

UNIVERSITY OF CALIFORNIA

Los Angeles

**FPGA Performance and Power Evaluation  
Considering Process Variation**

A dissertation prospectus for the degree  
Doctor of Philosophy in Electrical Engineering

by

**Lerong Cheng**

2005

© Copyright by  
Lerong Cheng  
2005

# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Preliminaries and Background for FPGA Architecture</b>	<b>4</b>
2.1	Conventional FPGA Architecture	4
2.2	Vdd gatable FPGA architecture	5
<b>3</b>	<b>Device and Architecture Co-Optimization</b>	<b>7</b>
3.1	Trace-based Estimation	7
3.1.1	Trace collection	7
3.1.2	Dynamic Power Model	8
3.1.3	Leakage Power Model	10
3.1.4	Delay Model	11
3.1.5	Validation of Ptrace	12
3.2	Hyper-arch Evaluation	13
3.2.1	Overview	13
3.2.2	Necessity of device and architecture co-optimization	15
3.2.3	Energy and delay tradeoff	17
3.2.4	ED and Area Tradeoff	21
<b>4</b>	<b>Power and Delay Optimization with Process Variation</b>	<b>24</b>
4.1	Leakage and Timing Variations	24
4.1.1	Leakage under Variation	24

4.1.2	Timing under Variation . . . . .	27
4.2	Yield Models . . . . .	27
4.2.1	Leakage Yield . . . . .	27
4.2.2	Timing Yield . . . . .	29
4.2.3	Leakage and Timing Combined Yield . . . . .	30
4.3	Leakage and Timing Yield Analysis . . . . .	31
4.3.1	Leakage Yield . . . . .	31
4.3.2	Timing Yield . . . . .	36
4.3.3	Leakage and Timing Combined Yield . . . . .	37
<b>5</b>	<b>Future Work . . . . .</b>	<b>40</b>
	<b>References . . . . .</b>	<b>41</b>

## LIST OF FIGURES

2.1	FPGA logic block and basic logic element. . . . .	5
2.2	(a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches. . . . .	5
2.3	(a) Vdd-gateable switch; (b) Vdd-gateable routing switch; (c) Vdd-gateable connection block; (d) Vdd-gateable logic block. . . . .	6
3.1	Cycle-accurate simulation versus trace-based estimation. . . . .	8
3.2	Comparison between Psim and Ptrace . . . . .	12
3.3	Hyper-archs within fix delay range. (a) Delay ranges from 9.5ns to 10.5ns. (b) Delay ranges from 19.5ns to 20.5ns. . . . .	16
3.4	Hyper-archs under different device setting. . . . .	18
3.5	Dom-archs. (a) <i>Homo-Vt</i> and <i>Hetero-Vt</i> (b) <i>Homo-Vt+G</i> and <i>Hetero-Vt+G</i> . . . . .	20
3.6	ED and area trade-off. ED and area are normalized with respect to the baseline. . . . .	23
4.1	Leakage power of baseline architecture (N=8, K=4) with ITRS device setting under intra-die and inter-die variations. . . . .	32
4.2	Energy-delay tradeoff among architectures in <i>Homo-Vt</i> using min-ED device setting. . . . .	35
4.3	Delay of baseline architecture (N=8, K=4) with ITRS device setting under intra-die and inter-die $\text{Leff}$ variation . . . . .	37
4.4	Leakage and delay of baseline architecture (N=8, K=4) with ITRS setting under process variations. . . . .	39

## LIST OF TABLES

3.1	Trace information, device and circuit parameters. . . . .	8
3.2	Switching activity comparison for different technology scale, Vdd and Vt. Architecture setting: $N = 10, K = 4$ . Unit: switch per clock cycle. . . . .	10
3.3	Short circuit power ratio comparison for different technology scale, Vdd and Vt. Architecture setting: $N = 10, K = 4$ . . . . .	10
3.4	MCNC benchmark list . . . . .	13
3.5	Baseline hyper-arch and evaluation ranges. . . . .	14
3.6	Summary of FPGA hyper-arch Classes. . . . .	14
3.7	Min-ED hyper arch of optimizing device and architecture separately and optimizing device and architecture simultaneously. . .	17
3.8	Power and delay ranges for different device settings. . . . .	18
3.9	Minimum delay and minimum energy hyper-archs. . . . .	19
3.10	Comparison between baseline and min-ED hyper-arch in <i>Homo-Vt</i> , <i>Hetero-Vt</i> , <i>Homo-Vt+G</i> , and <i>Hetero-Vt+G</i> . . . . .	21
3.11	Minimum ED-area product hyper-archs for different classes. ED, Area, and ED-area product are normalized with respect to Baseline. In the table, S refers to the sleep transistor size. ED, Area and AED are normalized with respect to baseline. . . . .	22
4.1	Comparison between analytical variation <i>models</i> and Monte Carlo ( <i>M-C</i> )simulation. . . . .	32
4.2	Comparison of Different Device Setting . . . . .	34

4.3	Comparison of leakage yield between classes. . . . .	36
4.4	Timing yield for Homo- $V_t$ +G . . . . .	38
4.5	Combined Leakage-delay yield between FPGA Classes. . . . .	39

## VITA

- 1979            Born, Guangzhou, China
- 2001            B.S. (Electrical Engineering), Zhongshan University,  
Guangzhou, China
- 2003            M.S. (Electrical Engineering), Portland State University
- 2003            Research Assistant, Electrical Engineering Department, CU  
Boulder
- 2004–present    Research Assistant, Electrical Engineering Department, UCLA

## PUBLICATIONS

P. Wong, L. Cheng, Y. Lin, and L. He, “FPGA Device and Architecture Co-Optimization Considering Process Variation”, *International Conference on Computer Aided Design*, June, 2005.

L. Cheng, P. Wong, F. Li, Y. Lin, and L. He, “Device and Architecture Co-Optimization for FPGA Power reduction”, *Design Automation Conference*, June, 2005.

F. He, X. Song, L. Cheng, G. Yang, Z. Tong, M. Gu, and J. Sun, “A Hierarchical Method for Wiring and Congestion Prediction”, *IEEE Computer Society Annual*

*Symposium on VLSI, May, 2005.*

L. Cheng, X. Song, G. Yang, and Z. Tang, “A Fast Congestion Estimator for Routing with Bounded Detours”, *Asian and South Pacific Design Automation Conference, Feb, 2004.*

L. Cheng, W. Hung, X. Song and G. Yang, “Congestion Estimation for Three-Dimensional Architectures”, *IEEE Computer Society Annual Symposium on VLSI, May, 2005.*

Y. Jeng and L. Cheng, “Digital Spectrum of The Non-Uniformly Sampled 2-Dimensional Signal And Its Reconstruction”, *IEEE Transactions on Instrumentations and Measurement, v.54, pp. 1180-1187, Jun 2005.*

L. Cheng, W. Hung, X. Song, and G. Yang, “Congestion Estimation for 3-D Circuit Architectures”, *IEEE Transactions on Circuits and Systems II, v.51, pp. 655-659, Dec 2004.*

W. Hung, X. Song, T. Kam, and L. Cheng, “Routability for Three-Dimensional Architecture”, *IEEE Transactions on VLSI, v.12, pp. 1371-1374, Dec 2004.*

ABSTRACT OF THE DISSERTATION PROSPECTUS

# FPGA Performance and Power Evaluation Considering Process Variation

by

**Lerong Cheng**

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2005

Professor Lei He, Chair

Power has become an increasingly important design constraint for FPGAs in nanometer technologies. Device optimization considering supply voltage  $V_{dd}$  and threshold voltage  $V_t$  has little chip area increase but has a great impact on power and performance in the nanometer technology. Moreover, process variations in nanometer technologies are becoming an important issue for FPGAs with a multi-million gate capacity. We develop an efficient yet accurate timing and power evaluation method, called trace-based model (Ptrace). And then extend it to consider process variation. With Ptrace, we perform device and architecture co-optimization. Compared to the baseline FPGA architecture which has the architecture same as the commercial FPGA used by Xilinx and has  $V_{dd}$  suggested by ITRS but  $V_t$  optimized by our device optimization, architecture and device co-optimization can reduce energy-delay product by 54.7% and chip area by 8.3%. Furthermore, if process variation is considered, architecture and device co-optimization increases leakage power yield by 73%. We also observe LUT size 4 gives the highest leakage yield, LUT size 7 gives the highest timing yield, and LUT size 5 achieves the maximum combined leakage and timing yield.

# CHAPTER 1

## Introduction

Field programmable gate array (FPGA) allows the same silicon implementation to be programmed or re-programmed for a variety of applications. It provides low NRE (non-recurring engineering) cost and short time to market. Due to the large number of transistors required for field programmability and the low utilization rate of FPGA resources, existing FPGAs consume more power compared to ASICs [KR98]. As the process advances to nanometer technology and low-energy embedded applications are explored for FPGAs, power consumption becomes a crucial design constraint for FPGAs.

Several recent papers have studied FPGA power modeling and optimization. [PYW02, LCH03, LH05] presented power evaluation frameworks for generic parameterized FPGA architectures and showed that both interconnect and leakage power are significant for FPGAs in nanometer technologies. [TL03] quantified the leakage power of a commercial FPGA architecture in 90nm technology. [DT05] presented a methodology for high-level FPGA power estimation leveraging existing commercial tools. FPGA power optimization involves CAD algorithms and novel circuits/architectures. [LW03] studied the interaction of a suite of power-aware FPGA CAD algorithms without changing the existing FPGAs. [ANT04] proposed configuration inversion method to reduce leakage power of multiplexers without any additional hardware. [MM04] presented a power-driven partition algorithm for mapping applications to FPGA with different V<sub>dd</sub>-levels.

[CCL04] presented a low-power technology mapping algorithm for dual-Vdd FPGAs. In addition, low power FPGA circuits and architectures have been proposed. [SGT05] designed a new type of routing multiplexer and proposes an input control method to reduce unused routing multiplexer leakage. [GTV04] developed region-based power-gating for unused FPGA logic blocks to reduce leakage power. [LCG05] studied low leakage interconnect circuitry, which combines gate biasing, body biasing and multi-threshold techniques to reduce interconnect leakage. [LLH05b] applied fine-grained power-gating to FPGA interconnects. [LLH04b, LLH04a] were the first work introducing dual-Vdd and field programmability of Vdd to FPGA. Vdd programmability has been applied to both FPGA logic blocks [LLH04b, LLH04a] and interconnects [GLV04, Fei04, Jas04].

Previously, conventional FPGA architecture evaluation has been performed using the metrics of area, delay and energy. [RFL90b] showed that LUT size 4 achieves the smallest area, and [SRC92] showed that LUT size 5 or 6 leads to the best performance in non-clustered FPGAs. [AR00] evaluated cluster-based island style FPGAs using the metric of area-delay product in  $0.35\mu m$  technology, and showed that the range of LUT sizes from 4 to 6 and cluster sizes between 4 and 10 can produce the best area-delay product. The following work further extended FPGA architecture evaluation considering energy. [PYW02] showed that LUT size 3 consumes the smallest energy in  $0.35\mu m$  technology. [LH05] showed that LUT size 4 consumes the smallest energy and LUT size 7 leads to the best performance in  $100nm$  technology. The device setting such as supply voltage (Vdd) and threshold voltage (Vt) have great impact on power (especially leakage power) and delay in nanometer technologies. [LLH04b] applied high-Vt to the configuration SRAM cells to reduce leakage without degrading system performance.

