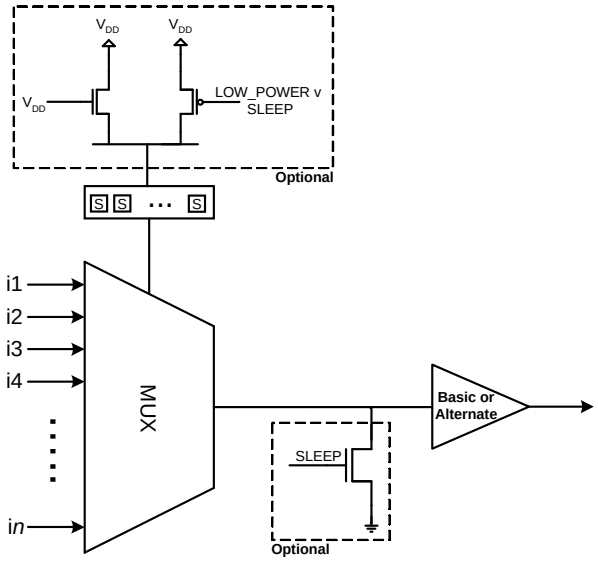


Figure 4.7: Family of routing switch designs.

this case, because of observation #1, no level converters are required when a switch in low-power mode drives a switch in high-speed mode.

We envision that the selection between low-power and high-speed modes can be realized through an extra configuration SRAM cell in each routing switch. Alternately, to save area, the extra SRAM cell could be shared by a number of switches, all of which must operate in the same mode. We expect that today's commercial FPGA routing switches already contain configuration circuitry to place them into a known state when they are unused. This circuitry can be used to select sleep mode, as appropriate. A key advantage of the proposed designs is that they have no impact on FPGA router complexity – the mode selection can be made at the post-routing stage, when timing slacks are accurately known.

The relatively low hardware cost and negligible software impact make the proposed switch designs quite practical. It is expected that they can be deployed in place of most existing routing switches in commercial FPGAs.

4.4 Slack Analysis

The benefits of a routing switch that offers a low-power (slow) mode depend on there being a sufficient fraction of routing resources that may actually operate in this mode, without violating design performance constraints. This depends directly on the amount of "timing slack" present in typical FPGA designs. In custom ASICs, any available slack is generally eliminated by sizing down transistors, saving silicon area and cost. In the FPGA domain, however, the device fabric is fixed, and therefore, it is conceivable that for many designs, the available timing slack is substantial.

To motivate the proposed switch designs, timing slack was evaluated in 22 routed industrial designs implemented in the Xilinx Spartan-3 FPGA [Spar 04] (described in Section 3.3.1). The Xilinx placement and routing tools were used to generate a performance-optimized layout for each design using the iterative constraint tightening process described in Section 3.3.1 (see page 53). Timing slack was evaluated in the layout solution corresponding to the most aggressive, yet achievable, constraint observed throughout the entire iterative process. Evaluating slack with respect to such aggressive constraints ensures that the picture of available timing slack generated is not overly optimistic.

To gauge slack, the algorithm in [Wang 02] was implemented, which finds a maximal set of a design's driver/load connections that may be slowed down by a *pre-specified* percentage without violating timing constraints. The algorithm was originally used to select sets of transistors to have high-V_{TH} in a dual-V_{TH} ASIC design framework. Since the aim here is to maximize the number of routing switches that operate in low-power mode, the algorithm was altered slightly to establish a preference for selecting connections (to be slowed down) that use larger numbers of routing switches in their routing solutions. In [Wang 02], each driver/load connection can be viewed as having "unit weight". In our implementation, a simple heuristic is employed: each connection is assigned a weight corresponding to the number of routing switches in its routing solution. Instead of finding a maximum size set of connections that may be slowed down (as

in [Wang 02]), the same algorithm is applied to find a maximum weight set of connections that may be slowed down. The interested reader is referred to [Wang 02] for complete details.

Three slack analyses were performed for each design and sets of connections that may be slowed down by 25%, 50%, and 75% were computed. Then, the fraction of routing resources that were used in the routing of the selected connections was determined; that is, the fraction of *used* routing resources that may be slowed down. The results are shown in Figure 4.8. The vertical axis shows the fraction of routing resources that may be slowed down by a specific percentage, averaged across all 22 designs. The horizontal axis shows the main routing resource types in Spartan-3. For each resource type, three bars represent the fraction of used routing resources of that type that may be slowed down. For example, the left-most set of bars indicate that roughly 80%, 75%, and 70% of used DOUBLE resources may be slowed down by 25%, 50%, and 75%, respectively. The right-most set of bars in Figure 4.8 provides average results across all resource types. Observe, for example, that ∼75% of all routing resources can be slowed down by 50%, on average. Interestingly, the results observed here agree closely with prior work by Betz and Hutton et. al, which showed that only 20% of an FPGA's routing resources need to be high-speed [Betz 98, Hutt 02]. The considerable slack in typical FPGA designs bodes well for the proposed routing switch designs.

4.5 Experimental Study

4.5.1 Methodology

Unless noted otherwise, all HSPICE simulation results reported in this chapter were produced at 85°C using the Berkeley Predictive Technology Models (BPTM) for a 70nm technology [Berk 04]. The technology models were enhanced to account for gate oxide leakage using four voltage controlled current sources, as shown in Figure 4.9, and described in [Aziz 04]. Both *direct tunneling* current, in an ON transistor, as well as *edge-directed tunneling*, in an OFF transistor, are modeled through current sources I_{GON_GS}, I_{GON_GD} and I_{EDT_SG}, I_{EDT_DG}, respectively. The results presented correspond to an oxide thickness of 1.2nm [Inte 02].

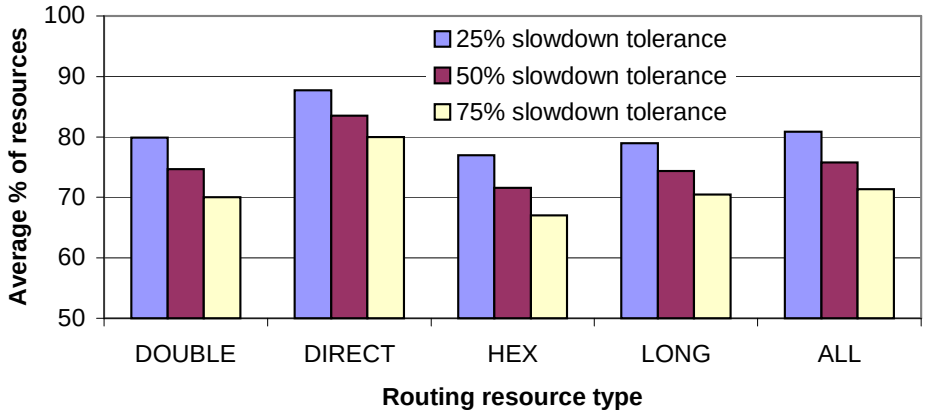


Figure 4.8: Timing slack in industrial FPGA designs.

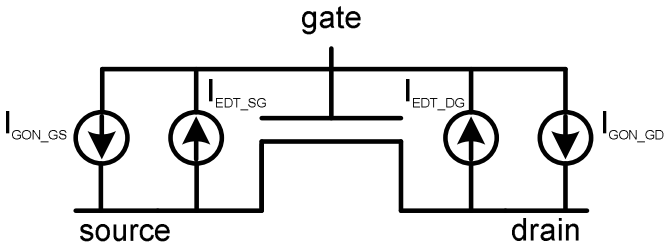


Figure 4.9: Model for transistor gate oxide leakage [Aziz 04].

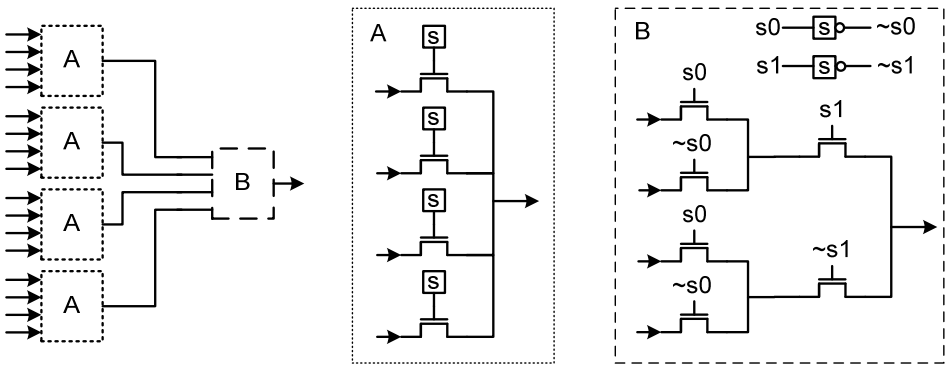


Figure 4.10: 16-to-1 multiplexer implementation.

To study the proposed switch designs, the first step was to develop a 16-input traditional routing switch [see Figure 4.2(b)], representative of those in current commercial FPGAs [Spar 04][1]. The buffer was sized for equal rise and fall times, with the second inverter stage being 3 times larger than the first stage. The 16-to-1 input multiplexer was constructed as shown in Figure 4.10, and it is believed to reflect a reasonable trade-off between speed and area. Two stages of 4-to-1 multiplexers are used to form the 16-input multiplexer. Input-to-output paths through the multiplexer consist of three NMOS transistors. As in [Rahm 04], SRAM configuration cells are assumed to be shared amongst the four 4-to-1 multiplexers in the first stage. Thus, the entire 16-to-1 multiplexer requires 6 SRAM cells to select a path from one of its inputs to its output.

The 16-input traditional switch was then used as a basis for developing the proposed switch designs. Specifically, in the *basic* design, transistor MPX (see Figure 4.3) was sized to provide high-speed mode performance within 5% of the traditional switch. Interconnect delay typically comprises about half of total path delay in FPGAs, and therefore, a 5% increase in interconnect delay would produce a 2.5% performance degradation overall. Transistor MNX was sized to achieve 50% slower speed performance in low-power versus high-speed mode. From Figure 4.8, one can expect that ∼75% of routing switches designed as such could operate in low-power

[1] A 16-input switch was selected as it is similar to the switches driving DOUBLE-length segments in Xilinx Spartan-3 [Spar 04].

mode in a typical design. Certainly, the sizes of sleep transistors MNX and MPX can be adjusted to realize different area/power/performance trade-offs, as desired.

Both a *basic* version of the proposed switch (Figure 4.3), as well as an *alternate* version (Figure 4.4) were developed. Both versions have the same performance characteristics; however, in the *alternate* version, it was possible to reduce the total width of the sleep transistors by 36% compared to the *basic* version. The *basic* and *alternate* switches were then extended to create two additional switch types: *basic+MUX* and *alternate+MUX* (see Figure 4.5). In these designs, where the programmable mode concept is applied to the multiplexer, the low-power mode is 80% slower than high-speed mode. Therefore, if these designs are used, slightly fewer routing switches would be permitted to operate in low-power mode.

To study the power characteristics of the proposed switch designs, the conditions of a used switch in an actual FPGA were simulated using the test platform shown in Figure 4.11. The test platform corresponds to a contiguous path of three switches through an FPGA routing fabric; the multiplexers in all three switches are configured to pass input $i1$ to their outputs. Power and performance measurements are made for the second switch, labeled "test switch", in Figure 4.11. The power measurements include current drawn from all sources, including gate oxide leakage in the multiplexer and sleep transistors. Subthreshold leakage current through the *inputs* of the test switch is not included, as this is attributable to the buffer(s) in the preceding switch stage(s). As in Chapter 3, this work ignores leakage power dissipated in the SRAM configuration cells, since such cells can be slowed down and their leakage reduced or eliminated.

4.5.2 Leakage Power Results

We first examine the difference in leakage power in low-power versus high-speed mode. For this task, two instances of the test platform were used: one in which all three switches are in high-speed mode, and one in which all three switches are in low-power mode. This configuration produced the most pessimistic power results for low-power mode. Both the high-speed and low-power platforms were simulated with identical vector sets, consisting of 2,000 random input

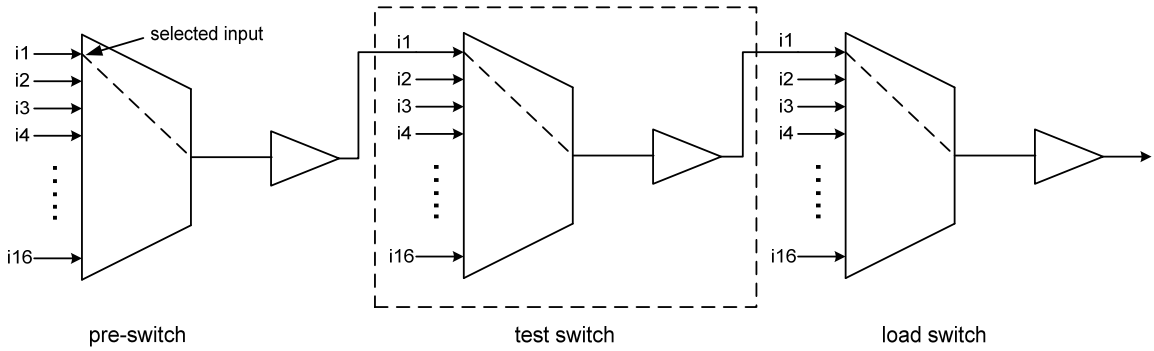


Figure 4.11: Baseline test platform.

vectors[2]. The leakage power consumed in the test switch was captured for each vector in both platforms. The results for the *basic* switch design are shown in Figure 4.12(a). The horizontal axis shows the percentage reduction in leakage power in the low-power switch versus the high-speed switch. The vertical axis shows the number of vectors that produced a leakage reduction in a specific range. Observe that larger leakage reductions are realized when the switch output signal is logic-0 versus logic-1, due primarily to the different leakage characteristics of NMOS versus PMOS devices. On average, in the *basic* design, low-power mode offers a 36% reduction in leakage power compared with high-speed mode.

Figure 4.12(b) gives results for the *alternate* switch design. Observe that, as expected, leakage reductions in the logic-0 state are smaller than in the *basic* design [Figure 4.12(a)], due to the lower threshold voltage, and increased subthreshold leakage of the PMOS transistors when the *alternate* switch operates in low-power mode. On average, the low-power mode of the *alternate* switch design offers a 28% reduction in leakage versus high-speed mode.

To evaluate sleep mode leakage, the test platform was altered by attaching the output of the test switch to a different, non-selected input of the load switch. Also, the multiplexer in the test switch was configured to disable all paths to the multiplexer output (SRAM cell contents are all 0s). As above, the modified platform was simulated with random vectors. The average

[2]Random signals were presented to all 46 inputs in each test platform. The same set of vectors were presented to each test platform.

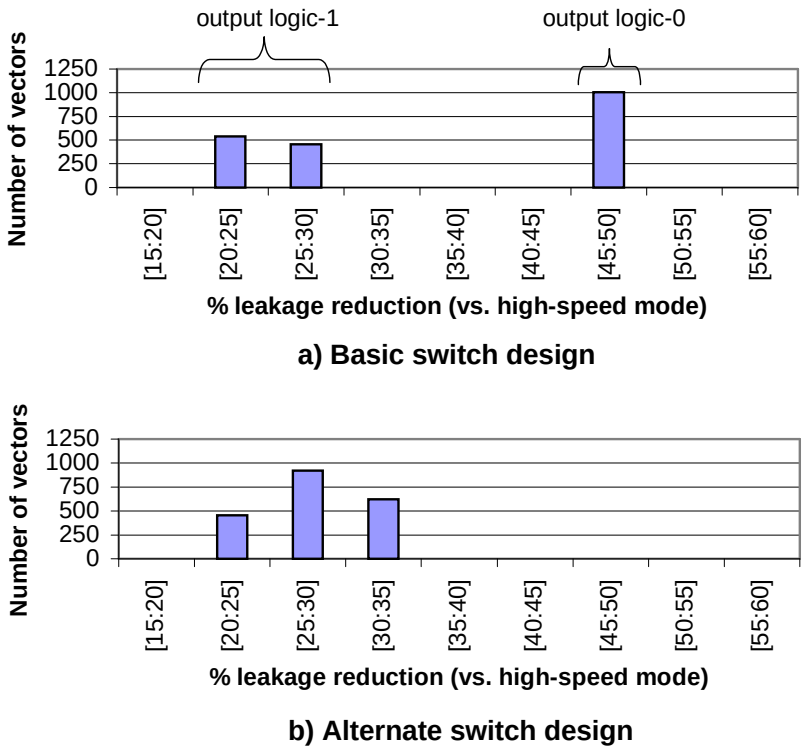


Figure 4.12: 85°C leakage reduction results (low-power mode versus high-speed mode).

reduction in leakage power for sleep mode relative to high-speed mode was found to be 61%. Similar results were observed for both the *basic* and *alternate* switch designs.

Routing conductors in FPGAs have multiple used and unused switches attached to them. Consequently, the sensitivity of the low-power mode results to multi-fanout conditions was studied. In one scenario, the test platform was augmented to include 5 unused switches in sleep mode on the test switch output. In a second scenario, the test platform was augmented to include 5 used switches on the test switch output. Average leakage power reduction results for all scenarios considered are summarized in Table 4.1, which gives the average percentage reduction in leakage power for each scenario versus the proposed switch in high-speed mode. The unshaded portion of the table gives results for the *basic* switch design; the shaded portion of the table gives results for the *alternate* design. Observe that the dependence of the low-power mode results on fanout is relatively weak – the results are slightly better in the more realistic multi-fanout scenarios.

Row 6 of Table 4.1 gives data comparing the average leakage power of the proposed switch designs with that of the *traditional* routing switch used as the development basis. The leakage of the proposed switch designs in high-speed mode is roughly equivalent to that of the traditional switch. Thus, there is no significant penalty for deploying the proposed switch designs from the leakage viewpoint, even if they are operated in high-speed mode.

An FPGA implementation of a circuit uses only a fraction of the FPGA's available hardware resources [Tuan 03]. It is therefore possible that large regions of an FPGA may be lightly utilized, and that the die temperature in lightly utilized regions is somewhat lower than in heavily utilized regions. To gain insight into how the leakage results presented above scale with temperature, the leakage characteristics of the proposed designs were evaluated at low temperature (25°C). The results are summarized in Table 4.2. The interpretation of the rows and columns in Table 4.2 is the same as that of Table 4.1.

Looking first at the 25°C simulation results for the single fanout scenario (row 2 of Table 4.2), one can see that the difference between the two designs is less pronounced than at high

Table 4.1: 85°C leakage power reduction results for *basic* design (unshaded) and *alternate* design (shaded).

Test scenario	**Avg. leakage pwr reduction (%) vs. high-speed mode (basic)**	**Avg. leakage pwr reduction (%) vs. high-speed mode (alternate)**
low-power mode (single fanout)	36.0%	27.6%
sleep mode	60.8%	61.3%
low-power mode (+ unused fanout)	39.7%	28.7%
low-power mode (+ used fanout)	38.7%	29.5%
traditional switch (single fanout)	0.3%	0.25%

temperature. This is explained by recalling that the superior leakage characteristics of the *basic* switch design are primarily due to its smaller subthreshold leakage current (see Section 4.3). Subthreshold leakage increases exponentially with temperature, whereas gate oxide leakage is almost insensitive to temperature [Agar 04]. At low temperature, gate oxide leakage comprises a larger fraction of total leakage. Gate oxide leakage is smaller in the *alternate* versus the *basic* design, due to its smaller sleep transistors. This leads to a narrower "gap" between the two designs from the leakage perspective at low temperature. Similar results are evident for the multi-fanout scenarios.

In sleep mode (row 3 of Table 4.2), the *alternate* design actually offers lower leakage than the *basic* design at low temperature, again due to the increased significance of gate oxide leakage. At high temperature, where subthreshold leakage dominates, the two designs exhibit roughly equivalent leakage (see row 3 of Table 4.1). Thus, although the smaller sleep transistors in the *alternate* design result in lower gate oxide leakage, they do not appear to yield a significant reduction in subthreshold leakage in sleep mode.

Tables 4.3 and 4.4 present the leakage power results for the *basic+MUX* and *alternate+MUX* designs at 85°C and 25°C, respectively. At 85°C, the leakage improvements over the original *basic* and *alternate* designs are modest. For example, in the single fanout case, the low-power

Table 4.2: 25°C leakage power reduction results for *basic* design (unshaded) and *alternate* design (shaded).

Test scenario	**Avg. leakage pwr reduction (%) vs. high-speed mode (basic)**	**Avg. leakage pwr reduction (%) vs. high-speed mode (alternate)**
low-power mode (single fanout)	33.3%	30.0%
sleep mode	64.7%	72.3%
low-power mode (+ unused fanout)	34.8%	31.4%
low-power mode (+ used fanout)	33.3%	31.4%
traditional switch (single fanout)	1.6%	1.3%

mode of the *basic+MUX* (*alternate+MUX*) design offers a 42% (33%) leakage reduction versus high-speed mode. This is a moderate improvement over the *basic* (*alternate*) design, which yields a 36% (28%) leakage reduction in low-power mode.

At 25°C, the benefits of reduced gate oxide leakage in the multiplexer (in *basic+MUX* and *alternate+MUX*) are more apparent. Consider row 2 of Table 4.4, which gives the low-power mode leakage results for the single fanout scenario. Leakage is reduced by 52% in *basic+MUX* and 48% in *alternate+MUX* versus high-speed mode. This can be compared with the *basic* and *alternate* designs which offer 33% and 30% leakage reductions, respectively (row 2 of Table 4.2). Subthreshold and gate oxide leakage exhibit different technology scaling trends. Should gate oxide leakage come to dominate total leakage in deep sub-100nm technologies, the benefits of the *basic+MUX* and *alternate+MUX* designs will be amplified.

Finally, we consider the benefits of the variant sleep mode, depicted in Figure 4.6. Leakage reduction results for the variant sleep mode relative to high-speed mode are shown in column 3 of Table 4.5. For comparison, column 2 of the table summarizes the sleep results already presented above for the original sleep mode. Observe that, for all but one of the switch types at both temperatures, the variant sleep mode offers better leakage results. The only exception is the *alternate* design at low temperature, for which similar leakage results are observed for both

Table 4.3: 85°C leakage power reduction results for *basic+MUX* design (unshaded) and *alternate+MUX* design (shaded).

Test scenario	**Avg. leakage pwr reduction (%) vs. high-speed mode (basic+MUX)**	**Avg. leakage pwr reduction (%) vs. high-speed mode (alternate+MUX)**
low-power mode (single fanout)	42.2%	33.0%
sleep mode	67.4%	64.0%
low-power mode (+ unused fanout)	42.0%	32.9%
low-power mode (+ used fanout)	41.3%	32.6%

Table 4.4: 25°C leakage power reduction results for *basic+MUX* design (unshaded) and *alternate+MUX* design (shaded).

