University of California

Los Angeles

Statistical Analysis and Optimization for Timing and Power of VLSI Circuits

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical Engineering

by

Lerong Cheng

2010

The dissertation of Lerong Cheng is approved.

Lieven Vandenberghe

Glenn Reinman

Sudhakar Pamarti

Lei He, Committee Co-chair

Puneet Gupta, Committee Co-chair

University of California, Los Angeles

2010

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgments

Many people offered me significant help during the development of this dissertation. I would like to acknowledge them here.

First of all, I would like to thank my adviser, Prof. Lei He and my co-adviser, Prof. Puneet Gupta. They not only helped me to develop skills to solve problems, but also motivated and enlightened me at all time. This dissertation would not exist without their enormous encouragement, support, and inspiration.

I am indebted to Prof. Sudhakar Pamarti, Prof. Glenn Reinman, and Prof. Lieven Vandenberghe for serving as members of my dissertation committee and their suggestions to improve this work.

I treasure the opportunity that I have worked with a group of wonderful people and would like to thank them for all the helps and support through my research work, especially those colleagues at the UCLA Design Automation Lab and NanoCAD Lab. In particular, I thank Dr. Fei Li, Dr. Yan Lin, Miss Phoebe Wong, and Dr. Jinjun Xiong. The work on device and architecture co-optimization for FPGA in Chapter 2 and Chapter 3 is a joint project with Fei Li, Yan Li, and Phoebe Wong. Yan Li also helped me on developing the transistor and circuit level power and delay model for FPGA process and architecture concurrent design in Chapter 4. Jinjun Xiong helped me to develop the experiment and revise the writing for the statistical static timing analysis work in Chapter 5. He also helped me to develop the chip-wise optimization flow to minimize FPGA delay considering process variation.

I sincerely thank Dr. Kevin Cao for his help with my research. He helped me to develop the FPGA device aging model, which is reported in Chapter 4.

I also indebted to Dr. Costas Spanos and Mr. Kun Qian for helping me to develop the across-wafer variation model, which is reported in Chapter 6.

I would also like to thank many researchers include Dr. Qi Xiang, Dr. Jeffery Tang, Mr. Tim Tuan, Miss Wenping Wang, for collaboration and helpful discussions.

Finally, and most of all, I am grateful to my family. I would like to thank my wife Wen Wang for her constant understanding and supporting me during my Ph.D. study. I would also like to thank my parents, my brother, and my sister in law, for their dedication, and support. This dissertation becomes so humble compared to their constant inspiration, support, and love.

Vita

1979	Born in People's Republic of China.
1997-2001	B.S. in Electronic and Communication Engineering, Zhongshan University, Guangzhou, P.R. China.
2001-2003	M.S. in Electrical and Computer Engineering, Portland State University, Portland, Oregon, USA. Teaching Assistant, Analogue Circuit Design, 2002. Graduate Student Researcher, 2002-2003.
2003-2004	First year Ph.D. program, in Electrical and Computer Engineering, University of Colorado, Boulder, Colorado, USA. Graduate Student Researcher, 2003-2004.
2004-present	P.D. program, in Electrical Engineering, University of California, Los Angeles, California, USA. Teaching Assistant, Advanced Computer Architecture, 2006. Graduate Student Researcher, UCLA. Design Automation Lab, 2004-2008. Graduate Student Researcher, UCLA. Nano CAD Lab, 2008-present.
2006	Intern Engineer, Nvidia Corp., San Jose, California, USA. Worked on GPU testing.
2006-2007	Research Intern, Altera Corp., San Jose, California, USA. Worked on statistical power and delay estimation.

PUBLICATIONS

L. Cheng, F. Li, **Y. Lin**, P. Wong and L. He, "Device and Architecture Co-Optimization for FPGA Architecture Power Reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, volume: 26, issue: 7, July 2007, pp: 1211-1221.

L. Cheng, P. Gupta, and L. He , "On Confidence in Characterization and Application of Variation Models," accepted by *IEEE/ACM Asia South Pacific Design Automation Conference (ASPDAC)*, February 2010.

L. Cheng, P. Gupta, and L. He , "Accounting for Non-linear Dependence Using Function Driven Component Analysis," *IEEE/ACM Asia South Pacific Design Automation Conference (ASPDAC)*, February 2009, pp: 474-479.

L. Cheng, P. Gupta, and L. He , "Efficient Additive Statistical Leakage Estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, volume: 28, issue: 11, November 2009, pp:1777-1781.

L. Cheng, P. Gupta, C. Spanos, K. Qian, and L. He , "Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across Wafer Variability," *Design Automation Conference (DAC)*, July 2009, pp: 104-109.

L. Cheng, W. Hung, X.Song, and G. Yang, "Congestion Estimation for 3D Routing," *IEEE Computer Society annual symposium on VLSI (ISVLSI)*, February 2004, pp:

239-240.

L. Cheng, W. Hung, X.Song, and G. Yang, "Congestion Estimation for Three-Dimensional Architectures," *IEEE Transactions on Circuits and Systems II: Express Briefs (TCAS II)*, volume: 51, issue: 12, December 2004, pp: 655- 659.

L. Cheng, Y. Lin, L. He, and Y. Cao, "Trace-Based Framework for Concurrent Development of Process and FPGA Architecture Considering Process Variation and Reliability," *International Symposium on Field-Programmable Gate Arrays (ISFPGA)*, February 2008, pp: 159-168.

L. Cheng, X.Song, G. Yang, W. Hung, Z. Tang, and S. Gao, "A Fast Congestion Estimator for Routing with Bounded Detours," *Integration, the VLSI Journal (Integrat VLSI J)*, volume: 41, issue: 3, May 2008, pp: 360-370.

L. Cheng, X.Song, G. Yang, and Z. Tang, "A Fast Congestion Estimator for Routing with Bounded Detours," *IEEE/ACM Asia South Pacific Design Automation Conference (ASPDAC)*, February 2004, pp: 666-670.

L. Cheng, P. Wong, F. Li, Y. Lin, and L. He, "Device and Architecture Co-Optimization for FPGA Power Reduction," *Design Automation Conference (DAC)*, June 2005, pp: 915-920.

L. Cheng, J. Xiong, and L. He, "FPGA Performance Optimization via Chipwise Placement Considering Process Variations," *International Conference on Field Programmable Logic and Applications (FPL)*, August 2006, pp:1-6.

L. Cheng, J. Xiong, and L. He, "Non-Linear Statistical Static Timing Analysis for Non-Gaussian Variation Sources," *ACM/IEEE International Workshop on Timing issue: in the Specification and Synthesis of Digital Systems(TAU)*, February 2007, pp: 80-85.

L. Cheng, J. Xiong, and L. He, "Non-Linear Statistical Static Timing Analysis for Non-Gaussian Variation Sources," *Design Automation Conference (DAC)*, June 2007, pp: 250-255.

L. Cheng, J. Xiong, and L. He, "Non-Gaussian Statistical Timing Analysis Using Second-Order Polynomial Fitting," *IEEE International Workshop on Design for Manufacturability and Yield (DFM&Y)*, October 2007.

L. Cheng, J. Xiong, and L. He, "Non-Gaussian Statistical Timing Analysis Using Second-Order Polynomial Fitting," *IEEE/ACM Asia South Pacific Design Automation Conference (ASPDAC)*, February 2008, pp: 298-303.

L. Cheng, J. Xiong, and L. He, "Non-Gaussian Statistical Timing Analysis Using Second-Order Polynomial Fitting," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, volume: 28, issue: 1, January 2009, pp: 130-140.

F. He, **L. Cheng**, X. Song, and G. Yang, "An Analytical Congestion Model With Bounded-Based Detours,". accepted by *Journal of Circuits, Systems, and Computers (JCSC)*.

F. He, **L. Cheng**, G. Yang, X. Song, M. Gu, and J. Sun, "On Theoretical Upper Bounds for Routing Estimation," *Journal of Universal Computer Science (J.UCS)*, volume: 11, Number 6, June 2005, pp: 916-925.

F. He, M. Gu, X. Song, Z. Tang, and **L. Cheng**, "Probabilistic Estimation for Routing Space," *The Computer Journal (TCJ)*, May 2005, pp: 667-676.

F. He, X. Song, **L. Cheng**, G. Yang, Z. Tang, M. Gu, and J. Sun, "A Hierarchical Method for Wiring and Congestion Prediction," *IEEE Computer Society annual symposium on VLSI (ISVLSI)*, May 2005, pp: 307-308.

F. He, X. Song, M. Gu, **L. Cheng**, G. Yang, Z. Tang, and J.Sun, "A Combinatorial Congestion Estimation Approach with Generalized Detours," *Computers & Mathematics with Applications (CAM)*, volume: 51, : 6-7, March 2006, pp: 1113-1126.

W. Hung, X.Song, T. Kam, **L. Cheng**, and G. Yang, "Routability Checking For Three-Dimensional Architectures," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, volume: 12, issue: 12, December 2004, pp: 1371-1374.

Y. Jeng and **L. Cheng**, "Digital Spectrum of a Nonuniformly Sampled Two-Dimensional Signal and Its Reconstruction," *IEEE Transaction on Instrumentation and Measurement (TIM)*, volume: 54, issue: 3, June 2005, pp: 1180-1187.

P. Wong, **L. Cheng**, Y. Lin, and L. He, "FPGA Device and Architecture Evaluation Considering Process Variation," *International Conference on Computer Aided Design (ICCAD)*, November 2005, pp: 19-24.

Abstract of the Dissertation

Statistical Analysis and Optimization for Timing and Power of VLSI Circuits

by

Lerong Cheng

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2010

Professor Puneet Gupta, Co-chair

Professor Lei He, Co-chair

As CMOS technology scales down, process variation introduces significant uncertainty in power and performance to VLSI circuits and significantly affects their reliability. If this uncertainty is not properly handled, it may become the bottleneck of CMOS technology improvement. This dissertation proposes novel techniques to model, analyze, and optimize power and performance of FPGAs and ASICs considering process variation. This dissertation focuses on two aspects: (1) Process and architecture concurrent optimization for FPGAs; (2) Statistical timing modeling and analysis.

To perform process and architecture concurrent optimization, an efficient and accurate FPGA power, delay, and variation evaluator, *Ptrace*, is proposed. With Ptrace, we present the first in-depth study on device and FPGA architecture co-optimization to minimize power, delay, area, and variation considering hundreds of device and architecture combinations. Furthermore, to enable early stage process and architecture co-optimization without stable device models, we develop transistor level and circuit level power, delay, and reliability models and incorporate them with Ptrace. With the extended Ptrace, we perform architecture and process parameters concurrent optimiza-

tion for FPGA power, delay, variation, and reliability.

To perform statistical timing modeling and analysis, we first present an efficient and accurate statistical static timing analysis (SSTA) flow for non-linear cell delay model with non-Gaussian variation sources. All operations in this flow are performed by analytical equations without any time consuming numerical approach. Then, to further improve the efficiency and accuracy of statistical timing analysis, we develop a new die-level spatial variation model which accurately models the across-wafer variation. Besides modeling spatial variation, mean and variance uncertainty introduced by limited number of samples is another problem in SSTA. To solve this problem, we evaluate the confidence for statistical analysis and estimate the guardband value to ensure a target confidence.

To the best of our knowledge, this dissertation is the first novel study of device, process, and architecture concurrent co-optimization for FPGA power, delay, variation, and reliability; and is the first work to model across-wafer variation at die-level and to consider confidence guardband in statistical analysis.

CHAPTER 1

Introduction

*C*omplementary *M*etal *O*xide *S*emiconductor (*CMOS*) manufacturing in nanometer region has resulted in great achievements in the world of electronics. The performance and capabilities of *V*ery *L*arge *S*cale *I*ntegration (*VLSI*) circuits improve rapidly with the aggressive scaling of transistors in CMOS. However, the increase of leakage power and process variation have emerged to slow down this trend.

Since leakage power is exponentially related to channel length, it increases rapidly as CMOS technology scales down. Figure 1.1 [114] illustrates the power density for different technology nodes in the past thirty years. It can be seen that power density is increasing at a dramatic rate. Therefore, power consumption becomes a crucial design constraint for nano-scale VLSI designs.

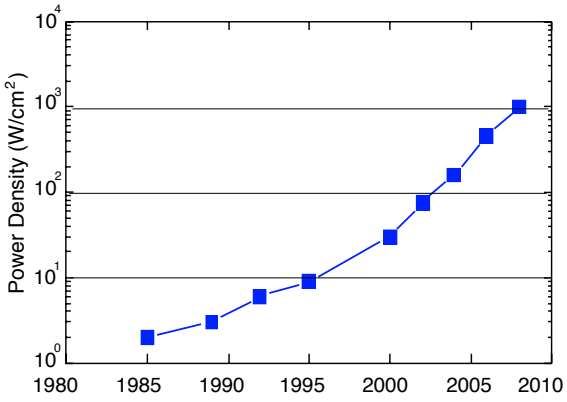


Figure 1.1: Increasing of power density.

In comparison to challenges created by power, process variation is more difficult to handle in VLSI design. Due to the inevitable manufacturing fluctuations and imperfections, transistors on different copies of the same design, or even at different locations on the same chip will vary significantly. Process variation is the uncertainty of physical process parameters, such as channel length, doping density, oxide thickness, channel width, metal thickness, metal width, and so on. According to the causes of variation, process variation can be classified into two types [162]:

- *Catastrophic defects* are caused by isolated random events (such as particles or other contaminations) during manufacturing, which render chips non-functional.
- *Parametric variations* are caused by random fluctuations in process conditions so that the physical properties of some parameters on a chip differ from the original design. The fluctuations may include aberrations in stepper lens, doping density, and manufacturing temperature.

Process variations introduce significant uncertainty for both circuit performance and leakage power. It has been shown in [25] that even for the 180nm technology, process variation can lead to 1.3X variation in frequency and 20X variation in leakage power, as illustrated in Figure 1.2. Such impact will become even larger in future technology generations. In recent years, many methods, such as statistical modeling, analysis, and optimization for VLSI circuits, have been developed to alleviate the variation effects. This research is called "*D*esign *for M*anufacturability " (*DFM*). *I*nternation *T*echnology *R*oadmap for *S*emiconductors (*ITRS*) predicts the DFM technology requirement for process variation in the near-term future, as shown in Table 1.1 [69]. From the table, it is predicted that leakage power variation will increase to 331% and performance variation will increase to 88% in the year 2015. Therefore, as CMOS technology scales down to nano-meter region, process variation will become a potential show-stopper if it is not appropriately handled.

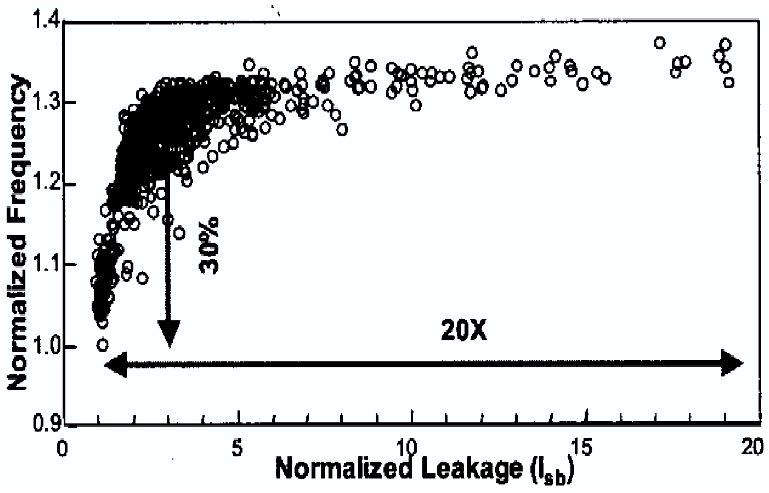


Figure 1.2: Leakage power and frequency variation.