However all the aforementioned architecture evaluation assumed fixed  $V_{dd}$ ,  $V_t$ , and sleep transistor size, and have not conducted simultaneous evaluation on device optimization such as  $V_{dd}$  and  $V_t$  tuning and architecture optimization on LUT and cluster size tuning. Device setting and FPGA architecture have a significant impact on performance, area, and power. Architecture and device co-optimization may obtain better power and performance tradeoff compared to architecture tuning alone.

Moreover, modern VLSI designs see a large impact from process variation as devices scale down to nanometer technologies. Variability in effective channel length, threshold voltage, and gate oxide thickness incurs uncertainties in both chip performance and power consumption. For example, measured variation in chip-level leakage can be as high as 20X compared to the nominal value for high performance microprocessors [BKN03]. In addition to meeting the performance constraint under timing variation, dies with excessively large leakage due to such a high variation have to be rejected to meet the given power budget. There have been a few studies on parametric yield estimation considering both timing [Nas01, GND01] and leakage [RDB04, ZWB04] variations in ASICs. However, the parametric yield study for FPGAs is largely unexplored in the literature.

In this proposal, we will first perform the device and architecture co-optimization to reduce FPGA power and delay, and then we extend the model to evaluate power and delay considering process variation.

The rest of the proposal is organized as follows: Chapter 2 introduce the background for the FPGA circuit, Chapter 3 presents the device and architecture co-optimization to minimize FPGA power, Chapter 4 presents the optimization considering process variation, finally, Chapter 5 discusses the future work.

## CHAPTER 2

# Preliminaries and Background for FPGA Architecture

### 2.1 Conventional FPGA Architecture

The cluster-based island style FPGA architecture such as that in [BRM99, LCH03] is shown in Figure 2.1, which includes  $N$  fully connected Basic Logic Elements (BLEs). Each BLE includes one  $K$ -input lookup table (LUT) and one flip-flop (DFF). The routing structure is of the island style shown in Figure 2.2. The logic blocks are surrounded by routing channels consisting of wire segments. The input and output pins of a logic block can be connected to the wire segments in routing channels via a *connection block* (see Figure 2.2 (b) ). A routing *switch block* is located at the intersection of a horizontal channel and a vertical channel. Figure 2.2 (c) shows a subset switch block [LB93], where the incoming track can be connected to the outgoing tracks with the same track number. The connections in a switch block (represented by the dashed lines in Figure 2.2 (c)) are programmable routing switches. The routing switches is implemented by tri-state buffers or pass transistors. Two tri-state buffers are used for each connection so that it can be programmed independently for either direction.

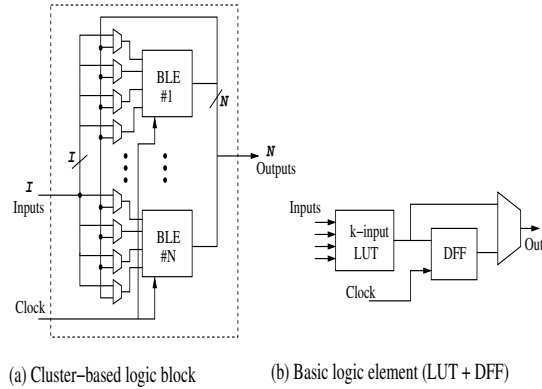


Figure 2.1: FPGA logic block and basic logic element.

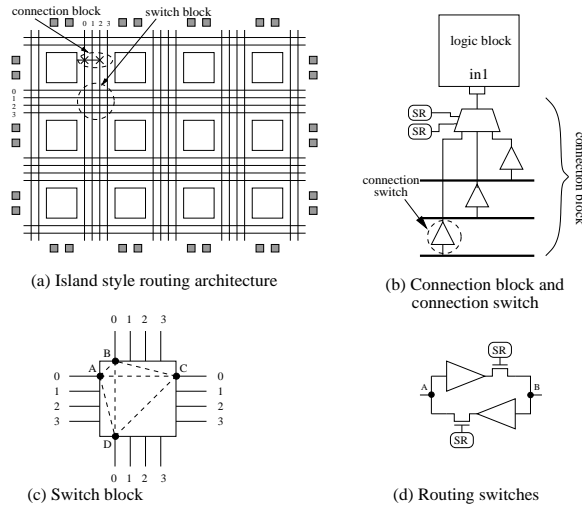


Figure 2.2: (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches.

## 2.2 Vdd gatable FPGA architecture

Power gating can be applied to interconnects and logic blocks to reduce FPGA power. Figure 2.3 illustrates the circuit design of Vdd-gateable interconnect switch and logic block from [Y 05]. A PMOS transistor (called sleep transistor) is inserted between the power rail and the buffer (or logic block) to provide the

power-gating capability. When a buffer or logic block is not used, sleep transistor is turned off by the configuration cell. SPICE simulation shows that power-gating can reduce the leakage power of an unused buffer or logic block by a factor of over 300.

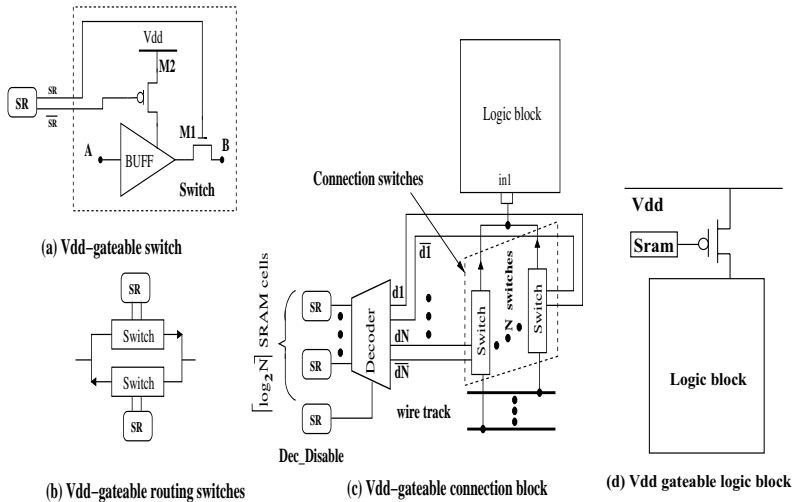


Figure 2.3: (a) Vdd-gateable switch; (b) Vdd-gateable routing switch; (c) Vdd-gateable connection block; (d) Vdd-gateable logic block.

There are power and delay overhead associated with the sleep transistor insertion. The dynamic power overhead is almost negligible. This is because the sleep transistor stay either ON or OFF after configuration and there is no charging and discharging at their source/drain capacitors. The delay overhead associated with the sleep transistor insertion can be bounded when the sleep transistor is properly sized. Moreover, when power gating is applied, the input MUX for the connection box is no longer needed, as shown in Figure 2.3(c). This is because one can select one wire segment to connect to the logic block by turning one of the connection buffer on and power gating all the others. Since we remove the input MUX, the delay of the connection box can be significantly reduced. Therefore, applying power gating may improve the performance in some cases.

## CHAPTER 3

# Device and Architecture Co-Optimization

### 3.1 Trace-based Estimation

#### 3.1.1 Trace collection

Cycle accurate power estimation (in short *Psim*) is very time consuming because a large number of the input vectors needs to be simulated using detailed delay model. Also, in order to obtain FPGA delay, static timing analysis has to be conducted for the entire circuit mapped to the FPGA fabric. The cycle-accurate simulation is not practical for architecture and device co-optimization because the total hyper-arch combinations can be easily over a few hundreds. We develop a runtime efficient trace-based estimation method. Figure 3.1 illustrates the relation between the cycle-accurate simulation and trace-based estimation. For a given benchmark set and a given FPGA architecture, we collect statistical information of switching activity, critical path structure and circuit element utilization rate by profiling the benchmark circuits using *Psim*. These statistical information is called the *trace* of the given benchmark set. Table 3.1 summarizes the trace information we collect as well as the device and circuit parameters. In the table, trace parameters, including  $N_i^u$ ,  $N_i^t$ ,  $S_i^u$ ,  $N_i^p$  and  $\alpha_{sc}$ , are what only depend on FPGA architecture, device parameters, including Vdd and Vt, are what depend on technology scale, and circuit parameters, including  $P_i^s$ ,  $C_i^u$  and

$D_i$ , are what depend on circuit design and device. In the following of this section, we will show that the trace information is insensitive to the device parameters and discussed our trace-based models.

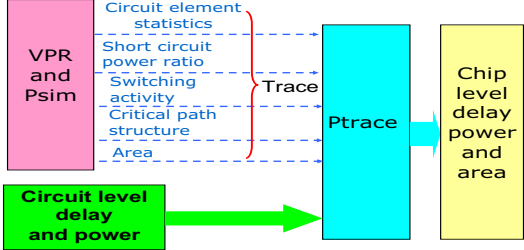


Figure 3.1: Cycle-accurate simulation versus trace-based estimation.

Trace Parameters (depend on architecture)	
$N_i^u$	# of <i>used</i> circuit elements of resource type $i$
$N_i^t$	total # of circuit elements in resource type $i$
$S_i^u$	avg. switching activity for a <i>used</i> ckt element of type $i$
$N_i^p$	# of circuit elements of type $i$ on the critical path
$\alpha_{sc}$	ratio between short circuit power and switching power
Device Parameters (depend on technology)	
$V_{dd}$	power supply voltage
$V_t$	threshold voltage
Circuit Parameters (depend on circuit design and device)	
$P_i^s$	avg. leakage power for a circuit element in resource type $i$
$C_i^u$	avg. load capacitance of a circuit element of resource type $i$
$D_i$	avg. delay of a circuit element in resource type $i$

Table 3.1: Trace information, device and circuit parameters.

### 3.1.2 Dynamic Power Model

Dynamic power includes switching power and short-circuit power. A circuit implemented on an FPGA fabric cannot utilize all circuit elements in FPGA because of the programmability. Dynamic power is only consumed by the utilized FPGA resources. Our trace-based switching power model distinguishes different types

of used FPGA resources and applies the following formula:

$$P_{sw} = \sum_i \frac{1}{2} N_i^u \cdot f \cdot V_{dd}^2 \cdot C_i^{sw} \quad (3.1)$$

The summation is over different types of circuit elements, i.e., LUTs, buffers, input pins and output pins. For circuit elements in FPGA resource type  $i$ ,  $C_i^{sw}$  is the average switching capacitance and  $N_i^u$  is the number of used circuit elements,  $f$  is the operating frequency. In this paper, we assume that the circuit works in its maximum frequency, i.e., the reciprocal of the critical path delay. The switching capacitance is further calculated as:

$$\begin{aligned} C_i^{sw} &= \left( \sum_{j \in El_i} C_{i,j} / N_i^u \right) \cdot S_i^u \\ &= C_i^u \cdot S_i^u \end{aligned} \quad (3.2)$$

For the type  $i$  circuit elements,  $C_i^u$  is the average load capacitance of used circuit elements, which is average over  $C_{i,j}$ , the local load capacitance for used circuit element  $j$ ,  $El_i$  is the set of used type  $i$  circuit elements, and  $S_i^u$  is the average switching activity of used type  $i$  circuit elements. We assume that the average switching activity of the circuit elements is determined by the circuit logic functionality and FPGA architecture. The device parameters of Vdd and Vt have a limited effect on switching activity. We verify this assumption in Table 3.2 by demonstrating the average switching activity of five benchmarks at different technology scale, Vdd, and Vt levels.