Test scenario	**Avg. leakage pwr reduction (%) vs. high-speed mode (basic+MUX)**	**Avg. leakage pwr reduction (%) vs. high-speed mode (alternate+MUX)**
low-power mode (single fanout)	52.1%	47.8%
sleep mode	68.7%	75.8%
low-power mode (+ unused fanout)	51.4%	47.6%
low-power mode (+ used fanout)	50.2%	46.5%

Table 4.5: Sleep mode leakage results 85°C (unshaded) and 25°C (shaded).

Switch type	Orig. sleep mode % leakage reduction	Sleep mode variant % leakage reduction
basic	60.8%	77.1%
basic+MUX	67.4%	79.0%
alternate	61.3%	73.1%
alternate+MUX	64.0%	75.0%
basic	64.7%	66.9%
basic+MUX	68.7%	73.0%
alternate	72.3%	71.8%
alternate+MUX	75.8%	77.8%

sleep modes (72% leakage reduction). Pulling internal buffer nodes to a known voltage state in sleep mode ensures that there are at least two OFF transistors on each path from supply to ground in the buffer. This significantly reduces subthreshold leakage due to the stack effect (see Section 2.2.2). At high temperature, the variant sleep mode offers a 73-79% leakage reduction versus high-speed mode, whereas the original sleep mode offers a 61-67% leakage reduction.

4.5.3 Dynamic Power Results

The dynamic power characteristics of the switch designs in low-power mode were evaluated; the results are given in Table 4.6. Switching energy was computed by using HSPICE to the integrate the supply current drawn during logic transitions. The dynamic energy benefits of all of the switch designs are similar, ranging from 28-31%, and due chiefly to the reduced output swing and smaller short-circuit current in the buffer. Note, however, that this may represent an optimistic estimate of the dynamic power reduction. The area overhead of the new switch designs versus a traditional switch will lead to a larger base FPGA tile, resulting in longer wire segment lengths and increased metal capacitance (higher dynamic power). A precise measurement of the area overhead for incorporating the new switch designs into a commercial FPGA is difficult, as it depends on available layout space and existing transistor sizings, both of which are proprietary. Nevertheless, a rough estimate of the area overhead is attempted below.

As mentioned previously, the 16-input traditional switch used as the development basis

Table 4.6: Dynamic power results for all switch designs.

Switch type	Switching energy reduction in low-power vs. high-speed mode
basic	28.2%
basic+MUX	28.8%
alternate	30.9%
alternate+MUX	31.2%

requires 6 SRAM configuration cells. An additional cell to control the switch mode increases the SRAM cell count by ∼17%. Based on transistor width, the area overhead for the remainder of the *basic* switch design, versus the traditional switch, is estimated as ∼31%, mainly due to the need for relatively large sleep transistors. Certainly, routing switches in commercial FPGAs have additional configuration and test circuitry beyond that shown in Figure 4.2(b), which will reduce the area overhead of the proposed switches. Pessimistically, we can assume that deploying the *basic* switch design increases an FPGA's interconnect area by 30%, and that interconnect accounts for ∼2/3 (66%) of an FPGA's base tile area [Rahm 04]. Given this, the overall tile area increase to include the proposed *basic* switch amounts to ∼20%. Assuming a square tile layout, the tile length in each dimension would increase by ∼9.5%. However, the metal wire segment represents only a fraction of the capacitance seen by a switch output. Significant capacitance is due to fanout switches that *attach* to the metal segment. This "attached switch capacitance" is unaffected by a larger tile length. Thus, 9.5% is a loose upper bound on the potential increase in capacitance seen by a switch output. The capacitance increase is surpassed considerably by the dynamic power reductions offered by the *basic* switch. The projected tile area breakdowns for the traditional and *basic* switch types are summarized graphically in Figures 4.13(a) and (b), respectively.

As mentioned previously, the *alternate* switch design has a considerably lower area overhead compared to the *basic* design. Applying the same rough analysis used above, we expect that incorporating the *alternate* switch design into an FPGA would increase the base tile length in each dimension by only ∼6.5% [see Figure 4.13(c)]. The area overheads of the *basic+MUX* and *alternate+MUX* designs are similar those of the *basic* and *alternate* designs, since sleep transis-

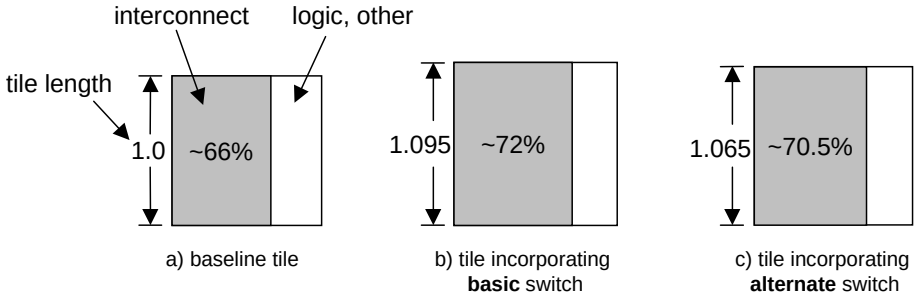


Figure 4.13: Projected tile area breakdown for traditional and proposed switch types.

tors MNX_M and MPX_M (see Figure 4.5) can be made small for the reasons mentioned in Section 4.3.

4.5.4 Summary of Results

In summary, the results show that all of the proposed switch designs have attractive qualities: the *basic* design offers large leakage reductions at high speed; the *alternate* design requires less silicon area; the *basic+MUX* and *alternate+MUX* designs offer the largest reduction in gate oxide leakage. The leakage/area/speed trade-offs between the switch designs are illustrated in Figure 4.14; the data values in the figure are normalized to those of the traditional switch design.

The leakage results presented above were for a single routing switch. A "back of the envelope" analysis can be used to project the overall leakage reductions offered by the proposed switch designs in an FPGA tile, which contains both logic and interconnect. Based on prior work, one can assume that ∼40% of leakage in a tile is in unused circuitry and ∼60% in used circuitry [Tuan 03]. Furthermore, as above, one can assume that about ∼66% of leakage in the used and unused circuitry is due to interconnect. Consider first the *basic* design with the variant sleep mode, operating at 85°C. In this design, leakage is reduced by 77.1% in sleep versus high-speed mode, and by ∼40% in low-power versus high-speed mode (see Tables 4.1 and 4.5). Assuming that all of the unused interconnect can be put into sleep mode, leakage

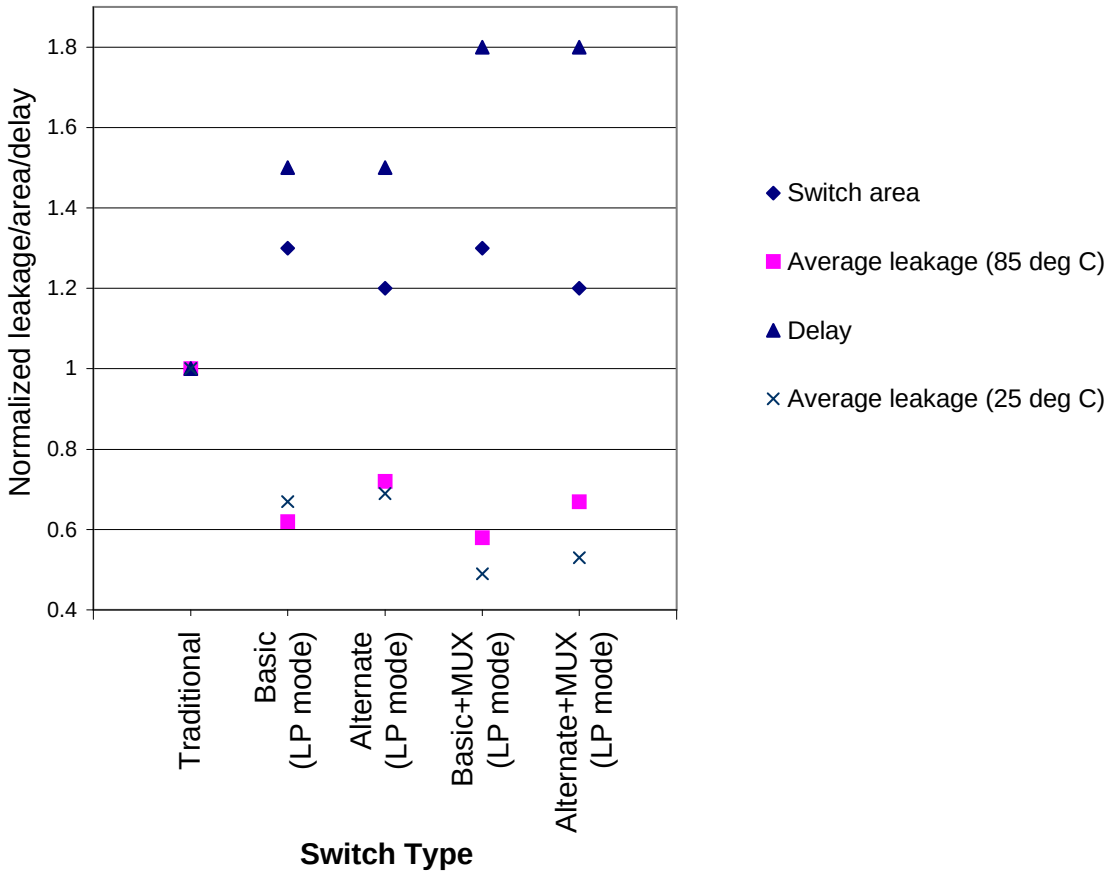


Figure 4.14: Leakage, area, and speed of switch designs.

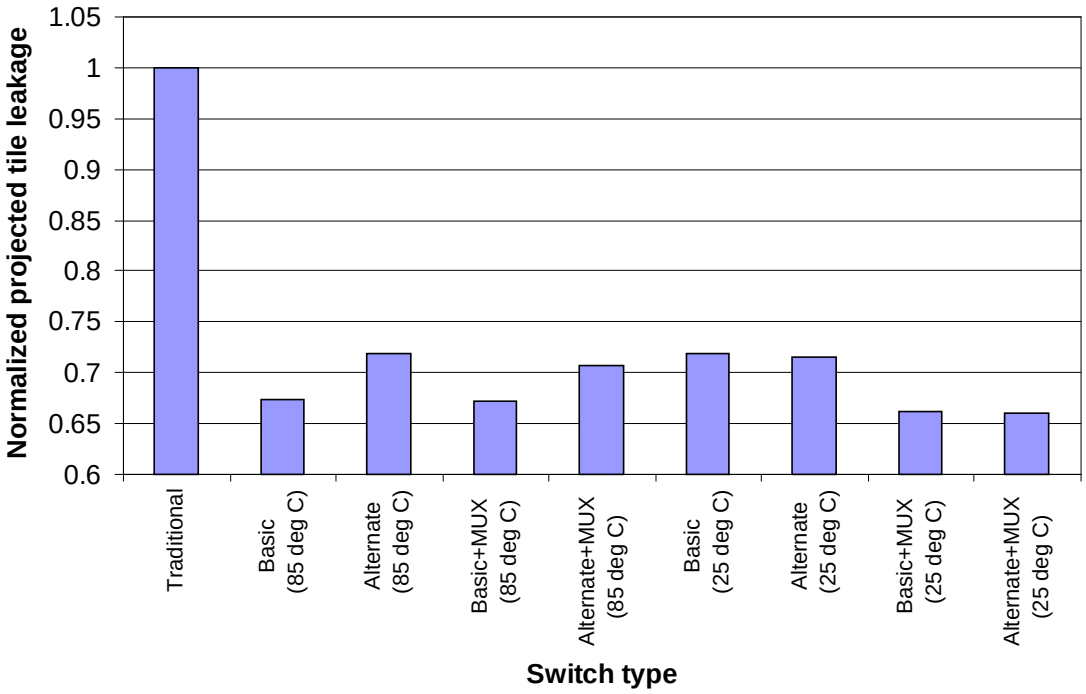


Figure 4.15: Overall leakage in FPGA tile.

in the unused part of a tile is reduced by: $0.66 \cdot 77.1\% = 50.9\%$. Leakage in the used part of a tile is reduced by: $0.66 \cdot 0.75 \cdot 40\% = 19.8\%$, where the "0.75" represents the average fraction of used interconnect that may be slowed down and operate in low-power mode (from the slack analysis). Given these partial results, the projected reduction in overall tile leakage for deploying the basic switch design is: $0.4 * 50.9\% + 0.6 * 19.8\% = 32\%$. Applying the same analysis to all the switch types produces the data in Figure 4.15. Note that the data in the figure corresponds to use of the variant sleep mode, which consistently offers better leakage.

4.6 Summary

Static and dynamic power dissipation in FPGAs is dominated by consumption in the interconnection fabric, making low-power interconnect a mandatory feature of future low-power

FPGAs. In this chapter, we proposed a number of new FPGA routing switch designs that can be programmed to operate in high-speed, low-power, or sleep mode. Each of the proposed designs offers a different power/area/speed trade-off. At high temperature, leakage reductions in low-power versus high-speed mode range from 28-42%. Depending on the design selected, such leakage reductions come with varying levels of performance and/or area overhead. At low temperature, leakage reductions range from 30-52% in low-power versus high-speed mode. Sleep mode leakage reductions range from 61-79% relative to high-speed mode. All of the proposed designs reduce dynamic power by up to 28-31%. An analysis of the timing slack in commercial FPGA benchmark circuits showed that the proposed switch designs are well-motivated. A majority of routing switches can be slowed down, and operated in low-power mode. The switch designs require only minor changes to a traditional FPGA routing switch and have no impact on router complexity, making them easy to deploy in current commercial FPGAs.

5 Power-Aware Technology Mapping

5.1 Introduction

In contrast to the two preceding chapters, which addressed leakage power optimization, this chapter focuses on optimizing an FPGA's *dynamic* power. Specifically, we concentrate on power reduction during the technology mapping step of the FPGA CAD flow. A number of techniques have been proposed in the literature for reducing FPGA power during technology mapping (see Section 2.5.3). A limitation of the prior work is that it has not explored the trade-offs between power and other optimization criteria. This chapter presents a new technology mapping algorithm that offers two key benefits: 1) It produces technology mapping solutions that consume less power than those produced by other, previously-published algorithms, and, 2) The proposed algorithm allows one to explore the depth/power curve, and therefore, to trade-off one criterion for the other. A novel feature of the algorithm is its approach to logic replication, which is shown to be generally undesirable from the power perspective. Furthermore, as part of this work, we demonstrate that different technology mapping approaches can produce solutions with widely varying power characteristics, despite having similar area and performance.

The chapter is organized as follows: Section 5.2 presents necessary background material. The technology mapping approach is described in Section 5.3. Section 5.4 presents an experimental study and the associated results. The impact of this work on other related research is briefly discussed in Section 5.5. A summary is provided in Section 5.6.

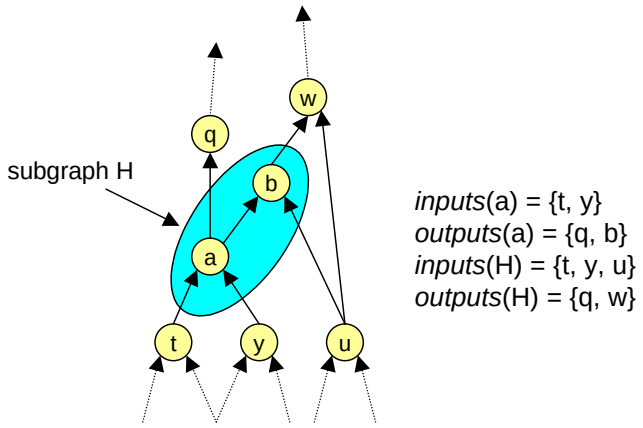


Figure 5.1: Circuit DAG definitions.

5.2 Preliminaries

Before presenting the technology mapping algorithm, we review some terminology. This chapter uses terminology similar to that in [Cong 94a].

The combinational part of a logic circuit can be represented as a Boolean network, which is a directed acyclic graph (DAG), $G(V, E)$, in which each node, $z \in V$, represents a single-output logic function and edges between nodes, $e \in E$, represent input/output dependencies between the corresponding logic functions. A primary input node is a node with an in-degree of 0; a primary output node has an out-degree of 0. For a node z in a circuit DAG, let $inputs(z)$ represent the set of nodes that are fanins of z, and $outputs(z)$ represent the nodes that are fanouts of z. For a subgraph, H, of a DAG, let $inputs(H)$ represent the set of nodes outside of H that are fanins of nodes in H; let $outputs(H)$ be the set of nodes that are outside of H that are fanouts of nodes in H. Figure 5.1 gives an example illustrating the subgraph, fanin, and fanout concepts. The *depth* of a node z, $D(z)$, is defined as the length (in nodes) of the longest path from any primary input to z.

A node x is said to be a predecessor of node z if there exists a directed path in the graph

from x to z. The subgraph consisting of a node z and all of its predecessors is referred to as the subgraph *rooted* at z. For any node z in a network, a *K-feasible cone* at z, N_z, is defined to be a subgraph consisting of z and some of its predecessors such that $|inputs(N_z)| \leq K$. A K-input LUT, or K-LUT, can implement any logic function with less than or equal to K inputs. Consequently, the technology mapping problem for K-LUTs can be thought of as "covering" an input Boolean network with K-feasible cones. Generally, there are many K-feasible cones for each node in the network, each having different area, delay, or power characteristics.

A concept closely related to K-feasible cone is that of *K-feasible cut*. A K-feasible cut for a node z is a partition, $(X, \overline{X})$, of the nodes in the subgraph rooted at z such that $z \in \overline{X}$, and the number of nodes in X that fanout to a node in $\overline{X}$ is $\leq K$. Figure 5.2(a) shows a network having an output node z. The figure shows two 4-feasible cuts for node z. There is a one-to-one correspondence between K-feasible cuts and K-feasible cones. Given a cut, $(X, \overline{X})$, the K-feasible cone is simply the subgraph induced by the nodes in $\overline{X}$. The problem of finding all possible K-LUTs that generate a node z's logic function is equivalent to the problem of enumerating all K-feasible cuts for node z.

To simplify the presentation of the proposed algorithm, for a K-feasible cut, $C_z = (X, \overline{X})$, for a node z, $Nodes(C_z)$ is used to represent the set $\overline{X}$, where $z \in \overline{X}$. $Support(C_z)$ is used to represent subset of nodes in X that fanout to a node in $\overline{X}$. For example, for cut 2 in Figure 5.2(a), $Nodes$(cut 2) = {c, z} and $Support$(cut 2) = {b, $i5$, $i6$}. Finally, $Cuts(z)$ is used to represent the set of all feasible cuts for a node z.

5.2.1 Power and Logic Replication

The aim of this work is dynamic power reduction, as computed by (2.7), and repeated here for convenience:

$$P_{avg} = \frac{F_{clk}}{2} \sum_{i \in signals} C(i) \cdot F(i) \cdot {V_{DD}}^2 \tag{5.1}$$

More specifically, the objective here is to reduce power dissipated in the FPGA interconnection network, since, as noted in Section 2.4, this accounts for roughly 60% of an FPGA's

dynamic power. For the remainder of this chapter, the term "power" is used to mean dynamic power dissipated in the interconnect.

Logic replication or duplication is an important concept in FPGA technology mapping. It is performed implicitly when a LUT covers a node that has fanout nodes both *inside* and *outside* the LUT. It is widely known that logic replication is necessary for depth minimization. Consider again the network shown in Figure 5.2. Figure 5.2(b) shows a mapping solution without logic replication, assuming LUTs with 4 inputs, where LUTs are shown as shaded, dashed regions. The duplication-free mapping has depth 2 and uses 3 LUTs. Figure 5.2(c) shows a mapping solution in which logic replication is permitted. This solution has a depth of 1, which is achieved by replicating node b. Observe that node b is covered by two different LUTs in the mapping solution.

When a node in a circuit is replicated for depth minimization, a connection from the node to one of its fanouts is "covered" within a LUT. In Figure 5.2(c), for example, both of the connections from node b to its fanouts are covered within LUTs. Such covered connections are not routed through the FPGA interconnection network, and therefore, do not contribute to interconnect power dissipation. The other consequence of node replication is that it generally increases the fanout of nodes that fanin to the replicated node. Referring again to Figure 5.2(b), primary inputs $i3$ and $i4$ each have one fanout LUT. In Figure 5.2(c), node b is replicated and therefore, primary inputs $i3$ and $i4$ must drive two LUTs. It is evident that replicating a node for depth minimization has two effects: 1) Connections from the replicated node to its fanouts may be covered within LUTs, and, 2) the fanout of nodes that fanin to the replicated node is generally increased.

Equation (5.1) specifies that power consumption depends linearly on switching activity. An important characteristic of switching activity in combinational circuits is that it typically decreases with circuit depth. Previous empirical research has shown, in fact, that activity falls quadratically with depth, on average [Nema 96]. This suggests that a node is likely to have fanins with higher switching activity than the switching activity at its output. Replicating

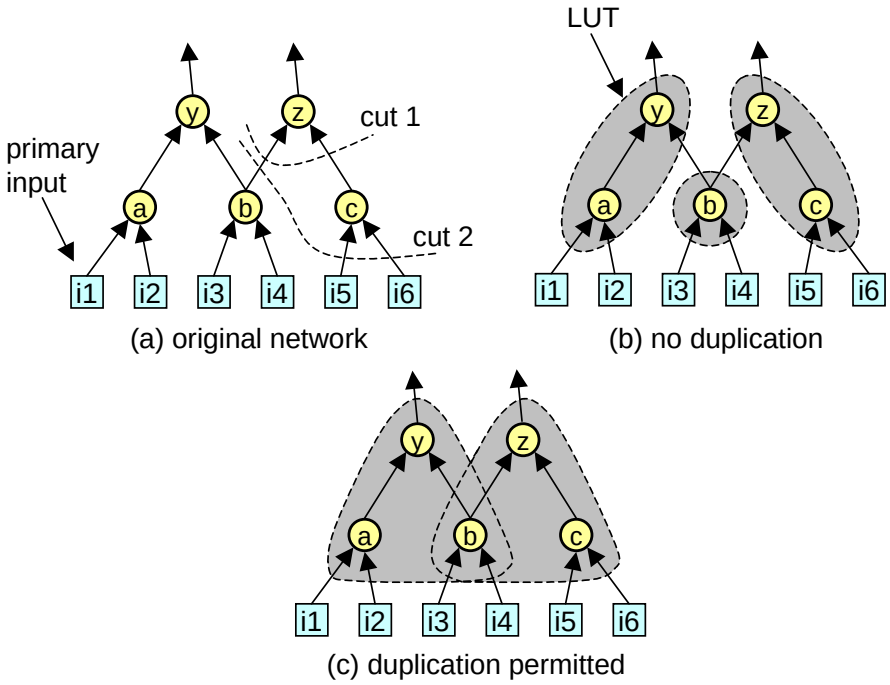


Figure 5.2: Illustration of feasible cuts; effect of logic replication in LUT mapping.

a node for depth minimization covers a fanout of the node within a LUT, but increases the fanout of the node's fanins. The activity/depth relationship implies that the activity on the signals whose fanout (and capacitance) has been increased is likely higher than the activity on the signal whose fanout has been decreased (by covering a connection within a LUT). Consequently, it is apparent that logic replication in LUT mapping is generally undesirable from a power perspective, except in specific cases, depending on the switching activities local to a node. This notion is applied in the proposed technology mapping algorithm, which includes an activity-based penalty for the replication of a node.