Year	2007	2008	2009	2010	2011	2012	2013	2014	2015
% of V_{dd} variability	10	10	10	10	10	10	10	10	10
% of V_{th} variability (minimum size)	33	37	42	42	42	58	58	81	81
% of V_{th} variability (typical size)	16	18	20	20	20	26	26	36	36
% of CD (L_{gate}) variability	12	12	12	12	12	12	12	12	12
% circuit performance variability	46	48	49	51	60	63	63	63	63
% circuit total power variability	56	57	63	68	72	76	80	84	88
% circuit leakage power variability	124	143	186	229	255	281	287	294	331

Table 1.1: Impact of process variation in near-term future.

There are four common design styles in current VLSI: *Application S*pecific *I*ntegrated *C*ircuit (*ASIC*), *F*ield-*P*rogrammable *G*ate *A*rray (*FPGA*), *A*pplication *S*pecific *I*nstruction set *P*rocessor (*ASIP*), and general purpose processor or microprocessor. Different design styles may have different power consumption requirement and vulnerability to process variation. Therefore, different modeling, analysis, and optimization techniques should be developed for different design styles.

Among the design styles, FPGA is the most flexible one. Since FPGA allows the same silicon implementation to be programmed or re-programmed for a variety of applications, it provides low non-recurring engineering (NRE) cost and short time to

market. Due to this advantage, FPGA industry has grown rapidly since its invention in 1984. However, in order to achieve programmability, FPGA pays the penalty with decrease of performance, power, and area. Previous studies [83, 82] have shown a 100X energy difference, a 4.3X delay difference, and a 40X area difference between FPGA designs and their ASIC counterparts. As discussed before, power consumption is a crucial design constraint for nano-scale VLSI circuits. The power problem is more significant for FPGAs than other design styles because FPGA has higher power consumptions. On the other hand, since FPGAs have lots of regularity, the impact of process variation on FPGAs are not as significant as other design styles. Nevertheless, the impact of process variation on FPGAs should not be ignored and should be properly handled. To solve the above problems, this dissertation focuses on optimizing power and performance for FPGAs considering process variation. The first objective of this dissertation is:

Objective 1: Concurrently evaluate FPGA architecture and process parameters to optimize power, delay, and area considering process variation and reliability.

In comparison to FPGAs, ASICs have higher performance, lower power, and smaller area. However, due to the increased design complexity, ASICs has higher NRE cost, design cost, and longer time to market than FPGAs. Due to the above advantages and disadvantages, ASICs are usually used in high performance or low power applications, wherein the impact of process variation is very significant. Therefore, variation modeling and analysis in ASICs are more important than in FPGAs.

Process variation may cause *manufacturing yield loss*. Manufacturing yield loss is defined as the ratio between the number of chips that fail to meet the design specifications and the total number of manufactured chips [112]. There are several reasons causing chip failure, such as catastrophic defects and parametric variations. Figure 1.3

illustrates the performance variation caused by parametric variations. It can be seen that chip delay is spread over a wide range and the delay of some chips is higher than the specification value. The chips failing to meet the delay specification contribute to yield loss.

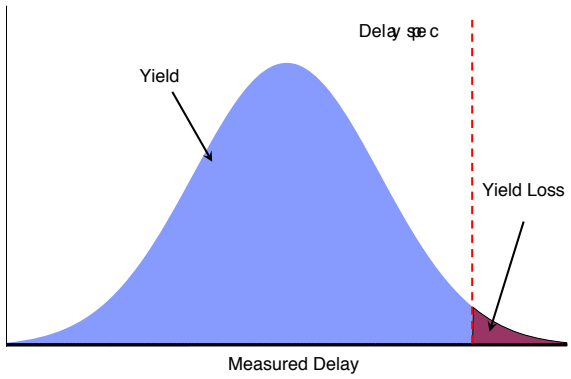


Figure 1.3: Delay yield.

In order to analyze and optimize performance yield, *S*tatistical *S*tatic *T*iming *A*nalysis (*SSTA*) is proposed. Instead of estimating the chip delay deterministically as in the traditional *S*tatic *T*iming *A*nalysis (*STA*), SSTA estimates the chip delay statistically. SSTA not only estimates the nominal value of chip delay but also provides the delay distribution. Figure 1.4 shows the SSTA flow. In this flow, there are three key components [122]:

- Modeling of process variation from silicon process characterization.
- Modeling of sensitivities of delay for standard cell libraries with regards to process parameter variations (library modeling).
- Statistics aware engines for STA and optimization.

In this dissertation, we mainly focus on modeling of process variation and developing an accurate SSTA engine. The second objective of this dissertation is:

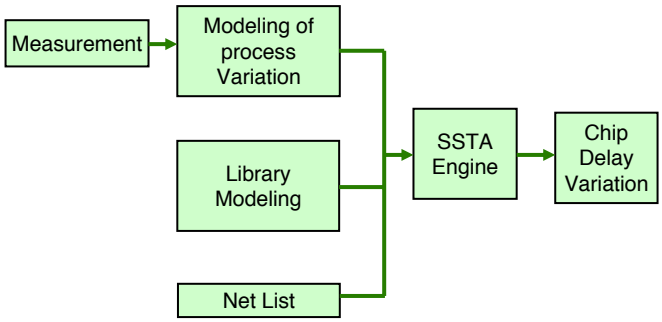


Figure 1.4: SSTA flow.

Objective 2: Statistical Timing Modeling and Analysis.

In the following of this chapter, Section 1.1 reviews some related research works; Section 1.2 discusses the major contributions of this dissertation; and Section 1.3 outlines this dissertation.

1.1 Literature Review

1.1.1 FPGA Power, Performance, and Area Optimization

It is well known that architecture, including logic block (or cluster) size, *L*ook-*U*p *T*able (*LUT*) size, and routing wire segment length, has great impact on FPGA power, performance, and area. Therefore, architecture evaluation for FPGA optimization attracts lots of concerns. In early 1990s, architecture evaluation mainly focuses on optimizing delay [143] and area [130]. These works showed that for non-clustered FPGAs, LUT size of 4 achieves the smallest area and LUT size 5 or 6 minimizes delay. After cluster-based island style FPGA became popular, architecture evaluation for this type of FPGA was performed. [10] showed that LUT sizes ranging from 4 to 6 and cluster sizes between 4 and 10 produce the best area-delay product.

As discussed before, power consumption is one of the most important design constraints of FPGA design when technology scales down to nano-meter region. Therefore, after the year 2000, FPGA power modeling became a hot research topic. [158] quantified the leakage power of commercial FPGA architecture and [48] introduced a high level FPGA power estimation methodology. [123, 89, 92] presented power evaluation frameworks for generic parameterized FPGAs and showed that leakage power takes higher portion of total power consumption in FPGAs than in ASICs. It was also shown that interconnects consume even more power than logic elements in FPGA. At the same time, FPGA power optimization was also studied. Power aware FPGA CAD algorithms was proposed in [85]; power driven partition method was presented in [117]; a leakage power saving routing multiplexer and an input control method were developed [150]; and a configuration inversion method to save the leakage power was proposed in [12]. Architecture evaluation for power and delay minimization is then studied in [93, 123, 92].

Due to programmability, utilization rate of circuit elements in FPGA is very low. To reduce leakage power consumption of unused circuit elements, body bias [110] and power gating [59, 102] were applied. To further reduce power on used circuit elements, programmable dual-V_{dd} was first applied to logic blocks [93, 90, 98, 91] and then extended to interconnects [58, 51, 71]. Architecture evaluation considering power gating and dual-V_{dd} was performed [101]. Besides leakage power minimization, a glitch minimization technique [84] was proposed to reduce dynamic power.

Recently, FPGA modeling and optimization considering process variation was studied. [31, 43, 100, 115, 147, 99, 135, 79, 115, 154, 137, 28] presented techniques to statistically optimize FPGA performance. [136] performs parametric yield estimation for FPGA considering within-die variation.

1.1.2 Statistical Timing Modeling, Analysis, and Optimization

Process Variation Modeling- Process variation introduces significant uncertainty to circuit power and delay. In order to analyze the power and delay uncertainty, process variation modeling is needed. An early work was proposed in [25] whereby process variation is separated into inter-die variation and within-die variation. All transistors on the same die share the same inter-die variation and different dies may have different inter-die variation; each transistor has its own within-die variation and the within-die variation for different transistors are assumed to be independent. Later on, it was observed that within-die variation is not independent but spatially correlated and the correlation depends on the distance between two within-die locations. In this case, spatial variation was modeled as correlated random variables [6, 32] and principle component analysis was applied to perform statistical timing analysis. In this model, a chip is divided into several grids and each grid has its own spatial variation. The spatial variations of different grids are correlated and the correlation coefficients depend on the distance between two grids. However, obtaining the correlation coefficient between two grids became a problem for this model. [172, 173, 105] solved this problem by modeling the spatial variation as a random field [174] and assuming the correlation coefficient to be a function of distance. After that, several more complicated spatial variation models were proposed [45, 105, 189, 55, 64].

Statistical Analysis- With the model of process variation, statistical analysis and optimization are then performed. Statistical static timing [103, 72, 7, 120, 157, 128, 32, 5, 49, 9, 22, 8, 163, 86, 53, 182, 13, 181, 113, 75, 180, 178, 76, 3, 183, 2, 142, 19, 179, 40, 30, 52, 144, 165, 116, 36, 41, 56, 63, 97, 35, 104, 189, 145, 141, 88, 161, 81, 107, 65, 61, 169, 138, 146, 70, 155, 29, 171, 118, 42] and leakage [26, 111, 21, 37, 176, 140, 109, 15, 156, 33, 95, 62, 73, 44, 119, 151, 188, 20, 133] analysis are hot research topics in recent years. There are two major approaches in

Statistical static timing analysis techniques: path-based [103, 72, 7, 120, 157, 128] and block-based [32, 5, 49, 9, 22, 8, 163, 86, 53, 182, 13, 181, 113, 75, 180, 178, 76, 3, 183, 2, 142, 19, 179, 40, 41, 42] SSTA. Since path-based SSTA is not scalable to large circuit sizes, block-based SSTA is more commonly used. Since Gaussian random variables are easy to be handled, the early block-based SSTA flows [32, 163] modeled the gate delay as linear functions of variation sources and assumed all the variation sources are mutually independent Gaussian random variables. Later, it was observed that the linear delay model is no longer accurate when the scale of variation becomes larger [94] and a higher-order delay model is thus used [180, 178]. Moreover, it was also observed that some variation sources do not follow a Gaussian distribution. For example, the via resistance has an asymmetric distribution [13], while dopant concentration is more suitably modeled as a Poisson distribution [142] rather than Gaussian. The nonlinearity of delay model and non-Gaussian variation sources make SSTA much more difficult. [142] applied independent component analysis to de-correlate the non-Gaussian random variables, but it was still based on a linear delay model. Some recent works [76, 13, 19, 179, 40, 41, 42] considered both non-linear delay model and non-Gaussian variation sources.

Confidence Analysis- In all the statistical modeling and analysis methods discussed above, the statistical characteristics (such as mean and variance) are assumed to be given and reliable. However, when the number of production samples is not large, the production statistical characteristics may significantly deviate from their population values. Therefore, uncertainty in the statistics of measured data as well as production data should be considered in statistical analysis. [177] modeled the uncertainty of mean, variance, and correlation coefficients as an interval, and then estimated the range of mean and variance of circuit performance.

1.2 Contributions of the Dissertation

The contributions of this dissertation are two-fold:

1. Device and architecture concurrent optimization for FPGAs.

2. Statistical timing modeling and analysis.

For device and architecture concurrent optimization for FPGAs, we have the following contributions:

- We first develop an efficient yet accurate power and delay estimator for FPGAs, which we refer to as *Ptrace*. Experimental results show that compared to cycle accurate power simulator and VPR [18], *Ptrace* predicts chip level power and delay within 4% and 5% error, respectively. Based on such framework, we perform device and architecture co-optimization for FPGA circuits. Compared to the baseline, which uses the VPR architecture model [18] with the same LUT size and cluster size as the commercial FPGAs used by Xilinx Virtex-II [170], and the device settings from ITRS roadmap[66], our co-optimization reduces energy-delay product by 18.4% and chip area by 23%. Furthermore, considering FPGA architecture with power-gating capability, our architecture and device co-optimization reduces energy-delay product by 55.0% and chip area by 8.2% compared to the baseline. We also study the impact of utilization rate and interconnect structure.

- We then extend *Ptrace* to estimate power and delay variation of FPGAs. We proposed closed-form models to estimate chip level leakage and timing variations for FPGA. It has been shown that the mean and standard deviation computed by our models are within 3% error compared to Monte-Carlo simulation. With the

extended Ptrace, we perform FPGA device and architecture evaluation considering process variations. Compared to the baseline, device and architecture tuning improves leakage yield by 4.8%, timing yield by 12.4%, and leakage and timing combined yield by 9.2%. We also observe that LUT size of 4 gives the highest leakage yield, LUT size of 7 gives the highest timing yield, but LUT size of 5 achieves the maximum leakage and timing combined yield.

- We further extend (*Ptrace*) to consider process parameters directly so that FPGA circuit and architecture evaluation can be conducted when only the first order process parameters are available. Such evaluation may be used to select circuits and architectures that are less sensitive to process changes or process variations. We call the resulting framework as *Ptrace2*. With *Ptrace2*, we incorporate analytical calculations for two types of FPGA reliability, device aging (*N*egative-*B*ias-*T*emperature-*I*nstability, *NBTI* [11, 149, 160, 23] and *H*ot-*C*arrier-*I*njection, *HCI* [34, 149, 166]) and permanent soft error rate (*SER*) [60], again in the 'from device to chip' fashion. We observe that device aging reduces standard deviation of leakage by 65% over 10 years while it has relatively small impact on delay variation. Moreover, we also find that neither device aging due to NBTI and HCI nor process variation has significant impact on SER.

For statistical timing modeling and analysis, we have the following contributions:

- We introduce an efficient and accurate SSTA flow for non-linear SSTA with non-Gaussian variation sources. All operations in our flow are based on efficient closed-form formulae. Experimental results show that compared to Monte-Carlo simulation, our approach predicts the mean, standard deviation, skewness, and 95-percentile point within 1%, 1%, 6%, and 1% error, respectively.

- Our proposed SSTA flow solves the problem of computing the chip delay vari-

ation, but does not consider modeling of process variation. In order to improve the accuracy and efficiency of SSTA, we presents an accurate model for across-wafer variation. Compared to the exact value, the error of the traditional grid-based spatial variation model is up to 8% while the error of our model is only 2%. Morever, our new model is 6X faster than the traditional model.