The short circuit power is related to signal transition time, which is difficult to obtain without detailed simulation under real delay model. In our trace-based model, we model the short circuit power as:

$$P_{sc} = P_{sw} \cdot \alpha_{sc} \quad (3.3)$$

Where  $\alpha_{sc}$  is the ratio between short circuit power and switching power. Such ratio value is a circuit parameter depending on FPGA circuit design and archi-

benchmark	70nm Vdd=1.1 Vt=0.25		100nm Vdd=1.3 Vt=0.32		70nm Vdd=1.0 Vt=0.20	
	logic	interconnect	logic	interconnect	logic	interconnect
alu4	2.06	0.55	2.01	0.54	2.03	0.59
apex2	1.73	0.47	1.75	0.47	1.70	0.47
apex4	1.23	0.27	1.19	0.26	1.16	0.29
bigkey	1.75	0.56	1.96	0.59	1.71	0.55
clma	0.90	0.21	0.87	0.21	0.91	0.23

Table 3.2: Switching activity comparison for different technology scale, Vdd and Vt. Architecture setting:  $N = 10, K = 4$ . Unit: switch per clock cycle.

texture. We assume that  $\alpha_{sc}$  does not depend on device and technology scale. We verify this assumption in Table 3.3 by showing the average short circuit power ratio at different technology scale, Vdd, and Vt levels.

benchmark	70nm Vdd=1.1 Vt=0.25		100nm Vdd=1.3 Vt=0.32		70nm Vdd=1.0 Vt=0.20	
	logic	interconnect	logic	interconnect	logic	interconnect
alu4	1.43	1.12	1.44	1.16	1.46	1.15
apex2	1.44	0.89	1.42	0.93	1.48	0.92
apex4	1.08	0.86	1.15	0.79	1.18	0.82
bigkey	0.74	1.64	0.76	1.71	0.72	1.68
clma	1.11	1.72	1.21	1.62	1.16	1.63

Table 3.3: Short circuit power ratio comparison for different technology scale, Vdd and Vt. Architecture setting:  $N = 10, K = 4$ .

For a given FPGA architecture (i.e.,  $N$  and  $K$ ), we profile each MCNC benchmark circuit to get the average switching activity for each resource type in the FPGA. The trace parameters  $\alpha_{sc}$ ,  $N_i^u$  and  $C_i^u$  depend only on the FPGA architecture and application benchmark set.

### 3.1.3 Leakage Power Model

The leakage power is modeled as follows,

$$P_{static} = \sum_i N_i^t P_i^s \quad (3.4)$$

For resource type  $i$ ,  $N_i^t$  is the total number of circuit elements, and  $P_i^s$  is the leakage power for a type  $i$  element. Notice that usually  $N_i^t > N_i^u$  because the resource utilization rate is low in FPGAs. For an FPGA architecture with power-gating capability, an unused circuit element can be power-gated to save leakage power. In that case, the total leakage power is modeled by the following formula:

$$P_{static} = \sum_i N_i^u P_i + \alpha_{gating} \cdot \sum_i (N_i^t - N_i^u) P_i \quad (3.5)$$

where  $\alpha_{gating}$  is the average leakage ratio between a power-gated circuit element and a circuit element in normal operation. SPICE simulation shows that sleep transistors can reduce leakage power by a factor of 300 and  $\alpha_{gating} = 0.003$  is used in this paper.

### 3.1.4 Delay Model

To avoid the static timing analysis for the whole circuit implemented on a given FPGA fabric, we obtain the structure of the ten longest circuit paths including the critical path for each circuit. The path structure is the number of elements of different resource types, i.e., LUT, wire segment and interconnect switch, on one circuit path. We assume that the new critical path due to different Vdd and Vt levels is among these ten longest paths found by our benchmark profiling. When Vdd and Vt change, we can calculate delay values for the ten longest paths under new Vdd and Vt levels, and choose the largest one as the new critical path delay. Therefore, the FPGA delay can be calculated as follows:

$$D = \sum_i N_i^p D_i \quad (3.6)$$

For resource type  $i$ ,  $N_i^p$  is the number of circuit elements that the critical path goes through, and  $D_i$  is the average delay of such a circuit element.  $D_i$  is the circuit parameter depending on Vdd, Vt, process technology, and FPGA architecture.

To get the path statistical information  $N_i^P$ , we only need to place and route the circuit *once* for a given FPGA architecture.

### 3.1.5 Validation of Ptrace

To validate Ptrace, we compare it to Psim for ITRS [Int02] 70nm technologies. We assume  $V_{dd}=1.0$  and  $V_t=0.2$ . We map 20 MCNC benchmarks, as illustrates in Table 3.4 to 2 architecture,  $N=8, K=4$ , and  $N=6, K=7$ . We collect trace using Psim in 70nm technology,  $V_{dd}=1.0$  and  $V_t=0.2$ . Figure 3.2 compares energy and delay between Psim and Ptrace for each benchmark. The average energy error of Ptrace is 1.3% and average delay error is 0.8%. From the figure, the Ptrace has the same trend of energy and delay as Psim, the delay error is within 4% and the power error is within 5%. Therefore, Ptrace has a high fidelity. Moreover the run time of Ptrace is 2s, while that of Psim is 120 hours.

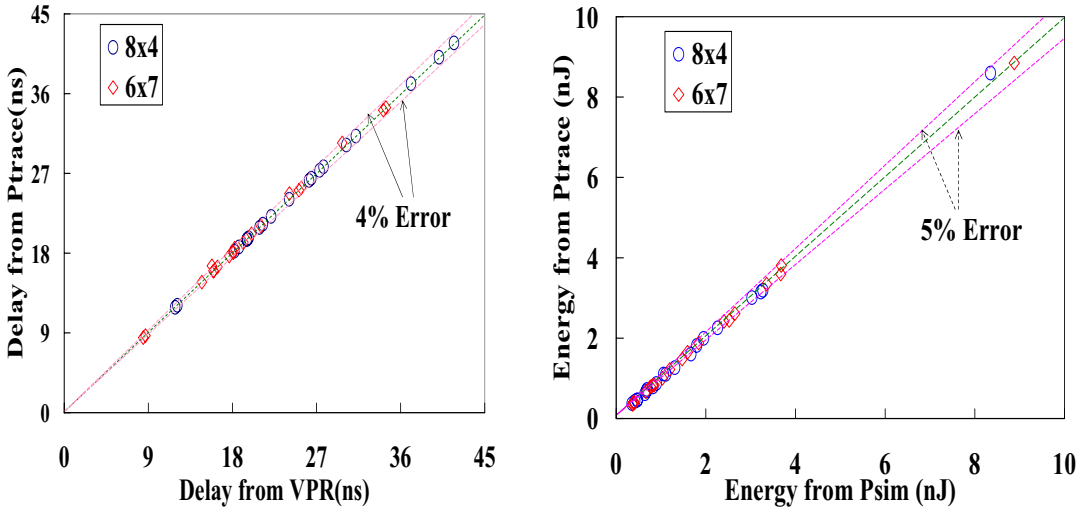


Figure 3.2: Comparison between Psim and Ptrace

alu4	des	ex5p	s38417
apex2	diffeq	frisc	s38584.1
apex4	dsip	misex3	seq
bigkey	elliptic	pdcc	spla
clma	ex1010	s298	tseng

Table 3.4: MCNC benchmark list

## 3.2 Hyper-arch Evaluation

### 3.2.1 Overview

In this section, we will use Ptrace to perform device and architecture evaluation. We will consider 70nm ITRS technology and will evaluate four FPGA hyper-arch classes: *Homo-Vt*, *Hetero-Vt*, *Homo-Vt+G* and *Hetero-Vt+G*. *Homo-Vt* is the conventional FPGA using homogeneous-Vt for both interconnect and logic block (in short, homogeneous-Vt). *Hetero-Vt* applies different Vt to logic blocks and interconnects (in short heterogeneous-Vt). *Homo-Vt+G* and *Hetero-Vt+G* are the same as *Homo-Vt* and *Hetero-Vt*, respectively, except that unused logic blocks and interconnects are power-gated [Y 05]. All these hyper-arch classes are summarized in Table 3.6. We compare them with the baseline hyper-arch, which has the same LUT size and cluster size as those used by Xilinx Virtex II [Xil02] (cluster size of 8, LUT size of 4), Vdd suggested by ITRS [Int02] (0.9v), and Vt of 0.3v that is optimized with respect to the above architecture and Vdd. The base line hyper-arch and evaluation ranges for device and architecture are shown in Table 3.5. Note that a high Vt is applied to all SRAM cells for configuration to reduce their leakage power as suggested by [LLH04b]. In the rest of this section, we assume that all benchmark circuits work in their highest frequency (1/critical path delay) and for each hyper-arch, we compute the energy, delay and area as the geometric mean of 20 MCNC benchmarks.

In the rest of this section we assume the utilization rate (utilization rate is defined as the utilization rate of logic blocks, i.e., number of used logic blocks over total available logic blocks) to be 0.5. Moreover, in the rest of this paper, we will use the symbol  $CV_t$  to represent the  $V_t$  of logic blocks and  $IV_t$  to represent the  $V_t$  for interconnects, notice that  $CV_t=IV_t$  in *Homo-Vt* and *Homo-Vt+G*. To illustrate the tradeoff between energy and delay, we introduce the concept *dominant hyper-arch* (in short, dom-arch): If hyper-arch  $A$  has less energy consumption and a smaller delay than hyper-arch  $B$ , then we say that  $B$  is inferior to  $A$ . We define the *dominant hyper-arch* as the set of hyper-archs that are not inferior to any other hyper-archs.

We organize the rest of this section as follows: First Section 3.2.2 illustrates the necessity of device and architecture co-optimization. Then, Section 3.2.3 presents the energy and delay tradeoff and ED reduction achieved by device and architecture co-optimization. Finally, Section 3.2.4 discusses the device and architecture co-optimization considering area.

Baseline FPGA device/arch parameter values			
Vdd	Vt	N	K
0.9v	0.3v	8	4
Value range for device/arch optimization			
Vdd	Vt	N	K
0.8v-1.1v	0.2v-0.4v	6-12	3-7

Table 3.5: Baseline hyper-arch and evaluation ranges.

hyper-arch Class	Case to study
Homo-Vt	homogeneous-Vt w/o power-gating
Hetero-Vt	heterogeneous-Vt w/o power-gating
Homo-Vt+G	homogeneous-Vt w/ power-gating
Hetero-Vt+G	heterogeneous-Vt w/ power-gating

Table 3.6: Summary of FPGA hyper-arch Classes.

### 3.2.2 Necessity of device and architecture co-optimization

In this section, we will show the necessity of device and architecture co-optimization. We will first present the power difference between hyper-archs with similar delay, and then compare the results of optimizing device and architecture separately and optimizing device and architecture simultaneously.