5.3 Algorithm Description

The proposed technology mapping algorithm operates in three phases, and has a high-level flow similar to that used in [Cong 99] and [Wang 01]. In phase 1, the set of K-feasible cuts for each node in the network is computed. In phase 2, the costs for the cuts generated in phase 1 are computed, and a "best cut" is selected for each node. In phase 3, the best-cost cuts are used to transform the Boolean network to produce the final LUT mapping solution. Each of these phases is described below in detail.

5.3.1 Generating K-Feasible Cuts

The input to the cut generation process is a K-*bounded* Boolean network, $G(V, E)$, where K-bounded implies that:

$$\forall_{v \in V} |inputs(v)| \leq K \tag{5.2}$$

Traversing the input Boolean network from primary inputs to primary outputs, the cuts for each node z are generated by merging cuts from its fanin nodes, using the method described in [Cong 99, Schl 94]. At a high level, this works as follows: Consider generating the K-feasible cuts for a node z with two fanin nodes, a and b. The list of K-feasible cuts for a and b have already been computed, owing to the network traversal order. Say that node a has two K-

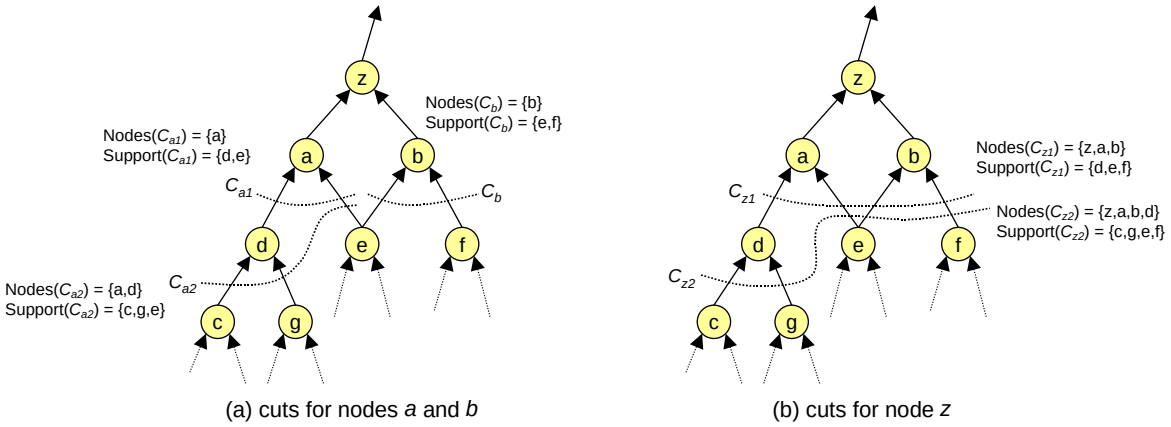


Figure 5.3: Generating the K-feasible cut sets.

feasible cuts, C_{a1} and C_{a2}, and node b has one K-feasible cut, C_b, as shown in Figure 5.3(a). We can merge C_{a1} and C_b to create a cut, C_{z1}, for node z, such that $Support(C_{z1}) = Support(C_{a1}) \cup Support(C_b)$ and $Nodes(C_{z1}) = z \cup Nodes(C_{a1}) \cup Nodes(C_b)$ [see Figure 5.3(b)]. Clearly, if $|Support(C_{z1})| > K$, the resulting cut is not K-feasible, and it is therefore discarded. Similarly, one can merge C_{a2} and C_b to create another candidate cut, C_{z2}, for node z. This provides a general picture of how the cut generation procedure works; however, there are several special cases to consider, and the interested reader is referred to [Schl 94] for complete details.

Note that other LUT-based technology mapping algorithms prune the cut-set for each node in order to reduce run-time [Cong 99]. Although an upper bound on the number of cuts for a node is $O(n^K)$, where n is the number of nodes in the circuit ($n = |V|$), we have observed that in actual circuits, the set of all cuts can be computed quickly when the target LUTs are small, as in commercial FPGAs. Thus, for this work, it was not necessary to implement and apply pruning techniques such as [Cong 99], although this is certainly possible.

5.3.2 Costing Cuts

After computing the set of K-feasible cuts for each node in the network, the network is again traversed from primary inputs to primary outputs. The objective of this second traversal is to select a best cut for each node. The best cut for a node is determined using a cost function with several components, reflecting depth ($DCost$), power ($PCost$), as well as a logic replication cost ($RCost$), to be described below. As will be shown, the cost components are defined in a way that freely permits logic replication in depth-critical portions of a circuit, and penalizes logic replication in non-depth-critical portions of a circuit. For a node z with a K-feasible cut, C_z, the cost of the cut is defined as:

$$Cost(C_z) = \alpha \cdot DCost(C_z) + \beta \cdot PCost(C_z) + \gamma \cdot RCost(C_z) \tag{5.3}$$

where the parameters α, β, and γ are coefficients reflecting the relative importance of each term. After computing the cost of the K-feasible cuts for node z, the best cut is selected to be the one with the minimum cost. We refer to the best cut for a node z as $BestCut(z)$.

We now elaborate on the terms of (5.3). The depth cost of a cut, C_z, is defined to be the depth of the LUT mapping solution of the subgraph rooted at node z, if $Nodes(C_z)$ is implemented as a LUT in the mapping solution. That is:

$$DCost(C_z) = 1 + \max_{v \in Support(C_z)} \{DCost[BestCut(v)]\} \tag{5.4}$$

Thus, to compute the depth cost of cut C_z, (5.4) considers the depth cost of the best cut for each node, v, that fans out to a node in $Nodes(C_z)$. For each of these support nodes, the best cut has already been selected since the network is being traversed in an input-to-output fashion. Primary input nodes are assigned a depth cost of zero.

For power optimization, the goal is to keep connections with high switching activity out of the FPGA interconnect. The algorithm aims to "capture" as many high-activity connections as possible within LUTs, leaving only low-activity connections between LUTs. The power cost

of cut C_z, for node z, is therefore defined as:

$$PCost(C_z) = \sum_{v \in Support(C_z)} [F(v) + PCost(BestCut(v))] - \sum_{w \in Nodes(C_z)} [F(w) \cdot |outputs(w) \cap Nodes(C_z)|] \quad (5.5)$$

where $F(x)$ represents the switching activity of the signal driven by a node x. The first summation tallies the activities of the connections in the mapping solution of the subgraph rooted at z. The first term in the first summation represents the switching activities of nodes that fanout to a node in $Nodes(C_z)$. The signals driven by these nodes will need to be routed through the interconnect if $Nodes(C_z)$ is implemented as a LUT in the mapping solution; hence, they contribute to higher cost. The second term in the first summation represents the power cost of the mapping solutions rooted at each of the support nodes. The second summation term, whose sign is negative, represents the sum of the fanout-weighted switching activity on the connections that have been captured inside a LUT, namely, the LUT implementing the functionality of $Nodes(C_z)$. For each node w in $Nodes(C_z)$, the summation counts the number of w's fanouts that are in $Nodes(C_z)$, and multiplies this count by the activity of the signal driven by w.

Prior to defining the replication cost term of (5.3), several additional concepts must be introduced, including slack, slack weight, and the notion of "replicated nodes". The slack of a node z, $Slack(z)$, is defined to be the number of levels in the circuit's Boolean network DAG by which the depth of node z may be increased, without increasing the overall depth of the DAG. For example, if a node has slack 0, then its depth cannot be increased without also increasing the overall depth of the DAG. A node with slack 1 can have its depth increased by 1 level without affecting the overall depth of the DAG. Formally,

$$Slack(z) = MaxDepth - [D(z) + DO(z)] \quad (5.6)$$

where $MaxDepth$ is the maximum combinational depth of any primary output, and $DO(z)$

is the length of the longest path from z to any primary output. The slack of each node in a DAG can be computed in $O(|V|)$ time by first performing a forward breadth-first search from the DAG's primary inputs to its primary outputs, and computing depth for each node and $MaxDepth$. Then, a reverse-order breadth-first search from the DAG's primary outputs to its primary inputs is performed, and the value of DO for each node is computed. The maximum slack, among all nodes in the network, is represented by $MaxSlack$. Using the slack and maximum slack, the slack weight of a node z is defined as:

$$SlackWeight(z) = 1 + \kappa \cdot \left[\frac{MaxSlack - Slack(z)}{MaxSlack}\right] \tag{5.7}$$

where κ is a positive real number. The definition of slack weight implies that nodes with 0 slack have a slack weight of $1 + \kappa$, and nodes with $MaxSlack$ have a slack weight of 1. The slack values and slack weights of nodes are computed up-front in the circuit's unmapped DAG. We have observed that a node's slack in the unmapped DAG generally correlates well with the slack of the node's covering LUT in the mapped network.

For a cut, C_z, for node z, the replicated nodes, $RNodes(C_z)$, in $Nodes(C_z)$ are those nodes that fanout to a node outside of $Nodes(C_z)$[1]. This is illustrated in Figure 5.4, where $Nodes(C_z) = \{z, a, b\}$ and $RNodes(C_z) = \{a\}$. If $Nodes(C_z)$ is implemented as a LUT in the mapping solution, then the logic function of node a must be replicated in a second LUT, since it has fanouts outside the first LUT. The formal definition of the replicated nodes in a cut C_z is:

$$RNodes(C_z) = \{v \in Nodes(C_z) \mid v \neq z, (\exists\, u \mid u \in outputs(v), u \notin Nodes(C_z))\} \tag{5.8}$$

We are now ready to define the replication cost of a cut C_z:

$$RCost(C_z) = \frac{\sum_{v \in RNodes(C_z)}[(\sum_{u \in inputs(v)} F(u)) - \lambda \cdot F(v) \cdot |outputs(v) \cap Nodes(C_z)|]}{SlackWeight(z)} \tag{5.9}$$

[1]Root node z is not included in the set of replicated nodes.

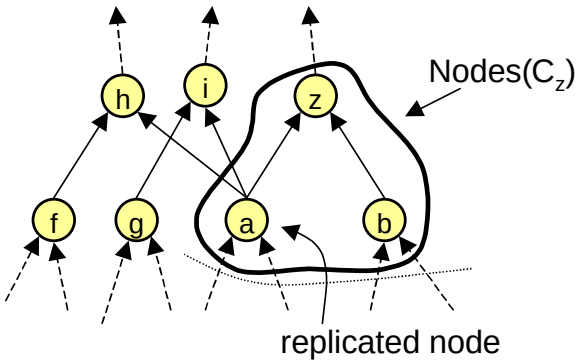


Figure 5.4: Identifying the replicated nodes.

The first summation in the numerator is over the replicated nodes. For each replicated node, v, the activities of the signals driven by v's fanins are tallied. Recall that in Section 5.2.1, we showed that replicating a node generally increases the fanout and load capacitance of its fanin nodes. The consequences of this on power consumption depend on the activities of the signals driven by these fanin nodes, and hence, *RCost* is increased in proportion to these activities. The second term in the square brackets is negative, and its intent is similar to the second summation term in (5.5). By including a replicated node v in a LUT, a subset of its fanout connections are captured within the LUT. The *RCost* is reduced in proportion to the product of the number of captured connections and activity of these captured connections. λ is a coefficient whose value was determined empirically by manually trying different values and evaluating the mapping results produced. Its value was set to 0.5 for the experiments.

It is important to recognize that to minimize the overall depth of a mapped circuit, it is not mandatory that all nodes in the circuit be mapped with minimal depth. Only some nodes play a significant role in influencing overall depth. The rationale for dividing by the slack weight in (5.9) is to reduce the replication cost for nodes whose depth in the mapping solution is likely to impact the overall depth of the mapped circuit. Nodes with low slack will have a slack weight that is substantially larger than one. For such "critical" nodes, it may be more important to

select a best cut that optimizes depth, rather than one that avoids logic replication. This is achieved by reducing the replication cost through dividing by the larger slack weight.

5.3.3 Mapping

The mapping phase of the algorithm is similar to that of FlowMap [Cong 94a]. A FIFO queue is initialized to contain all of the primary output nodes. A node, v, is removed from the queue and its best cut, $C_v = BestCut(v)$, is recovered. The subnetwork corresponding to $Nodes(C_v)$ is implemented as a LUT in the mapping solution. Each node in $Support(C_v)$ is then added to the end of the FIFO queue, if it is not already in the queue. The process of removing nodes from the queue, using their best cuts to establish LUTs in the mapping solution, and adding the support of these cuts to the end of the queue continues until the queue contains only primary inputs. When this condition is met, the input Boolean network has been fully mapped into LUTs.

5.4 Experimental Study and Results

5.4.1 Methodology

The algorithm described in Section 5.3 has been implemented in the C language within the Berkeley SIS framework [Sent 92]. To evaluate the algorithm, 29 of the largest MCNC combinational circuits were used (each circuit uses $>$ 300 LUTs). Prior to technology mapping, each benchmark circuit was optimized in SIS using *script.rugged* [Sent 92], and then transformed into a network of 2-bounded functions using *dmig* [Chen 92].

The technology mapper was compared with two publicly available mappers: 1) FlowMap [Cong 94a], which maps circuits in a depth-optimal manner; and, 2) FlowMap-r [Cong 94b], which optimizes both depth and area by relaxing the depth optimality on portions of a circuit that are not depth-critical. FlowMap-r performs duplication-free mapping on non-critical portions of a circuit. Additionally, this study evaluates the effect on power of using various area-reducing

post-processing routines, including FlowPack (FP) [Cong 94a], and MP-Pack (MP)[2] [Chen 92].

In the proposed algorithm, parameters α and β in (5.3) were set to be 1 and 0.0001, respectively, reflecting a preference for optimizing depth over power. An iterative, automatic approach was implemented to select a value for parameter γ (the replication cost weight) individually for each circuit, such that power was minimized, while meeting certain depth constraints, described below. To compute the value of γ for each circuit, γ was initially set to a small value, and then increased gradually. For each value of γ considered, steps 2 and 3 of the algorithm were invoked (costing and mapping), and the mapping solution with the best power characteristics was tracked. Generally, as γ is increased, more logic replication is eliminated, reducing power and increasing depth. In practice, a binary search could be used to select a value for γ, most likely with a small reduction in quality. Following mapping, MP-Pack is called as a post-processing routine on each mapped circuit.

To estimate power consumption using (5.1), the capacitance of each net is required. Actual net capacitance is not known until after layout is complete. Therefore, capacitance was estimated through an empirically-derived model based on structural properties of the circuit. The model estimates each net's capacitance based on its fanout, and is similar to ASIC wire load models (e.g., [Synopsys 04]), and the capacitance models applied in prior FPGA technology mapping work [Li 01]. The capacitance model was developed by using VPR [Betz 97b] to place and route the benchmark circuits, mapped using the FlowMap-r algorithm. The logic block in the targeted FPGA architecture contained a cluster of four 4-LUT/flip-flop pairs. The routing network was comprised of wire segments of length 4 (span 4 logic blocks), with half of the routing switches being buffered, and half unbuffered. This FPGA architecture was shown to be efficient from both the area and delay perspective [Betz 99b]. VPR's built-in interconnect model was used with resistance and capacitance values based on a $180n$m TSMC process [TSMCPROC 02]. Following the placement and routing of each circuit, the capacitance data for each net was extracted, including all metal and transistor capacitance. In generating

[2]MP-Pack was executed with node duplication off.

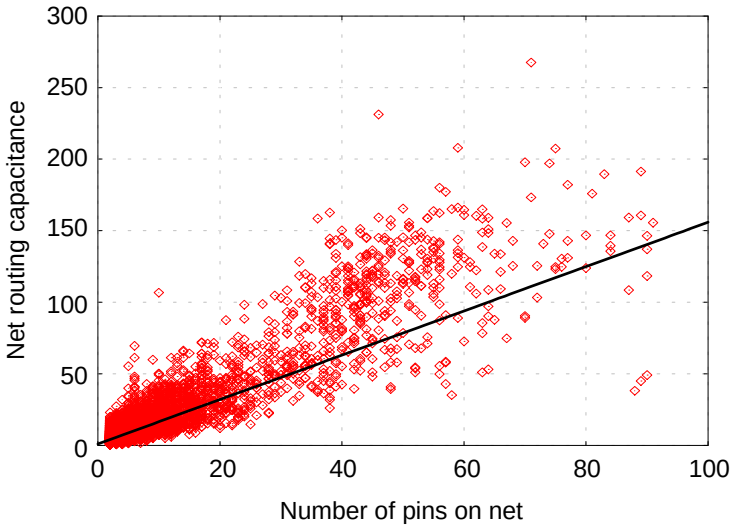


Figure 5.5: Routing capacitance versus # of net pins.

the capacitance model, only the routing of nets between logic blocks was considered; the connections *within* a logic block were ignored during model generation. The results of the capacitance analysis and model construction are shown in Figure 5.5. The horizontal axis represents the number of pins per net; the vertical axis shows total net capacitance. Each point in the figure represents the capacitance of a single net in one of the benchmark circuits. In addition to the raw data, the figure shows a line of best fit, which has the following equation:

$$C_i = 1.05 + 1.55 \cdot [FO(i) + 1] \tag{5.10}$$

where $FO(i)$ represents the fanout of net i. Equation (5.10) is used in computing the power results presented below. However, as is evident in Figure 5.5, a net with a given number of pins can have a range of capacitance values. Consequently, power estimates made using (5.10) may be inaccurate. Section 5.5 describes follow-up research work that has validated the power results after physical layout.

To measure power using (5.1), one also needs a switching activity value for each net. This

value was computed using the power characterization capabilities that are built-in to SIS. Specifically, for each primary input, i, SIS allows one to specify the input's *static probability*, $P(i)$, which, as described in Section 3.3, is the probability that the value at the primary input is logic-1 during circuit operation. SIS propagates primary input static probabilities through the network to yield a probability for each internal signal. The activity value for the signal driven by an internal node, n, is computed by SIS using $F(n) = 2 \cdot P(n) \cdot [1 - P(n)]$ [Ciri 87]. For most of the results in this paper, the static probability of all primary inputs was set to 0.5, corresponding to a primary input activity of $2 \cdot (0.5) \cdot (1 - 0.5) = 0.5$. For a limited set of results, the effect of this choice was investigated by re-computing the power of already-mapped solutions using randomly chosen primary input signal probabilities.

5.4.2 Results

The first experimental scenario considered is that of mapping circuits into 4-LUTs in a depth-optimal manner. Figure 5.6 shows the average power, area (# of LUTs), and number of connections between LUTs for this case. The power for a circuit was computed by first estimating the capacitance of each net (between LUTs) using (5.10); total circuit power was then computed using (5.1). Figure 5.6 includes the number of connections since this metric correlates with overall routing complexity – it represents the total number of net load pins to be routed by the routing step of the CAD flow, which roughly corresponds to routing demand. The numbers in Figure 5.6 were computed by first determining the increase in power, area, and number of connections for each mapped circuit in comparison with the mapping solution produced by the proposed algorithm. The increases were then averaged across all circuits for a given mapping approach.

Figure 5.6 shows that the mapping solutions produced by the proposed approach have substantially less power dissipation than those produced by other approaches. The most competitive alternate method is FlowMap-r followed by MP-Pack; the solutions produced using this technique require 14.2% more power than the proposed algorithm, on average. Table 5.1 shows

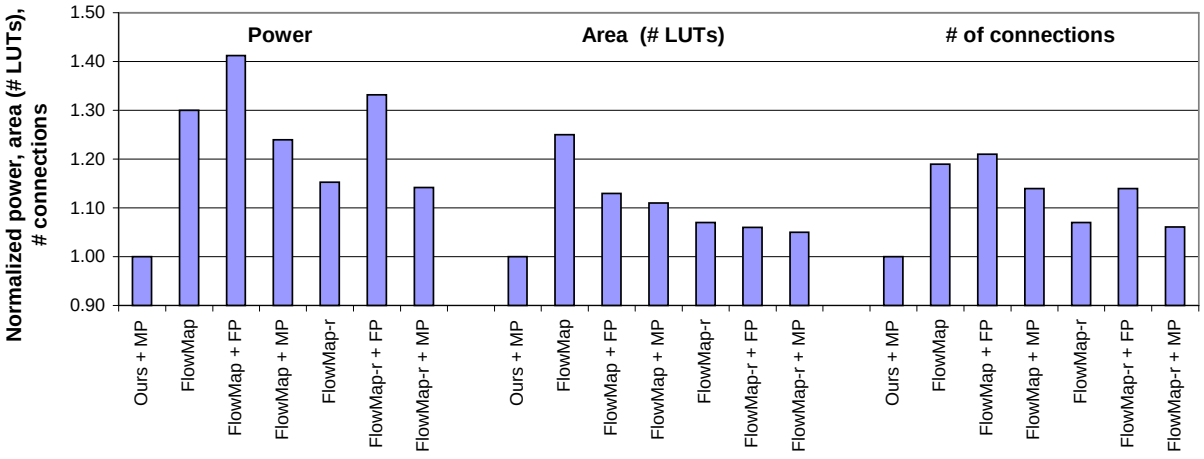


Figure 5.6: Power, area, number of connections in depth-optimal 4-LUT mapping solutions.

detailed results comparing the proposed algorithm with FlowMap-r + MP-Pack for each of the 29 circuits. Observe that the gains offered by the new algorithm are consistent. Specifically, the power dissipation was equal to, or better than FlowMap-r + MP-Pack for 26 of 29 circuits, with 2 of the 3 degradations being only 1%. Further, the algorithm also improves the area and number of connections slightly (≈5-6%); hence, it is likely that the results will remain valid after layout.