- The above SSTA flow also assumes that the statistical variation models are reliable. However, due to limited number of samples (especially in the case of lot-to-lot variation), calibrated models have low degree of confidence. The problem is further exacerbated when low production volumes cause additional loss of confidence in the statistical analysis (since production only sees a small snapshot of the entire distribution). We mathematically derive the confidence intervals for commonly used statistical measures (mean, variance, percentile corner) and analysis (SPICE corner extraction, statistical timing). Our estimations are within 2% of simulated confidence values. Our experiments indicate that for moderate characterization volumes (10 lots) and low-to-medium production volumes (15 lots), a significant guardband is needed (e.g., 34.7% of standard deviation for single parameter corner, 38.7% of standard deviation for SPICE corner, and 52% of standard deviation for 95%-tile point of circuit delay are needed) to ensure 95% confidence in the results. The guardbands are non-negligible for all cases when either production or characterization volume is not large.

1.3 Dissertation Outline

The remaining of this dissertation is organized as follows:

Part I, including Chapter 2, 3, 4, proposes techniques to statistically model and optimize FPGA power and delay.

Chapter 2 first introduces a time efficient trace-based delay and power estimation framework for FPGA circuits, *Ptrace*. With *Ptrace*, we simultaneously optimize device setting (supply voltage V_{dd} and threshold voltage V_{th}) and FPGA architecture (look-up table size K, cluster size N, and interconnect wire segment length W) to minimize power, delay and area.

Chapter 3 extends the architecture and device co-optimization flow in Chapter 2 to considering process variation. We first develop a set of closed-form formulas for chip level leakage and timing variations considering both die-to-die and within-die variations. Based on such model, we extend *Ptrace* to estimate the power and delay variation of FPGAs. Applying extended *Ptrace*, we optimize device setting and FPGA architecture to maximize delay yield, leakage yield, and delay and leakage combine yield.

Chapter 4 proposes a framework to perform concurrent process and architecture optimization for FPGAs in early stage without detailed process modeling. We first implement models to calculate electrical characteristics of advanced CMOS transistors based on the ITRS *MASTAR4* (*M*odel for *A*ssessment of cmo*S* *T*echnologies *A*nd *R*oadmaps) tool [67, 68, 148], and then develop circuit level models of delay, leakage power, and input/output capacitance for FPGA basic circuit elements. With this circuit level model, we further extend *Ptrace* in Chapter 2 and 3 to evaluate chip level power, delay, and reliability. We refer to the new trace-based model as *Ptrace2*. With *Ptrace2*, we can conduct FPGA circuit and architecture evaluation when only the first order process parameters are available. We may also consider reliability issues, such as device aging and permanent soft error rate, in addition to power and delay during evaluation.

Part II, including Chapter 5, Chapter 6, and Chapter 7 introduces statistical timing modeling and analysis.

Chapter 5 presents an efficient and accurate statistical static timing analysis flow for non-linear delay model with non-Gaussian variation sources. We first propose a second order polynomial approximation for the max operation of two random variables. Based on this approximation, we develop an SSTA flow for three different delay models: quadratic delay model, quadratic delay model without crossing terms (*semi-quadratic model*), and linear delay model.

Chapter 6 studies another key component of SSTA: the modeling of process variation. We analyze the impact of deterministic across-wafer variation and find that the spatially correlated within-die variation is mainly caused by deterministic across wafer variation. Then, we propose an efficient and accurate die-level variation model to model the across-wafer variation. We apply our proposed variation model to the SSTA flow in Chapter 5 to verify its efficiency and accuracy.

Chapter 7 studies the confidence interval of statistical characteristics, such as mean and variance, of statistical timing analysis. We estimates the confidence interval for a single variation sources, for the fast/slow corner of an inverter chain, and for full chip timing analysis.

Finally, Chapter 8 concludes this dissertation with discussion of the ongoing work and future work as well.

In Chapter 9, I briefly summarize the works I have finished in my Ph.D. program which are not included in this dissertation.

Part I

Statistical Modeling and Optimization for FPGA Circuits

CHAPTER 2

Deterministic Device and Architecture Co-Optimization for FPGA Power, Delay, and Area Minimization

Device optimization considering supply voltage V_{dd} and threshold voltage V_{th} has little chip area increase, but a great impact on power and performance in the nanometer technology. This chapter studies simultaneous evaluation of device and architecture optimization for FPGAs. We first develop an efficient yet accurate timing and power evaluation method, called trace-based model. By collecting trace information from cycle-accurate simulation of placed and routed FPGA benchmark circuits and re-using the trace for different V_{dd} and V_{th}, we enable device and architecture co-optimization considering hundreds of device and architecture combinations. Compared to the baseline FPGA architecture, which uses the VPR architecture model and the same LUT and cluster sizes as those used by the Xilinx Virtex-II, V_{dd} suggested by ITRS, and V_{th} optimized with respect to the above architecture and V_{dd}, architecture and device co-optimization can reduce energy-delay product by 20.5% and chip area by 23.3%. Furthermore, considering power-gating of unused logic blocks and interconnect switches (in this case sleep transistor size is a parameter of device tuning), our co-optimization reduces energy-delay product by 55.0% and chip area by 8.2% compared to the baseline FPGA architecture.

2.1 Introduction

FPGAs allow 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 (typically 62.5% [158]), existing FPGAs consume more power compared to ASICs [83]. As the process advances to nanometer technologies and low-energy embedded applications are explored for FPGAs, power consumption becomes a crucial design constraint for FPGAs.

Recent work has studied FPGA power modeling and optimization. The leakage power of a commercial FPGA architecture was quantified [158], and a high level FPGA power estimation methodology was presented [48]. Power evaluation frameworks were introduced for generic parameterized FPGA [123, 89, 92] and it was shown that both interconnect and leakage power are significant for FPGAs in nanometer technologies. As to power optimization, the interaction of a suit of power-aware FPGA CAD algorithms without changing the existing FPGAs was studied in [85]. Power-driven partition algorithm for mapping applications to FPGAs with different V_{dd}-levels [117] was proposed. A configuration inversion method to reduce the leakage power of multiplexers without any additional hardware [12] was investigated. Besides the power optimization CAD algorithms, low power FPGA circuits and architectures have also been studied. Region based power gating for FPGA logic blocks [59] and fine-grained power-gating for FPGA interconnects [102] were proposed, and V_{dd} programmability was applied to both FPGA logic blocks [93, 90, 98, 91] and interconnects [58, 51, 71]. A new type of routing multiplexer and an input control method were developed [150] to reduce leakage of unused routing multiplexers, and a circuitry combining gate biasing, body biasing and multi-threshold techniques was designed [110] to reduce interconnect leakage. Besides leakage power reduction, a glitch minimization technique [84]

was proposed to reduce dynamic power.

Architecture evaluation has been performed first for area and delay. For non-clustered FPGAs, it was shown that an LUT size of 4 achieves the smallest area [130] and an LUT size of 5 or 6 leads to the best performance [143]. Later on, the cluster-based island style FPGA was studied to optimize area-delay product and it showed that LUT sizes ranging from 4 to 6 and cluster sizes between 4 and 10 can produce the best area-delay product [10]. Besides area and delay, FPGA architecture evaluation considering energy was studied recently [93, 123, 92]. It was shown that in $0.35\mu m$ technology, an LUT size of 3 consumes the smallest energy [123]. In $100nm$ technology, an LUT size of 4 consumes the smallest energy [92]. [101] further evaluated FPGA architectures with field programmable dual-V_{dd} and power gating, and considering area, delay, and energy.

However, all the aforementioned architecture evaluation assumed fixed supply voltage V_{dd} and threshold voltage V_{th}, and sleep transistor size (if power gating is applied), and have not conducted simultaneous evaluation of device optimization such as V_{dd} and V_{th} tuning and architecture optimization such as tuning LUT and cluster sizes. Architecture and device co-optimization may obtain a better power and performance tradeoff compared to architecture tuning alone. We define *hyper-architecture* as the combination of device and architectural parameters. The co-optimization requires the exploration of the following dimensions: cluster size N, LUT size K, V_{dd}, V_{th}, and sleep transistor size if power gating is applied. The total number of hyper-architecture combinations can be easily over a few hundreds considering the interaction between these dimensions. In the existing power evaluation frameworks, such as cycle-accurate simulation [89] and transition density based estimation [123], timing and power are calculated for each circuit element. Therefore, it is time-consuming to explore the huge hyper-architecture solution space using methods from [89, 123]. In order to perform

device and architecture co-optimization, an accurate yet extremely efficient timing and power evaluation method is required.

The first contribution of this chapter is that we develop a trace-based estimator (called *Ptrace*) for FPGA power, delay, and area. We profile placed and routed benchmark circuits and collect statistical information (called trace) on switching activity, short circuit power, near-critical path structure, and circuit element utilization rate for a given set of benchmark circuits (MCNC [175] benchmark set in this chapter). We show that the trace is independent of device tuning and it can be used to calculate FPGA chip-level performance and power for a number of device designs. Compared to performing placement-and-routing by VPR [18] followed by cycle-accurate simulation (called *Psim*) from [89], Ptrace has a high fidelity and an average error of 1.3% for energy and of 0.8% for delay. The trace collecting has the same runtime as evaluating FPGA architecture for one combination of V_{dd}, V_{th} and sleep transistor size using VPR and Psim. It took one week to collect the trace for the MCNC benchmark set using eight 1.2GHz Intel Xeon servers while all the hyper-architecture evaluation reported in this chapter with over hundreds of hyper-architecture combinations took only a few minutes on one server.

The second contribution is that we perform the architecture and device co-optimization for conventional FPGAs and FPGAs with power gating capability. We explore different V_{dd}, V_{th}, and sleep transistor size combinations in addition to cluster size and LUT size combinations. For comparison, we obtain the baseline FPGA hyper-architecture which uses the VPR architecture model [18] and the same LUT size and cluster size as the commercial FPGAs used by Xilinx Virtex-II [170], and V_{dd} suggested by ITRS [66], but V_{th} optimized by our device optimization. Such baseline is significantly better than the ones with no device optimization. Compared to the baseline hyper-architecture, architecture and device co-optimization can reduce energy-delay product

(product of energy per clock cycle and critical path delay, in short, ED) by 18.4% and chip area by 23.3%. Furthermore, considering FPGA architecture with power-gating capability, our architecture and device co-optimization reduces ED by 55.0% and chip area by 8.2% compared to the baseline. We also study the impact of utilization rate and interconnect structure.

The rest of the chapter is organized as follows. Section 2.2 presents the background of FPGA architecture and existing FPGA architecture evaluation flow. Section 2.3 introduces our trace-based estimation models. Section 2.4 applies the new estimation models to architecture and device co-optimization. Section 2.5 concludes this chapter.

2.2 Preliminaries

2.2.1 Conventional FPGA Architecture

We assume cluster-based island style FPGA architecture such as that in [18] for all classes of FPGAs studied in this chapter. A cluster-based logic block (see Figure 2.1) includes N fully connected Basic Logic Elements (BLEs). Each BLE includes one K-input lookup table (LUT) and one flip-flop (DFF). The combination of cluster size N and LUT size K is the architectural issue we evaluate in this chapter. 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 2.1 (b)). A routing *switch block* is located at the intersection of a horizontal channel and a vertical channel. Figure 2.1 (c) shows a subset switch block [87], where the incoming track can be connected to the outgoing tracks with the same track number.[1] The connections in a switch block (represented by the dashed lines in Figure 2.1 (c)) are programmable routing switches. We implement

[1] Without loss of generality, we assume subset switch block in this chapter.

routing switches by tri-state buffers and use two tri-state buffers for each connection so that it can be programmed independently for either direction. We define an *interconnect segment* as a wire segment driven by a tri-state buffer or a buffer.[2] In this chapter, we investigate both uniform length-4 interconnect wire segments and mixed-length interconnects. We decide the routing channel width CW in the same way as the architecture study in [18], i.e., $CW = 1.2CW_{min}$ where CW_{min} is the minimum channel width required to route the given circuit successfully.

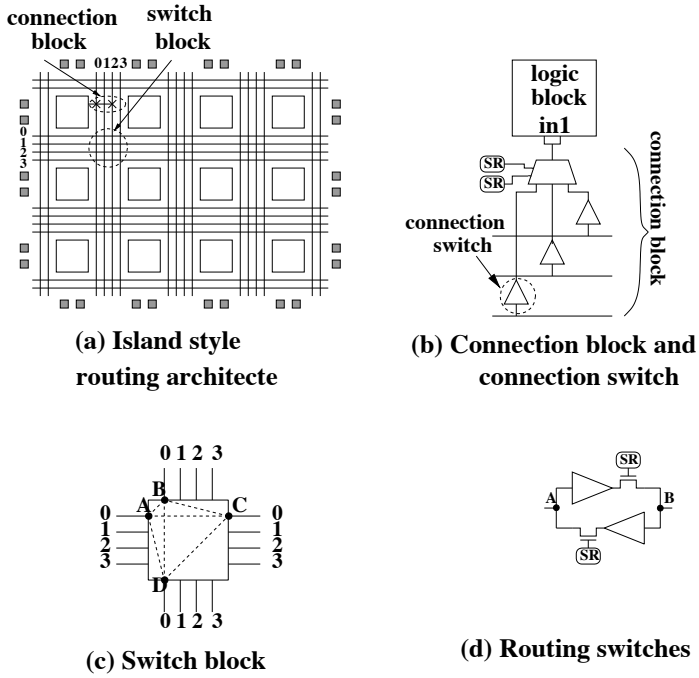


Figure 2.1: (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches.

[2]We interchangeably use the terms of switch and buffer/tri-state buffer.

2.2.2 V_{dd} Gatable FPGA Architecture

Power gating can be applied to interconnects and logic blocks to reduce FPGA power. Figure 2.2 illustrates the circuit design of the V_{dd}-gateable interconnect switch and logic block from [101]. We insert a PMOS transistor (called a sleep transistor) 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, the 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 a logic block by a factor of over 300.