Figure 3.3 (a) and (b) plot the energy versus delay for *Homo-Vt* within delay range from 9.5ns to 10.5ns and delay range from 19.5ns to 20.5ns, respectively. From the figures, we find that the maximum energy hyper-archs consume more than 4 times energy than the minimum energy hyper-archs. Power between different hyper-archs within similar performance differs significantly. For other classes, we will have similar energy difference between maximum and minimum energy hyper-archs within a performance range. Therefore, it is necessary to perform hyper-arch optimization.

There are three methods to perform device and architecture optimization. The first method is that we first optimize the architecture within one device setting and then optimize the device setting according to such architecture, we call such method *device-arch*. The second method is that we first optimize device using one architecture and then optimize the architecture under such device setting, we call such method *arch-device*. The third method is that we optimize architecture and device simultaneously, we call such method *full-search*. For the methods *arch-device* and *device-arch*, we do not need to test too many hyper-archs, but we cannot guarantee that we find the optimum solution. However, for the method *full-search*, we may need to test all hyper-arch combinations, but we can get the optimum solution. Table 3.7 compares the min-ED hyper-archs for *Homo-Vt* found by three different methods. For the method *device-arch* we first optimize architecture under device setting  $V_{dd}=0.9$ ,  $V_t=0.3$ , which is suggested by ITRS,

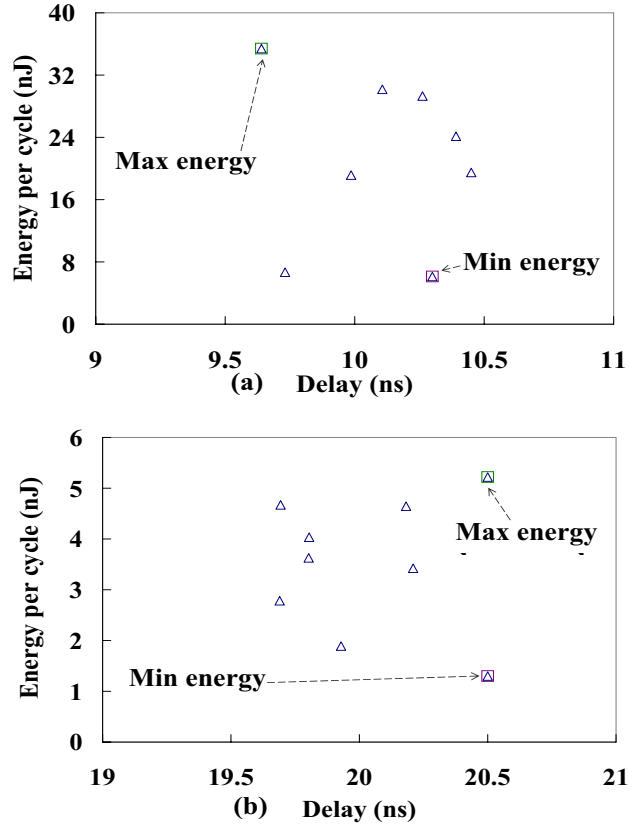


Figure 3.3: Hyper-archs within fix delay range. (a) Delay ranges from 9.5ns to 10.5ns. (b) Delay ranges from 19.5ns to 20.5ns.

and then choose the best device setting for the optimized architecture. And for the method *arch-device*, we first optimize hyper-archs for architecture:  $N=8$ ,  $K=4$ , which is the same as commercial FPGA used by Xilinx Virtex II [Xil02], and then choose the best architecture for the optimized device. From the table, we see that *arch-device* and *device-arch* will give the same optimization result, however, *full-search* can reduced ED by 13.3% compared to *arch-device* and *device-arch*. Although *full-search* requires a large amount of experiments, considering the time efficiency of Ptrace, it is still worthwhile doing so.

	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	
arch-device	0.9	0.30	0.30	(6,7)	1.38	19.8	27.3	
device-arch	0.9	0.30	0.30	(6,7)	1.38	19.8	27.3	
full-search	1.0	0.30	0.30	(10,4)	1.37	17.5	24.1 (-13.3%)	

Table 3.7: Min-ED hyper arch of optimizing device and architecture separately and optimizing device and architecture simultaneously.

### 3.2.3 Energy and delay tradeoff

In this section, we will first compare the impact of device tuning and architecture tuning, and then we will present min-energy and min-delay hyper-archs, finally we will also discuss the energy and delay tradeoff. In this section, for the classes with power gating *Homo-Vt+G* and *Hetero-Vt+G*, we assume the following fixed sleep transistor size: 210X PMOS for a logic block, 10X PMOS for a switch buffer, and 1X PMOS for a connection buffer. We will discuss the sleep transistor tuning in Section 3.2.4.

Architecture evaluation has been studied by previous research [RFL90a, KG92, AR00, LCH03, PYW02, Y 05]. However, device tuning has not been studied yet. Our experiments show that device tuning has a much greater impact on delay and energy than architecture tuning does, which is demonstrated in Figure 3.4 and Table 3.8. Each set of data points in Figure 3.4 is the dom-archs for a given device setting. (dom-archs for a given device setting are the architecture that are no inferior to any other architecture under such device setting ) For example, set D4 is the dom-archs under Vdd=1.0 and Vt=0.25. From the figure, we observe that a change on the device level leads to a more significant change in energy and delay than architecture change does. Within a certain device setting, architecture tuning can only change delay and energy in a very small range. For example, for device setting Vdd=0.9v and Vt=0.25v, energy for different architectures changes

from 1.82nJ to 2.11nJ, and delay changes from 13.68ns to 16.46ns. However, if we increase  $V_t$  by 0.05v, i.e.,  $V_{dd}=0.9v$  and  $V_t=0.3v$ , the energy ranges from 1.17nJ to 1.32nJ and the delay ranges from 18.99ns to 23.24ns. Therefore, it is important evaluating both device and architecture instead of evaluating architecture only.

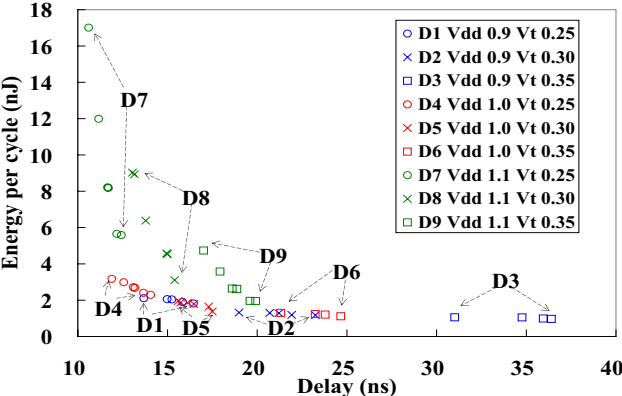


Figure 3.4: Hyper-archs under different device setting.

Set	Vdd (V)	Vt (V)	Min energy (nJ)	Max energy (nJ)	Min delay (ns)	Max delay (ns)
D1	0.9	0.25	1.82	2.11	13.68	16.46
D2	0.9	0.30	1.17	1.32	18.99	23.24
D3	0.9	0.35	0.97	1.05	31.01	36.40
D4	1.0	0.25	2.30	3.17	11.90	14.06
D5	1.0	0.30	1.37	1.95	15.60	17.50
D6	1.0	0.35	1.11	1.30	21.31	24.66
D7	1.1	0.25	5.58	17.01	10.60	12.43
D8	1.1	0.30	3.10	9.03	13.05	15.40
D9	1.1	0.35	1.95	4.73	17.01	19.92

Table 3.8: Power and delay ranges for different device settings.

Table 3.9 summarizes the minimum delay and minimum energy hyper-archs for each class. The minimum delay hyper-archs have cluster size 6 and LUT size 7 which are the same for all classes. The minimum energy hyper-archs have

LUT size 4 for all classes. This is very similar to the previous evaluation result [Y 05, LCH03]. The min-delay hyper archs have the highest Vdd and lowest Vt. However, the min-energy hyper-archs have the lowest Vdd but not highest Vt. This is because we assume that each circuit works in its highest possible frequency (1/critical path delay). The energy is calculated as  $E = delay \cdot power$ . When Vt is too high, the delay will be so large that the energy per clock cycle will increase.

Hyper-arch Class	Minimum delay hyper-arch						Minimum energy hyper-arch					
	Vdd (V)	CVt (V)	Ivt (V)	(N,K)	Energy (nJ)	Delay (ns)	Vdd (V)	CVt (V)	Ivt (V)	(N,K)	Energy (nJ)	Delay (ns)
Hemo-Vt	1.1	0.20	0.20	(6,7)	31.22	8.86	0.8	0.35	0.35	(10,4)	0.942	59.2
Hetero-Vt	1.1	0.20	0.20	(6,7)	31.22	8.86	0.8	0.30	0.35	(12,4)	0.920	43.6
Hemo-Vt+G	1.1	0.20	0.20	(6,7)	15.98	9.45	0.8	0.30	0.30	(12,4)	0.550	30.5
Hetero-Vt+G	1.1	0.20	0.20	(6,7)	15.98	9.45	0.8	0.30	0.25	(10,4)	0.549	24.3

Table 3.9: Minimum delay and minimum energy hyper-archs.

Usually, higher performance hyper-archs will consume more energy. To illustrate the energy and delay tradeoff, we present the dom-archs of *Homo-Vt* and *Hetero-Vt* in Figure 3.5 (a) and those of *Homo-Vt+G* and *Hetero-Vt+G* in Figure 3.5 (b). From the figure, we find that the energy difference between *Homo-Vt+G* and *Hetero-Vt+G* is smaller than the energy difference between *Homo-Vt* and *Hetero-Vt*. This is because leakage power is significantly reduced by field programmable power-gating and therefore the more detailed Vt tuning such as heterogeneous-Vt has a smaller impact for *Homo-Vt+G* and *Hetero-Vt+G*. Moreover, with the dom-arch figure, we can obtain the minimum energy solution for a given performance range. For example, if we want to find the minimum energy solution for *Homo-Vt* with delay limit 15ns, we only need to pick the dom-arch whose delay is closest to 15ns.