Figure 5.6 shows that for optimal depth 4-LUT mapping solutions, power can vary by as much as 40% on average, depending on the mapping approach used. The power variation is considerably larger than the variation in area or the number of connections, which can change by about 25% and 20%, respectively. Thus, it is apparent that simply knowing a circuit has been mapped in a depth-optimal manner does not allow one to make inferences regarding power.

Another interesting feature of Figure 5.6 is that it shows the effect of the post-processing routines. One can see that FlowPack (FP) reduces the number of LUTs in mapping solutions substantially; however, it increases the power, as well as the number of connections to route. FlowPack is a variation on FlowMap that maximizes cut volume; that is, it maximizes the number of nodes that are packed into each LUT, permitting logic replication. The logic replication

Table 5.1: Detailed results for depth-optimal 4-LUT mapping solutions.

Circuit	**Power increase FlowMap-r + MP vs. ours + MP**	**Area increase FlowMap-r + MP vs. ours + MP**	**# conns increase FlowMap-r + MP vs. ours + MP**
C3540	1.11	1.09	1.05
C5315	1.02	1.03	1.04
alu4	1.20	1.11	1.11
apex1	1.14	1.00	1.00
apex2	1.15	1.08	1.07
apex3	1.19	1.10	1.09
apex4	1.03	1.01	1.01
apex5	1.05	1.05	1.03
cordic	1.08	1.04	1.06
cps	1.16	1.06	1.05
dalu	1.16	1.03	1.07
des	1.06	1.01	1.02
ex1010	1.73	1.17	1.17
ex4p	1.07	1.04	1.07
ex5p	1.26	0.97	0.97
frg2	1.00	1.00	1.03
i10	0.87	0.80	0.79
i8	1.23	1.19	1.18
i9	1.42	1.59	1.57
k2	1.03	0.98	0.97
misex3	1.24	1.05	1.08
misex3c	1.20	1.12	1.11
pair	0.99	0.97	0.96
pdc	1.34	1.03	1.02
rot	0.99	1.00	0.99
seq	1.07	1.02	1.04
spla	1.14	1.00	1.01
table3	1.10	1.03	1.04
table5	1.09	0.99	1.04
Average:	1.14	1.05	1.06

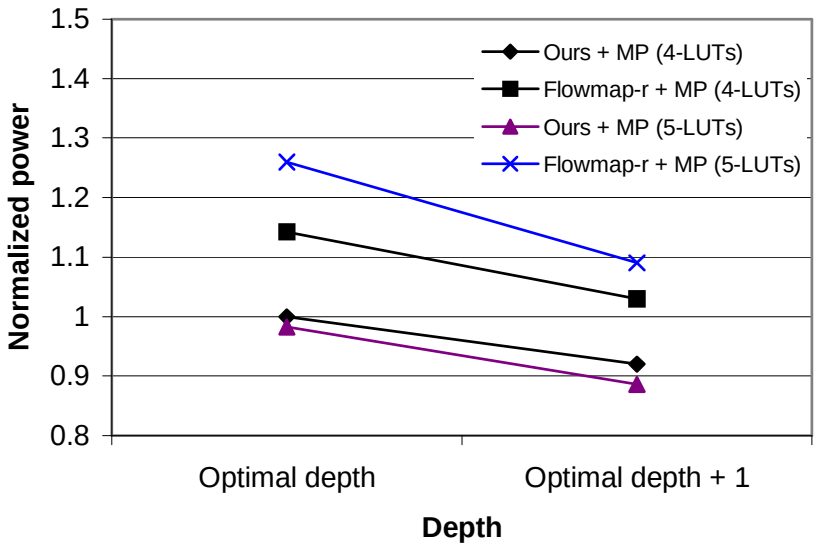


Figure 5.7: Power results for other depths, 5-LUTs.

performed by FlowPack may lead to increased net fanout and higher power. On the other hand, the data in Figure 5.6 show that MP-Pack is about as effective as FlowPack in reducing area, and also reduces both power and the number of connections to route.

Figure 5.7 shows how power varies when the optimal depth constraint is relaxed and circuits are mapped with optimal depth + 1. Results are given for the new algorithm, as well as FlowMap-r + MP-Pack for 4-LUTs, and larger, 5-LUTs. Again, the numbers in the figure are normalized to the 4-LUT, depth-optimal solutions produced by our algorithm. As depth is increased, greater amounts of logic replication can be eliminated, which permits further reductions in power. Specifically, relaxing the depth constraint by one level allows the new algorithm to reduce power by about 8% over the depth-optimal case for 4-LUTs, and 10% for 5-LUTs. Note that the 4-LUT mapping solutions produced by FlowMap-r + MP-Pack with relaxed depth use more power than the depth-optimal solution produced by the new approach.

Modern commercial FPGAs contain 4-input LUTs; however, some devices allow two 4-LUTs to be combined into a 5-LUT [Virt 03]. Hence, mapping to 5-LUTs is also an important

problem. The results in Figure 5.7 show that for FlowMap-r + MP-Pack, 5-LUT mapping solutions actually require more power than the 4-LUT solutions; whereas, for the new algorithm, the 5-LUT solutions require slightly less power than the 4-LUT solutions. The depth of the 5-LUT mapping solutions is generally smaller than the depth of the 4-LUT mapping solutions. It appears to be more expensive from the power viewpoint for FlowMap-r to achieve this smaller depth. Figure 5.7 shows that the improvements offered by the new algorithm over FlowMap-r are larger for 5-LUTs than 4-LUTs. The larger LUTs can cover a larger portion of the input network, and thus, appear to offer more potential for power optimization. For one of the circuits, *spla*, the depth-optimal 5-LUT solution produced by FlowMap-r + MP-Pack used 7 times more power than the solution produced by the proposed algorithm. Since this circuit affected the average substantially, it was not included in the data used to create Figure 5.7. If this circuit is included, the average power of the 5-LUT mapping solutions of FlowMap-r + MP-Pack increases to nearly 1.5 for the depth optimal case, and 1.16 for the relaxed depth case.

The power results presented above were computed based on the primary inputs of each circuit being set to have identical switching activities (0.5). A problem that arises frequently in low-power synthesis is that input switching activities are not known at synthesis time, or the set of switching activities used during synthesis do not reflect the stimulus applied to a circuit in actual field operation. To investigate the dependence of the results on switching activities used during technology mapping, the power for already-mapped circuits was re-computed using randomly chosen input switching activities. Specifically, the static probability for each primary input was set to a random number between 0.1 and 0.9, corresponding to a randomly chosen switching activity between 0.18 and 0.5. The power of two sets of depth-optimal 4-LUT mapping solutions was re-computed: those produced by the new algorithm, as well as those produced by FlowMap-r + MP-Pack. The results showed that the improvements offered by the new algorithm over FlowMap-r + MP-Pack degraded by less than 1%, on average. Thus, the power benefits remain considerable, even when switching activities deviate from those used during technology mapping.

When this work was first published in [Ande 02], perhaps the most appropriate existing algorithm to compare against was PowerMap [Li 01], which, like our algorithm, optimizes depth and power. Unfortunately, it was not possible to compare directly with PowerMap. However, the results in the PowerMap paper compare PowerMap with FlowMap for LUTs with 5-inputs, showing a power improvement of about 15%. The results in Figure 5.6 and 5.7 show that the proposed approach provides gains over FlowMap that exceed this margin.

5.5 Impact of Research

It is worth mentioning that, since this work was published in [Ande 02], several researchers have cited it, extended it, and improved slightly on the results presented above.

In the experimental study above, circuit power was estimated based on pre-layout predictions of interconnect capacitance, as no power-aware FPGA layout tools were available with which to place and route the generated mapping solutions. In [Lamo 03], Lamoureux and Wilton incorporate the proposed techniques into a complete power-aware CAD flow, including layout tools. Their results validate the projected power reductions given above, and show that the power benefits largely do "hold up" when post-layout capacitance data is available.

In [Chen 04b], UCLA researchers cite and apply the activity-based logic replication concept presented here, and include this in a mapping algorithm for FPGAs with dual supply voltages. The mapping algorithm they propose is similar to ours, though the cost function used to select the best cut is simplified. They compared their approach directly with ours, and achieved additional power reductions of 1-2%.

5.6 Summary

This chapter focused on reducing dynamic power dissipation and presented a new algorithm for power-aware mapping that allows one to trade-off depth and power. A novel aspect of the proposed approach is that it takes an activity-aware approach to logic replication, which

has been shown to significantly affect the power of technology mapped circuits. Experimental results show the proposed algorithm produces solutions that require less power than competing techniques. It was also shown that, for a given depth of mapping solution, circuit power can vary considerably, depending on the technology mapping approach used and the choice of area-reducing post-processing routine. Results show a clear trade-off between mapping depth and power. Additional power reductions of 8-10% are possible when the requirement of depth optimality is relaxed by one logic level.

6 Power Prediction Techniques

6.1 Introduction

Designing complex digital systems with millions of gates often involves teams of engineers and a lengthy design cycle. Power-aware design for large systems requires not only power optimization techniques, such as those described in the previous chapters, but also requires efficient estimation tools that can assess power at early design phases. The availability of accurate, early power estimates will reduce the required number of iterations through the entire CAD flow, allowing design trade-offs to be evaluated at an abstract level, decreasing design time and cost.

Recall that the dynamic power consumed by a signal is proportional to the signal's interconnect capacitance, as well as the rate of logic transitions on the signal – the signal's switching activity. One can conceive of several different views of switching activity, depending on how circuit delays are accounted for. First, activity values can be computed assuming logic and routing delays are zero (*zero-delay activity*). Second, activity values can be computed considering logic delays, but not routing delays (*logic-delay activity*). Third, activity values can be computed considering complete logic and routing delays (*routed-delay activity*). Section 2.2 outlined different approaches for computing switching activity and described the notion of glitching, which leads to increased activity.

An understanding of how switching activity changes when delays are considered is important for several reasons. First, since FPGA power dissipation is dominated by interconnect, the consequences of glitching on total power consumption may be more severe in FPGAs versus

ASICs. In addition, due to the presence of programmable switches in the interconnection network, path delays in FPGAs are generally dominated by interconnect rather than logic delays, suggesting that the severity of glitching could conceivably be greater in FPGAs than in ASICs[1]. Another reason to study switching activity is that low-power synthesis techniques may perform optimizations on the basis of zero-delay switching activity data [Li 01, Farr 94], with the assumption that such data correlates well with routed-delay activity data. It is unknown whether this assumption is valid for FPGA technology.

In addition to switching activity, computing dynamic power using (2.7) (see page 11) requires the capacitance of each net. Early capacitance prediction for FPGAs is not well-studied and the pre-fabricated, programmable nature of FPGA interconnect makes the capacitance prediction problem for FPGAs significantly different from the corresponding problem in ASICs. The dominant role of interconnect in total FPGA power consumption implies that characterization and management of net capacitance is a crucial part of a power-aware FPGA CAD flow.

This chapter focuses on estimating the dynamic power consumed by FPGA interconnect, and specifically, two separate problems in FPGA power estimation are studied: switching activity prediction and interconnect capacitance prediction. Models are proposed for predicting these parameters prior to routing completion, using the Xilinx Virtex-II PRO commercial FPGA [Virt 03] as the investigation vehicle. The proposed models can be applied in a variety of scenarios, such as low-power synthesis systems, power-aware layout synthesis, and early power prediction, when accurate routing data is incomplete or unavailable. The chapter is organized as follows: Section 6.2 provides relevant background and gives an overview of the prediction methodology. Section 6.3 considers *pre-layout* switching activity prediction. To motivate the work, we study switching activity and examine whether zero-delay activity values can be used reliably as estimates of routed-delay activity. An activity prediction model is presented that estimates the routed-delay activity of a net using the net's zero-delay or logic-delay activity, together with functional and structural properties of a circuit. Section 6.4 deals with interconnect

[1] Path delays in modern ASICs are also dominated by interconnect delays [Cong 97]. However, this property is even more pronounced in FPGAs due to the interconnect switch delays and long wire segments.

capacitance prediction at the *placement* stage. One of the main results here is that capacitance is not well-approximated by generic parameters, such as a net's bounding box half-perimeter. Prediction accuracy is improved significantly when architectural aspects of the FPGA interconnect are considered. An important contribution of this work is the observation of significant "noise" in both the capacitance and activity of nets. In this context, the term "noise" does not refer to signal integrity issues, such as crosstalk. Rather, in this chapter, "noise" refers to an inherent uncertainty in activity and capacitance that imposes limits on the accuracy achievable by any predictor. A summary is given in Section 6.5.

6.2 Background

6.2.1 Target FPGA Architecture

The Virtex-II PRO FPGA targeted in this study is similar to the Virtex-4 FPGA described in Section 2.3. Like Virtex-4, Virtex-II PRO consists of a two-dimensional array of programmable logic and interconnect resources. A Virtex-II PRO logic block (CLB) contains four SLICEs and is similar to the Virtex-4 CLB depicted in Figure 2.9. A SLICE contains two 4-LUTs, called the F-LUT and the G-LUT. Virtex-II PRO interconnect is similar to Virtex-4 interconnect, with the exception that instead of there being length-24 wire segments (as in Virtex-4), Virtex-II PRO contains wire segments that span the entire length/width of the device.

It is worth mentioning that although the prediction models proposed here are based on Virtex-II PRO, we believe the techniques are generic and can easily be adapted to other popular FPGA families. For example, the Altera Stratix FPGA [Stra 03] has logic and routing structures similar to Virtex-II. A basic tile in Stratix is called a logic array block (LAB), and it contains 10 4-LUT/FF pairs (versus 8 4-LUT/FF pairs in a Virtex-II CLB). Stratix interconnect consists of variable-length wire segments and buffered routing switches. The LAB LOCAL and DIRECT interconnect in Stratix correspond to the CLB LOCAL and DIRECT interconnect in Virtex-II. Furthermore, Stratix has routing resources that span lengths of 4, 8, and 24/16 LAB tiles, which roughly resemble the DOUBLE, HEX and LONG resources in Virtex-II. Due to such

architectural similarities, we expect the proposed techniques to be widely applicable and not limited to use with Virtex-II.

6.2.2 Prediction Methodology Overview

One objective of this work is the construction of models for early prediction of a net's routed-delay activity and interconnect capacitance, which are referred to as the *target parameters*. The proposed prediction models are mathematical functions of a second set of parameters called *prediction parameters*, whose values are known prior to routing completion. The prediction parameters for activity and capacitance prediction are described in subsequent sections. The following set of steps convey the general methodology taken to build the prediction models:

1. A set of benchmark circuits were selected and mapped into Virtex-II PRO. Circuits were synthesized from VHDL using Synplicity's Synplify Pro tool (version 7.0), and then technology mapped, placed, and routed using Xilinx tools (version M5.2i)[2]. Each circuit was mapped into the smallest FPGA device able to accommodate it. Table 6.1 provides detail on the benchmark circuits.

2. The circuits were arbitrarily divided into two sets, a *characterization* set and a *test* set. Shading in Table 6.1 differentiates the characterization circuits. The characterization circuits are used to derive models for predicting switching activity and interconnect capacitance.

3. The characterization circuits were analyzed, and prediction and target parameter values were extracted.

4. The prediction and target parameter values were fed into the GNU R statistical analysis framework [GnuR 03]. Multi-variable regression analysis was employed to establish an empirical relationship between the target and prediction parameter values. Through this approach, a prediction model was tuned to a particular FPGA device and CAD flow. In

[2]The placement and routing tools were run at the highest effort level, without performance constraints.

Table 6.1: Characteristics of benchmark circuits.

Circuit	LUTs	SLICEs
misex3	257	131
C3450	638	327
pair	464	240
ex1010	1112	567
spla	229	116
pdc	609	308
apex2	400	204
alu4	500	252
seq	1193	605
apex4	1078	548
ex5p	557	286
cps	524	271
dalu	323	165
C2670	233	123

practice, such model characterization would be done by an FPGA vendor to produce a prediction model incorporated into CAD tools used by engineers in the field.

5. Following the characterization step (#4), the prediction models were applied to predict capacitance and routed-delay activity values for nets in the *test* circuit set. Predicted values were verified by comparing with actual values (routed-delay activity and routed interconnect capacitance).

6.3 Switching Activity Prediction

In this study, a simulation-based approach is used to gather switching activity data. The CAD flow employed is shown in Figure 6.1. The Synopsys VHDL System Simulator (VSS) is used for simulation. VSS has built-in capabilities for capturing the number of logic transitions on each net during a simulation, as well as the proportion of time each net spends in the high and low logic states. Simulation with zeroed or logic delays can be done after the technology mapping step. Simulation with full routing delays must be done after placement and routing. In all cases, the VHDL simulation netlist was generated using the Xilinx tools, *ngdanno* and *ngd2vhdl*. The netlist is comprised of interconnected physical primitives which correspond

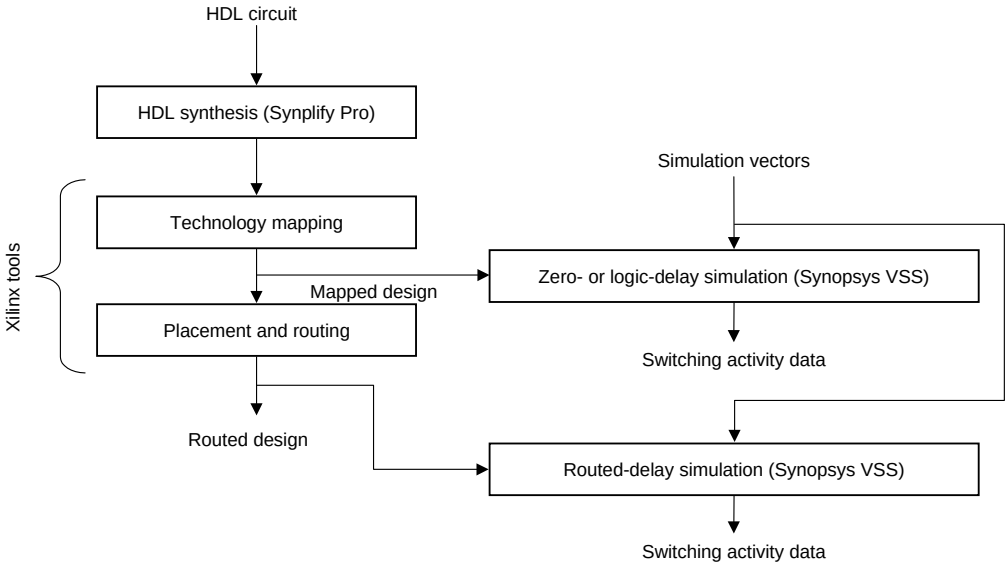


Figure 6.1: CAD flow for activity analysis.

to hardware resources in the FPGA, such as 4-input look-up-tables (4-LUTs), flip-flops, and multiplexers. For the delay-based simulations, an SDF (standard delay format) file, generated by the Xilinx annotation tool, is provided to VSS.

Circuits were simulated using 10,000 randomly chosen input vectors. Two different vector sets were generated for each circuit: one representing high input activity and a second representing low input activity. In the high (low) activity vector set, the probability of an individual input toggling between successive vectors is 50% (25%).

6.3.1 Activity Analysis

Using the flow of Figure 6.1, it is possible to investigate the amount by which switching activity changes when delays are considered. Columns 2-5 of Table 6.2 compare the total number of transitions in the logic-delay and routed-delay simulations of each circuit with the number of transitions in the zero-delay simulation. Columns 2 and 3 (4 and 5) of the table present data for the high (low) activity vector set simulations. Each table entry represents, for a given circuit,

Table 6.2: Effect of glitching on switching activity.

	High activity vector set		**Low activity vector set**		**High activity vector set**	
Circuit	% increase in transitions for logic-dly sim vs. zero-dly sim	% increase in transitions for routed-dly sim vs. zero-dly sim	% increase in transitions for logic-dly sim vs. zero-dly sim	% increase in transitions for routed-dly sim vs. zero-dly sim	Zero-delay net activity μ error (σ)	Logic-delay net activity μ error (σ)
misex3	24.3	52.3	12.5	29	38.7 (22.6)	21.6 (15.8)
C3540	20.6	125.2	15.2	93.8	46.3 (24.3)	18.7 (13.7)
pair	15.8	49	9.5	10.9	30.7 (22.8)	14.3 (15.8)
ex1010	60.3	114.9	33.9	60.9	59.7 (19.4)	41.7 (17.3)
spla	21.5	27.6	11.3	15.1	26.2 (19)	19.2 (14.2)
pdc	38.7	75.2	19.8	43.2	45.4 (19.5)	30.1 (15.9)
apex2	17.8	39.1	9.3	21.8	32.2 (19.5)	17.7 (14.1)
alu4	33.4	54.8	17.1	28.8	36.5 (19.3)	25.1 (14.5)
seq	23.5	44.7	11.6	23	35.1 (18.0)	22.4 (14.5)
apex4	40.6	96.3	22.6	51.9	40.0 (19.5)	27.8 (15.5)
ex5p	52.7	131.5	35.3	97.5	45.0 (14.5)	29.5 (11.7)
cps	32.9	50.2	17.4	27.6	34.9 (17.1)	26.6 (14.4)
dalu	24.8	48.3	15.3	31	44.1 (19.4)	24.9 (15.0)
C2670	27.3	51.8	17.1	36.1	43.1 (18.7)	25.3 (15.6)

the percentage increase in the number of transitions in the circuit's logic-delay or routed-delay simulation, versus the circuit's zero-delay simulation. Note that partial glitches were filtered out of this analysis and do not register as transitions[3].

From Table 6.2, it is apparent that there is a significant increase in activity when delays are considered. For the high activity vector set, when logic delays are used, the percentage increase in transition count versus the zero-delay simulation ranges from 16% to 60%. When routing delays are used, overall circuit delay increases and becomes more variable, leading to more glitching and higher activity. In the routed-delay simulations, the increase in transition count versus the zero-delay simulations ranges from 28% to 131%. Comparing the data for the two vector sets, one can see that the increases in activity are somewhat less drastic when the low activity vector set is used. Specifically, the activity increases are about 1/2 to 2/3 of that seen with the high activity vector set. In the low activity vector set, fewer inputs switch simultaneously between successive vectors, which reduces the potential for logic transitions on

[3]A partial glitch on a net is a glitch of short duration, shorter than the logic delay of the net's driving gate.

multiple unequal delay paths to a net, leading to reduced glitching.