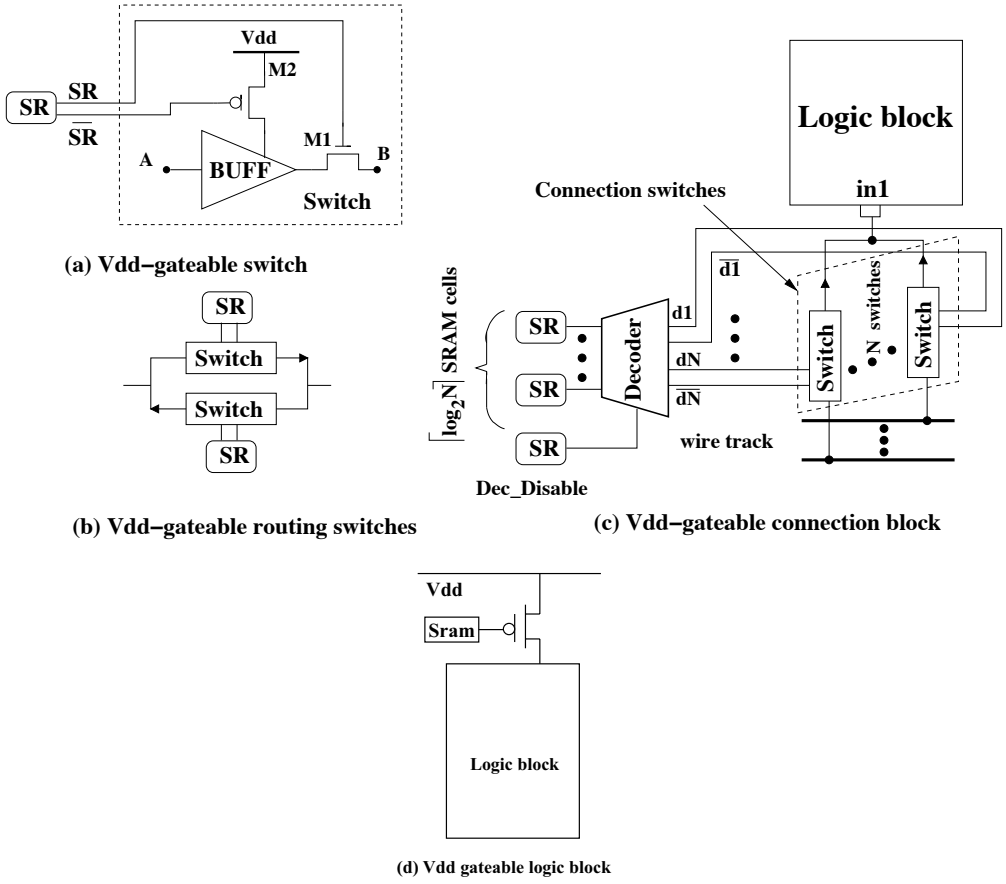


Figure 2.2: (a) V_{dd}-gateable switch; (b) V_{dd}-gateable routing switch; (c) V_{dd}-gateable connection block; (d) V_{dd}-gateable logic block.

There is power and delay overhead associated with the sleep transistor insertion. The dynamic power overhead is almost negligible. This is due to the fact that the sleep transistors 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, the connection box with power gating is different from that in the conventional architecture. For the conventional FPGA, an input MUX, which is effectively an implicit decoder [18], is used in the connection box as illustrated in Figure 2.1(b). However, for the power gating architecture, an explicit decoder is used in the connection box to enable both the signal path and power supply for the single input, as illustrated in Figure 2.2(c) [101]. Because the decoder is no longer in the critical path, the delay of connection box in Figure 2.2(c) is smaller than that in Figure 2.1(b). The connection box in Figure 2.2(c) also has a much smaller leakage power but a larger area and a slightly larger dynamic power compared to the conventional architecture.

2.2.3 FPGA Architecture Evaluation Flow

The existing FPGA architecture evaluation flow considering area, delay, and power [92] is illustrated in Figure 2.3. For a given benchmark set, we first optimized the logic then map the circuit to a given LUT size. TV-Pack is used to pack the mapped circuit to a given cluster size. After packing, we place and route the circuit using VPR [18] and obtain the chip level delay and area. Finally, cycle-accurate power simulator [89] (in short *Psim*) is used to estimate the chip level power consumption.

The architecture evaluation flow discussed above is time consuming because we need to place and route every circuit under different architectures and a large number of randomly generated input vectors need to be simulated for each circuit. However, the device and architecture co-optimization requires the exploration of the following

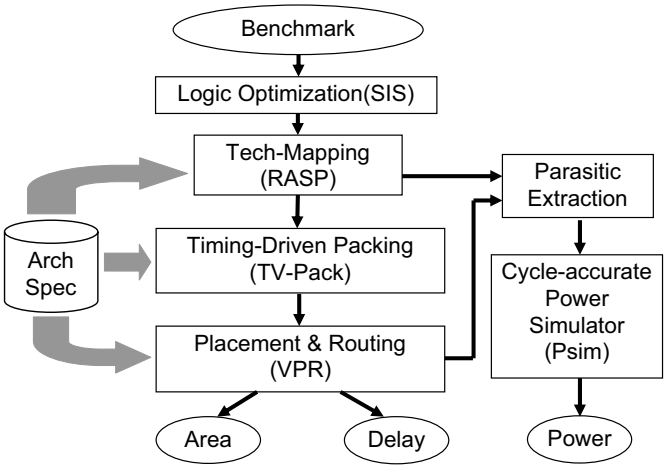


Figure 2.3: Existing FPGA architecture evaluation flow for a given device setting.

dimensions: cluster size N, LUT size K, supply voltage V_{dd}, threshold voltage V_{th}, and sleep transistor size if power gating is applied. The total number of hyper-architecture combinations can be easily over a few hundreds considering the interaction between these dimensions. Therefore, the conventional evaluation flow is not practical for device and architecture co-optimization due to its time inefficiency. In order to perform device and architecture co-optimization, a fast yet accurate FPGA power and delay estimator is required. Such estimator is just the one we are going to introduce in Section 2.3, the trace-based estimator.

2.3 Trace-Based Estimation

In this section, we will introduce the efficient power and delay estimation model, trace-based estimation method (Ptrace). We speculate that during hyper-architecture evaluation, there are following two classes of information as summarized in Table 2.1. The first class only depends on architecture and is called trace of the architecture. The second class only depends on device setting and circuit design. The basic idea of Ptrace is

as follows: For a given benchmark set, we profile placed and routed benchmark circuits and collect trace information under one device setting and for one FPGA architecture. We then obtain chip-level performance and power for a set of device for a given architecture parameter values based on the trace information. Figure 2.4 illustrates the flow of trace-based evaluation.

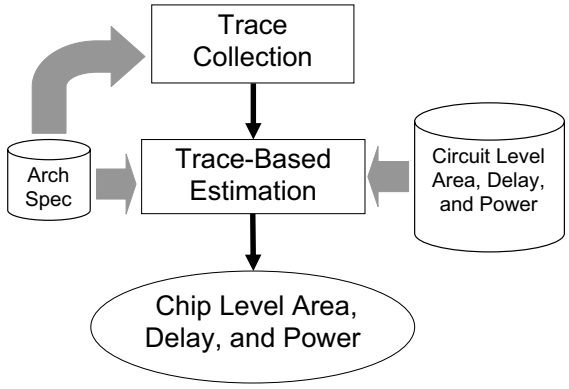


Figure 2.4: New trace-based evaluation flow. We perform the same flow as Figure 2.3 under one device setting to collect the trace information.

2.3.1 Trace Collection

As mentioned before, the trace information only depends on architecture and remains the same when device setting changes. For a given benchmark set and a given FPGA architecture, the trace includes number of used type i circuit elements (N_i^u), total number of type i circuit elements (N_i^t), average switching activity of used type i circuit elements (S_i^u), short circuit power ratio (α_{sc}), and near-critical path structure (see Table 2.1). Near-critical path structure is the number of each type of circuit elements (N_i^p) on the near-critical path. Device parameters include V_{dd} and V_t, which depend on technology scale, and average leakage power of type i circuit elements (P_i^s), average load capacitance of type i circuit elements (C_i^u), and average delay of type i circuit

elements (D_i), which depend on circuit design. The device parameters, such as circuit level delay and power, can be collected by SPICE simulation or by measurement. After collecting the trace and device level delay and power, we can perform Ptrace to estimate the chip level delay, power, and area. Below, we will show that the trace information is insensitive to the device parameters and discuss our trace-based models.

Trace Parameters (depend on architecture)	
N_i^u	# of *used* type i circuit elements
N_i^t	total # of type i circuit elements
S_i^u	avg. switching activity for *used* type i circuit elements
N_i^p	# of type i circuit elements on the near-critical path
α_{sc}	ratio between short circuit power and switch power
Device Parameters (depend on processing technology and circuit design)	
V_{dd}	power supply voltage
V_{th}	threshold voltage
P_i^s	avg. leakage power for type i circuit elements
C_i^u	avg. load capacitance of type i circuit elements
D_i	avg. delay of type i circuit elements

Table 2.1: Trace information, device and circuit parameters.

2.3.2 Power and Delay Model

2.3.2.1 Dynamic Power Model

Dynamic power includes switch power and short-circuit power. A circuit implemented on an FPGA cannot utilize all circuit elements. Dynamic power is only consumed by the used FPGA resources. Our trace-based switch power model distinguishes different types of used FPGA resources and applies the following formula:

$$P_{sw} = \sum_i \frac{1}{2} N_i^u \cdot f \cdot V_{dd}^2 \cdot C_i^{sw} \tag{2.1}$$

The summation is over different types of circuit elements, i.e., LUTs, buffers, input pins and output pins. For type i circuit elements, C_i^{sw} is the average switch capacitance

and N_i^u is the number of used circuit elements, f is the operating frequency. In this chapter, we assume that the circuit works at its maximum frequency, i.e., the reciprocal of the near-critical path delay. The switch capacitance is further calculated as:

$$\begin{aligned} C_i^{sw} &= (\sum_{j \in El_i} C_{i,j}/N_i^u) \cdot S_i^u \\ &= C_i^u \cdot S_i^u \end{aligned} \tag{2.2}$$

For type i circuit elements, C_i^u is the average load capacitance of used circuit elements, which is averaged 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 V_dd and V_{th} have a limited effect on switching activity. We verify this assumption in Table 2.2 by demonstrating the average switching activity of five benchmarks at different technology nodes, and V_dd and V_{th} levels.

benchmark	70nm V_dd=1.1 V_{th}=0.25		100nm V_dd=1.3 V_{th}=0.32		70nm V_dd=1.0 V_{th}=0.20	
	logic	inter-connect	logic	inter-connect	logic	inter-connect
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 2.2: switching activities for different technology nodes, V_dd and V_{th}. Architecture setting: $N = 10, K = 4$. Unit: switch per clock cycle.

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

$$P_{sc} = P_{sw} \cdot \alpha_{sc} \tag{2.3}$$

Where α_{sc} is the ratio between short circuit power and switch power. Although this ratio value at the circuit level depends on FPGA circuit design and architecture, we assume that α_{sc} does not depend on device and technology at the chip level. We verify this assumption in Table 2.3[3] by showing the average short circuit power ratio at different technology nodes, $V_d d$, and V_{th} levels.

benchmark	70nm $V_d d$=1.1 V_{th}=0.25		100nm $V_d d$=1.3 V_{th}=0.32		70nm $V_d d$=1.0 V_{th}=0.20	
	logic	inter-connect	logic	inter-connect	logic	inter-connect
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 2.3: Short circuit power ratios for different technology nodes, $V_d d$ and V_{th}. Architecture setting: $N = 10, K = 4$.

2.3.2.2 Leakage Power Model

The leakage power is modeled as follows,

$$P_{static} = \sum_i N_i^t P_i^s \tag{2.4}$$

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 (typically 62.5% [158]). For an FPGA architecture with power-gating capability, an unused circuit element can be power-gated to reduce leakage

[3]Because the delay between buffers is usually large for FPGA circuits, the short circuit power ratio could be big.

power.[4] In this case, the total leakage power is modeled by the following formula:

$$P_{static} = \sum_i N_i^u P_i + \alpha_{gating} \cdot \sum_i (N_i^t - N_i^u) P_i \tag{2.5}$$

where α_{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 chapter.

2.3.2.3 Delay Model

To avoid the static timing analysis for the entire circuit implemented on a given FPGA fabric, we obtain the structure of the ten longest circuit paths including the near-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 near-critical path due to different V_dd and V_{th} levels is among these ten longest paths found by our benchmark profiling. When V_dd and V_{th} change, we can calculate delay values for the ten longest paths under new V_dd and V_{th} levels, and choose the largest one as the new near-critical path delay. Therefore, the FPGA delay can be calculated as follows:

$$D = \sum_i N_i^p D_i \tag{2.6}$$

For resource type i, N_i^p is the number of circuit elements that the near-critical path goes through, and D_i is the delay of such a circuit element. D_i is a circuit parameter depending on V_dd, V_{th}, process technology, and FPGA architecture.[5] To get the path

[4] In this chapter, we assume the wire segment length change introduce by power gating can be ignored. In our experiment, wire length is calculated as $4 \times \sqrt{area_{tile}}$ [18], where $area_{tile}$ is the area of a logic block with the interconnect surrounding it. The area overhead introduced by power gating is less than 30%. Therefore, the change of wire segment length is less than 14%. Moreover the load capacitance of a routing buffer is mainly determined by the input and output capacitances of buffers. The wire capacitance is a small portion (about 20%) of the load capacitance. Therefore, the area increase introduced by power gating will have small impact on the delay and power estimation.

[5] Delay of each circuit element is measured with the worst case switch. When power gating is applied, a single gating transistor is used for the entire logic block. In order to measure the delay of

statistical information N_i^p, we only need to place and route the circuit *once* for a given FPGA architecture.

2.3.3 Validation of Ptrace

2.3.3.1 Verification of Deterministic Model

To validate Ptrace, we compare it to the conventional architecture evaluation flow for ITRS [66] 70nm technologies. We assume V_dd=1.0V and V_{th}=0.2V and map 20 MCNC benchmarks, as illustrated in Table 2.4, to two architectures: $\{N = 8, K = 4\}$ and $\{N = 6, K = 7\}$. In Table 2.4, the number of LUTs and flip flops are counted under the architecture *N=8* and *K=4*. Notice that we can map the benchmarks to different LUT and cluster sizes. We collect trace using the conventional architecture evaluation flow in 70nm technology, V_dd=1.0V and V_{th}=0.2V. Figure 2.5 compares energy and delay between the conventional evaluation flow 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 trends for energy as Psim does and for delay as VPR does. Therefore, Ptrace has a high fidelity. Moreover, the run time of Ptrace is 2*s*, while that of the conventional evaluation flow is 120 hours.

each circuit element in a logic block with gating transistor, a series of randomly generated vectors are used as the inputs to the block and the worst case delay is measured for each circuit element under such random input vectors.

benchmark	# LUTs	# Flip flops	benchmark	# LUTs	# Flip flops
alu4	1607	0	ex5p	1201	0
apex2	2118	0	frisc	5926	900
apex4	1291	0	misex3	1501	0
bigkey	2935	224	pdc	5608	0
clma	13537	33	s298	2548	8
des	2172	0	s38417	8458	1463
diffeq	1933	377	s38584	7040	1260
dsip	1619	224	seq	1953	0
elliptic	4185	1138	spla	3938	0
ex1010	4721	0	tseng	1299	382

Table 2.4: MCNC benchmark list. The number of LUTs and flip flops are counted under the architecture *N=8* and *K=4*.