In order to achieve the best energy and delay tradeoff, we find out the hyper-

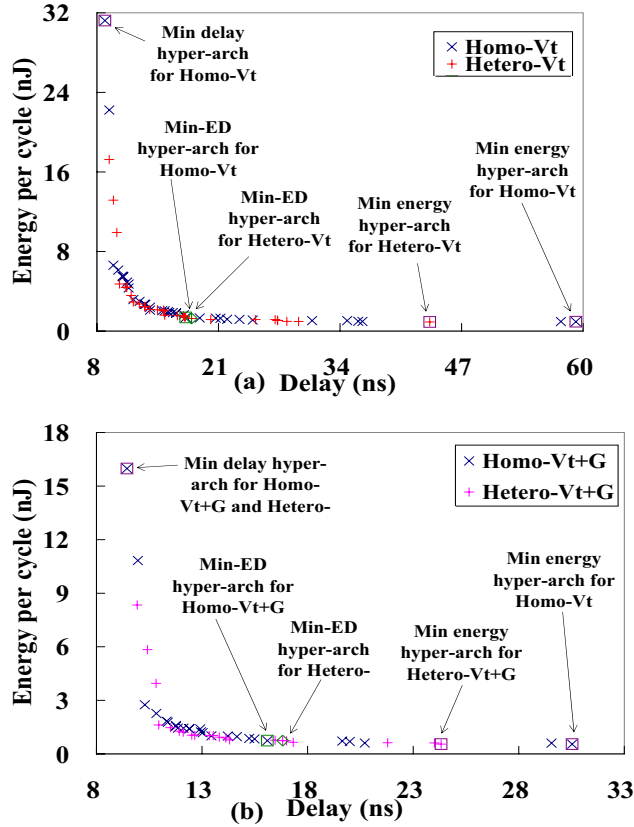


Figure 3.5: Dom-archs. (a) *Homo-Vt* and *Hetero-Vt* (b) *Homo-Vt+G* and *Hetero-Vt+G*

archs with the *minimum energy delay product* (in short *min-ED*). Table 3.10 summarizes the min-ED hyper-archs for each class. From the table, we see that compared to the baseline case, device and architecture tuning can significantly reduce ED. Compared to the baseline, ED reduction is 14.54% and 18.44% for *Homo-Vt* and *Hetero-Vt*, respectively. If power gating is applied, ED can be reduced by about 60% for both *Homo-Vt+G* and *Hetero-Vt+G*. The similar ED reduction for power gating classes is because leakage power is greatly reduced by power-gating and therefore the more detailed Vt tuning such as heterogeneous-Vt has a smaller impact on power reduction as discussed before. We also see

that compared to the min-ED hyper-archs of the classes without power gating, the min-ED hyper-archs for the classes with power gating has lower  $V_t$ . This is because when power gating is applied, leakage power is greatly reduced, therefore lower  $V_t$  can be applied to improve performance.

Hyper-arch Class	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	ED Reduction %	Normalized Area %
Baseline	0.9	0.30	0.30	(8,4)	1.20	23.5	28.2	-	100.00
Hemo-Vt	1.0	0.30	0.30	(10,4)	1.37	17.5	24.1	14.54	81.90
Hetero-Vt	0.9	0.25	0.30	(12,4)	1.27	18.1	23.0	18.44	79.52
Hemo-Vt+G	0.9	0.25	0.25	(12,4)	0.74	16.10	11.9	57.80	127.43
Hetero-Vt+G	0.8	0.25	0.20	(10,4)	0.65	17.30	11.2	60.28	126.19

Table 3.10: Comparison between baseline and min-ED hyper-arch in *Homo-Vt*, *Hetero-Vt*, *Homo-Vt+G*, and *Hetero-Vt+G*.

### 3.2.4 ED and Area Tradeoff

In the previous sections, we assume fix sleep transistor sizes for *Homo-Vt* and *Hetero-Vt+G* and discuss hyper-arch evaluation to minimize ED without considering area. However area is important for FPGA design. Power-gating using sleep transistors may change delay and area tradeoff for FPGA architecture. Usually, the larger the sleep transistor size, the smaller the delay will be. In this section, we will perform device and architecture co-optimization to achieve the best ED and area tradeoff. Although dual- $V_t$  may change layout area due to extra diffusion well area, such change depends on technology and is often very small. Therefore in this section, we assume that Vdd and  $V_t$  change does not affect area.

For *Homo-Vt* and *Hetero-Vt*, because no power gating is applied, we do not need to tune sleep transistor size. To achieve the best ED-area tradeoff, we find out the hyper-archs with minimum *product of energy, delay, and area* (in short

*AED product*), which are summarized in Table 3.11. We see that compared to the baseline, the min-AED hyper-arch of *Homo-Vt* reduces ED by 14.7% and area by 18.3%, and the min-AED hyper-arch of *Hetero-Vt* reduces ED by 18.4% and area by 23.3%. Figure 3.6 presents the chip-level ED and area tradeoff. We prune inferior solutions with both ED and area larger than any alternative solutions. From the figure, we see that for the classes without power gating, the min-ED hyper-arch is exactly the min-AED hyper-arch. This is because when no power gating is applied, the larger the area, the more leakage power will be consumed. Therefore, the min-ED hyper-archs use less area than other hyper-archs.

Hyper-arch Class	Vdd (V)	CVt (V)	IVt (V)	(N,K)	S	ED	Area	AED	AED reduction%
Baseline	0.9	0.30	0.30	(8,4)	-	1	1	-	-
Hemo-Vt	1.0	0.30	0.30	(10,4)	-	0.853	0.817	0.697	30.33
Hetero-Vt	0.9	0.25	0.30	(12,4)	-	0.816	0.767	0.626	37.44
Hemo-Vt+G	0.9	0.25	0.25	(12,4)	2	0.470	0.918	0.432	56.85
Hetero-Vt+G	0.9	0.25	0.20	(12,4)	2	0.450	0.918	0.413	58.70

Table 3.11: Minimum ED-area product hyper-archs for different classes. ED, Area, and ED-area product are normalized with respect to Baseline. In the table, S refers to the sleep transistor size. ED, Area and AED are normalized with respect to baseline.

For *Homo-Vt+G* and *Hetero-Vt+G* with power gating, the sleep transistor size has to be considered. Because only one sleep transistor is used for one logic block, as illustrated in Figure 2.3(d), we assume the 210X PMOS for the sleep transistor with negligible area overhead. Moreover, we observe that a 1X PMOS as the sleep transistor for one switch in connection box provides good performance, any further increase of the sleep transistor size will not improve the performance much. Therefore, we use a 1X PMOS as sleep transistor for one switch in connection box.



## CHAPTER 4

# Power and Delay Optimization with Process Variation

### 4.1 Leakage and Timing Variations

#### 4.1.1 Leakage under Variation

We extend the leakage model in FPGA power and delay estimation framework *Ptrace* [CWL05] to consider variations. In *Ptrace*, the total leakage of an FPGA chip is calculated as follows,

$$I_{chip} = \sum_i N_i^t \cdot I_i \quad (4.1)$$

where  $N_i^t$  is the number of FPGA circuit elements in FPGA resource type  $i$ , i.e., an interconnect switch, buffer, LUT, configuration SRAM cell, or flip-flop, and  $I_i$  is the leakage of an element. Different sizes of interconnect switches and buffers are considered as different circuit elements.

The leakage current  $I_i$  of a circuit element  $i$  is the sum of the sub-threshold and gate leakages:

$$I_i = I_{sub} + I_{gate} \quad (4.2)$$

Variation in  $I_{sub}$  mainly sources from variation in  $L_{eff}$  and  $V_{th}$ . Variation in  $I_{gate}$  mainly sources from variation in  $T_{ox}$ .

Different from [RDB04] that models sub-threshold leakage and gate leakage separately, we model the total leakage current  $I_i$  of circuit element in resource type  $i$  as follows,

$$I_i = I_n(i) \cdot e^{f_i(\Delta L_{eff})} \cdot e^{f_i(\Delta V_{th})} \cdot e^{f_i(\Delta T_{ox})} \quad (4.3)$$

where  $I_n(i)$  is the leakage of a circuit element in resource type  $i$  in the absence of any variability and  $f$  is the function that represents the impact of each type of process variation on leakage. The interdependency between these functions has been shown to be negligible in [RDB04]. From SPICE simulation, we find that it is sufficient to express these functions as simple linear functions. To make the presentation simple, we denote  $\Delta L_{eff}$ ,  $\Delta V_{th}$ , and  $\Delta T_{ox}$  as  $L$ ,  $V$ , and  $T$ , respectively. We can express these functions with this simple notation as follows,

$$f(L) = -c_{i1} \cdot L \quad f(V) = -c_{i2} \cdot V \quad f(T) = -c_{i3} \cdot T \quad (4.4)$$

where  $c_{i1}, c_{i2}, c_{i3}$  are fitting parameters decided by SPICE simulations. The negative sign in the exponent indicates that the transistors with shorter channel length, lower threshold voltage, and smaller oxide thickness lead to higher leakage current. We rewrite (4.3) as follows by decomposing  $L$ ,  $V$  and  $T$  in to local  $(L_l, V_l, T_l)$  and global  $(L_g, V_g, T_g)$  components.

$$I_i = I_n(i) \cdot e^{-(c_{i1}L_g + c_{i2}V_g + c_{i3}T_g)} \cdot e^{-(c_{i1}L_l + c_{i2}V_l + c_{i3}T_l)} \quad (4.5)$$

To extend the leakage model (4.1) under variations, we consider that each element has unique local variations but all elements in one die share the same global variations. Both global and local variations are modeled as normal random variables. The leakage distribution of a circuit element is a lognormal distribution. The total leakage is the sum of all lognormals. The state-of-the-art FPGA chip usually has a large number of circuit elements and therefore the relative random

variance of the total leakage approaches zero. Same as [RDB04], we apply the Central Limit Theorem and use the mean of the distribution to approximate the distribution of the sum of lognormals. After integration, we can write the expression of the chip-level leakage as the follows,

$$\begin{aligned}
I_{chip} &\approx \sum_i N_i^t \cdot E[I_i] \\
&= \sum_i N_i^t S_i I_{L_g, V_g, T_g}(i) \\
S_i &= e^{(c_{i1}\sigma_{L_l}^2 + c_{i2}\sigma_{V_l}^2 + c_{i3}\sigma_{T_l}^2)/2} \\
I_{L_g, V_g, T_g}(i) &= I_n(i) e^{-(c_{i1}L_g + c_{i2}V_g + c_{i3}T_g)}
\end{aligned} \tag{4.6}$$

where  $S_i$  is the scale factor introduced due to local variability in  $L, V$ , and  $T$ .  $I_{L_g, V_g, T_g}(i)$  is the leakage as a function of global variations.  $\sigma_{L_l}$ ,  $\sigma_{V_l}$  and  $\sigma_{T_l}$  are the variances of  $L_l$ ,  $V_l$ , and  $T_l$ , respectively.

For an FPGA architecture with power-gating capability, an unused circuit element can be power-gated to reduce leakage power. In this case, *Ptrace* calculates the total leakage current as follows,

$$I_{chip} = \sum_i N_i^u I_i + \alpha_{gating} \sum_i (N_i^t - N_i^u) I_i \tag{4.7}$$

where  $N_i^u$  is the number of used circuit elements in FPGA resource type  $i$  and  $\alpha_{gating}$  is the average leakage ratio between a power-gated circuit element and a circuit element in normal operation. Same as [CWL05], 1/300 is used for  $\alpha_{gating}$  in this paper. Similar to (4.6), (4.7) can be easily extended to consider variations as follows,

$$I_{chip} \approx \sum_i N_i^u E[I_i] + \alpha_{gating} \sum_i (N_i^t - N_i^u) E[I_i] \tag{4.8}$$

where  $E[I_i]$  is still defined as in (4.6).