To investigate whether the increase in activity due to glitching is distributed uniformly amongst the nets of a circuit, the zero-delay and logic-delay transition count for a net are viewed as estimates of the net's routed-delay transition count, allowing the absolute value of the percentage error in the estimates to be measured. For example, the error in a net n's zero-delay activity estimate is:

$$EZ(n) = 100 \cdot \frac{|TR_z(n) - TR_r(n)|}{TR_r(n)} \tag{6.1}$$

where $TR_r(n)$ and $TR_z(n)$ represent the number of transitions on net n in the routed-delay and zero-delay simulations, respectively.

Error analysis results for the high activity vector set are given in columns 6 and 7 of Table 6.2, which shows the average and standard deviation of error for each circuit. Note that for this analysis, the error was ignored on nets that transitioned on fewer than 3% of the simulation vectors, as the error data for such low activity nets was not considered statistically significant. Table 6.2 indicates that the mean error (μ) in the zero-delay activity values falls in the 26-60% range. The mean error in logic-delay activity ranges from 14-42%. Also, observe that coupled with these large mean errors are large error deviations (σ), ranging from 14-24% for the zero-delay case, and 12-17% for the logic-delay case. One can therefore conclude that zero-delay and logic-delay activity values do not necessarily correlate strongly with routed-delay activity values. Error data for the low activity vector set simulations was also computed. Smaller errors for this vector set were observed, with the mean and deviation of error for each circuit being about 1/2 to 2/3 of that observed for the high activity vector set.

The results in Table 6.2 show activity can change considerably when delays are brought into the picture and motivates the need for early prediction of a net's routed-delay activity. Prior to presenting the prediction approach, the noise in the prediction problem is analyzed.

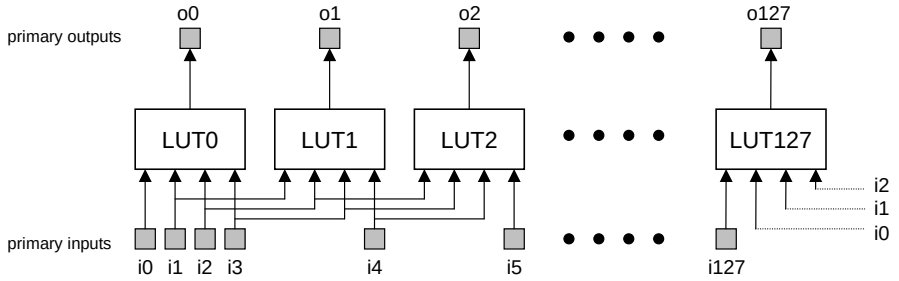


Figure 6.2: Circuit with regularity.

6.3.2 Noise in Switching Activity

In an effort to understand the difficulty of generating accurate pre-layout activity estimates, the noise in routed-delay activity values is studied by performing an activity analysis on the circuit shown in Figure 6.2. The circuit is highly regular in terms of structure and functionality and consists of 128 inputs driving 128 4-input look-up-tables (LUTs), which in turn drive 128 outputs. Each LUT in the circuit is programmed to implement a 4-input logical AND function.

The circuit in Figure 6.2 was mapped into Virtex-II PRO, and then simulated with both the high and low activity vector sets. The change in activity on the LUT output signals was examined in the routed-delay simulation versus the zero-delay simulation. Note that the circuit's regularity implies that variability in the activity change on the LUT output signals is largely a result of variable path delays, known only after layout is complete. The results of the analysis are shown in Figure 6.3. The figure shows the percentage increase in activity on each LUT output signal for both simulation vector sets. For each vector set, 128 data points are shown – one point for each LUT output signal. Observe that, despite the circuit's regularity, the variability in the activity increase on the LUT output nets is considerable, due to the wide variety of routing resources and delay paths in the FPGA routing fabric, and the different delays associated with the four input-to-output paths through a LUT[4]. For the low activity

[4]LUT input pins are logically equivalent. The selection of a LUT input pin for a particular LUT fanin signal is made by the router.

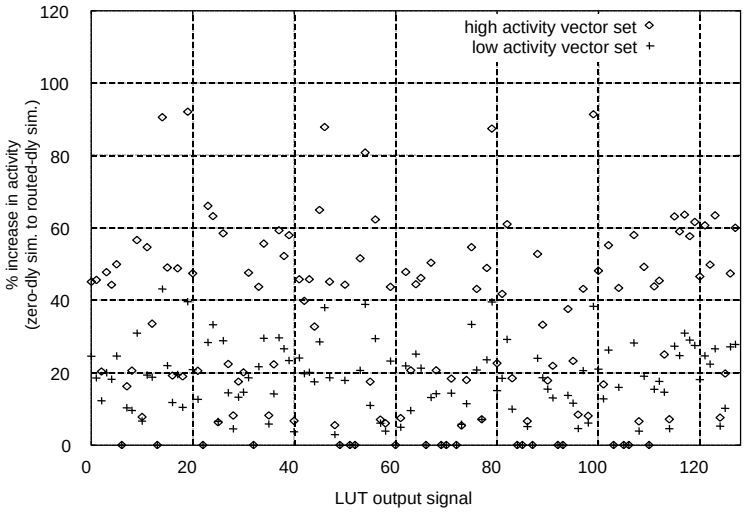


Figure 6.3: Activity change in regular circuit.

vector set, the activity increase for most nets is in the range of 0 to 40%; for the high activity vector set, the increase ranges from 0 to 90%.

Real circuits are likely to be much less regular than the circuit of Figure 6.2, and therefore, we conclude it will be difficult to achieve a high degree of accuracy in activity prediction at the pre-layout stage. Nevertheless, the next section describes a prediction approach that produces activity values that, in comparison with zero-delay or logic-delay activity values, are superior estimates of routed-delay activity.

6.3.3 Prediction Model

Section 5.2 (see page 100) described how a digital circuit can be represented by a directed acyclic graph (DAG) and introduced terminology relevant to a circuit's DAG representation. This section uses the same terminology as Section 5.2. Note also that in this section, a DAG node in a circuit and the net driven by the node are referred to interchangeably; for example, a node y drives net y.

The proposed prediction approach accepts a technology mapped, Virtex-II PRO circuit as

input. At this level of abstraction, internal DAG nodes correspond to the LUTs and other logic elements in the target FPGA device. In FPGA technology, path depth (# of LUTs) is frequently used as a predictor of path delay at the pre-layout stage [Cong 94a, Fran 91b]. The reason for this is that, unlike in ASIC technologies such as standard cell, the logic blocks in FPGAs are uniform and have equal drive capability. Furthermore, the programmable routing switches in an FPGA's routing fabric are typically buffered, making connection delay relatively independent of fanout. Consequently, without access to more accurate delay information extracted from physical layout, depth is viewed as a reasonable estimate of delay. This FPGA-specific property is leveraged in the proposed activity prediction approach, which incorporates delay estimation into a simple model of net glitching severity.

The approach to activity prediction is analogous to the generate and propagate notion that defines how carry signal values are assigned in arithmetic circuits. In such circuits, the carry value for a particular bit may either be *generated* by the bit, or it may be *propagated* from a lower-order bit. For activity prediction, consider a node y with logic function $y = f(x_1, x_2, ..., x_n)$. Similar to carry signal operation, glitches on y's output may come from two sources: they may be propagated from one of y's inputs, $x_1, x_2, ..., x_n$, or they may be generated by y itself. A prediction function that quantifies the severity of glitching on y's output is defined as follows:

$$PR(y) = f[GEN(y), PROP(y), D(y)] \tag{6.2}$$

where f is a function defined below, and $GEN(y)$ and $PROP(y)$ represent the amount of glitching generated by y and the amount of glitching propagated from y's inputs, respectively. $PR(y)$ is the predicted percentage change in the activity of net y due to glitching. The depth term, $D(y)$, of (6.2) is included to reflect the intuition that glitching severity typically increases with combinational depth. All things being equal, one can expect that a node with shallow depth will experience less glitching than a deep node. Note that the prediction values for the nodes of a circuit are computed in a specific order, *from* primary inputs *to* primary outputs.

Prior to defining the generate term of (6.2), we introduce a new parameter. Let $PL(y)$

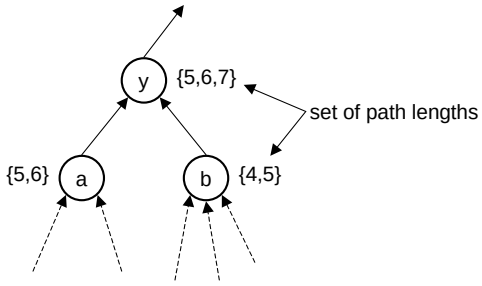


Figure 6.4: Finding the set of path lengths for y.

represent the *set* of different path lengths from a primary input to node y. This parameter can be computed in linear time by an input-to-output DAG traversal that maintains a set of path lengths for each node. When a node is traversed, its path length set is populated by taking the union of incremented path lengths of each of its immediate fanin nodes. More formally:

$$PL(y) = \bigcup_{x_i \ \epsilon \ inputs(y)} \{p + 1 \mid p \ \epsilon \ PL(x_i)\} \tag{6.3}$$

Observe that a given node may have a larger set of path lengths than any of its immediate fanins. Consider the example shown in Figure 6.4, in which a node y has two fanin nodes, a and b. The set of path lengths for each node is shown adjacent to the node. Observe that node y has three path lengths, whereas its fanins have only two path lengths. It can be said that one path length is *introduced* at y. The generate term of (6.2) is defined to be the number of path lengths introduced at node y:

$$GEN(y) = \min_{x_i \ \epsilon \ inputs(y)} \{|PL(y)| - |PL(x_i)|\} \tag{6.4}$$

The rationale for incorporating the number of path lengths to a node into the prediction function is that variable path lengths to a node generally correlate with variable, unequal path delays to the node, leading to glitching at the node's output.

The propagate term of (6.2) borrows ideas from the concept of transition density [Najm 93]

and uses the notions of *Boolean difference* and *static probability*, briefly reviewed here. The Boolean difference of a function, $y = f(x_1, x_2, ..., x_n)$, with respect to one of its inputs, x_i, is defined as:

$$\frac{\partial y}{\partial x_i} = f_{x_i} \oplus f_{\overline{x}_i} \tag{6.5}$$

where f_{x_i} ($f_{\overline{x}_i}$) is the Boolean function obtained by setting $x_i = 1$ ($x_i = 0$) in $f(x_1, x_2, ..., x_n)$, and $\oplus$ denotes the exclusive-OR operation. When the Boolean difference function, $\frac{\partial y}{\partial x_i}$, is true (logic-1), a transition on x_i will cause a transition on y.

Recall that the static probability of a signal is the fraction of time that the signal is in the logic-1 state. Thus, the static probability of a Boolean difference function, $P(\frac{\partial y}{\partial x_i})$, represents the probability that a transition on x_i will cause a transition on y. Clearly, the ability of glitches on an input signal, x_i, to propagate to y depends on $P(\frac{\partial y}{\partial x_i})$. Furthermore, it seems likely that the influence of a node input on the node's output will depend partly on the input's switching activity. The propagate function is therefore defined as:

$$PROP_y = \frac{\sum_{x_i \ \epsilon \ inputs(y)} P(\frac{\partial y}{\partial x_i}) \cdot TR_z(x_i) \cdot PG(x_i)}{\sum_{x_i \ \epsilon \ inputs(y)} P(\frac{\partial y}{\partial x_i}) \cdot TR_z(x_i)} \tag{6.6}$$

The $P(\frac{\partial y}{\partial x_i}) \cdot TR_z(x_i)$ in the numerator can be viewed as a weight quantifying the influence of glitching on x_i to glitching on y. $PG(x_i)$ is defined as $PROP(x_i) + GEN(x_i)$ and represents the amount of glitching on input x_i. The denominator of (6.6) normalizes the values computed by the propagate function so they are relatively independent of the transition counts and probabilities involved. Note that TR_z in (6.6) can be replaced with TR_l, the logic-delay transition count, if logic-delay activity data is available. In the experimental study, the probability and transition data needed to compute (6.6) is extracted from zero-delay or logic-delay circuit simulation (see below). However, such data need not be derived from simulation; it can be computed efficiently using probabilistic approaches, such as those in [Yeap 98]. Thus, simulation is not a requirement for the use of the prediction model.

6.3.4 Results and Discussion

Following the methodology outlined in Section 6.2.2, the characterization circuits were used to derive a model relating the activity change on a net due to glitching to the *prediction function* (6.2). In this case, the prediction parameters are the $PROP$, GEN, and D terms in (6.2). To begin with, a linear function for f was used, and gradually, its complexity was increased by adding higher-order terms. Eventually, a quadratic function was settled upon, having the following form:

$$\begin{aligned} PR_y = {} & \alpha \cdot GEN(y) + \beta \cdot GEN(y)^2 + \gamma \cdot PROP(y) + \\ & \upsilon \cdot PROP(y)^2 + \nu \cdot D(y) + \xi \cdot D(y)^2 + \eta \cdot PROP(y) \cdot GEN(y) + \\ & \iota \cdot GEN(y) \cdot D(y) + \rho \cdot PROP(y) \cdot D(y) + \phi \end{aligned} \tag{6.7}$$

where α, β, γ, etc. are scalar coefficients whose values are determined through regression analysis. The use of higher-order models was also considered but were found to not substantially improve accuracy. Two prediction models were constructed: one that predicts routed-delay activity from *zero*-delay activity, and one that predicts routed delay activity from *logic*-delay activity. For completeness, Section A.1 of Appendix A gives the details of the regression analysis, including the parameter coefficients.

Following characterization, to apply the prediction model derived from the characterization circuits, the model's value was computed for each net in the test circuits. This yields a predicted percentage change in activity for each net versus the net's zero-delay (or logic-delay) activity. The predicted percentage change is used to compute an estimate of each net's routed-delay activity.

The model was evaluated numerically by computing the percentage error in predicted activity values relative to routed-delay activity values using (6.1). The error data for the test circuits is shown in Table 6.3. The table gives the average of the absolute values of percentage error across all nets for each circuit. Column 2 of the table shows the error results for the

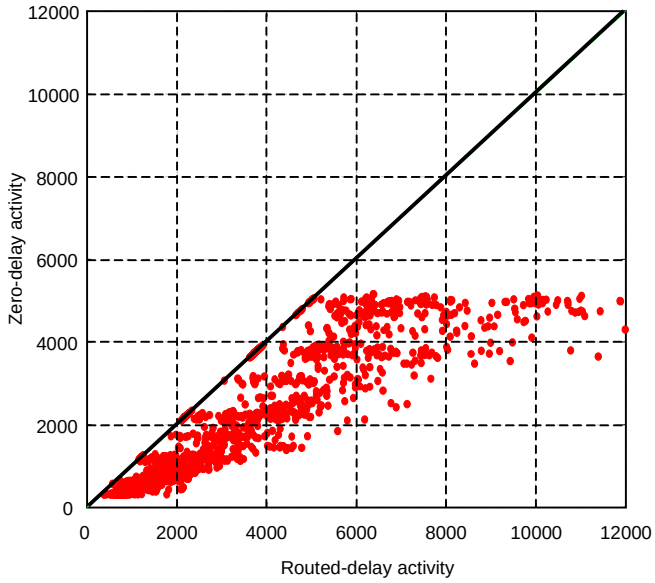
Table 6.3: Error in predicted activity values.

Circuit	Predicted net activity mean error (from zero-dly activity)	Predicted net activity mean error (from logic-dly activity)
alu4	17.8	16
seq	23.8	19.8
apex4	21.4	20.2
ex5p	17.1	13.3
cps	19	16.7
dalu	23.9	16.4
C2670	22.9	17.8

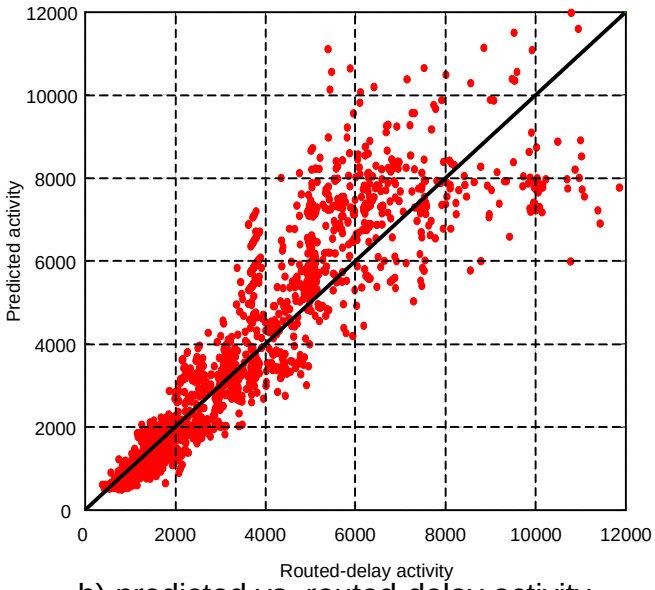
model that predicts routed-delay activity from zero-delay activity; column 3 gives results for the model that predicts routed-delay activity from logic-delay activity. In Section 6.3.1, it was observed that logic-delay activities are "closer" to routed-delay activities than are zero-delay activities. Table 6.3 confirms that using logic-delay activities as the basis of a prediction model yields smaller error values. The error data for each circuit in the table can be compared with the error data in columns 6 and 7 of Table 6.2. Observe that the average error of the predicted activities is significantly less than the error of the zero-delay or logic-delay activities; the error is reduced by a factor of 2 for many of the circuits.

Figures 6.5 (a) and (b) show zero-delay and (zero-delay-based) predicted activity values versus routed-delay activity values. Each point in these plots corresponds to a net in one of the test circuits. The vertical axis of Figure 6.5(a) represents zero-delay activity (transition count); the horizontal axis represents routed-delay activity. The vertical axis of Figure 6.5(b) represents predicted activity. Observe, in Figure 6.5(a), that the absence of glitching creates a "ceiling" in the zero-delay activity values at about 5000 transitions. This ceiling is eliminated in the predicted activity values [Figure 6.5(b)], which more closely resemble the routed-delay activity values. Figure 6.6 gives analogous plots for logic-delay and (logic-delay-based) predicted activity values. Both sets of predicted values exhibit a considerable "spread" about the $y = x$ line shown. This is expected and is in line with the noise analysis in Section 6.3.2.

Figures 6.5 and 6.6 show that, as expected, the errors in the zero-delay and logic-delay activity values are largely one-sided (under estimation), whereas the errors in the predicted

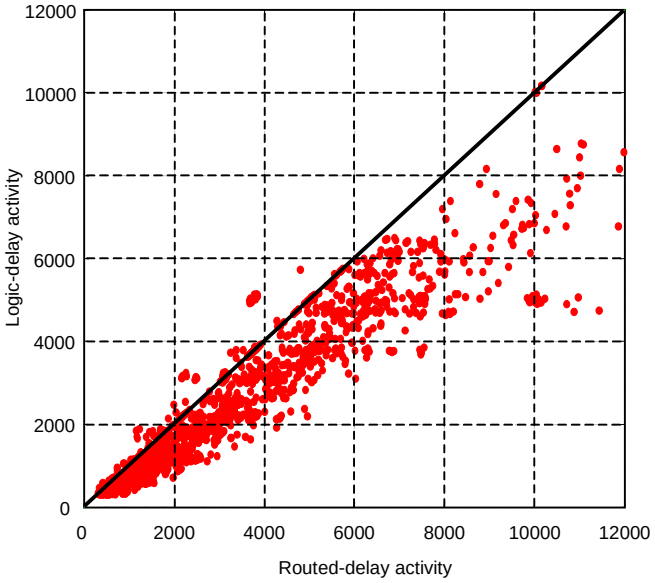


a) zero- vs. routed-delay activity

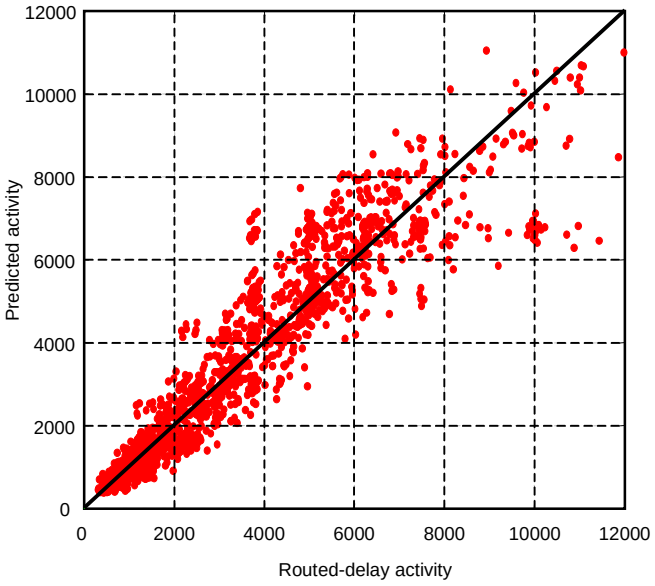


b) predicted vs. routed-delay activity

Figure 6.5: Zero-delay activity and predicted activity versus routed-delay activity.



a) logic- vs. routed-delay activity



b) predicted vs. routed-delay activity

Figure 6.6: Logic-delay activity and predicted activity versus routed-delay activity.

activity values are balanced about the $y = x$ line. The use of zero-delay or logic-delay activity values in power estimation will lead to significant underestimates of circuit power. Conversely, since the proposed approach under-predicts activity for some nets and over-predicts for others, we expect that the use of the predicted activity values will produce average power estimates that are much closer to actual average circuit power. Eliminating the one-sided bias in error is one of the key advantages of the prediction method, making it attractive for use in applications such as early power estimation.

To investigate the dependence of the prediction results on the division of benchmarks into the characterization and test sets, an *alternate* benchmark division was created by swapping three of the seven circuits in each set. The prediction models were reconstructed using the new characterization set and then applied to predict activity values for nets in the new test set. The average of the absolute values of percentage error in the predicted activity values is shown in Table 6.4. The columns of the table are analogous to those of Table 6.3. As with the original benchmark division, the average errors of the predicted activity values are considerably less than the errors of zero-delay or logic-delay activities (compare with columns 6 and 7 of Table 6.2). Shading in Table 6.4 is used to show the circuits that are common between the original and alternate test circuit sets. The error results for these circuits can be compared with the results in Table 6.3 – the results for the original benchmark division. Observe that, despite the use of different characterization sets, the prediction errors for these circuits are fairly similar. This illustrates that the prediction model is not tied to a specific choice of characterization and test set. We expect that the dependence of the prediction model on the benchmark division can be reduced further through the use of a larger characterization set, as is available to commercial FPGA vendors.

The prediction results presented above were based on the simulation data for the high activity vector set. However, we expect that the prediction method can be applied effectively for a range of input switching activities. A direction for future work is to augment the prediction approach to automatically account for various amounts of primary input switching activity.