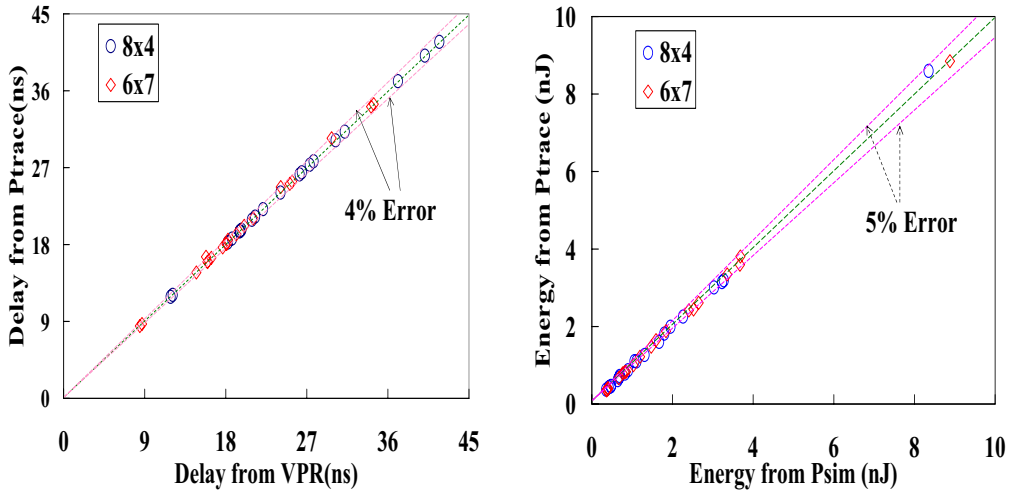


Figure 2.5: Comparison between Psim and Ptrace.

2.4 Hyper-Architecture Evaluation

2.4.1 Overview

In this section, we use Ptrace to perform device and architecture evaluation. We consider 70nm ITRS technology and evaluate four FPGA hyper-architecture classes: *Homo-*V_{th}, *Hetero-*V_{th}, *Homo-*V_{th}*+G* and *Hetero-*V_{th}*+G*. *Homo-*V_{th} is the conventional FPGA using homogeneous V_{th} for interconnects and logic blocks. *Hetero-*V_{th} applies different V_{th} to logic blocks and interconnects. *Homo-*V_{th}*+G* and *Hetero-*V_{th}*+G* are the same as *Homo-*V_{th} and *Hetero-*V_{th}, respectively, except that unused logic blocks and interconnects are power-gated [101]. We compare them with the baseline hyper-architecture, which uses the VPR architecture model [18] and has the same LUT size and cluster size as those used by the Xilinx Virtex-II [170] (cluster size of 8, LUT size of 4), V_{dd} suggested by ITRS [66] (0.9v), and V_{th} of 0.3v that is optimized with respect to the above architecture and V_{dd}. The baseline hyper-architecture and evaluation ranges for device and architecture are presented in Table 2.5. Note that a high V_{th} is applied to all SRAM cells for configuration to reduce their leakage power as suggested by [93].

In the subsections 2.4.2 to 2.4.4, we assume the following:

- The utilization rate (defined as the utilization rate of logic blocks, i.e., number of used logic blocks over the number of total available logic blocks) is 0.5.
- All interconnect wire segments span 4 logic blocks with fully buffered routing switches.
- The routing channel width is 1.2 times of the minimum channel width that allows the FPGA circuit being routed.
- All benchmark circuits work at their highest frequency (1/critical path delay).

The impact of utilization rate and interconnect architecture will be discussed in subsections 2.4.5 and 2.4.6, respectively. For each hyper-architecture, we compute the energy, delay and area as the geometric mean of 20 MCNC benchmarks in Table 2.4. Moreover, in the rest of this chapter, we use CV_t for V_{th} of logic blocks and IV_{th} for V_{th} for global interconnects. Note that $CV_t = IV_{th}$ in *Homo-*V_{th} and *Homo-*V_{th}*+G*. To illustrate the tradeoff between energy and delay, we introduce the concept of *dominant hyper-architecture*: If hyper-architecture *A* has less energy consumption and a smaller delay than hyper-architecture *B*, then we say that *B* is inferior to *A*. We define the *dominant hyper-architectures* as the set of hyper-architectures that are not inferior to any other hyper-architectures.

We organize the rest of this section as follows: First, Section 2.4.2 illustrates the necessity of device and architecture co-optimization. Then Section 2.4.3 presents the energy and delay tradeoff and ED reduction achieved by device and architecture co-optimization. Section 2.4.4 discusses the device and architecture co-optimization considering area. Finally, Sections 2.4.5 and 2.4.6 analyze the impact of utilization rate and compare the evaluation result of different routing architectures, respectively.

Baseline FPGA device/arch parameter values			
V_{dd}	V_{th}	N	K
0.9v	0.3v	8	4
Value range for device/arch optimization			
V_{dd}	V_{th}	N	K
0.8v-1.1v	0.2v-0.4v	6-12	3-7

Table 2.5: Baseline hyper-architecture and evaluation ranges.

2.4.2 Necessity of Device and Architecture Co-Optimization

In this section, we show the necessity of device and architecture co-optimization. We first discuss the need of device tuning, then compare the results of optimizing device

and architecture separately and simultaneously.

Architecture evaluation has been studied in previous research [131, 80, 10, 89, 123, 101]. However, device tuning has not been reported in the literature. Our experiments show that device tuning has a much greater impact on delay and energy than architecture tuning does, which is demonstrated in Figure 2.6 and Table 2.6. Each set of data points in Figure 2.6 is the dominant hyper-architectures for a given device setting.[6] For example, set D4 is the dominant hyper-architectures under V_{dd}=1.0V and V_{th}=0.25V. From the figure, we observe that a change on the device leads to a more significant change in energy and delay than architecture change does. For example, for device setting $V_{dd} = 0.9V$ and V_{th}=0.25v, energy for different architectures ranges from 1.82nJ to 2.11nJ, and delay ranges from 13.68ns to 16.46ns. However, if we increase V_{th} by 0.05v, i.e., $V_{dd} = 0.9V$ and $V_{th} = 0.3V$, the energy ranges from 1.17nJ to 1.32nJ and the delay ranges from 18.99ns to 23.24ns. Therefore, it is important to evaluate both device and architecture instead of evaluating architecture only.

Set	V_{dd} (V)	V_{th} (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 2.6: Power and delay ranges for different device settings.

There are three methods to perform device and architecture optimization. In the first method, we first optimize device using one architecture then optimize the ar-

[6]Dominant hyper-architectures for a given device setting are the hyper-architectures that are not inferior to any other hyper-architectures under such device setting.

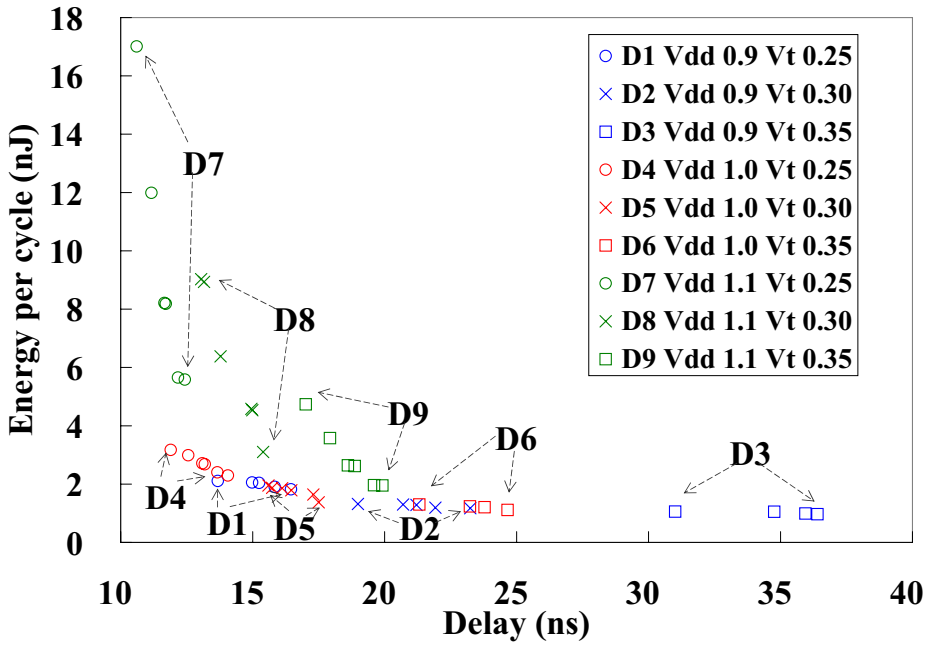


Figure 2.6: hyper-architectures under different device settings.

chitecture under the optimized device setting and call it *device-arch* method. In the second method, we first optimize the architecture within one device setting then optimize the device setting according to the optimized architecture and call it *arch-device* method. In the third method, we optimize architecture and device simultaneously and call it *simultaneous* method. Both methods *arch-device* and *device-arch* cannot guarantee the optimal solution. Table 2.7 compares the min-ED hyper-architectures for *Homo-*V_{th} found by three different methods. For both *arch-device* and *device-arch*, we start search from the baseline case $\{N = 8, K = 4, V_{dd} = 0.9V$, and $V_{th} = 0.3V\}$. In our experiment, we find that there are two local optimal hyper-architectures in the whole solution space: $\{N = 6, K = 7, V_{dd} = 0.9V, V_{th} = 0.3V\}$ and $\{N = 10, K = 4, V_{dd} = 1.0V, V_{th} = 0.3V\}$ which is also the global optimal. In this particular example, both *arch-device* and *device-arch* achieve the same local optimal hyper-architecture $\{N = 6, K = 7, V_{dd} = 0.9V, V_{th} = 0.3V\}$. If we start search from some other hyper-

architectures, *arch-device* and *device-arch* may achieve other local optimum but there is no guarantee of the global optimum. In order to obtain the global optimal solution, we have to perform *simultaneous* method. From Table 2.7, we observe that *simultaneous* method can reduce ED by 13.3% compared to *arch-device* and *device-arch*. We also see that the runtime of *arch-device* and *device-arch* is shorter than that of *simultaneous* method. However, due to the time efficiency of Ptrace, the runtime of *simultaneous* method is also small (only 34.1s). Therefore, it is still worthwhile performing simultaneous device and architecture optimization.

	V_{dd} (V)	CV_t (V)	IV_{th} (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	runtime (s)
arch-device	0.9	0.30	0.30	(6,7)	1.38	19.8	27.3	5.1
device-arch	0.9	0.30	0.30	(6,7)	1.38	19.8	27.3	3.4
simultaneous	1.0	0.30	0.30	(10,4)	1.37	17.5	24.1 (-13.3%)	34.1

Table 2.7: Min-ED hyper-architecture of optimizing device and architecture separately and simultaneously.

2.4.3 Energy and Delay Tradeoff

In this section, we first compare the impact of device tuning and architecture tuning, then present min-energy and min-delay hyper-architectures, and finally discuss the energy and delay tradeoff. For the classes *Homo-*V_{th}*+G* and *Hetero-*V_{th}*+G* with power-gating, 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 then discuss the sleep transistor tuning in Section 2.4.4.

Table 2.8 summarizes the minimum delay and minimum energy hyper-architectures for each class. The minimum delay hyper-architectures have cluster size of 6 and LUT size of 7 which are the same for all classes. The minimum energy hyper-architectures have LUT size of 4 for all classes. This is similar to the previous evaluation result [101, 89]. As expected, the min-delay hyper-architectures have the highest V_{dd} and

lowest V_{th}. However, the min-energy hyper-architectures have the lowest V_{dd} but not the highest V_{th}. This is because we assume that each circuit works at its highest possible frequency (1/critical path delay). The energy is calculated as $E = delay \cdot power$. When V_{th} is too high, the delay is so large that the energy per clock cycle increases.

Hyper-	Minimum delay hyper-architecture						Minimum energy hyper-architecture					
Arch Class	V_{dd} (V)	CV_t (V)	IV_{th} (V)	(N,K)	E (nJ)	D (ns)	V_{dd} (V)	CV_t (V)	IV_{th} (V)	(N,K)	E (nJ)	D (ns)
Homo-V_{th}	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-V_{th}	1.1	0.20	0.20	(6,7)	31.22	8.86	0.8	0.30	0.35	(12,4)	0.920	43.6
Homo-V_{th}+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-V_{th}+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 2.8: Minimum delay and minimum energy hyper-architectures.

Usually, higher performance hyper-architectures consume more energy. To illustrate the energy and delay tradeoff, we present the dominant hyper-architectures of *Homo-V_{th}* and *Hetero-V_{th}* in Figure 2.7 (a) and those of *Homo-V_{th}+G* and *Hetero-V_{th}+G* in Figure 2.7 (b). From the figure, we find that the energy difference between *Homo-V_{th}+G* and *Hetero-V_{th}+G* is smaller than that between *Homo-V_{th}* and *Hetero-V_{th}*. This is because leakage power is significantly reduced by power-gating and therefore more detailed V_{th} tuning such as heterogeneous-V_{th} has a smaller impact. We also see that dominant hyper-architectures of *Homo-V_{th}+G* and *Hetero-V_{th}+G* have smaller delay than that of *Homo-V_{th}* and *Hetero-V_{th}*. This is due to the fact that the connection box with power gating has smaller delay than that without power gating as discussed in Section 2.2.2. Moreover, with the dominant hyper-architecture 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-V_{th}* with delay limit 15ns, we only need to pick the dominant hyper-architecture whose delay is closest to 15ns.

In order to achieve the best energy and delay tradeoff, we find the hyper-architectures with the *minimum energy delay product* (in short *min-ED*) in Table 2.9. Compared to

the baseline, ED reduction is 14.5% and 18.4% for *Homo-V_{th}* and *Hetero-V_{th}*, respectively. If power gating is applied, ED can be reduced by about 60% for both *Homo-V_{th}+G* and *Hetero-V_{th}+G*. The similar ED reduction for power gating classes is due to the fact that leakage power is greatly reduced by power-gating and therefore the more detailed V_{th} tuning such as heterogeneous-V_{th} has a small impact on power reduction as discussed before. We also see that, compared to the min-ED hyper-architectures without power gating, the min-ED hyper-architectures with power gating has a lower V_{th}. This is because leakage power is greatly reduced when power gating is applied, therefore a lower V_{th} can improve performance without much penalty on leakage.

hyper-architecture.Class	V_{dd} (V)	CV_t (V)	IV_{th} (V)	(N, k)	E(nJ)	D (ns)	ED (nJ· ns)	Area %
Baseline	0.9	0.30	0.30	(8,4)	1.20	23.5	28.2	100.00
Homo-V_{th}	1.0	0.30	0.30	(10,4)	1.37	17.5	24.1 (-14.5%)	81.90
Hetero-V_{th}	0.9	0.25	0.30	(12,4)	1.27	18.1	23.0 (-18.4%)	79.52
Homo-V_{th}+G	0.9	0.25	0.25	(12,4)	0.74	16.10	11.9 (-57.8%)	127.43
Hetero-V_{th}+G	0.8	0.25	0.20	(10,4)	0.65	17.30	11.2 (-60.3%)	126.19

Table 2.9: Comparison between baseline and min-ED hyper-architecture in *Homo-V_{th}*, *Hetero-V_{th}*, *Homo-V_{th}+G*, and *Hetero-V_{th}+G*.

2.4.4 ED and Area Tradeoff

In the previous sections, we assume fixed sleep transistor sizes for *Homo-V_{th}* and *Hetero-V_{th}+G* and discuss hyper-architecture 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 is. In this section, we perform device and architecture co-optimization to achieve the best ED and area tradeoff. Although dual-V_{th} may change the 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 V_{dd} and V_{th} change does not affect area.