### 4.1.2 Timing under Variation

The performance depends on  $L_{eff}$ ,  $V_{th}$ , and  $T_{ox}$ , but its variation is primarily affected by  $L_{eff}$  variation[RDB04]. Below we extend the delay model in *Ptrace* to consider global and local variations of  $L_{eff}$ . The structure of the critical path for each benchmark is obtained for timing analysis. The path delay can be calculated as follows,

$$D = \sum_i d_i(L_g, L_l) \quad (4.9)$$

For circuit element  $i$  in the path,  $d_i(L_g, L_l)$  is the delay considering global variation  $L_g$  and local variation  $L_l$ .  $L_g$  is the same for all the circuit elements in the critical path. Given  $L_g$ , we evenly sample a few (eleven in this paper) points within range of  $[L_g - 3\sigma_{L_l}, L_g + 3\sigma_{L_l}]$ . We then perform SPICE simulation to obtain the delay for each circuit element with these variations. As the delay monotonically decreases when  $L_{eff}$  increases, we can directly map the probability of a channel length to the probability of a delay and obtain the delay distribution of a circuit element. We assume that the local channel length variation of each element is independent from each other. Therefore, we can obtain the distribution of the critical path delay for a given  $L_g$  as follows by convolution operation,

$$PDF(D) = PDF(d_1) \otimes PDF(d_2) \otimes \dots \otimes PDF(d_i) \otimes \dots \otimes PDF(d_n) \quad (4.10)$$

## 4.2 Yield Models

### 4.2.1 Leakage Yield

The leakage yield is calculated on a bin-by-bin basis where each bin corresponds to a specific value  $L_g$ . For a particular bin, the value  $L_g$  is constant. We can

rewrite (4.6) for chip-level leakage current as follows,

$$\begin{aligned} I_{chip} &= \sum_i A_i \cdot e^{-c_{i2}V_g} \cdot e^{-c_{i3}T_g} \\ A_i &= N_i I_n(i) S_i e^{-c_{i1}L_g} \end{aligned} \quad (4.11)$$

where  $A_i$  is the leakage current for all circuit elements of resource type  $i$  at a value of  $L_g$  and includes the scale factor  $S_i$  due to the local variability. Let  $X_i$  be the leakage consumed by the elements of resource type  $i$  and it is a lognormal variable. The chip-level leakage current  $I_{chip}$  is the sum of each lognormal variable  $X_i$  [RDB04] and it can be expressed as follows,

$$\begin{aligned} I_{chip} &= \sum_i X_i \\ X_i &\sim \text{Lognormal}(\log(A_i), ((c_{i2}\sigma_{V_g})^2 + (c_{i3}\sigma_{T_g})^2)) \end{aligned} \quad (4.12)$$

Same as [RDB04], we model  $I_{chip}$ , the sum of the lognormal variables  $X_i$ , as another lognormal random variable. The lognormal variable  $X_i$  shares the same random variables  $\sigma_{V_g}$  and  $\sigma_{T_g}$ , and therefore these variables are dependent of each other. Considering the dependency, we calculate the mean and variance of the new lognormal  $I_{chip}$  as follows,

$$\begin{aligned} \mu_{I_{chip}} &= \sum_i \left\{ \exp\left[\log(A_i) + \frac{(c_{i2}\sigma_{V_g})^2}{2} + \frac{(c_{i3}\sigma_{T_g})^2}{2}\right] \right\} \\ \sigma_{I_{chip}}^2 &= \sum_i \left\{ \exp\left[2\log(A_i) + (c_{i2}\sigma_{V_g})^2 + (c_{i3}\sigma_{T_g})^2\right] \right. \\ &\quad \cdot \left. \left[ \exp(c_{i2}^2\sigma_{V_g}^2 + c_{i3}^2\sigma_{T_g}^2) - 1 \right] \right\} \\ &\quad + \sum_{i,j} 2\text{COV}(X_i, X_j) \end{aligned} \quad (4.13)$$

where the mean of  $I_{chip}$ ,  $\mu_{I_{chip}}$ , is the sum of means of  $X_i$  and the variance of  $I_{chip}$ ,  $\sigma_{I_{chip}}$ , is the sum of variance of  $X_i$  and the covariance of each pair of  $X_i$ . The covariance is calculated as follows,

$$\text{COV}(X_i, X_j) = E[X_i X_j] - E[X_i]E[X_j] \quad (4.15)$$

$$\begin{aligned}
E[X_i X_j] &= \exp[\log(A_i A_j) + \frac{(c_{i2} + c_{j2})^2 \sigma_{V_g}^2}{2} \\
&\quad + \frac{(c_{i3} + c_{j3})^2 \sigma_{T_g}^2}{2}] \\
E[X_i] &= \exp[\log(A_i) + \frac{(c_{i2} \sigma_{V_g})^2}{2} + \frac{(c_{i3} \sigma_{T_g})^2}{2}]
\end{aligned}$$

We then use the method from [RDB04] to obtain the mean and variance  $(\mu_{N,I_{chip}}, \sigma_{N,I_{chip}}^2)$  of the normal random variable corresponding to the lognormal  $I_{chip}$ . As the exponential function that relates the lognormal variable  $I_{chip}$  with the normal variable  $I_{N,chip}$  is a monotone increasing function, the CDF of  $I_{chip}$  can be expressed as follows using the standard expression for the CDF of a lognormal random variable,

$$\begin{aligned}
\mu_{N,I_{chip}} &= \frac{\log[\mu_{I_{chip}}^4 / (\mu_{I_{chip}}^2 + \sigma_{I_{chip}}^2)]}{2} \\
\sigma_{N,I_{chip}}^2 &= \log[1 + (\sigma_{I_{chip}}^2 / \mu_{I_{chip}}^2)] \\
Y_{leak}(I_{chip} | L_g) &= CDF(I_{chip}) \\
&= \frac{1}{2} [1 + \operatorname{erf}(\frac{\log(I_{chip}) - \mu_{N,I_{chip}}}{\sqrt{2} \sigma_{N,I_{chip}}})] \\
Y_{leak}(I_{chip}) &= \int_{-\infty}^{+\infty} Y_{leak}(I_{chip} | L_g) \cdot PDF(L_g) \cdot dL_g \tag{4.16}
\end{aligned}$$

where  $\operatorname{erf}()$  is the error function. Given a leakage limit  $I_{cut}$  for  $I_{chip}$ ,  $[CDF(I_{cut}) \times 100\%]$  gives the leakage yield rate  $Y_{leak}(I_{cut} | L_g)$ , i.e., the percentage of FPGA chips that is smaller than  $I_{cut}$  in a particular  $L_g$  bin. Similarly, the yield for the FPGA chip with power-gating capability can be easily calculated using (4.8).

### 4.2.2 Timing Yield

The timing yield is again calculated on a bin-by-bin basis where each bin corresponds to a specific value  $L_g$ . We further consider local variation of channel length in timing yield analysis. Given the global channel length variation  $L_g$ , (4.10) gives the PDF of the critical path delay  $D$  of the circuit. We can obtain

the CDF of delay,  $CDF(D|L_g)$ , by integrating for a given  $L_g$ . Given a cutoff delay ( $D_{cut}$ ) and  $L_g$ ,  $CDF(D_{cut}|L_g)$  gives the probability that the path delay is smaller than  $D_{cut}$  considering  $L_{eff}$  variations. However, it is not sufficient to only analyze the original critical path in the absence of process variations. The close-to-be critical paths may become critical considering variations and an FPGA chip that meets the performance requirement should have the delay of all paths no greater than  $D_{cut}$ .

We assume that the delay of each path is independent and we can calculate the timing yield for a given  $L_g$  as follows,

$$Y_{perf}(D_{cut}|L_g) = \prod_{i=1}^n CDF_i(D_{cut}|L_g) \quad (4.17)$$

where  $CDF_i(D_{cut}|L_g)$  gives the probability that the delay of the  $i^{th}$  longest path is no greater than  $D_{cut}$ . In this paper, we only consider the ten longest paths, i.e.,  $n = 10$  because the simulation result shows that the ten longest paths have already covered all the paths with a delay larger than 75% of the critical path delay under the nominal condition. We then integrate  $Y_{perf}(D_{cut}|L_g)$  to calculate the performance yield  $Y_{perf}$  as follows,

$$Y_{perf} = \int_{-\infty}^{+\infty} PDF(L_g) \cdot Y_{perf}(D_{cut}|L_g) \cdot dL_g \quad (4.18)$$

### 4.2.3 Leakage and Timing Combined Yield

To analyze the yield of a lot, we need to consider both leakage and delay limit. Given a specific global variation of channel length  $L_g$ , the leakage variability only depends on the variability of random variable  $V_g$  and  $T_g$  as shown in (4.6), and the timing variability only depends on the variability of random variable  $L_l$ . Therefore, we assume that the leakage yield and timing yield are independent of each other. The yield considering the imposed leakage and timing limit can be

calculated as follows,

$$Y_{com} = \int_{-\infty}^{+\infty} PDF(L_g)Y_{leak}(I_{cut}|L_g)Y_{perf}(D_{cut}|L_g) \cdot dL_g \quad (4.19)$$

## 4.3 Leakage and Timing Yield

### Analysis

For the total power and leakage power we report the arithmetic mean of 20 MCNC benchmarks within and among three FPGA architecture classes: *Homo- $V_t$*  is the conventional FPGA using the same and optimized  $V_t$  for both logic blocks and interconnect; *Hetero- $V_t$*  optimizes  $V_t$  separately for logic blocks and interconnect; and *Homo- $V_t+G$*  is the same as *Homo- $V_t$*  except that unused logic blocks and interconnect are power-gated as studied in[LLH05a]. We assume 10% of the nominal value as  $3\sigma$  for all the process variations.

#### 4.3.1 Leakage Yield

Figure 4.1 shows the full chip leakage power simulated by Monte Carlo simulation and  $\sigma$ , in the presence of inter-die and intra-die variations. Leakage may change significantly due to process variations. When there is a  $\pm 3\sigma$  inter-die variation of  $L_{eff}$ , the leakage power has a  $3X$  span. When no  $L_g$  variation is present, there is still a  $2X$  span in leakage power due to  $V_{th}$  and  $T_{ox}$  variation. Therefore it is important to consider the impact of process variations on leakage when determining the yield.

We further validate our chip-level analytical model for leakage by Monte Carlo simulation to estimate the full chip leakage power in Table 4.1, where global variations are all set to  $\pm 3\sigma$ , and local variations are set to 0,  $\pm 1\sigma$ , and  $\pm 2\sigma$ . The mean calculated from our analytical method has a less than 3% difference

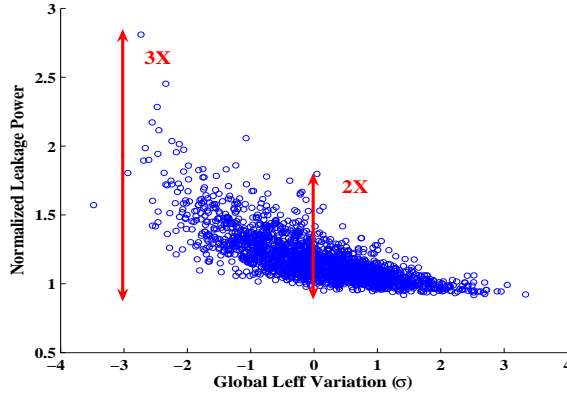


Figure 4.1: Leakage power of baseline architecture ( $N=8, K=4$ ) with ITRS device setting under intra-die and inter-die variations.

from the simulation and the standard deviations differed by 1% of the mean value. In the rest of the paper, we always report the standard deviation as a relative value with respect to the mean and use our analytical model to calculate the yield.