Table 6.4: Error in predicted activity values for alternate benchmark division.

Circuit	Predicted net activity mean error (from zero-dly activity)	Predicted net activity mean error (from logic-dly activity)
pdc	19.5	14.3
apex2	26.0	15.9
ex1010	22.7	28.5
ex5p	16.1	15.6
cps	18.1	15.1
dalu	23.7	17.0
C2670	22.0	17.8

6.4 Interconnect Capacitance Prediction

We now turn to the second topic of this chapter, namely, interconnect capacitance prediction at the placement stage. We begin with a brief review of related work.

6.4.1 Related Work

Several papers have considered capacitance estimation in the context of power-aware FPGA CAD tools. At the technology mapping level (pre-layout), net capacitance has been estimated using a linear function of fanout [Farr 94, Li 01]. Previous placement-based capacitance estimates have appeared in [Roy 99, Kumt 00b]. At this level, the approach taken has been to use a combination of a net's bounding box half-perimeter and its fanout to estimate its routed capacitance. These prior works use generic, non-architecture-specific parameters to predict capacitance.

A problem related to capacitance estimation is FPGA delay estimation, which is well-studied in the literature. The problems differ from each other in that delay estimates are needed for individual driver/load connections, whereas capacitance estimates are needed for entire multi-fanout nets. In [Marq 00], a delay estimation model is constructed during placement by executing a pre-routing step in which "dummy" routes having known x and y distances are made. The dummy routes are used to construct a table that relates delay to distance; table values are then used as delay estimates during placement. The delay estimation model devel-

opment approach used in [Karn 95, Maid 03] is similar to the one used here. Routed designs are analyzed and connection delays are correlated with placement parameters, producing an empirically-derived estimation model. In [Hutt 03], characteristics of a target FPGA's interconnect architecture are used to predict delay within a partitioning-based placement system. In that case, the FPGA interconnect is hierarchical and the placer's partitioning levels are chosen to match the underlying FPGA interconnect hierarchy. As such, the placer has knowledge of the interconnect resources likely to be used in the routing of nets that are cut and uncut at a given partitioning level. Like [Hutt 03], the proposed capacitance estimator also considers architecture-specific criteria to improve estimation accuracy.

6.4.2 Noise in Interconnect Capacitance

To gauge the inherent noise in capacitance estimation, an approach similar to that used in [Boda 00] is taken. Specifically, each benchmark circuit is placed using Xilinx tools, and a placed netlist is generated. Then, a copy of the placed netlist is made, and the copy is modified, reversing the *order of the nets* but leaving all other aspects of the design intact, including the placement[5]. The order of the nets in the netlist is arbitrary and generally not under user-control. The original placed netlist and the modified netlist for each design are then routed to produce *baseline* and *alternate* routing solutions, respectively. Interconnect capacitance values for nets are determined by running XPower [XPower 03], the Xilinx power estimation tool, on the routed circuits[6]. XPower produces a log file containing the capacitance of each net in femtofarads (fF). The capacitance values for each net in the two routing solutions can be compared to assess routing variability, since both routing solutions have the same placement. Differences in net capacitance between the baseline and alternate routing represent noise that one cannot correct or account for in estimation. Note, however, that differences in the two

[5] Netlist modifications were made by reversing the order of instances in each placed design's XDL (Xilinx Design Language) ASCII representation.

[6] FPGA vendors do not provide users with a transistor-level or layout-level FPGA representation. Commercial capacitance extraction tools, commonly used in custom ASIC design, cannot be applied by FPGA users to determine capacitance values.

routing solutions for a design generally do not represent problems with the routing tool. The tool aims to minimize total routing resource usage, which involves trade-offs between the FPGA resources allocated to each net; such trade-offs may be resolved arbitrarily in some cases.

Figure 6.7 shows the results of the noise analysis. Each point in the figure represents a net in one of the benchmark circuits. The horizontal axis represents net capacitance in the baseline routing solution; the vertical axis represents net capacitance in the alternate routing solution. Ideally, in the absence of variability, all points would lie on the line shown ($y = x$). However, Figure 6.7 shows substantial noise in net capacitance. Notice that the results in Figure 6.7 illustrate that one routing solution is unlikely superior to the other: the symmetry in spread about the $y = x$ line suggests that the number of nets for which net capacitance increased in the alternate routing solution is approximately equal to the number of nets for which capacitance decreased. This assertion is also true at the individual circuit level, as evidenced by the data in Column 3 of Table 6.5. Column 3 shows, for each circuit, the number of nets for which capacitance increased and decreased (in parentheses) in the alternate routing solution versus the baseline. The number of increases and decreases are roughly equal for each circuit.

The absolute value of the percentage change in capacitance for each net was computed in the alternate routing versus the baseline. Column 2 of Table 6.5 shows the average change for each circuit. The average change across all circuits is 22%. Certainly, the baseline and alternate routing solutions represent just two of the possible routing solutions for a given placement. Different net orderings could be used to produce additional routing solutions. The capacitance of a given net in such routing solutions may differ from its capacitance in the baseline and alternate routing solutions. Thus, the noise results presented above represent a statistical lower bound on the error in capacitance prediction; prediction accuracy cannot be improved beyond this noise floor error limit.

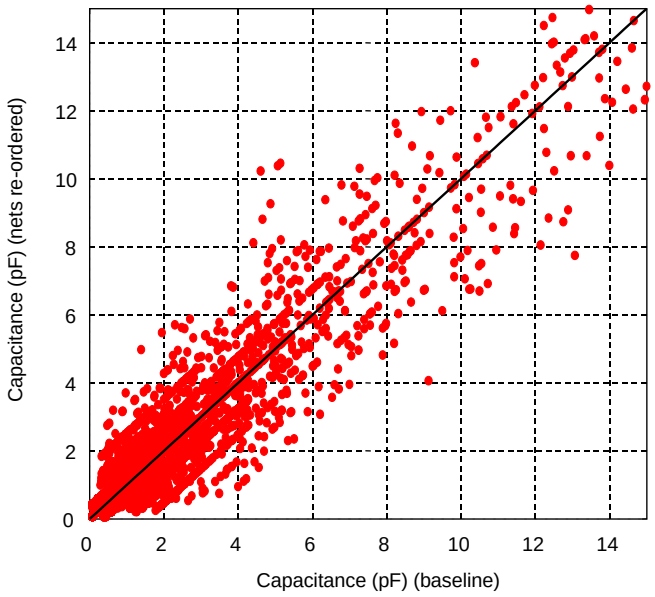


Figure 6.7: Noise in interconnect capacitance.

Table 6.5: Noise in individual circuits.

Circuit	**Avg. absolute % change in cap.**	**# of nets with cap increase (decrease)**
misex3	14.9	27 (38)
C3540	18.2	115 (117)
pair	19.1	112 (103)
ex1010	23.7	234 (255)
spla	22.8	33 (28)
pdc	23.4	146 (112)
apex2	23.7	77 (95)
alu4	20.0	97 (86)
seq	23.1	246 (259)
apex4	23.7	226 (235)
ex5p	24.0	124 (128)
cps	23.1	88 (78)
dalu	23.0	72 (65)
C2670	22.8	49 (65)
Average:	21.8	

6.4.3 Prediction Model

Having analyzed the noise in capacitance, we now introduce the prediction model. In comparison with prediction at the pre-layout stage, a larger number of prediction parameters are available for prediction at the placement stage, for example, physical design data. Consequently, a slightly different approach to capacitance prediction is taken as compared with that taken for activity prediction. First, a set of candidate prediction parameters for the model are defined. Following this, the rationale for why the chosen parameters may correlate with interconnect capacitance is explained. Section 6.4.4 presents experimental results showing which of the parameters are best for use in a capacitance prediction model.

CAD applications such as power-aware placement and early power planning require that capacitance estimates be produced quickly, as they are typically needed within the inner loop of design optimization. With this in mind, the focus here is on parameters with low computational requirements. Considering a net n, the following are known at the placement stage:

- $FO(n)$: The fanout of net n.
- $BB(n)$: The half-perimeter of net n's bounding box, as measured in CLB tiles.
- $XS(n)$, $YS(n)$: The span of net n in the x and y-dimensions, respectively.
- $NT(n)$: The number of CLB (or I/O) tiles in which net n has at least 1 pin.
- $X6(n)$, $Y6(n)$: Defined as $XS(n)$ *mod* 6 and $YS(n)$ *mod* 6, respectively.
- $FP(n)$, $GP(n)$: The number of load pins on net n that are F-LUT and G-LUT inputs, respectively.
- $CG(n)$: The average estimated routing congestion in net n's bounding box.

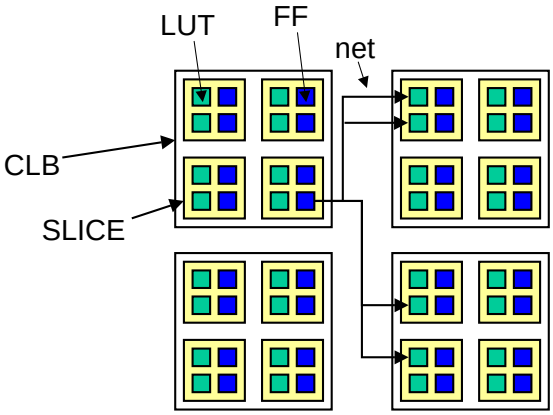


Figure 6.8: Illustration of parameter NT.

The fanout and bounding box of net n are generic parameters, frequently used to predict capacitance in the ASIC domain. Breaking the bounding box into its x and y spans through $XS(n)$ and $YS(n)$ allows one to evaluate whether there is a capacitance bias associated with the use of horizontal versus vertical routing resources.

In contrast to the fanout and distance terms, parameters $NT(n)$, $X6(n)$, $Y6(n)$, $FP(n)$, and $GP(n)$ are specific to the Virtex-II PRO FPGA. As mentioned in Section 6.2.1, the FPGA contains an array of CLB tiles. Most of the interconnect resources connect CLB tiles to one another, with the exception of the LOCAL interconnect that is internal to a CLB. A CLB contains 4 SLICEs (8 LUTs/FFs) and therefore, a net n may have multiple pins placed in a single CLB. In such cases, some of the net's routing *between* CLBs may be shared by the net's pins *within* in a single CLB. The sharing of routing resources amongst pins may influence net capacitance, and the $NT(n)$ term aims to account for this possibility. To illustrate, Figure 6.8 shows a net with 5 pins, but spanning only 3 CLB tiles ($NT = 3$).

An important routing resource in the FPGA interconnect is the HEX wires spanning 6 CLB tiles. Long nets may be routed using a sequence of HEXes, with the "left over" distance being composed of the shorter DOUBLE, DIRECT, or LOCAL resources. HEX resources likely have different capacitance than shorter resources. $X6(n)$ and $Y6(n)$ represent the left over distance

in the x and y dimensions, respectively, and roughly correspond to the number of short resources needed for a net. Similarly, the different types of pins on logic and I/O blocks will likely have different capacitance values associated with them. The $FP(n)$ and $GP(n)$ parameters allow the F and G-LUT input pins (see Section 6.2.1) to be differentiated from other types of pins.

Routing congestion may lead to nets with long circuitous paths and excess capacitance. The congestion for a net n, $CG(n)$, is estimated using a probabilistic method similar to that described in [Lou 02], chosen for its simplicity and computational efficiency. The approach is summarized here; the interested reader is referred to [Lou 02] for details. Nets are first converted into a set of two-pin connections by finding their minimum spanning tree using Prim's algorithm. The routing demand of a two-pin connection is then computed probabilistically, considering its potential routing topologies. An example for a two-pin connection with a 3-by-3 CLB tile bounding box is shown in Figure 6.9. As illustrated, only routing topologies that have *at most* two jogs are included. There are 4 possible route options for the connection. Generally, the number of route options for a connection having a bounding box with c columns and r rows is:

$$N_{route\_opt} = \begin{cases} c + r - 2, & \text{for } c > 1, r > 1, \\ 1, & \text{otherwise.} \end{cases} \tag{6.8}$$

Similarly, the number of a connection's route options that cross a *specific* CLB tile edge can be expressed analytically. Dividing the number of a connection's route options that cross a specific CLB tile edge by the total number of route options for the connection yields the probability that the connection's route will traverse the CLB tile edge. This probability can be viewed as the *demand* exerted by the connection on a tile edge (see Figure 6.9). The routing demands contributed by each two-pin connection in each net are tallied to produce a total routing demand on each CLB tile. The $CG(n)$ term represents the *average* routing demand across all CLB tile edges within net n's bounding box.

The capacitance of nets in the characterization circuit set were fit to a mathematical function of the parameters described above. The result is a mathematical model that may be applied to predict capacitance values of nets in the test circuit set. Separate estimation models were

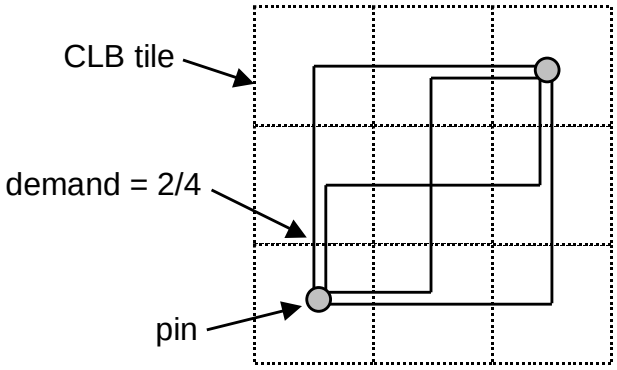


Figure 6.9: Routing congestion estimation.

developed for high-fanout nets (> 10 loads) and low-fanout nets (≤ 10 loads), and each model was applied accordingly in the experimental study (in the next section). A range of models were evaluated and the labels *lin*, *quad*, and *cubic* are used to represent linear, quadratic, and cubic functions, respectively. Models are specified using a function type, followed by a parameter list in parentheses. Using this terminology, a model specified as $lin(FO, BB)$ would predict the capacitance of a net n, $C(n)$, using a linear function of the net's fanout and its bounding box half-perimeter:

$$lin(FO, BB) : C(n) = \alpha \cdot FO(n) + \beta \cdot BB(n) + \gamma \tag{6.9}$$

where α, β, and γ are scalar coefficients with values determined through regression analysis. Note that cross-variable terms [e.g., $FF(n) \cdot BB(n)$] are omitted, unless explicitly included in the parameter list.

6.4.4 Results and Discussion

Figure 6.10 gives error results for some of the estimation models evaluated in this study. The vertical axis gives the average error in capacitance estimate for a given estimation model, shown on the horizontal axis. The error for a model was computed by averaging the absolute values of percentage estimation errors of all nets in the test circuit set. Models are labeled from M1

to M10, in order of increasing complexity.

Model M1 estimates capacitance using a linear function of fanout, yielding an error of about 84%. This represents the error one could expect in capacitance estimation at the *pre-layout* stage. M2 incorporates physical data, namely, bounding box half-perimeter, and reduces error to 66%. In M3, the bounding box parameter is partitioned into separate x and y domains. Estimation accuracy is not improved and, therefore, there is evidently very little directional bias in Virtex-II: the capacitance "cost" of using horizontal routes is approximately equal to that of vertical routes. Previous work on FPGA capacitance estimation, such as [Roy 99], used models equivalent to M2 or M3.

Beginning with M4, architecture-specific parameters are inserted into the model. M4 includes $NT(n)$, which is the number of CLB tiles in which a net n has pins. Incorporating this parameter reduces error from 66% to 54%. In model M5, the $X6(n)$ and $Y6(n)$ parameters are brought in (related to the HEX resources in the interconnect) and error is further reduced, to about 50%. M6 considers the pin types on a net [through $FP(n)$ and $GP(n)$] and yields an average error of 46%. Comparing the results for M6 to those for M3, the considerable benefits of tying model parameters to the underlying FPGA interconnect architecture are apparent.

In model M7, congestion is introduced and surprisingly, very little benefit to error reduction is observed. There are a number of potential explanations for this. First, it is possible that there are sufficient routing resources in Virtex-II PRO such that routing congestion is not a problem and circuitous routes are not needed to achieve routability. This is a probable explanation, as the routing stress imposed by the MCNC circuits is likely relatively low in comparison with modern industrial designs. A second possibility is that the congestion metric employed does not accurately reflect routing congestion in Virtex-II PRO. The impact of congestion on routing in commercial FPGAs is not well-studied and is likely to be highly architecture dependent. Interestingly, a very recent paper by Jariwala and Lillis considered the accuracy of known congestion prediction techniques in ASICs and FPGAs [Jari 04]. Considerable shortcomings were discovered in the ability of the techniques to accurately predict congested "hot spots".

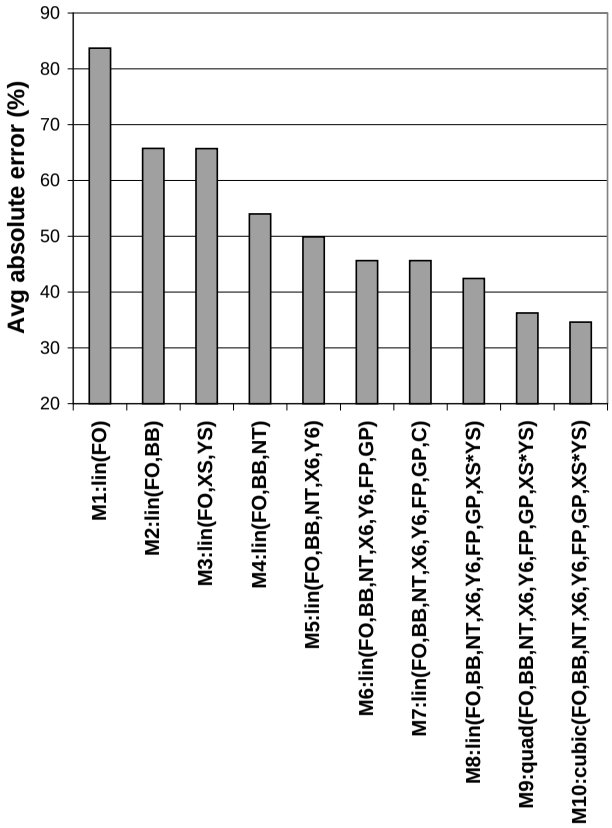


Figure 6.10: Average error for a variety of prediction models.

Model M8 includes a cross term, the x-span of a net multiplied by its y-span $[XS(n) \cdot YS(n)]$. The intuition behind this is to differentiate between nets that span both dimensions from those that span only a single dimension. With this parameter included, error is reduced somewhat, from 46% to 42%. Models M9 and M10 have the same parameter set as M8, but estimate capacitance using quadratic and cubic functions, respectively. Observe that using a quadratic function (M9) reduces error to about 36%. The benefits of moving to a cubic function (M10) are minimal. Higher-order models were also investigated but found to not significantly improve estimation accuracy. The regression analysis details for model M10 are given in Section A.2 of Appendix A.

Model M10 yields average error values of about 35%. Error results for the individual test circuits are shown in column 2 of Table 6.6. Table 6.5 shows that the noise floor errors for these circuits fall in the 20-24% range. The difference between the prediction and noise floor errors limits the potential for improvement in prediction accuracy. Given the range of routing resource types available in the FPGA, we consider the prediction accuracy to be quite good.

Figure 6.11 plots the predicted (vertical axis) and actual (horizontal axis) capacitance values for all nets in the test circuit set, as predicted using model M10. Observe that capacitance is under-predicted for some nets and over-predicted for others, leading to under and overestimates of a net's power. The under and over-predictions are roughly equally distributed, and consequently, it is expected that average power estimates made using the proposed model will be close to actual power values. Note also the similarity between the estimation results and noise results (Figures 6.11 and 6.7), which is quite interesting as the noise error cannot be resolved in estimation.

As with activity prediction, the dependence of the capacitance prediction model on the division of benchmarks into the characterization and test circuit sets was investigated. The model was reconstructed using the alternate benchmark division described in Section 6.3.4. Column 3 of Table 6.6 lists the benchmarks in alternate test set. Shading is used to show the benchmarks that are common to the two test sets. Column 4 gives error results for the

Table 6.6: Errors for individual circuits; results for alternate characterization/test benchmark division).

Circuit	Mean absolute prediction error (%)	Alternate test circuit	Mean absolute prediction error (%)
alu4	32.9	pdc	37.4
seq	34.4	apex2	34.6
apex4	33.1	ex1010	34.9
ex5p	34.1	ex5p	33.9
cps	39.4	cps	39.3
dalu	32.2	dalu	33.6
C2670	38.8	C2670	38.9
Average:	35.0		36.1

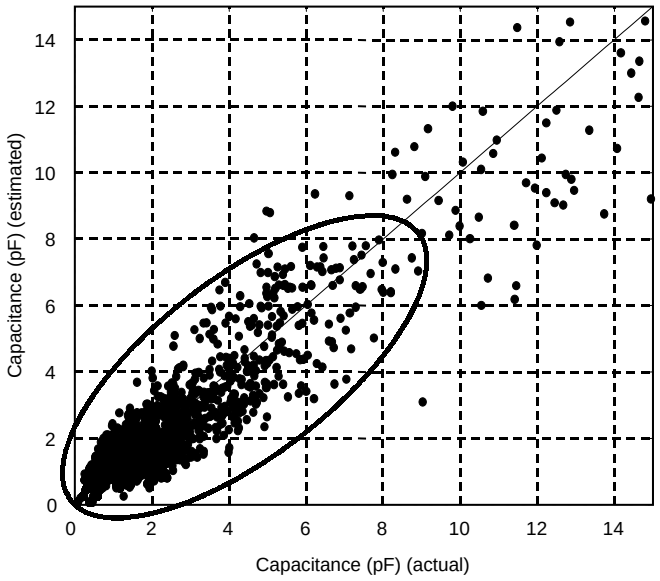


Figure 6.11: Estimated versus actual values (approx. 4000 points in ellipse).

alternate test circuit set. The error data for the common circuits can be compared with that in column 2 of the table. Observe that the error results for these circuits are similar, even though the characterization sets used for model construction differ. This provides evidence that the prediction model is robust and not strongly dependent on the benchmark division.

6.5 Summary

The dominance of interconnect in overall FPGA power consumption implies that understanding and managing activity and capacitance is a crucial part of power-aware FPGA CAD. This chapter considered activity and capacitance prediction for FPGAs and proposed models for the early prediction of these parameters. The activity prediction approach estimates the routed-delay activity value for a net using its zero-delay or logic-delay activity value, as well as circuit functional and structural properties. The proposed capacitance prediction model uses generic parameters, such as fanout and bounding box length, as well as parameters that are specific to the underlying FPGA routing fabric. A noise analysis was conducted and limits were established on the potential accuracy achievable in prediction. The prediction models work well given the noise limitations. We expect that the models will be useful in applications such as low-power synthesis, early power estimation, and power-aware layout.