For *Homo-V_{th}* and *Hetero-V_{th}*, 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-architectures with minimum *product of energy, delay, and area* (in short *AED*), which are summarized in Table 2.10. Compared to the baseline, the min-AED hyper-architecture of *Homo-V_{th}* reduces ED by 14.7% and area by 18.3%,[7] and the min-AED hyper-architecture of *Hetero-V_{th}* reduces ED by 18.4% and area by 23.3%. Figure 2.8 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-architecture is exactly the min-AED hyper-architecture. This is because that when no power gating is applied, the larger the area, the more leakage power is consumed. Therefore, the min-ED hyper-architectures use less area than other hyper-architectures.

	V_{dd} (V)	CV_t (V)	IV_{th} (V)	(N,K)	S	ED	Area	AED	AED reduction %
Baseline	0.9	0.30	0.30	(8,4)	-	1	1	-	-
Homo-V_{th}	1.0	0.30	0.30	(10,4)	-	0.853	0.817	0.697	30.3
Hetero-V_{th}	0.9	0.25	0.30	(12,4)	-	0.816	0.767	0.626	37.4
Homo-V_{th}+G	0.9	0.25	0.25	(12,4)	2	0.470	0.918	0.432	56.9
Hetero-V_{th}+G	0.9	0.25	0.20	(12,4)	2	0.450	0.918	0.413	58.7

Table 2.10: Minimum ED-area product hyper-architectures for different classes. ED, Area, and ED-area product are normalized with respect to the baseline.

For *Homo-V_{th}+G* and *Hetero-V_{th}+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 2.2(d), we assume the 210X PMOS for the sleep transistor with negligible area overhead. Moreover, we observe that a 1X PMOS as the sleep

[7] In our evluation, we assume that the area depends only on architecture and but not device setting. Our experimental result shows that $N = 12$, $K = 4$ is most area efficient architecture, this architecture reduces area by 23.6% compared to the baseline ($N = 8$, $K = 4$). The architecture $N = 10$, $K = 4$ is the second area efficient architecture which reduces area by 18.3% compared to the baseline.

transistor for one switch in connection box (see the circuit in Figure 2.2(c)) provides good performance, and any further increase of the sleep transistor size cannot improve the performance much.

The sleep transistors for the switches in the routing box, however, may affect delay greatly. We consider four sleep transistor sizes: 2X, 4X, 7X, and 10X PMOS for a routing switch. From the Figure 2.8, we find that device and architecture co-optimization can reduce ED and area simultaneously even when power gating is applied. This is due to the fact that tuning device setting and architecture offers a bigger solution space to explore in chip level. [8] Table 2.10 summarizes the minimum AED product hyper-architectures. Compared to the baseline case, the minimum AED product hyper-architecture of *Homo-V_{th}+G* reduces ED by 53.0% and area by 8.2% and the minimum AED product hyper-architecture in *Hetero-V_{th}+G* reduces ED by 55.0% and area by 8.2%.

2.4.5 Impact of Utilization Rate

In the previous part of this section, we assume fixed utilization rate (0.5). In this subsection, we will further discuss the impact of utilization rate on FPGA architecture evaluation. We compare the min-ED and min-AED hyper-architectures under three different utilization rates: 0.3, 0.5, and 0.8 in Tables 2.11 and 2.12. We see that the min-ED and min-AED hyper-architectures under different utilization rates are the same. Therefore, we conclude that utilization rate in practice does not affect hyper-architecture evaluation. We guess that the reason why the utilization rate does not affect hyper-architecture evaluation is as follows: In our models the performance and

[8] As discussed before, the most area efficient architectures ($N = 12$ and $K = 4$) reduces area by about 20% compared to the baseline. The area increase introduced by power gating leads to only about 15% area increase. Therefore, for the minimum AED hyper-architecture of the power gating classes consumes less area compared to the baseline.

dynamic power of the circuit under different utilization rate is the same, [9] and the only difference is leakage power. For the classes with power gating, the leakage power is determined by the active elements and the leakage power of the unused circuit elements is negligible (as we can see from Table 2.11, in classes *Homo-V_{th}+G* and *Hetero-V_{th}+G*, ED under different utilization rate is very close). When no power gating is applied, for given benchmark circuits, leakage power is the dominant power component and changes inversely proportional to the utilization rate. Therefore the relationship between the leakage power of different hyper-architectures does not change with respect to the change of the utilization rate.

Utilization	0.3					0.5					0.8				
rate	V_{dd}	CV_{th}	IV_{th}	(N, k)	ED	V_{dd}	CV_{th}	IV_{th}	(N, k)	ED	V_{dd}	CV_{th}	IV_{th}	(N, k)	ED
Homo-V_{th}	1.0	0.30	0.30	(10,4)	32.0	1.0	0.30	0.30	(10,4)	24.1	1.0	0.30	0.30	(10,4)	19.4
Hetero-V_{th}	0.9	0.25	0.30	(12,4)	31.5	0.9	0.25	0.30	(12,4)	23.0	0.9	0.25	0.30	(12,4)	18.1
Homo-V_{th}+G	0.9	0.25	0.25	(12,4)	12.3	0.9	0.25	0.25	(12,4)	11.9	0.9	0.25	0.25	(12,4)	11.8
Hetero-V_{th}+G	0.8	0.25	0.20	(10,4)	11.6	0.8	0.25	0.20	(10,4)	11.2	0.8	0.25	0.20	(10,4)	11.0

Table 2.11: Min-ED hyper-architecture under different utilization rates.

Utilization	0.3						0.5						0.8					
rate	V_{dd}	CV_{th}	IV_{th}	(N, k)	S	AED	V_{dd}	CV_{th}	IV_{th}	(N, k)	S	AED	V_{dd}	CV_{th}	IV_{th}	(N, k)	S	AED
Homo-V_{th}	1.0	0.30	0.30	(10,4)	-	0.649	1.0	0.30	0.30	(10,4)	-	0.697	1.0	0.30	0.30	(10,4)	-	0.698
Hetero-V_{th}	0.9	0.25	0.30	(12,4)	-	0.637	0.9	0.25	0.30	(12,4)	-	0.626	0.9	0.25	0.30	(12,4)	-	0.617
Homo-V_{th}+G	0.9	0.25	0.25	(12,4)	2	0.330	0.9	0.25	0.25	(12,4)	2	0.432	0.9	0.25	0.25	(12,4)	2	0.533
Hetero-V_{th}+G	0.8	0.25	0.20	(10,4)	2	0.324	0.8	0.25	0.20	(10,4)	2	0.413	0.8	0.25	0.20	(10,4)	2	0.510

Table 2.12: Min-AED hyper-architecture under different utilization rates. Note: AED is normalized with respect to the baseline.

2.4.6 Impact of Interconnect Structure

In the previous discussion, we always assume all the wire segments span four logic blocks (i.e. length-4 interconnect wires). In this section, we will compare two routing

[9] We assume the placement and route is the same for different utilization rate.

structures to show the impact of routing structure on delay and power. We compare a structure with uniform length-4 wire segments (in short *uniform-interconnect*) and a routing structure with 60% length-4 wire segments and 40% length-8 wire segments (in short *mixed-interconnect*). The *mixed-interconnect* is optimal for minimizing delay [47]. Tables 2.13, 2.14, and 2.15 compare the min delay, min energy and min ED hyper-architectures between the two routing structures, respectively. We see that the min-delay hyper-architectures for both interconnect structures are same. The min-energy and min-ED hyper-architectures for the two routing structure have the same the device setting and LUT size, but mixed-interconnect tends to use smaller cluster size as the interconnect delay is reduced in mixed-interconnect. We also see that *uniform-interconnect* has lower energy but higher delay than *mixed-interconnect*. This is due to the fact that *mixed-interconnect* applies length-8 wire-segments and therefore a buffer with a larger size is used, which reduces delay but increases power.

uniform-interconnect						
hyper-architecture Class	V_{dd} (V)	CV_{th} (V)	IV_{th} (V)	(N,K)	Energy (nJ)	Delay (ns)
Homo-V_{th}	1.1	0.20	0.20	(6,7)	31.22	8.86
Hetero-V_{th}	1.1	0.20	0.20	(6,7)	31.22	8.86
Homo-V_{th}+G	1.1	0.20	0.20	(6,7)	15.98	9.45
Hetero-V_{th}+G	1.1	0.20	0.20	(6,7)	15.98	9.45
mixed-interconnect						
Homo-V_{th}	1.1	0.20	0.20	(6,7)	35.56	8.42
Hetero-V_{th}	1.1	0.20	0.20	(6,7)	35.56	8.42
Homo-V_{th}+G	1.1	0.20	0.20	(6,7)	16.25	9.13
Hetero-V_{th}+G	1.1	0.20	0.20	(6,7)	16.25	9.13

Table 2.13: Min-delay hyper-architecture under different routing structures.

uniform-interconnect						
hyper-architecture Class	V_{dd} (V)	CV_{th} (V)	IV_{th} (V)	(N,K)	Energy (nJ)	Delay (ns)
Homo-V_{th}	0.8	0.35	0.35	(10,4)	0.942	59.2
Hetero-V_{th}	0.8	0.30	0.35	(12,4)	0.920	43.6
Homo-V_{th}+G	0.8	0.30	0.30	(12,4)	0.550	30.5
Hetero-V_{th}+G	0.8	0.30	0.25	(10,4)	0.549	24.3
mixed-interconnect						
Homo-V_{th}	0.8	0.35	0.35	(8,4)	1.052	55.6
Hetero-V_{th}	0.8	0.35	0.35	(8,4)	1.052	55.6
Homo-V_{th}+G	0.8	0.30	0.30	(12,4)	0.610	29.3
Hetero-V_{th}+G	0.8	0.30	0.25	(8,4)	0.602	22.8

Table 2.14: Min-energy hyper-architecture under different routing structures.

uniform interconnect							
hyper-architecture Class	V_{dd} (V)	CV_{th} (V)	IV_{th} (V)	(N,K)	Energy (nJ)	Delay (ns)	ED (nJ·ns)
Homo-V_{th}	1.0	0.30	0.30	(10,4)	1.37	17.50	24.1
Hetero-V_{th}	0.9	0.25	0.30	(12,4)	1.27	18.10	23.0
Homo-V_{th}+G	0.9	0.25	0.25	(12,4)	0.74	16.10	11.9
Hetero-V_{th}+G	0.8	0.25	0.20	(10,4)	0.65	17.30	11.2
mixed-interconnect							
Homo-V_{th}	1.0	0.30	0.30	(8,4)	1.51	17.00	25.7
Hetero-V_{th}	0.9	0.25	0.30	(12,4)	1.39	18.40	25.6
Homo-V_{th}+G	0.9	0.25	0.25	(8,4)	0.82	16.40	13.4
Hetero-V_{th}+G	0.8	0.25	0.20	(8,4)	0.74	16.67	12.3

Table 2.15: Min-ED hyper-architecture under different routing structures.

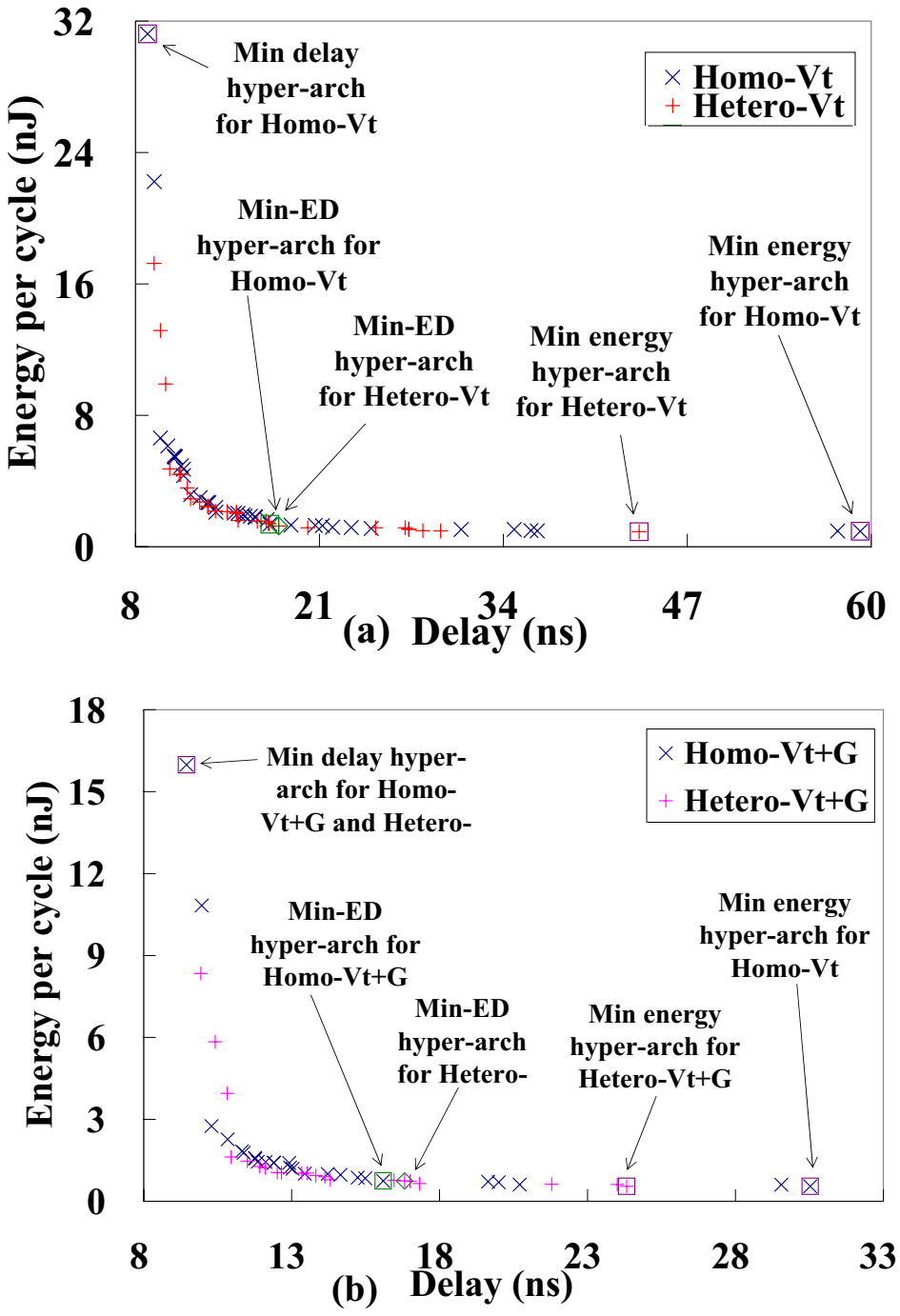


Figure 2.7: Dominant hyper-architectures. (a) *Homo-V_{th}* and *Hetero-V_{th}*; (b) *Homo-V_{th}+G* and *Hetero-V_{th}+G*.