Variations( $\sigma$ )			Mean(W)		SD(%)	
$(L_g, L_l)$	$(V_g, V_l)$	$(T_g, T_l)$	<i>M-C</i>	<i>Model</i>	<i>M-C</i>	<i>Model</i>
$(\pm 3, 0)$	$(\pm 3, 0)$	$(\pm 3, 0)$	1.24	1.20	14	13
$(\pm 3, \pm 1)$	$(\pm 3, \pm 1)$	$(\pm 3, \pm 1)$	1.41	1.37	14	13
$(\pm 3, \pm 2)$	$(\pm 3, \pm 2)$	$(\pm 3, \pm 2)$	2.07	2.00	13	12

Table 4.1: Comparison between analytical variation *models* and Monte Carlo (*M-C*)simulation.

#### 4.3.1.1 Impact of Architecture and Device Tuning

In this section we consider combinations of device and architecture parameters, called as *hyper-architecture* (in short, *hyper-arch*). Table 4.2 shows the yield, mean leakage, and standard deviation from two different device settings, sorted by the yield. Columns 1-4 use ITRS device setting. Our baseline FPGA has

$N = 8$  and  $K = 4$ , which is the architecture used by Xilinx Virtex-II Pro. Yield is calculated using the nominal leakage of each architecture plus an offset of 30% of the nominal leakage of baseline architecture,  $P_{base}^L$ , as the leakage limit. As shown in column 1 of Table 4.2, the yield ranges from 24% to 70%, which shows that architecture tuning has a significant impact on the yield. Among all architectures,  $N = 6$  and  $K = 5$  gives the maximum yield, which is 12% higher than the baseline. The yield is affected by both the mean and variance. When the mean leakage is close to the leakage limit, the variance gains importance in determining the yield. However, when the mean is not close to the limit, the variance does not have that much impact on the yield. In this case, the lower the mean leakage is, the higher the yield is (see columns 5 – 8). It is also noticeable that larger LUT sizes have larger mean leakage, thus yield becomes smaller.

Device tuning also affects the yield. In Columns 5 – 8 of Table 4.2, we use a device setting that provides the minimum energy-delay product (minimum product of energy per clock cycle and critical path delay, in short, min-ED) given in [CWL05]. Column 5 shows that optimizing Vdd and  $V_t$  can increase the yield rate of each architecture by an average of 39%. Therefore, device tuning has a great impact on yield rate and it is important to evaluate different Vdd and  $V_t$  levels while considering process variations. Comparing the yield of architecture (12, 7) in ITRS device setting and architecture (6, 4) in Min-ED device setting shows that combining device tuning with architecture tuning can increase the yield by up to 73%. From the Table, architectures with  $K=4$  generally provides the highest yield rate, and they have the minimum area as reported in previous work such as [CWL05]. In the rest of the paper, we will only consider *dominant architectures*. Dominant architectures are defined as the group of architectures that either has smaller delay or less energy consumption than others [CWL05]. Fig 4.2 presents the energy and delay tradeoff between dominant architectures

1	2	3	4	5	6	7	8
ITRS Vdd0.80V/ $V_t$ 0.20V				Min ED Vdd0.90V/ $V_t$ 0.30V			
Y (%)	Mean (W)	SD (%)	(N,K)	Y (%)	Mean (W)	SD (%)	(N,K)
70	0.40	39	(6, 5)	97	0.07	48	(6, 4)
68	0.50	40	(8, 3)	97	0.08	48	(8, 4)
64	0.58	39	(10, 3)	96	0.08	48	(10, 4)
61	0.55	38	(12, 3)	96	0.08	49	(6, 5)
60	0.43	64	(6, 4)	94	0.10	48	(8, 3)
58	0.45	63	(8, 4)	93	0.12	48	(10, 3)
55	0.47	62	(10, 4)	92	0.11	48	(12, 3)
43	0.55	34	(8, 5)	89	0.11	49	(12, 4)
43	0.56	34	(10, 5)	88	0.11	49	(8, 5)
42	0.60	34	(12, 5)	87	0.11	49	(10, 5)
40	0.58	37	(3, 6)	87	0.12	48	(3, 6)
39	0.62	53	(12, 4)	86	0.12	49	(12, 5)
37	0.71	40	(8, 6)	78	0.15	49	(6, 6)
37	0.71	40	(6, 6)	78	0.15	49	(8, 6)
37	0.78	39	(10, 6)	76	0.16	49	(10, 6)
36	0.82	39	(12, 6)	75	0.17	49	(12, 6)
26	0.92	47	(6, 7)	72	0.17	49	(6, 7)
25	0.98	46	(8, 7)	70	0.18	49	(8, 7)
25	1.32	46	(10, 7)	68	0.25	49	(10, 7)
24	1.22	44	(12, 7)	65	0.23	49	(12, 7)

Table 4.2: Comparison of Different Device Setting

assuming  $Homo-V_t$  class.

#### 4.3.1.2 Impact of Heterogeneous- $V_t$ and Power-gating

It has been shown that heterogeneous- $V_t$  and power-gating may have great impact on energy delay tradeoff [CWL05]. Here we further consider the impact of heterogeneous- $V_t$  on the yield by comparing  $Homo-V_t$  and  $Hetero-V_t$  in min-ED device setting. Table 4.3 shows the results of the dominant architectures in all classes. The average yield for each class is presented in the last row of the ta-

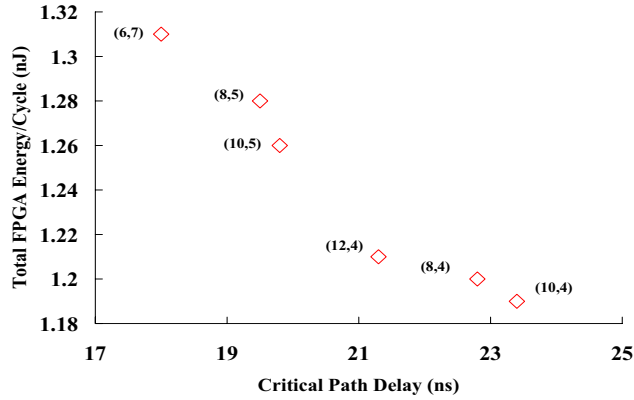


Figure 4.2: Energy-delay tradeoff among architectures in  $Homo-V_t$  using min-ED device setting.

ble. Comparing the yield of  $Homo-V_t$  and  $Hetero-V_t$ , we can see that the average yield is improved by 5% via applying different  $V_t$  for logic blocks and interconnect. Therefore, introducing heterogeneous- $V_t$  could improve yield with no or little area increase (due to an increase in doping well area).

Furthermore, power-gating can be applied to unused FPGA logic blocks and interconnect to reduce leakage power. As only one sleep transistor is used for one logic block, we use a 210X PMOS as the sleep transistor for each logic block. For interconnect, the area overhead associated with sleep transistors is more significant. We therefore use a 2X PMOS as the sleep transistor for each interconnect switch. Comparing the yield of  $Homo-V_t$  and  $Homo-V_t+G$  in Table 4.3, applying power-gating can improve the yield by 8%. Comparing the yield of  $Hetero-V_t$  and  $Homo-V_t+G$ , power-gating can obtain more yield improvement than heterogeneous- $V_t$  at the cost of chip-level area overhead between 10% to 20%. As leakage power can be greatly reduced by power-gating, little benefit can be introduced by applying simultaneous heterogeneous- $V_t$  and power-gating, and we will not present the results here. Again, with heterogeneous- $V_t$  or power-gating,

LUT size  $K=4$  is the best for leakage yield rate.

(N,K)	Homo- $V_t$					Hetero- $V_t$						Homo- $V_t+G$				
	Vdd (V)	$V_t$ (V)	Y (%)	M (W)	SD (%)	Vdd (V)	$CV_t$ (V)	$IV_t$ (V)	Y (%)	M (W)	SD (%)	Vdd (V)	$V_t$ (V)	Y (%)	M (W)	SD (%)
(6,4)	0.90	0.30	97	0.07	48	0.90	0.30	0.35	99	0.06	46	0.90	0.30	99	0.04	48
(8,4)	0.90	0.30	97	0.08	48	0.90	0.30	0.35	99	0.06	46	0.90	0.30	99	0.04	48
(10,4)	0.90	0.30	96	0.08	48	0.90	0.30	0.35	98	0.06	46	0.90	0.30	99	0.04	48
(12,4)	0.90	0.30	89	0.11	49	0.90	0.30	0.35	96	0.08	45	0.90	0.30	99	0.05	48
(6,5)	0.90	0.30	96	0.08	49	0.90	0.30	0.35	98	0.06	46	0.90	0.30	99	0.05	48
(8,5)	0.90	0.30	88	0.11	49	0.90	0.30	0.35	95	0.08	46	0.90	0.30	98	0.05	48
(10,5)	0.90	0.30	87	0.11	49	0.90	0.30	0.35	95	0.08	46	0.90	0.30	98	0.05	48
(6,6)	0.90	0.30	78	0.15	49	0.90	0.30	0.35	86	0.11	46	0.90	0.30	92	0.08	48
(8,6)	0.90	0.30	78	0.15	49	0.90	0.30	0.35	85	0.12	46	0.90	0.30	91	0.08	48
(6,7)	0.90	0.30	72	0.17	49	0.90	0.30	0.35	77	0.14	47	0.90	0.30	83	0.11	48
Avg	0.90	0.30	88	0.11	49	0.90	0.30	0.35	93	0.08	46	0.90	0.30	96	0.06	48

Table 4.3: Comparison of leakage yield between classes.

### 4.3.2 Timing Yield

For timing yield analysis, we only analyze the delay of the largest MCNC benchmark *clma*. Similarly, the timing yield is often studied using selected test circuit such as ring oscillator for ASIC in the literature. Figure 4.3 shows the delay with intra-die and inter-die channel length variation at baseline architecture (8,4) with ITRS device setting. As shown in the figure, there is a 1.9X span with  $\pm 3\sigma$   $L_g$  variation, and a 1.1X span without  $L_g$  variation. Clearly, delay is more sensitive to inter-die variation than within-die variation. This is because of the independence of local  $L_{eff}$  variation between each element. Therefore the effect of within-die  $L_{eff}$  variation tends to average out when the critical path is long enough.

For timing yield, we discard dies with critical delay larger than the cutoff delay, which is 1.1X of the nominal critical path delay of each architecture. Table 4.4

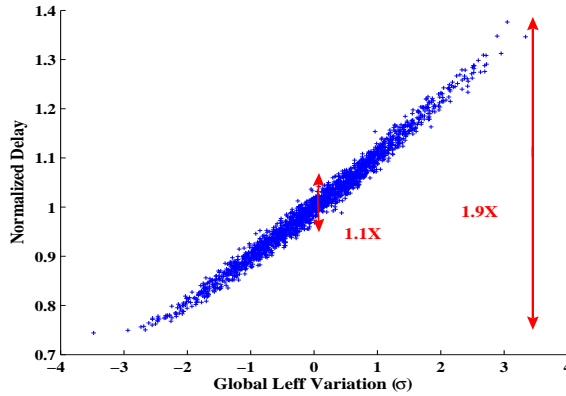


Figure 4.3: Delay of baseline architecture ( $N=8$ ,  $K=4$ ) with ITRS device setting under intra-die and inter-die Leff variation .

shows the delay yield of  $Homo-V_t+G$ . One can see from this table that a larger LUT size will give a higher yield rate. This is because a larger LUT size generally gives a smaller mean delay with a shorter critical path (see Fig 4.2), i.e., smaller number of elements in the path, which leads to a smaller variance. Therefore, a larger LUT size leads to a higher timing yield. As the timing specification may be relaxed for certain applications that are not timing-critical, the cutoff delay may be relaxed in this case. In this table, we also show the yield with the cutoff delay as  $1.2X$  of the nominal delay. The yield rate under a higher cutoff still has the same trend as that under a lower cutoff. Note that the other architecture classes have similar trends on timing yield.