7 Conclusions

7.1 Summary and Contributions

Trends in technology scaling imply a drastic increase in leakage power and a steady increase in dynamic power with each successive process generation. Field-programmable gate arrays (FPGAs) require considerable hardware overhead to offer programmability, making them less power-efficient than custom ASICs for implementing a given logic circuit. The huge number of transistors on the largest FPGA chips suggest that the power trends associated with scaling may impact FPGAs more severely than custom ASICs. Despite this, until recently, the majority of published research on FPGA CAD and architecture, as well as the focus of the commercial vendors, has been on improving FPGA speed and density. Power management in FPGAs will be mandatory at the 65nm technology node and beyond to ensure correct functionality, provide high reliability, and to reduce packaging costs. Furthermore, lower power is needed if FPGAs are to be a viable alternative to ASICs in low-power applications, such as battery-powered electronics.

This dissertation has contributed new computed-aided design (CAD) and circuit-level techniques for the optimization and prediction of FPGA power consumption:

- Chapter 3 looked at active leakage power and proposed two novel "no cost" CAD techniques for active leakage reduction. The first technique involves changing the polarity of logic signals to place hardware structures into their low leakage states as much as possible. The second technique alters the routing step of the CAD flow, with the aim of encouraging more frequent use of routing resources that consume less leakage. Combined, the two techniques provide an average leakage power reduction of 30%, across a

suite of benchmark circuits. The proposed techniques have been published in [Ande 04f] and [Ande 05a]. To the author's knowledge, this represents the first published work on active leakage reduction in FPGAs.

- Previous work on FPGA power characterization has shown that the majority of an FPGA's dynamic and static power is dissipated in the interconnect. To address this, Chapter 4 considered the circuit-level design of low-power FPGA interconnect. Several new routing switch designs were proposed. The designs are programmable to operate in one of three modes: high-speed, low-power, or sleep mode. In high-speed mode, switch power and performance are similar to that of a traditional routing switch. In low-power mode, dynamic and leakage power are reduced versus high-speed mode, albeit at the expense of speed. Sleep mode is a low leakage state suitable for unused routing switches. Each of the different proposed designs offers a different area/speed/power trade-off. All of the designs involve only minor changes to a traditional routing switch, and have minimal impact on CAD complexity, making them straightforward to incorporate into current FPGAs. A portion of this work has been published in [Ande 04c], [Ande 04d], and [Ande 05b].

- Chapter 5 presented a new power-aware technology mapping algorithm for FPGAs that permits trade-offs between power and performance (mapped circuit depth). An important aspect of this work is its recognition of the consequences of logic replication on power, and notably, that logic replication is usually undesirable from the power viewpoint. Based on this, an activity-conscious approach to logic replication is taken during technology mapping. An experimental study showed that different publicly-available mapping approaches produce solutions with widely varying power characteristics, and that the proposed algorithm yields solutions with considerably less power than competing techniques. This work has been published in [Ande 02].

- Finally, Chapter 6 dealt with predicting the dynamic power consumed in FPGA interconnect, and specifically, with the early prediction of interconnect capacitance and switching

activity. Empirical estimation models were developed for these parameters. The models have many applications; for example, they could be applied within an early power planning framework, during low-power synthesis, power-aware layout, or in other instances where power estimates are required but final routing data is incomplete or unavailable. A noise analysis showed there to be considerable variability in both the activity and capacitance of nets that cannot be accounted for in estimation. The proposed estimation models perform well given the noise limitations. This work has been published in [Ande 03], [Ande 04b], and [Ande 04e].

7.2 Future Work

There are several promising power-related FPGA research directions. Some of these are extensions of the work appearing in the preceding chapters, while others are not specifically linked to the work presented here. Suggestions of both types are highlighted below.

7.2.1 Extensions of this Research Work

One of the techniques proposed for leakage power reduction in Chapter 3 is to change the polarity of logic signals with low static probability (≤ 0.5), thus placing circuit structures into low leakage states. This optimization relies on knowledge of the static probability of all design signals. A direction for future research is to alter upstream CAD tools to make them "static probability-aware". It was observed, in Section 3.3.1, that the polarity selection optimization is more effective in some circuits versus others, and that this is partly due to the distribution of static probabilities amongst circuit signals. Low leakage is achieved when many signals have static probability close to 0 or 1. The synthesis steps in the FPGA CAD flow could be adapted to produce mapping solutions such that signals between LUTs have static probabilities favourable to low leakage hardware states.

Leakage-aware routing (Section 3.4) reduces active leakage by discouraging the use of routing resources with high leakage power consumptions. A future research direction is to investigate

the impact of applying this idea earlier in the CAD flow. Since most leakage is consumed in the interconnect, reducing overall interconnect usage may benefit leakage. Interconnect usage is strongly impacted by decisions made during front-end synthesis and technology mapping. Interconnect usage optimization has been previously considered in the context of routability enhancement for early FPGAs [Schl 94], which had very limited routing flexibility. Perhaps such techniques should be revisited and evaluated in the leakage optimization context. Similarly, the leakage benefits of interconnect-driven clustering techniques (e.g, [Sing 02]) should be evaluated.

Many extensions of the routing circuitry design work in Chapter 4 are possible. A downside of the proposed switch designs is the area overhead needed to provide the programmable mode, which involves extra SRAM configuration cells and relatively large sleep transistors. This overhead can be addressed in a number of ways. One option is to look at *sharing* sleep transistors between two or more routing switches, instead of there being sleep transistors for each individual switch. It is possible that there are pairs of routing switches in the interconnect fabric that are never used or never transition concurrently. Such switches would be candidates for sharing sleep transistors. Likewise, the SRAM cell(s) needed to control the programmable mode selection could be shared amongst multiple switches.

A second option to mitigate the area overhead is to not make all routing switches "fully programmable". The results in Section 4.4 indicate that most routing switches can be slowed down and operated in low-power mode. Given this, one might design the interconnect such that most switches can only operate in low-power or sleep mode, with a limited number of switches able to operate in all three modes. Clearly, this idea would increase routing complexity, since the router would need to allocate the fully programmable switches to delay-critical connections. Placer complexity may also increase, as the placement solution should provide an adequate supply of fully programmable switches in the vicinity of delay-critical connections. Finally, another research direction is to extend the programmable mode concept into the FPGA logic blocks. A design trade-off here would involve appropriately partitioning logic blocks into sub-blocks, each having a separate programmable mode.

With regard to early power prediction, future work may include deploying the techniques proposed in Chapter 6 into power-aware FPGA CAD tools, and examining the power reduction benefits of basing CAD trade-offs on more accurate early power estimates. A second direction is to look at early leakage power estimation, which will likely become important as leakage dominance rises due to technology scaling. Since FPGAs contain a fixed number of pre-fabricated blocks, leakage power models for such blocks could be developed and used to gauge leakage early in the design flow.

The power optimization techniques proposed here were each described and evaluated independently. A future research direction is to combine all of the proposed techniques and examine their total benefit to dynamic and leakage power. It is unknown whether the benefits offered by each approach in isolation are cumulative when all approaches are applied simultaneously, or whether one given technique weakens the ability of another technique to reduce power.

7.2.2 Additional Power-Related Research Directions

The routing switch designs proposed in Chapter 4 offer a sleep mode that is suitable for unused routing switches. In the ASIC domain, sleep states are applied to reduce leakage in temporarily inactive circuitry, rather than in unused circuitry, as proposed here. The ASIC sleep concept could be applied to FPGAs, with portions of *used* FPGA circuitry being put to sleep when inactive. The sleep mode of the proposed switch designs could be selected/invoked via a sleep signal. Certainly, supplying a separate sleep signal to each routing switch would be impractical. Instead, one sleep signal could control all routing switches in a given FPGA region. The appropriate granularity of the sleep regions would need to be studied. Placement would need to be adapted to be conscious of the sleep regions. A related idea involves the selection of low-power mode via a control signal, instead of through configuration SRAM cells, as was assumed in Chapter 4. The "selectable" low-power mode could then be invoked when system workload requirements are low enough to permit slowing down the FPGA, as is done with frequency throttling in custom ASICs.

There likely exists significant potential for power reduction at early stages of the FPGA design flow, namely, in front-end synthesis or technology mapping. The unique, pre-fabricated features of FPGAs can be leveraged for power reduction. As a concrete example, consider that, in addition to LUTs and flip-flops, commercial FPGAs contain large blocks of SRAM. Previous work has examined the area and speed consequences of implementing combinational logic in an FPGA's SRAM blocks versus in LUTs (e.g., [Wilt 00]). The effect of this optimization on power-efficiency has not yet been examined. However, it holds promise since it may be possible to realize high activity logic in the SRAMs, thus avoiding the need to route some high activity signals through the power-dominant FPGA interconnect.

7.3 Closing Remarks

In summary, we believe that power optimization and prediction for FPGAs will be an active research area in the years ahead. The techniques proposed in this dissertation provide power reductions on the order of tens of percent, and this is certainly a good start. However, further improvements of the same or even larger magnitude will be needed if FPGAs are to truly compete with ASICs from the power perspective. The future research directions noted above leave us optimistic that there remains considerable room for improving FPGA power, thereby increasing the competitiveness of FPGAs relative to custom ASICs.

A Power Estimation Model Regression Analysis Results

A.1 Switching Activity Prediction

Table A.1 shows the prediction model and regression details. Shading is used to differentiate the logic-delay activity-based prediction model from the (separate) zero-delay activity-based prediction model. Column 1 of the table lists the parameters used in each model. Observe that all of the terms of (6.7) are not present for each model – the *step* function in R was used to prune (6.7) to contain only the required, significant terms [GnuR 03]. Column 2 presents the coefficient values for each parameter. Column 3 of the table gives the *P-value* for each parameter, which is a frequently-used significance metric in regression analysis. It represents, for a given parameter, the probability that the parameter's actual coefficient is zero rather than the value specified in the model. Thus, low P-values are generally associated with high parameter significance. Column 4 of the table gives the 95% confidence bounds for each parameter's coefficient.

A.2 Capacitance Prediction

The parameter coefficients for model M10 and other regression analysis data are provided in Table A.2. The unshaded portion of the table shows the low-fanout net prediction model (≤ 10 loads); the shaded portion of the table corresponds to the prediction model used for high-fanout nets. About 5% of the nets in the circuits are high-fanout. The columns of the table are analogous to those of Table A.1. Observe that a very simple model suffices for predicting the capacitance of high-fanout nets. Only the net's fanout, bounding box, the number of CLB tiles, and pin types are included. Further, the P-value for the bounding box term implies a

Table A.1: Prediction model and regression analysis details (zero-delay activity and logic-delay activity-based prediction models).

Parameter	**Coeff**	**P-value**	**Lower 95% / Upper 95%**
int	35.909	<2E-16	31.330 / 40.488
$PROP$	37.413	<2E-16	33.184 / 41.642
GEN	-3.653	0.129	-8.369 / 1.064
D	3.214	<2E-16	2.866 / 3.563
$PROP^2$	-0.728	0.135	-1.683 / 0.227
GEN^2	-0.950	0.109	-2.113 / 0.212
$PROP \cdot D$	-0.484	<2E-16	-0.593 / -0.374
$GEN \cdot D$	0.313	0.0033	0.104 / 0.523
int	48.875	<2E-16	46.008 / 51.742
$PROP$	21.361	<2E-16	18.014 / 24.709
GEN	-4.165	8.613E-05	-6.241 / -2.088
D	-2.276	<2E-16	-2.630 / -1.921
$PROP^2$	0.974	6.905E-04	0.412 / 1.536
D^2	0.042	<2E-16	0.034 / 0.050
$PROP \cdot D$	-0.452	3.1E-13	-0.573 / -0.331
$PROP \cdot GEN$	1.193	0.08697	-0.173 / 2.559

fairly weak significance. The effect of dropping this term was investigated and only a small decrease in prediction accuracy was observed (<1%). The use of more complex models for the high-fanout nets was found to result in "overfitting" to the characterization circuits that led to lower prediction accuracy in the test circuits. This is likely due to a greater noise presence in high-fanout versus low-fanout nets. The overfitting phenomenon was not observed for the low-fanout net modeling: better fitting during characterization for such nets resulted in better prediction accuracy.

Table A.2: Prediction model and regression analysis details (low-fanout and high-fanout prediction models).

Parameter	**Coeff**	**P-Value**	**Lower 95% / Upper 95%**
int	-544.425	<2E-16	-662.320 / -426.530
FO	359.435	1.48E-13	264.419 / 454.452
BB	53.297	6.72E-10	36.412 / 70.182
NT	521.498	3.39E-15	392.219 / 650.777
GP	-177.252	1.23E-9	-234.301 / -120.202
$X6$	74.155	2.5E-4	34.537 / 113.774
$Y6$	52.402	0.00903	13.073 / 91.731
$XS \cdot YS$	-10.159	6.66E-6	-14.574 / -5.743
FO^2	22.694	0.09255	-3.751 / 49.139
BB^2	-1.222	0.01139	-2.168 / -0.276
NT^2	-69.444	2.25E-5	-101.536 / -37.352
FP^2	-44.168	0.00323	-73.551 / -14.785
GP^2	-53.275	0.00272	-88.097 / -18.453
$X6^2$	-9.777	0.02367	-18.246 / -1.308
$Y6^2$	-9.391	0.02421	-17.558 / -1.225
$(XS \cdot YS)^2$	0.053	0.00088	0.022 / 0.085
FO^3	-1.703	0.06502	-3.513 / 0.106
NT^3	4.638	7.92E-5	2.337 / 6.940
FP^3	3.862	0.01468	0.761 / 6.964
GP^3	5.335	0.00387	1.716 / 8.954
int	2012.409	9.96E-5	1011.598 / 3013.220
FO	67.576	0.04597	1.218 / 133.934
BB	-42.029	0.18612	-104.479 / 20.422
NT	182.588	0.00087	75.991 / 289.184
FP	216.387	5.01E-6	125.225 / 307.548

References

[Abdo 02] A. Abdollahi, F. Fallah, and M. Pedram. "Runtime Mechanisms for Leakage Current Reduction in CMOS VLSI Circuits". In: *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 213–218, Monterey, CA, 2002.

[Agar 04] A. Agarwal, C. Kim, S. Mukhopadhyay, and K. Roy. "Leakage in Nano-Scale Technologies: Mechanisms, Impact and Design Considerations". In: *ACM/IEEE Design Automation Conference*, pp. 6–11, San Diego, CA, 2004.

[Ahme 02] E. Ahmed and J. Rose. "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 85–94, Monterey, CA, 2002.

[Amer 98] E. Amersaekera and F. Najm. *Failure Mechanisms in Semiconductor Devices*. John Wiley and Sons, Toronto, 1998.

[Ande 02] J. Anderson and F. Najm. "Power-Aware Technology Mapping for LUT-Based FPGAs". In: *IEEE International Conference on Field-Programmable Technology*, pp. 211–218, Hong Kong, 2002.

[Ande 03] J. Anderson and F. Najm. "Switching Activity Analysis and Pre-Layout Activity Prediction for FPGAs". In: *ACM/IEEE International Workshop on System-Level Interconnect Prediction*, pp. 15–21, Monterey, CA, 2003.

[Ande 04a] J. Anderson, S. Nag, K. Chaudhary, S. Kalman, C. Madabhushi, and P. Cheng. "Run-Time-Conscious Automatic Timing-Driven FPGA Layout Synthesis". In: *International Conference on Field-Programmable Logic and Applications*, pp. 168–178, Antwerp, Belgium, 2004.

[Ande 04b] J. Anderson and F. Najm. "Interconnect Capacitance Estimation for FPGAs". In: *IEEE/ACM Asia and South Pacific Design Automation Conference*, pp. 713–718, Yokohama, Japan, 2004.

[Ande 04c] J. Anderson and F. Najm. "Low-Power Programmable FPGA Routing Circuitry". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 602–609, San Jose, CA, 2004.

[Ande 04d] J. Anderson and F. Najm. "A Novel Low-Power FPGA Routing Switch". In: *IEEE Custom Integrated Circuits Conference*, pp. 719–722, Orlando, FL, 2004.

[Ande 04e] J. Anderson and F. Najm. "Power Estimation Techniques for FPGAs". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 12, No. 10, pp. 1015–1027, Oct. 2004.

[Ande 04f] J. Anderson, F. Najm, and T. Tuan. "Active Leakage Power Optimization for FPGAs". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 33–41, Monterey, CA, 2004.

[Ande 05a] J. Anderson and F. Najm. "Active Leakage Power Optimization for FPGAs". *Accepted to appear in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005.

[Ande 05b] J. Anderson and F. Najm. "Low-Power Programmable FPGA Routing Circuitry". *Submitted to IEEE Transactions on Very Large Scale Integration (VLSI) Systems (under review)*, 2005.

[Anis 02] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry. "Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gate Clustering Technique". In: *ACM/IEEE Design Automation Conference*, pp. 480–485, New Orleans, LA, 2002.

[Aziz 04] N. Azizi and F. Najm. "An Asymmetric SRAM Cell to Lower Gate Leakage". In: *IEEE International Symposium on Quality Electronic Design*, pp. 534–539, San Jose, CA, 2004.

[Basu 04] A. Basu, S.-C. Lin, V. Wason, A. Mehrotra, and K. Banerjee. "Simultaneous Optimization of Supply and Threshold Voltages for Low-Power and High-Performance Circuits in the Leakage Dominant Era". In: *ACM/IEEE Design Automation Conference*, pp. 884–887, San Diego, CA, 2004.

[Berk 04] *Berkeley Predictive Technology Model (http://www.device.eecs.berkeley.edu/∼ptm/).* University of California, Berkeley, 2004.

[Betz 96] V. Betz and J. Rose. "Directional Bias and Non-Uniformity in FPGA Global Routing". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 652–659, San Jose, CA, 1996.

[Betz 97a] V. Betz and J. Rose. "Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size". In: *IEEE Custom Integrated Circuits Conference*, pp. 551–554, Santa Clara, CA, 1997.

[Betz 97b] V. Betz and J. Rose. "VPR: A New Packing, Placement and Routing Tool for FPGA Research". In: *International Workshop on Field-Programmable Logic and Applications*, pp. 213–222, London, UK, 1997.

[Betz 98] V. Betz. "Architecture and CAD for the Speed and Area Optimization of FPGAs". In: *Ph.D. Thesis*, Department of Electrical and Computer Engineering, University of Toronto, Tornto, Ontario, Canada, 1998.

[Betz 99a] V. Betz and J. Rose. "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 140–149, Monterey, CA, 1999.

[Betz 99b] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Boston, MA, 1999.

[Boda 00] S. Bodapati and F. N. Najm. "Pre-Layout Estimation of Individual Wire Lengths". In: *ACM International Workshop on System-Level Interconnect Prediction*, pp. 91–96, San Diego, CA, 2000.

[Bork 99] S. Borkar. "Design Challenges of Technology Scaling". *IEEE Micro*, Vol. 19, No. 4, pp. 23–29, 1999.

[Brow 90] S. Brown, J. Rose, and Z. Vranesic. "A Detailed Router for Field Programmable Gate Arrays". In: *IEEE International Conference on Computer-Aided Design*, pp. 382–385, San Jose, CA, 1990.

[BSIM4 04] *UC Berkeley MOSFET Simulation Model*. University of California, Berkeley, 2004.

[Calh 03] B. Calhoun, F. Honore, and A. Chandrakasan. "Design Methodology for Fine-Grained Leakage Control in MTCMOS". In: *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 104–109, Seoul, Korea, 2003.

[Chan 92] A. Chandrakasan, S. Sheng, and R. Brodersen. "Low-Power CMOS Digital Design". *IEEE Journal of Solid State Circuits*, Vol. 27, No. 4, pp. 473–484, Apr. 1992.

[Chen 03] D. Chen, J. Cong, and Y. Fan. "Low-Power High-Level Synthesis for FPGA Architectures". In: *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 134–139, Seoul, Korea, 2003.

[Chen 04a] D. Chen and J. Cong. "Register Binding and Port Assignment for Multiplexer Optimization". In: *IEEE/ACM Asia and South Pacific Design Automation Conference*, pp. 68–73, Yokohama, Japan, 2004.

[Chen 04b] D. Chen, J. Cong, F. Li, and L. He. "Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 109–117, Monterey, CA, 2004.

[Chen 92] K. Chen, J. Cong, Y. Ding, A. Kahng, and P. Trajmar. "DAG-Map: Graph-Based FPGA Technology Mapping for Delay Optimization". *IEEE Design and Test of Computers*, pp. 13–26, Sep. 1992.

[Chou 97] T.-L. Chou and K. Roy. "Statistical Estimation of Combinational and Sequential CMOS Digital Circuit Activity Considering Uncertainty of Gate Delays". *IEICE (Japan) Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 95–100, 1997.

[Cicc 04] L. Ciccarelli, A. Lodi, and R. Canegallo. "Low Leakage Circuit Design for FPGAs". In: *IEEE Custom Integrated Circuits Conference*, pp. 715–718, Orlando, FL, 2004.

[Ciri 87] M. Cirit. "Estimating Dynamic Power Consumption of CMOS Circuits". In: *IEEE International Conference on Computer-Aided Design*, pp. 534–537, San Jose, CA, 1987.

[Clar 02] L. Clark, S. Demmons, N. Deutscher, and F. Ricci. "Standby Power Management for a 0.18um Microprocessor". In: *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 7–12, Monterey, CA, 2002.

[Cong 94a] J. Cong and Y. Ding. "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Look-up-Table Based FPGA Designs". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 1, pp. 1–12, 1994.

[Cong 94b] J. Cong and Y. Ding. "On Area/Depth Trade-Off in LUT-Based FPGA Technology Mapping". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 2, No. 2, pp. 137–148, 1994.

[Cong 97] J. Cong, Z. Pan, L. He, C.-K. Koh, and K.-Y. Khoo. "Interconnect Design for Deep Submicron ICs". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 478–485, San Jose, CA, 1997.

[Cong 99] J. Cong, C. Wu, and E. Ding. "Cut Ranking and Pruning: Enabling a General And Efficient FPGA Mapping Solution". In: *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 29–35, Monterey, CA, 1999.

[Dona 81] W. Donath. "Wire Length Distribution for Placement of Computer Logic". *IBM Journal of Research and Development*, Vol. 25, No. 152, 1981.

[Doyl 02] B. Doyle, R. Arghavani, D. Barlage, S. Datta, M. Doczy, J. Kavalieros, A. Murthy, and R. Chau. "Transistor Elements for 30nm Physical Gate Lengths and Beyond". *Intel Technology Journal*, Vol. 6, No. 2, pp. 42–54, May 16 2002.