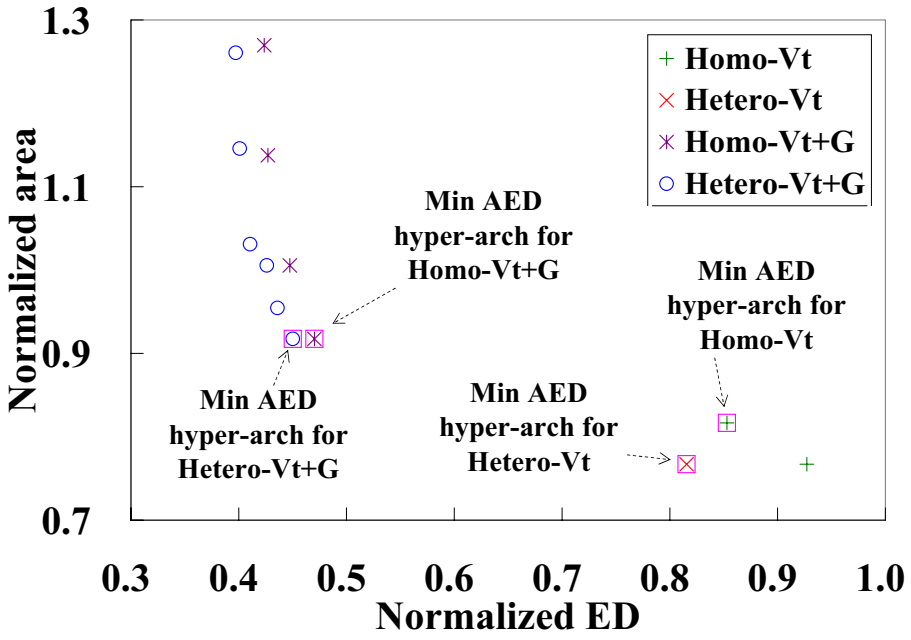


Figure 2.8: ED and area trade-off. ED and area are normalized with respect to the baseline ($N = 8$, $K = 4$, $V_{dd} = 0.9V$, and $V_{th} = 0.3V$).

2.5 Conclusions

In this chapter, we have developed trace-based power and performance evaluation (Ptrace) for FPGA. The one-time use of placement, routing and cycle-accurate power simulation is applied to collect the timing and power trace for a given benchmark set and a given FPGA architecture. The trace can then be re-used to calculate timing and power via closed-form formulae for different device parameters and technology scaling. Ptrace is much faster, yet accurate compared to the conventional evaluation based on placement and routing by VPR [18] followed by cycle-accurate simulation (Psim) [89].

Using the trace-based estimation, we have performed device (V_{dd}, V_{th} and sleep transistor size if power gating is applied) and architecture (cluster and LUT size) co-optimizations for low power FPGAs. We assume the ITRS [66] 70nm technology and use the following baseline for comparison: Cluster size of 8 and LUT size of 4 as in the Xilinx Virtex-II [170], V_{dd} of 0.9v suggested by ITRS, V_{th} of 0.3v which is optimized for min-ED (i.e., minimum energy delay product) with respect to the above architecture and V_{dd}. Compared to the baseline case, simultaneous optimization of FPGA architecture and device reduces the min-ED by 14.7% and area by 18.3% for FPGA using homogeneous-V_{th} for the logic blocks and interconnects without power gating. Optimizing V_{th} separately (i.e., heterogeneous-V_{th}) for the logic block and interconnect reduces min-ED by 18.4% and area by 23.3%. Furthermore, power gating unused logic and interconnect reduces the min-ED by up to 55.0% and reduces area by 8.2%. Compared to the classes without power gating, the min-ED hyper-architectures of the classes with power gating have lower V_{th}. This is due to the fact that, when power gating is applied, leakage power is significantly reduced and therefore a lower V_{th} can be applied to reduce delay.

We observe that min-ED hyper-architectures and min-AED hyper-architectures for

different utilization rate (between 30% and 80%) are the same. Therefore, the utilization rate in practice does not affect the device and architecture co-optimization result. Moreover, we also test two different routing structures, one with uniform length 4 wire segments (uniform-interconnect) and the other with 60% length 4 wire segments and 40% length 8 wire segments (mixed-interconnect). We observe that the min-ED, min-energy and min-delay hyper-architectures under two different interconnect structures are similar, except that mix-interconnect tends to use slightly smaller cluster size due to the reduced interconnect delay.

CHAPTER 3

FPGA Device and Architecture Co-Optimization Considering Process Variation

Process variations in nanometer technologies are becoming an important issue for cutting-edge FPGAs with a multi-million gate capacity. In this chapter, we extend *Ptrace* to handle process variation. We first develop closed-form models of chip level FPGA leakage and timing variations considering both die-to-die and within-die variations in effective channel length, threshold voltage, and gate oxide thickness. Experiments show that the mean and standard deviation computed by our models are within 3% from those computed by Monte Carlo simulation. We also observe that the leakage and delay variations can be up to 5.5X and 1.9X, respectively. We then derive analytical yield models considering both leakage and timing variations, and use such models to evaluate FPGA device and architecture considering process variations. Compared to the baseline, which uses the VPR architecture and device setting from ITRS roadmap, device and architecture tuning improves leakage yield by 10.4%, timing yield by 5.7%, and leakage and timing combined yield by 9.4%. We also observe that LUT size of 4 gives the highest leakage yield, LUT size of 7 gives the highest timing yield, but LUT size of 5 achieves the maximum leakage and timing combined yield.

3.1 Introduction

In the previous chapter, we have introduced a time efficient trace-based power and delay estimator for FPGA circuit. However, such estimator is only for deterministic values and does not consider process variation. 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 [25]. In addition to meeting the performance constraint under timing variation, dice with excessively large leakage due to such a high variation have to be rejected to meet the given power budget.

Recent work has studied parametric yield estimation for both timing and leakage power. Statistical timing analysis considering path correlation was studied in [120] [86, 178] and [180, 33] further introduced non-Gaussian variation and non-linear variation models. Timing yield estimation was discussed in [57, 53] and [127] purposed a methodology to improve timing yield. As device scale down, leakage power becomes a significant component of total power consumption and it is greatly affected by process variation. [129, 187, 152] studied the parametric yield considering both leakage and timing variations. Power minimization by gate sizing and threshold voltage assignment under timing yield constrains were studied in [114]. However, all these studies only focus on ASIC and do not consider FPGA.

FPGA has a great deal of regularity, therefore process variation may have smaller impact on FPGAs than on ASICs. Yet the parametric yield for FPGAs still should be studied. Some recent works [43, 100, 115, 136] has studied the impact of process variation and presented various statistical optimization methods. However, architecture and device co-optimization considering process variation has not be studied yet.

In this chapter,we first develop closed-form formulas of chip level leakage and timing variations considering both die-to-die and within-die variations. Based on such model, we extend the *Ptrace* discussed in Chapter 2 to estimate the power and delay variation of FPGAs. Experiments show that the mean and standard deviation computed by our models are within 3% from those copmuted by Monte Carlo simulation. We also observe that the leakage and delay variations can be up to 5.5X and 1.5X, respectively. With the extended Ptrace, we perform FPGA device and architecture evaluation considering process variations. The evaluation requires the exploration of the following dimensions: cluster size N, LUT size K, [1] supply voltage V_{dd}, and threshold voltage V_t. We defined the combinations of the above parameters as *hyper-architecture*. For comparison, we obtain the baseline FPGA hyper-architecture which uses the VPR architecture model [18] and the same LUT size and Cluster size as the commercial FPGAs used by Xilinx Virtex-II [170], and device setting from ITRS roadmap[66]. Compared to the baseline, device and architecture tuning improves leakage yield by 4.8%, timing yield by 12.4%, and leakage and timing combined yield by 9.2%. We also observe that LUT size of 4 gives the highest leakage yield, LUT size of 7 gives the highest timing yield, but LUT size of 5 achieves the maximum leakage and timing combined yield.

The rest of the chapter is organized as follows: Section 3.2 derives closed-form models for leakage and delay variations and develops the leakage and timing yield models. Section 3.3 perform device and architecture evaluation to improve yield rate. Finally, Section 3.4 concludes the chapter.

[1] In this chapter, N refers to cluster size and K refers to LUT size

3.2 Delay and Leakage Variation Model

In this chapter, we consider the variation in gate channel length (L_{gate}), threshold voltage (V_{th}), and gate oxide thickness (T_{ox}). According to [190], spatial correlation is not significant. Therefore, in this chapter, we assume each variation source is decomposed into global (inter-die) variation and local (intra-die) variation as follows:

$$L = L_g + L_l \tag{3.1}$$

$$V = V_g + V_l \tag{3.2}$$

$$T = T_g + T_l \tag{3.3}$$

where L, V, and T are variations of L_{gate}, V_{th}, and T_{ox} respectively, L_g, V_g, and T_g are inter-die variations, and L_l, V_l, and T_l are intra-die variations. In the rest of this chapter, we assume both inter-die (L_g, V_g, and T_g) and intra-die (L_l, V_l, and T_l) variations are normal random variables. And we also assume that inter-die variation and intra-die variation are independent, and all variation sources are also independent.

3.2.0.1 Leakage under Variation

We extend the leakage model in FPGA power and delay estimation framework *Ptrace* [38] to consider variations. In *Ptrace*, the total leakage current of an FPGA chip is calculated as follows:

$$I_{chip} = \sum_i N_i^t \cdot I_i \tag{3.4}$$

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

The leakage current I_i of a type i circuit element is the sum of the sub-threshold

and gate leakages:

$$I_i = I_{sub} + I_{gate} \tag{3.5}$$

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

Different from [129] which 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:

$$I_i = I_n(i) \cdot e^{f_{Li}(L)} \cdot e^{f_{Vi}(V)} \cdot e^{f_{Ti}(T)} \tag{3.6}$$

where $I_n(i)$ is the nominal value of the leakage current of type i circuit element and f is the function that represents the impact of each type of process variation on leakage. The dependency between these functions has been shown to be negligible in [129]. From MASTAR4 model [68], we find that it is sufficient to express these functions as simple linear functions as follows:

$$f_{Li}(L) = -c_{i1} \cdot L \tag{3.7}$$

$$f_{Vi}(V) = -c_{i2} \cdot V \tag{3.8}$$

$$f_{Ti}(T) = -c_{i3} \cdot T \tag{3.9}$$

where c_{i1}, c_{i2}, c_{i3} are fitting parameters obtained from MASTAR4 model. 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 (3.6) as follows by decomposing L, V and T in to intra-die (L_l, V_l, T_l) and inter-die (L_g, V_g, T_g) components.

$$I_i = I_n(i) \cdot e^{-(c_{i1}L_g + c_{i2}V_g + c_{i3}T_g)} \cdot e^{-(c_{i1}L_l + c_{i2}V_l + c_{i3}T_l)} \tag{3.10}$$

To extend the leakage model (3.4) under variations, we assume that each element has unique intra-die variations but all elements in one die share the same inter-die

variations. Both inter-die and intra-die 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, therefore the relative random variance of the total leakage due to intra-die variation approaches zero. Similar to [129], for given inter-die variations, we apply the Central Limit Theorem and use the sum of mean to approximate the total leakage current. After integration, we can write the expression of the chip-level leakage as the follows:

$$\begin{aligned} I_{chip} &\approx \sum_i N_i^t \cdot E[I_i | L_g, V_g, T_g] \\ &= \sum_i N_i^t S_i I_{L_g,V_g,T_g}(i) \qquad (3.11) \\ S_i &= e^{(c_{i1}{\sigma_{L_l}}^2 + c_{i2}{\sigma_{V_l}}^2 + c_{i3}{\sigma_{T_l}}^2)/2} \qquad (3.12) \\ I_{L_g,V_g,T_g}(i) &= I_n(i) e^{-(c_{i1}L_g + c_{i2}V_g + c_{i3}T_g)} \qquad (3.13) \end{aligned}$$

where S_i is the scale factor introduced by intra-die variability in L, V, and T. $I_{L_g,V_g,T_g}(i)$ is the leakage as a function of inter-die variations. σ_{L_l}, σ_{V_l} and σ_{T_l} are the variances of L_l, V_l, and T_l, respectively.

3.2.0.2 Leakage Yield

From (3.11), (3.12), and (3.13), we can see that the chip leakage current is a sum of log-normal random variables and it can be expressed as follows,

$$\begin{aligned} I_{chip} &= \sum_i X_i \qquad (3.14) \\ X_i &\sim Lognormal(log(A_i), ((c_{i1}\sigma_{L_g})^2 + (c_{i2}\sigma_{V_g})^2 + (c_{i3}\sigma_{T_g})^2)) \qquad (3.15) \\ A_i &= N_i I_n(i) \qquad (3.16) \end{aligned}$$

Same as [129], 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

σ_{L_g}, σ_{V_g}, and σ_{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,

$$\mu_{I_{chip}} = \sum_i \{exp[log(A_i) + \frac{(c_{i1}\sigma_{L_g})^2}{2} + \frac{(c_{i2}\sigma_{V_g})^2}{2} + \frac{(c_{i3}\sigma_{T_g})^2}{2}]\} \quad (3.17)$$

$$\sigma^2_{I_{chip}} = \sum_i \{exp[2log(A_i) + (c_{i1}\sigma_{L_g})^2 + (c_{i2}\sigma_{V_g})^2 + (c_{i3}\sigma_{T_g})^2] \cdot [exp({c_{ii}}^2\sigma^2_{L_g} + {c_{i2}}^2\sigma^2_{V_g} + c^2_{i3}\sigma^2_{T_g}) - 1]\} + \sum_{i,j} 2COV(X_i, X_j) \quad (3.18)$$

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,

$$COV(X_i, X_j) = E[X_iX_j] - E[X_i]E[X_j] \quad (3.19)$$

$$E[X_iX_j] = exp[log(A_iA_j) + \frac{(c_{i1} + c_{j2})^2{\sigma_{L_g}}^2}{2} + \frac{(c_{i2} + c_{j2})^2{\sigma_{V_g}}^2}{2} + \frac{(c_{i3} + c_{j3})^2{\sigma_{T_g}}^2}{2}] \quad (3.20)$$

$$E[X_i] = exp[log(A_i) + \frac{(c_{i1}\sigma_{L_g})^2}{2} + \frac{(c_{i2}\sigma_{V_g})^2}{2} + \frac{(c_{i3}\sigma_{T_g})^2}{2}] \quad (3.21)$$

We then use the method from [129] 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,

$$\mu_{N,I_{chip}} = \frac{log[{\mu_{I_{chip}}}^4/({\mu_{I_{chip}}}^2 + {\sigma_{I_{chip}}}^2)]}{2} \quad (3.22)$$

$${\sigma_{N,I_{chip}}}^2 = log[1 + ({\sigma_{I_{chip}}}^2/{\mu_{I_{chip}}}^2)] \quad (3.23)$$

$$CDF(I_{chip}) = \frac{1}{2}[1 + erf(\frac{log(I_{chip}) - \mu_{N,I_{chip}}}{\sqrt{2}\sigma_{N,I_{chip}}})] \quad (3.24)$$

where $erf(\cdot)$ is the error function. Given a leakage limit I_{cut} for I_{chip},

$$Y_{leak} = CDF(I_{cut}) \times 100\% \quad (3.25)$$

gives the leakage yield rate $Y_{leak}(I_{cut}|L_g)$, i.e., the percentage of FPGA chips that is smaller than I_{cut}.