### 4.3.3 Leakage and Timing Combined Yield

Figure 4.4 presents the leakage and delay variation for the baseline case using Monte Carlo simulation with *Ptrace*. It can be seen that a smaller delay leads to a larger leakage in general. This is because of the inverse correlation between circuit delay and leakage. A device with short channel length has a small delay

	Y 1.1X (%)	Y 1.2X (%)	Mean (ns)
(6,4)	69	86	39.9
(8,4)	70	86	40.7
(10,4)	69	86	41.5
(12,4)	71	88	38.3
(6,5)	75	91	36.4
(8,5)	74	90	34.6
(10,5)	74	90	34.7
(6,6)	77	93	30.8
(8,6)	78	94	29.9
(6,7)	79	95	27.7
Avg	75	90	35.4

Table 4.4: Timing yield for Homo- $V_t+G$

and consumes large leakage, which may lead to a high leakage. To calculate the leakage and delay combined yield, we set the cutoff leakage as the nominal leakage plus 30% that of the baseline, while the cutoff delay is 1.2X of each architecture’s nominal delay.

Table 4.5 presents the combined yield for *Homo- $V_t$*  with ITRS device setting and all classes with min-ED device setting. The area overhead introduced by power-gating is also presented in the table. Comparing *Homo- $V_t$*  with ITRS device setting and min-ED device setting, the combined yield is improved by 21%. Comparing the classes using min-ED device setting, *Hetero- $V_t$*  has a 3% higher yield than *Homo- $V_t$*  due to heterogeneous- $V_t$  while *Homo- $V_t+G$*  has a 8% higher yield than *Homo- $V_t$*  due to power-gating. *Homo- $V_t+G$*  has the highest combined yield with an average of 16% area overhead. Device tuning and power-gating improve yield by 29% comparing *Homo- $V_t+G$*  with min-ED setting to *Homo- $V_t$*  with ITRS setting. This table also shows that architectures with LUT size 5 gives the highest yield within each class. This is because it has both a relatively high leakage yield as well as timing yield.

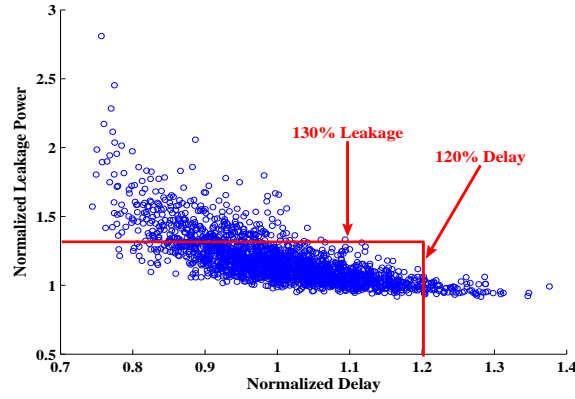


Figure 4.4: Leakage and delay of baseline architecture ( $N=8$ ,  $K=4$ ) with ITRS setting under process variations.

(N,K)	ITRS	Min-ED			
	Homo- $V_t$	Homo- $V_t$	Hetero- $V_t$	Homo- $V_t+G$	
	Y(%)	Y(%)	Y(%)	Y(%)	Area Inc(%)
(6,4)	71	83	83	86	18
(8,4)	67	81	81	86	14
(10,4)	65	81	81	86	17
(12,4)	48	77	81	87	20
(6,5)	79	85	84	90	14
(8,5)	55	81	86	89	15
(10,5)	55	81	86	89	19
(6,6)	49	77	82	88	15
(8,6)	49	75	80	88	16
(6,7)	45	73	77	86	10
Avg	58	79	82	87	16

Table 4.5: Combined Leakage-delay yield between FPGA Classes.

## CHAPTER 5

### Future Work

The Ptrace introduced in Chapter 3 is accurate and fast. But we still need to run VPR and PSim for trace correction. In the future, I will also study the high level FPGA power and delay estimation. That means given the logic function, estimate the power and delay for an FPGA without placement and route. The challenge of this work is that the switching activity and the interconnect architecture is hard to predict.

My other future topic is to improve the process variation model and perform place and route considering process variation. In the variation model discussed in Chapter 4, we ignore spatial correlation and assume the top 10 paths are independent. However in practice, there is overlap between paths and there is spatial correlation between elements in the same chip. Such correlation will greatly affect the accuracy of our yield estimation. In the future, we plan to extend the process variation model to consider spatial correlation and path convergence. Moreover, with the process variation model, we will perform power and delay optimization by customize placement considering process variation. Assuming the variation map of a chip is given, when doing placement and route, we can put the critical path in the fast region of the chip to reduce delay.

## REFERENCES

- [ANT04] Jason H. Anderson, Farid N. Najm, and Tim Tuan. “Active Leakage Power Optimization for FPGAs.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Februray 2004.
- [AR00] E. Ahmed and J. Rose. “The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, pp. 3–12, Feb 2000.
- [BKN03] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. “Parameter Variations and Impact on Circuits and Microarchitecture.” In *Proc. Design Automation Conf.*, June 2003.
- [BRM99] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Feb 1999.
- [CCL04] Deming Chen, Jason Cong, Fei Li, and Lei He. “Low-power technology mapping for FPGA architectures with dual supply voltages.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2004.
- [CWL05] Lerong Cheng, Phoebe Wong, Fei Li, Yan Lin, and Lei He. “Device And Architecture Co-Optimization for FPGA Power Reduction.” In *Proc. Design Automation Conf.*, June 2005.
- [DT05] Vijay Degalahal and Tim Tuan. “Methodology for High Level Estimation of FPGA Power Consumption.” In *Proc. Asia South Pacific Design Automation Conf.*, Jan 2005.
- [Fei04] Fei Li, Yan Lin and Lei He. “Vdd Programmability to Reduce FPGA Interconnect Power.” In *Proc. Intl. Conf. Computer-Aided Design*, November 2004.
- [GLV04] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan. “A Dual-Vdd Low Power FPGA Architecture.” In *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2004.
- [GND01] Anne Gattiker, Sani Nassif, Rashmi Dinakar, and Chris Long. “Timing Yield Estimation from Static Timing Analysis.” In *International Symposium on Quality of Electronic Design*, 2001.
- [GTV04] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan. “Reducing Leakage Energy in FPGAs Using

- Region-Constrained Placement.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2004.
- [Int02] International Technology Roadmap for Semiconductor. In <http://public.itrs.net/>, 2002.
- [Jas04] Jason H. Anderson and Farid N. Najm. “Low-Power Programmable Routing Circuitry for FPGAs.” In *Proc. Intl. Conf. Computer-Aided Design*, November 2004.
- [KG92] J. Kouloheris and A. El Gamal. “FPGA Area vs. Cell Granularity - Lookup Tables and PLA Cells.” In *1st ACM Workshop on FPGAs, Berkeley, CA*, Feb 1992.
- [KR98] E. Kusse and J. Rabaey. “Low-Energy Embedded FPGA Structures.” In *Proc. Intl. Symp. Low Power Electronics and Design*, pp. 155–160, August 1998.
- [LB93] G. G. Lemieux and S. D. Brown. “A Detailed Router for Allocating Wire Segments in Field-Programmable Gate Arrays.” In *Proceedings of the ACM Physical Design Workshop*, April 1993.
- [LCG05] Andrea Lodi, Luca Ciccarelli, and Roberto Giansante. “Combining Low-Leakage Techniques for FPGA Routing Design.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2005.
- [LCH03] Fei Li, Deming Chen, Lei He, and Jason Cong. “Architecture Evaluation for Power-Efficient FPGAs.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2003.
- [LH05] F. Li and L. He. “Power Modeling and Characteristics of Field Programmable Gate Arrays.” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, October 2005.
- [LLH04a] Fei Li, Yan Lin, and Lei He. “FPGA Power Reduction Using Configurable Dual-Vdd.” In *Proc. Design Automation Conf.*, June 2004.
- [LLH04b] Fei Li, Yan Lin, Lei He, and Jason Cong. “Low-power FPGA Using Pre-Defined Dual-Vdd/Dual-Vt Fabrics.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2004.
- [LLH05a] Yan Lin, Fei Li, and Lei He. “Power Modeling and Architecture Evaluation for FPGA with Novel Circuits for Vdd Programmability.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2005.

- [LLH05b] Yan Lin, Fei Li, and Lei He. “Routing Track Duplication with Fine-Grained Power-Gating for FPGA Interconnect Power Reduction.” In *Proc. Asia South Pacific Design Automation Conf.*, Jan 2005.
- [LW03] Julien Lamoureux and Steven J.E. Wilton. “On the Interaction Between Power-Aware FPGA CAD Algorithms.” In *Proc. Intl. Conf. Computer-Aided Design*, pp. 701–708, November 2003.
- [MM04] R. Mukherjee and S. O. Memik. “Power-Driven Design Partitioning.” In *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2004.
- [Nas01] Sani R. Nassif. “Modeling and Analysis of Manufacturing Variations.” In *Proc. IEEE Custom Integrated Circuits Conf.*, 2001.
- [PYW02] K. Poon, A. Yan, and S. Wilton. “A Flexible Power Model for FPGAs.” In *Proc. of 12th International conference on Field-Programmable Logic and Applications*, Sep 2002.
- [RDB04] R. Rao, A. Devgan, D. Blaauw, and D. Sylvester. “Parametric Yield Estimation Considering Leakage Variability.” In *Proc. Design Automation Conf.*, June 2004.
- [RFL90a] J. Rose, R. J. Francis, D. Lewis, and P. Chow. “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Functionality on Area Efficiency.” *Proc. IEEE Int. Solid-State Circuits Conf.*, 1990.
- [RFL90b] J. Rose, R.J. Francis, D. Lewis, and P. Chow. “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Functionality on Area Efficiency.” *IEEE Journal of Solid-State Circuits*, 1990.
- [SGT05] S. Srinivasan, A. Gayasen, and T. Tuan. “Leakage Control in FPGA Routing Fabric.” In *Proc. Asia South Pacific Design Automation Conf.*, January 2005.
- [SRC92] S. Singh, J. Rose, P. Chow, and D. Lewis. “The Effect of Logic Block Architecture on FPGA Performance.” *IEEE Journal of Solid-State Circuits*, 1992.
- [TL03] Tim Tuan and Bocheng Lai. “Leakage Power Analysis of a 90nm FPGA.” In *Proc. IEEE Custom Integrated Circuits Conf.*, 2003.
- [Xil02] Xilinx Corporation. “Virtex-II 1.5V Platform FPGA Complete Data Sheet.” July 2002.

- [Y 05] Y. Lin, F. Li and L. He. “Circuits and Architectures for Vdd Programmable FPGAs.” In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2005.
- [ZWB04] S. Zhang, V. Wason, and K. Banerjee. “A Probabilistic Framework to Estimate Full-chip Subthreshold Leakage Power Distribution Considering Within-Die and Die-to-Die P-T-V Variations.” In *ISLPED*, Aug 2004.