[Farr 94] A. H. Farrahi and M. Sarrafzadeh. "FPGA Technology Mapping for Power Minimization". In: *International Workshop on Field-Programmable Logic and Applications*, pp. 167–174, Prague, Czech Republic, 1994.

[Fran 02] D. Frank. "Power-Constrained CMOS Scaling Limits". *IBM Journal of Research and Development*, Vol. 46, No. 2/3, pp. 235–244, March 2002.

[Fran 90] R. Francis, J. Rose, and K. Chung. "Chortle: A Technology Mapping Program for Lookup Table-Based Field Programmable Gate Arrays". In: *ACM/IEEE Design Automation Conference*, pp. 613–619, Orlando, FL, 1990.

[Fran 91a] R. Francis, J. Rose, and Z. Vranesic. "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs". In: *ACM/IEEE Design Automation Conference*, pp. 227–233, San Francisco, CA, June 1991.

[Fran 91b] R. Francis, J. Rose, and Z. Vranesic. "Technology Mapping for Lookup Table-Based FPGAs for Performance". In: *IEEE International Conference on Computer-Aided Design*, pp. 568–571, 1991.

[Full 04] B. Fuller. "IC Vendors Lock Horns Over Design Approaches". *EE Times*, February 11 2004.

[Gaya 04a] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan. "A Dual-Vdd Low Power FPGA Architecture". In: *International Conference on Field-Programmable Logic and Applications*, pp. 145–157, Antwerp, Belgium, 2004.

[Gaya 04b] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan. "Reducing Leakage Energy in FPGAs Using Region-Constrained Placement". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 51–58, Monterey, CA, 2004.

[Geor 01] V. George and J. Rabaey. *Low-Energy FPGAs: Architecture and Design.* Kluwer Academic Publishers, Boston, MA, 2001.

[Geor 99] V. George, H. Zhang, and J. Rabaey. "The Design of a Low Energy FPGA". In: *ACM International Symposium on Low Power Electronics and Design*, pp. 188–193, San Diego, CA, 1999.

[GnuR 03] *The R Project for Statistical Computing (http://www.r-project.org).* GNU, 2003.

[Guin 03] R. Guindi and F. Najm. "Design Techniques for Gate-Leakage Reduction in CMOS Circuits". In: *IEEE International Symposium on Quality Electronic Design*, pp. 61–65, San Jose, CA, 2003.

[Halt 97] J. Halter and F. Najm. "A Gate-level Leakage Power Reduction Method for Ultra-Low-Power CMOS Circuits". In: *IEEE Custom Integrated Circuits Conference*, pp. 475–478, Santa Clara, CA, 1997.

[Hawk 03] D. Hawk and K. Kolwicz. "Design Your Own ASIC". *Advanced Packaging Magazine*, Sep. 2003.

[Hutt 02] M. Hutton, V. Chan, P. Kazarian, V. Maruri, T. Ngai, J. Park, R. Patel, B. Pedersen, J. Schleicher, and S. Shumarayev. "Interconnect Enhancements for a High-Speed PLD Architecture". In: *ACM/SIGDA International Symposium on FPGAs*, pp. 3–10, Monterey, CA, 2002.

[Hutt 03] M. Hutton, K. Adibsamii, and A. Leaver. "Adaptive Delay Estimation for Partitioning-Driven PLD Placement". *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 11, No. 1, pp. 60–63, Feb. 2003.

[Hwan 98] J.-M. Hwang, F.-Y. Chiang, and T.-T. Hwang. "A Re-Engineering Approach to Low Power FPGA Design Using SPFD". In: *ACM/IEEE Design Automation Conference*, pp. 167–174, San Francisco, CA, 1998.

[Inte 02] *International Technology Roadmap for Semiconductors (ITRS).* http://www.itrs.org, 2002.

[Jari 04] D. Jariwala and J. Lillis. "On the Interactions Between Routing and Detailed Placement". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 387–393, San Jose, CA, 2004.

[Jian 02] W. Jiang, V. Tiwari, E. de la Iglesia, and A. Sinha. "Topological Analysis for Leakage Prediction of Digital Circuits". In: *IEEE International Conference on VLSI Design*, pp. 39–44, Bangalore, India, 2002.

[John 02] M. Johnson, D. Somasekhar, L.-Y. Choiu, and K. Roy. "Leakage Control with Efficient Use of Transistor Stacks in Single Threshold CMOS". *IEEE Transactions on Very Large Scale Integared (VLSI) Systems*, Vol. 10, No. 1, pp. 1–5, Feb. 2002.

[Juan 01] J. Juan-Chico, M. Bellido, P. R. de Clavijo, C. Baena, C. Jimenez, and M. Valencia. "Switching Activity Evaluation of CMOS Digital Circuits Using

Logic Timing Simulation". *IEEE Electronics Letters*, Vol. 37, No. 9, pp. 555–557, April 26 2001.

[Kao 02] J. Kao, S. Narendra, and A. Chandrakasan. "Subthreshold Leakage Modeling and Reduction Techniques". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 141–148, San Jose, CA, 2002.

[Karn 95] T. Karnik and S.-M. Kang. "An Empirical Model for Accurate Estimation of Routing Delay in FPGAs". In: *IEEE International Conference on Computer-Aided Design*, pp. 328–331, 1995.

[Kesh 01] A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, and V. De. "Effectiveness of Reverse Body Bias for Leakage Control in Scaled Dual Vt CMOS ICs". In: *ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 207–211, Huntington Beach, CA, 2001.

[Kim 02] C. Kim and K. Roy. "Dynamic Vth Scaling Scheme for Active Leakage Power Reduction". In: *IEEE Design, Automation and Test in Europe Conference*, pp. 163–167, Paris, France, 2002.

[Kim 03] C. Kim, J.-J. Kim, S. Mukhopadhyay, and K. Roy. "A Forward Body-Biased Low-Leakage SRAM Cache: Device and Architecture Considerations". In: *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 6–9, Seoul, Korea, 2003.

[Kris 02] R. Krishnamurthy, A. Alvandpour, V. De., and S. Borkar. "High-Performance and Low-Power Challenges for Sub-70nm Microprocessor Circuits". In: *IEEE Custom Integrated Circuits Conference*, pp. 125–128, San Jose, CA, 2002.

[Kuma 98] K. Kumagai, H. Iwaki, H. Yoshida, H. Suzuki, T. Yamada, and S. Kurosawa. "A Novel Powering-Down Scheme for Low Vt CMOS Circuits". In: *IEEE Symposium on VLSI Circuits*, pp. 44–45, Honolulu, HI, 1998.

[Kumt 00a] B. Kumthekar, L. Benini, E. Macii, and F. Somenzi. "Power Optimisation in FPGA-based Design Without Rewiring". *IEE Proc. Comput. Digit. Tech.*, Vol. 147, No. 3, pp. 167–174, May 2000.

[Kumt 00b] B. Kumthekar and F. Somenzi. "Power and Delay Reduction via Simultaneous Logic and Placement Optimization in FPGAs". In: *ACM/IEEE Design, Automation and Test in Europe Conference*, pp. 202–207, 2000.

[Kuss 98] E. Kusse and J. Rabaey. "Low-Energy Embedded FPGA Structures". In: *ACM/IEEE International Symposium on Low-Power Electronics Design*, pp. 155–160, Monterey, CA, 1998.

[Lamm 03] D. Lammers. "Chip Industry Growing, But Experts Wonder For How Long". *EE Times*, December 2003.

[Lamo 03] J. Lamoureux and S. Wilton. "On the Interaction Between Power-Aware FPGA CAD Algorithms". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 701–708, San Jose, CA, 2003.

[Lee 03] D. Lee and D. Blaauw. "Static Leakage Reduction Through Simultaneous Threshold Voltage and State Assignment". In: *ACM/IEEE Design Automation Conference*, pp. 191–194, Anaheim, CA, 2003.

[Lee 04] D. Lee, D. Blaauw, and D. Sylvester. "Gate Oxide Leakage Current Analysis and Reduction for VLSI Circuits". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 12, No. 2, pp. 155–166, Feb. 2004.

[Lemi 02] G. Lemieux and D. Lewis. "Circuit Design of Routing Switches". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 19–28, Monterey, CA, 2002.

[Lemi 03] G. Lemieux. "Routing Architecture for Field-Programmable Gate Arrays". In: *Ph.D. Thesis*, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada, 2003.

[Lemi 04] G. Lemieux. "Directional and Single-Driver Wires in FPGA Interconnect". In: *IEEE International Conference on Field-Programmable Technology*, pp. 41–48, Brisbane, Australia, 2004.

[Lemi 93] G. Lemieux and S. Brown. "A Detailed Router for Allocating Wire Segments in FPGAs". In: *ACM/SIGDA Physical Design Workshop*, pp. 215–226, Lake Arrowhead, CA, 1993.

[Lewi 03] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose. "The Stratix Routing and Logic Architecture". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 12–20, Monterey, CA, 2003.

[Li 01] H. Li, W.-K. Mak, and S. Katkoori. "LUT-Based FPGA Technology Mapping for Power Minimization with Optimal Depth". In: *IEEE Computer Society Workshop on VLSI*, pp. 123–128, Orlando, FL, 2001.

[Li 03] F. Li, D. Chen, L. He, and J. Cong. "Architecture Evaluation for Power-Efficient FPGAs". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 175–184, Monterey, CA, 2003.

[Li 04a] F. Li and L.He. "Vdd Programmability to Reduce FPGA Interconnect Power". In: *IEEE/ACM International Conference on Computer-Aided Design*, pp. 760–765, San Jose, CA, 2004.

[Li 04b] F. Li, Y. Lin, and L. He. "FPGA Power Reduction Using Configurable Dual-Vdd". In: *ACM/IEEE Design Automation Conference*, pp. 735–740, San Diego, CA, 2004.

[Li 04c] F. Li, Y. Lin, L. He, and J. Cong. "Low-Power FPGA Using Pre-Defined Dual-Vdd/Dual-Vt Fabrics". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 42–50, Monterey, CA, 2004.

[Liu 02] Y. Liu and Z. Gao. "Timing Analysis of Transistor Stack for Leakage Power Saving". In: *IEEE International Conference on Electronics, Circuits and Systems*, pp. 41–44, Dubrovnik, Croatia, 2002.

[Lou 02] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng. "Estimating Routing Congestion Using Probabilistic Analysis". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No. 1, pp. 32–41, Jan. 2002.

[Maid 03] P. Maidee, C. Ababei, and K. Bazargan. "Fast Timing-driven Partitioning-based Placement for Island Style FPGAs". In: *ACM/IEEE Design Automation Conference*, pp. 598–603, Anaheim, CA, 2003.

[Marc 98] R. Marculescu, D. Marculescu, and M. Pedram. "Probabilistic Modeling of Dependencies During Switching Activity Analysis". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 2, pp. 73–83, Feb. 1998.

[Marq 00] A. Marquardt, V. Betz, and J. Rose. "Timing-Driven Placement for FPGAs". In: *ACM International Symposium on Field-Programmable Gate Arrays*, pp. 203–213, Monterey, CA, 2000.

[Marq 99] A. Marquardt, V. Betz, and J. Rose. "Using Cluster Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 37–46, Monterey, CA, 1999.

[Mart 02] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads". In: *IEEE International Conference on Computer-Aided Design*, pp. 721–725, San Jose, CA, 2002.

[McMu 95] L. McMurchie and C. Ebeling. "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 111–117, Monterey, CA, 1995.

[Meht 95] H. Mehta, M. Borah, R. M. Owens, and M. J. Irwin. "Accurate Estimation of Combinational Circuit Activity". In: *ACM/IEEE Design Automation Conference*, pp. 618–622, San Francisco, CA, 1995.

[Mont 93] J. Monterio, S. Devadas, and A. Ghosh. "Retiming Sequential Circuits for Low Power". In: *IEEE International Conference on Computer-Aided Design*, pp. 398–402, San Jose, CA, 1993.

[Najm 93] F. Najm. "Transition Density: A New Measure of Activity in Digital Circuits". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, pp. 310–323, Feb. 1993.

[Najm 94] F. N. Najm. "A Survey of Power Estimation Techniques in VLSI Circuits". *IEEE Transactions on Very-Large Scale Integration (VLSI) Systems*, Vol. 2, No. 4, pp. 446–455, Dec. 1994.

[Nare 01] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan. "Scaling of Stack Effect and its Application for Leakage Reduction". In: *ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 195–200, Huntington Beach, CA, 2001.

[Nema 96] M. Nemani and F. N. Najm. "Towards a High-Level Power Estimation Capability". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 6, pp. 588–598, June 1996.

[Nema 99] M. Nemani and F. N. Najm. "High-Level Area and Power Estimation for VLSI Circuits". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 6, pp. 697–713, June 1999.

[Nguy 03] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson, and K. Keutzer. "Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization". In: *ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 158–163, Seoul, Korea, 2003.

[Nowa 02] E. Nowak. "Maintaining the Benefits of CMOS Scaling when Scaling Bogs Down". *IBM Journal of Research and Development*, Vol. 46, No. 2/3, pp. 169–180, March 2002.

[Poon 02a] K. Poon. "Power Estimation for Field Programmable Gate Arrays". In: *M.A.Sc. Thesis*, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, British Columbia, Canada, 2002.

[Poon 02b] K. Poon, A. Yan, and S. J. E. Wilton. "A Flexible Power Model for FPGAs". In: *International Conference on Field-Programmable Logic and Applications*, pp. 312–321, Montpellier, France, 2002.

[Rahm 04] A. Rahman and V. Polavarapuv. "Evaluation of Low-Leakage Design Techniques for Field-Programmable Gate Arrays". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 23–30, Monterey, CA, 2004.

[Rose 89] J. Rose, R. Francis, P. Chow, and D. Lewis. "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays". In: *IEEE Custom Integrated Circuits Conference*, pp. 5.3.1–5.3.5, San Diego, CA, 1989.

[Rose 91] J. Rose and S. Brown. "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays". *IEEE Journal of Solid State Circuits*, Vol. 26, No. 3, pp. 277–282, 1991.

[Roy 03] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits". In: *Proceedings of the IEEE*, pp. 305–327, Feb. 2003.

[Roy 99] K. Roy. "Power-Dissipation Driven FPGA Place and Route Under Timing Constraints". *IEEE Transactions On Circuits and Systems*, Vol. 46, No. 5, pp. 634–637, May 1999.

[Saku 02] T. Sakurai. "Minimizing Power Across Multiple Technology and Design Levels". In: *IEEE International Conference on Computer-Aided Design*, pp. 24–27, San Jose, CA, 2002.

[Sant 03] M. Santarini. "Are Structured ASICs a Dead End for EDA?". *EE Times*, June 23 2003.

[Schl 94] M. Schlag, J. Kong, and P. Chan. "Routability-Driven Technology Mapping for Lookup Table-Based FPGAs". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 1, pp. 13–26, 1994.

[Sedr 97] A. Sedra and K. Smith. *Microelectronic Circuits*. Oxford University Press, Toronto, fourth Ed., 1997.

[Sent 92] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. "SIS: A

System for Sequential Circuit Synthesis". *University of California at Berkeley, Memorandum No. UCB/ERL M92/41*, 1992.

[Shan 02] L. Shang, A. Kaviani, and K. Bathala. "Dynamic Power Consumption in the Virtex-II FPGA Family". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 157–164, Monterey, CA, 2002.

[Shen 92] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer. "On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks". In: *IEEE International Conference on Computer-Aided Design*, pp. 402–407, Santa Clara, CA, 1992.

[Sing 02] A. Singh and M. Marek-Sadowska. "Efficient Circuit Clustering for Area and Power Reduction in FPGAs". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 59–66, Monterey, CA, Feb. 2002.

[Sing 90] S. Singh, J. Rose, D. Lewis, K. Chung, and P. Chow. "Optimization of Field-Programmable Gate Array Logic Block Architecture for Speed". In: *IEEE Custom Integrated Circuits Conference*, pp. 6.1.1–6.1.6, Boston, MA, 1990.

[Siri 02] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda, and D. Blaauw. "Duet: An Accurate Leakage Estimation and Optimization Tool for Dual-Vt Circuits". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 10, No. 2, pp. 79–90, Apr. 2002.

[Soel 00] H. Soeleman, K. Roy, and T.-L. Chou. "Estimating Circuit Activity in Combinational CMOS Digital Circuits". *IEEE Design and Test of Computers*, pp. 112–119, April-June 2000.

[Spar 04] *Spartan-3 FPGA Data Sheet*. Xilinx, Inc., San Jose, CA, 2004.

[Sriv 04a] A. Srivastava, D. Sylvester, and D. Blaauw. "Power Minimization Using Simultaneous Gate Sizing, Dual-Vdd and Dual-Vth Assignment". In: *ACM/IEEE Design Automation Conference*, pp. 783–787, San Diego, CA, 2004.

[Sriv 04b] A. Srivastava, D. Sylvester, and D. Blaauw. "Statistical Optimization of Leakage Power Considering Process Variations using Dual-Vth Sizing". In: *ACM/IEEE Design Automation Conference*, pp. 773–778, San Diego, CA, 2004.

[Stok 03] L. Stok and J. Cohn. "There is Life Left in ASICs". In: *ACM/IEEE International Symposium on Physical Design*, pp. 48–50, Monterey, CA, 2003.

[Stra 03] *Stratix FPGA Device Handbook*. Altera Corp., San Jose, CA, 2003.

[Stra 04] *Stratix-II FPGA Data Sheet.* Altera Corp., San Jose, CA, 2004.

[Sult 04] A. Sultania, D. Sylvester, and S. Sapatnekar. "Tradeoffs between Gate Oxide Leakage and Delay for Dual Tox Circuits". In: *ACM/IEEE Design Automation Conference*, pp. 761–766, San Diego, CA, 2004.

[Swar 98] J. Swartz, V. Betz, and J. Rose. "A Fast Routability-Driven Router for FPGAs". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 140–149, Monterey, CA, 1998.

[Synopsys 04] *Synopsys Design Compiler Reference Manual: Contraints and Timing.* Synopsys, Inc., 2004.

[Taur 02] Y. Taur. "CMOS Design Near the Limit of Scaling". *IBM Journal of Research and Development*, Vol. 46, No. 2/3, pp. 213–222, March 2002.

[Thom 98] S. Thompson, P. Packan, and M. Bohr. "MOS Scaling: Transistor Challenges for the 21st Century". *Intel Technology Journal*, Vol. Q3'98, 1998.

[Toga 98] N. Togawa, K. Ukai, M. Yanagisawa, and T. Ohtsuki. "A Simultaneous Placement and Global Routing Algorithm for FPGAs with Power Optimization". In: *IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 125–128, Chiangmai, Thailand, 1998.

[TSMCPROC 02] *0.18um Process Models.* Taiwan Semiconductor Manufacturing Corporation, 2002.

[Tuan 03] T. Tuan and B. Lai. "Leakage Power Analysis of a 90nm FPGA". In: *IEEE Custom Integrated Circuits Conference*, pp. 57–60, San Jose, CA, 2003.

[Usam 02] K. Usami, N. Kawabe, M. Koizumi, K. Seta, and T. Furusawa. "Automated Selective Multi-Threshold Design for Ultra-Low Standby Applications". In: *IEEE International Conference on Low-Power Electronics and Design*, pp. 202–206, Monterey, CA, 2002.

[Virt 03] *Virtex II PRO FPGA Data Sheet.* Xilinx, Inc., San Jose, CA, 2003.

[Virt 04] *Virtex-4 FPGA Data Sheet.* Xilinx, Inc., San Jose, CA, 2004.

[Wang 01] Z.-H. H. Wang, E.-C. Liu, J. Lai, and T.-C. Wang. "Power Minimization in LUT-Based FPGA Technology Mapping". In: *ACM/IEEE Asia and South Pacific Design Automation Conference*, pp. 635–640, Yokohama, Japan, 2001.

[Wang 02] Q. Wang and S. B. K. Vrudhula. "Algorithms for Minimizaing Standby Power in Deep Submicrometer, Dual-Vt CMOS Circuits". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No. 3, pp. 306–318, March 2002.

[Wang 97] C.-C. Wang and C.-P. Kwan. "Low Power Technology Mapping by Hiding High-Transition Paths in Invisible Edges for LUT-Based FPGAs". In: *IEEE International Symposium On Circuits and Systems*, pp. 1536–1539, Hong Kong, 1997.

[Wilt 00] S. Wilton. "Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 1, pp. 56–68, Jan. 2000.

[Wilt 04] S. Wilton, S.-S. Ang, and W. Luk. "The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays". In: *International Conference on Field Programmable Logic and Applications*, pp. 719–728, Antwerp, Belgium, 2004.

[Wrig 00] R. L. Wright and M. A. Shanblatt. "Improved Switching Activity Estimation for Behavioral and Gate Level Designs". In: *IEEE Midwest Symposium on Circuits and Systems*, pp. 172–175, Lansing, MI, 2000.

[X4K 02] *XC4000XLA FPGA Data Sheet*. Xilinx, Inc., San Jose, CA, 2002.

[XilinxCh 04] *Xilinx ChipScope Pro (http://www.xilinx.com/ise/verification/chipscope_pro.htm)*. Xilinx, Inc., 2004.

[XPower 03] *Xilinx Power Tools (http://www.xilinx.com/ise/power_tools)*. Xilinx, Inc., 2003.

[Ye 04] A. Ye, J. Rose, and D. Lewis. "Using Multi-Bit Logic Blocks and Automated Packing to Improve Field-Programmable Gate Array Density for Implementing Datapath Circuits". In: *IEEE International Conference on Field-Programmable Technology*, pp. 129–136, Brisbane, Australia, 2004.

[Yeap 98] G. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Publishers, Boston, MA, 1998.

[Yu 00] B. Yu, H. Wang, C. Riccobene, Q. Xiang, and M.-R. Lin. "Limits of Gate-Oxide Scaling in Nano-Transistors". In: *IEEE Symposium on VLSI Technology*, pp. 90–91, Honolulu, Hawaii, 2000.

[Zeit 04] P. Zeitzoff. "MOSFET Scaling Trends and Challenges Through the End of the Roadmap". In: *IEEE Custom Integrated Circuits Conference*, pp. 233–240, Orlando, FL, 2004.

[Zuch 02] P. Zuchowski, C. Reynolds, R. Grupp, S. Davis, B. Cremen, and B. Troxel. "A Hybrid ASIC and FPGA Architecture". In: *IEEE International Conference on Computer-Aided Design*, pp. 187–194, San Jose, CA, 2002.