3.2.0.3 Timing under Variation

The performance depends on L_{gate}, V_{th}, and T_{ox}, but its variation is primarily affected by L_{gate} and V_{th} variation [129]. Below we extend the delay model in *Ptrace* to consider inter-die and intra-die variations of L_{gate}. In *Ptrace*, the path delay is calculated as follows:

$$D = \sum_i d_i \tag{3.26}$$

where d_i is the delay of the i_{th} circuit element in the path. Considering process variation,the path delay is calculated as follows:

$$D = \sum_i d_i(L_g, L_l, V_g, V_l) \tag{3.27}$$

For circuit element i in the path, $d_i(L_g, L_l, V_G, V_l)$ is the delay considering inter-die variation L_g, V_g and intra-die variation L_l, V_l. L_g and V_g the same for all the circuit elements in the critical path. Given L_g and V_g, we evenly sample a few (eleven in this chapter) points within range of $[L_g - 3\sigma_{L_l}, L_g + 3\sigma_{L_l}]$. We then use circuit level delay model in [39] to obtain the delay for each circuit element with these variations. As the delay monotonically decreases when L_{gate} and V_{th} increase, 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 intra-die channel length and threshold voltage variation of each element is independent from each other. Therefore, we can obtain the *PDF* (probability density function) of the critical path delay for a given L_g and V_g as follows by convolution operation,

$$\begin{aligned} PDF(D|L_g, V_g) = PDF(d_1|L_g, V_g) \otimes PDF(d_2|L_g, V_g) \otimes \cdots \\ \otimes PDF(d_i|L_g, V_g) \otimes \cdots \otimes PDF(d_n|L_g, V_g) \end{aligned} \tag{3.28}$$

3.2.0.4 Timing Yield

The timing yield is calculated on a bin-by-bin basis where each bin corresponds to a specific value L_g and V_g. We further consider intra-die variation of channel length in timing yield analysis. Given the inter-die channel length variation L_g, and threshold voltage variation V_g, (3.28) gives the PDF of the critical path delay D of the circuit. We can obtain the CDF of delay, $CDF(D|L_g,V_g)$, by integrating $PDF(D|L_g,V_g)$. Given a cutoff delay (D_{cut}), $CDF(D_{cut}|L_g)$ gives the probability that the path delay is smaller than D_{cut} considering L_{gate} and V_{th} 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 for a given L_g the delay of each path is independent and we can calculate the timing yield as follows,

$$Y_{perf}(D_{cut}|L_g,V_g) = \prod_{i=1}^{n} CDF_i(D_{cut}|L_g,V_g) \tag{3.29}$$

where $CDF_i(D_{cut}|L_g,V_g)$ gives the probability that the delay of the i^{th} longest path is no greater than D_{cut}. In this chapter, 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 $Y_{perf}(D_{cut}|L_g,V_g)$ over L_g and V_g to calculate the performance yield Y_{perf} as follows,

$$Y_{perf} = \int\int_{-\infty}^{+\infty} PDF(L_g)PDF(V_g) \cdot Y_{perf}(D_{cut}|L_g,V_g) \cdot dL_g dV_g \tag{3.30}$$

3.2.0.5 Leakage and Timing Combined Yield

To analyze the yield of a lot, we need to consider both leakage and delay limit. In order to compute the leakage and delay combined yield, we first need to calculate the leakage

yield for a given inter-die variation of gate channel length L_g and threshold voltage V_g, $Y_{leak|L_g,V_g}$. Similar to Section 3.2.0.2, we first calculate the mean and variance of leakage current for given L_g and V_g,

$$\mu_{I_{chip|L_g,V_g}} = \sum_i \{exp[log(\bar{A}_{i|L_g,V_g}) + \frac{(c_{i3}\sigma_{T_g})^2}{2}]\} \quad (3.31)$$

$$\sigma^2_{I_{chip|L_g,V_g}} = \sum_i \{exp[2log(\bar{A}_{i|L_g,V_g}) + (c_{i3}\sigma_{T_g})^2] \cdot [exp(c^2_{i3}\sigma^2_{T_g}) - 1]\} + \sum_{i,j} 2COV(\bar{X}_{i|L_g,V_g}, \bar{X}_{j|L_g,V_g}) \quad (3.32)$$

where

$$\bar{A}_{i|L_g,V_g} = A_i \cdot exp(-c_{i1}L_g - c_{i2}V_g) \quad (3.33)$$

$$\bar{X}_{i|L_g,V_g} \sim Lognormal(\bar{A}_{i|L_g,V_g}, (c_{i3}\sigma_{T_g})^2) \quad (3.34)$$

Similar to X_i's, the covariance between $\bar{X}_{i|L_g,V_g}$'s are computed as:

$$COV(\bar{X}_{i|L_g,V_g}, \bar{X}_{j|L_g,V_g}) = E[\bar{X}_{i|L_g,V_g} \cdot \bar{X}_{j|L_g,V_g}] - E[\bar{X}_{i|L_g,V_g}]E[\bar{X}_{j|L_g,V_g}] \quad (3.35)$$

$$E[\bar{X}_{i|L_g,V_g} \cdot \bar{X}_{j|L_g,V_g}] = exp[log(\bar{A}_{i|L_g,V_g} \cdot \bar{A}_{j|L_g,V_g}) + \frac{(c_{i3} + c_{j3})^2 {\sigma_{T_g}}^2}{2}] \quad (3.36)$$

$$E[X_i] = exp[log(\bar{A}_{i|L_g,V_g}) + \frac{(c_{i3}\sigma_{T_g})^2}{2}] \quad (3.37)$$

Finally, the CDF of leakage current for given L_g and V_g, $I_{leak|L_g,V_g}$, is calculated as:

$$\mu_{N,I_{chip|L_g,V_g}} = \frac{log[\mu^4_{I_{chip|L_g,V_g}}/(\mu^2_{I_{chip|L_g,V_g}} + \sigma^2_{I_{chip|L_g,V_g}})]}{2} \quad (3.38)$$

$${\sigma_{N,I_{chip|L_g,V_g}}}^2 = log[1 + (\sigma^2_{I_{chip|L_g,V_g}}/\mu^2_{I_{chip|L_g,V_g}})] \quad (3.39)$$

$$CDF(I_{chip}|L_g,V_g) = \frac{1}{2}[1 + erf(\frac{log(I_{chip}) - \mu_{N,I_{chip|L_g,V_g}}}{\sqrt{2}\sigma_{N,I_{chip}}})] \quad (3.40)$$

With the CDF of $I_{leak|L_g,V_g}$, it is easy to compute the leakage yield for given L_g and V_g,

$$Y_{leak|L_g,V_g} = CDF(I_{cut}|L_g,V_g) \times 100\% \quad (3.41)$$

MC sim			Our model		
Y_{leak} %	Y_{perf} %	Y_{com} %	Y_{leak} %	Y_{perf} %	Y_{com} %
90.2	74.1	64.4	89.3 (-0.9)	72.6 (-1.5)	62.1 (-2.1)

Table 3.1: Verification of yield model.

Because for given a specific inter-die variation of channel length L_g and threshold voltage variation V_g, the leakage variability only depends on the variability of random variable T_g as shown in (3.31), and the timing variability only depends on the variability of random variable L_l and V_l as shown in (3.29). Therefore, we assume that the leakage yield and timing yield are independent of each other for given L_g and V_g. The yield considering the imposed leakage and timing limit can be calculated as follows,

$$Y_{com} = \iint_{-\infty}^{+\infty} PDF(L_g)PDF(V_g)Y_{leak}(I_{cut}|L_g,V_g)Y_{perf}(D_{cut}|L_g,V_g)\cdot dL_g dV_g \quad (3.42)$$

3.2.0.6 Verification of Yield Model

In this section, we verify our yield model by comparing it to 10,000 sample Monte-Carlo simulation. In our experiment, we assume 70nm ITRS technology and use device and architecture same as the baseline hyper-architecture as in Section 2.4. We also assume that all 20 MCNC benchmarks are put into one FPGA chip. The cut of leakage power is 2X of the nominal value and the cut of delay is 1.1X of nominal value. Table 3.1 compares the yield estimated from our model and that from the Monte-Carlo simulation. From the table, we see that our yield model is within 3% error compared to the Monte-Carlo simulation.

3.3 Hyper-Architecture Evaluation Considering Process Variation

In this section, we use our yield model to perform device and architecture evaluation for leakage and delay yield optimization. We consider ITRS 70nm technology and

	N	K	W	V_{th} (V)	V_{dd} (V)
Evaluation range	4,5,6,7	6,8,10,12	4	0.2 0.4	0.8 1.1
Baseline	8	4	4	0.3	0.9

Source	Distribution	$3\sigma_g$	$3\sigma_l$
L_{gate}	Normal	5.0%	3.0%
V_{th}	Normal	2.5%	1.9%
T_{ox}	Normal	2.5%	1.9%

Table 3.2: Experimental setting.

change V_{dd} and V_{th} around such setting. For architecture, we consider LUT size K from 4 to 7, and cluster size N from 6 to 12. For interconnect, we assume that all the global routing track (W) span 4 logic blocks with all buffer switch box. In the experiment, we assume that all 20 MCNC benchmarks are put into one chip and obtain the longest 10 critical paths from them. For comparison, we use the same baseline hyper-architecture as in Section 2.4. For process variation, we assume that all the variation sources has normal distribution. For all variation sources, we assume that the 3σ value of the inter-die variation is 10% of nominal value, and the 3σ value of the intra-die variation is 5% of nominal value.

3.3.1 Impact of Process Variation

In this section, we analyze the impact of process variation on FPGA leakage power and delay. Figure 3.1 illustrates the leakage and delay variation from Monte-Carlo simulation for the baseline hyper-arch. In the figure, each dot is sample of Monte-Carlo simulation. From the figure, we can see with process variation, the range of leakage power is up to 5.5X and the range of delay variation is up to 1.5X.

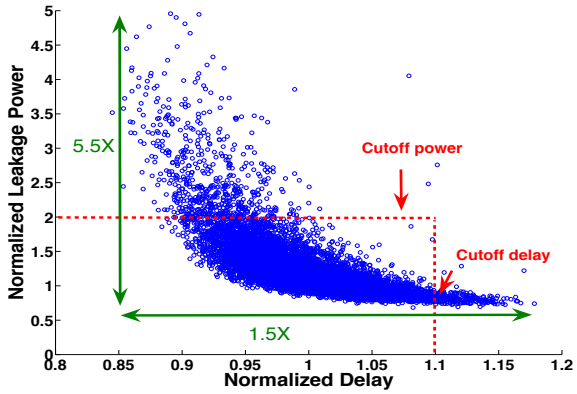


Figure 3.1: Leakage and delay of baseline architecture hyper-arch.

3.3.2 Impact of Device and Architecture Tuning

In this section, we perform device and architecture evaluation to optimize the delay and leakage power yield for two FPGA classes, *Homo-*V_{th} and *Hetero-*V_{th} as defined in in Section 2.4. In the rest of this section, we assume that the cutoff leakage power is 2X of the nominal value and the cutoff delay is 1.1X of the nominal value, as shown in Figure 3.1.

3.3.2.1 Leakage Yield

We first optimize leakage yield. Table 3.3 illustrates the hyper-archs with maximum leakage yield for both two classes.

In the table, CV_{th} refers to the V_{th} of logic blocks and IV_{th} refers to the V_{th} of interconnect. Notice that for *Homo-*V_{th}, $CV_{th} = IV_{th}$. From the table, we see that *Homo-*L_{gate} and *Hetero-*L_{gate} give the same maximum leakage yield result. That is, the optimum L_{gate} for logic blocks and interconnect is the same.

This is because the larger L_{gate} gives better leakage yield, both logic block and interconnect using largest L_{gate} (33nm) results in optimum leakage yield. Moreover,

	N	K	CV_{th} (V)	IV_{th} (V)	V_{dd} (V)	Y_{leak} %	Y_{perf} %	Y_{com} %
Baseline	8	4	0.3	0.3	0.9	89.5	85.5	67.1
Homo-V_{th}	10	4	0.4	0.4	0.9	94.3(+4.8)	79.3	69.1
Hetero-V_{th}	10	4	0.4	0.4	0.9	94.3(+4.8)	79.3	69.1

Table 3.3: Optimum leakage yield hyper-architecture.

	N	K	CV_{th} (V)	IV_{th} (V)	V_{dd} (V)	Y_{leak} %	Y_{perf} %	Y_{com} %
Baseline	8	4	0.3	0.3	0.9	89.5	85.5	67.1
Homo-V_{th}	6	7	0.2	0.2	1.1	63.5	97.9(+12.4)	57.9
Hetero-V_{th}	6	7	0.2	0.2	1.1	67.6	97.9(+12.4)	57.9

Table 3.4: Optimum Timing yield hyper-architecture.

we can also find that $K = 4$ gives the optimum leakage yield, which improve leakage yield by 4.8% compared to the baseline.

3.3.2.2 Timing Yield

Secondly, we analyze the 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. Table 3.4 illustrates the optimum timing yield hyper-arch. Similar to leakage yield analysis, both *Homo-*L_{gate} and *Hetero-*L_{gate} achieve the same hyper-arch for optimum timing yield. The reason is similar to the leakage yield. That is the smaller V_{th} gives better timing yield, therefore both logic block and interconnect using smallest V_{th} (0.2V) results in optimum timing yield. From the table, we can also find that the optimum timing yield hyper-arch has $K = 7$ and improve timing yield by 12.4% compared to the baseline.

3.3.2.3 Leakage and Timing Combined Yield

Finally, we discuss about leakage and timing combined yield. Table 3.5 illustrates the optimum leakage and timing combined yield hyper-arch. From the table, we see that

	N	K	CV_{th} (V)	IV_{th} (V)	V_{dd} (V)	Y_{leak} %	Y_{perf} %	Y_{com} %
Baseline	8	4	0.3	0.3	0.9	89.5	85.5	67.1
Homo-V_{th}	10	5	0.3	0.3	1.0	87.6	86.5	75.2 (+8.1)
Hetero-V_{th}	8	5	0.35	0.3	1.0	91.6	84.1	76.3 (+9.2)

Table 3.5: Optimum leakage and timing combined yield hyper-architecture.

compared to the baseline, the optimum hyper-arch for *Homo-V_{th}* improves combined yield by 8.1% and the optimum hyper-arch for *Hetero-V_{th}* improves the combined yield by 9.2%. Unlike the leakage yield and timing yield analysis, *Homo-V_{th}* and *Hetero-V_{th}* give different result for combined yield optimization. This is because both leakage and timing should be considered to optimize combined yield, the largest (or smallest) V_{th} not necessary gives the optimum combined yield. We also find that *Hetero-V_{th}* gives better result than *Homo-V_{th}*. This is because *Hetero-V_{th}* provides larger search space. But for both *Homo-V_{th}* and *Hetero-V_{th}*, $K = 5$ gives the best combined yield.

3.4 Conclusions

In this chapter, we have developed efficient models for chip-level leakage variation and system timing variation in FPGAs. Experiments show that our models are within 3% from Monte Carlo simulation, and the FPGA chip level leakage and delay variations can be up to 5.5X and 1.5X, respectively. We have shown that architecture and device tuning has a significant impact on FPGA parametric yield rate. Compared to the baseline, the optimum hyper-architecture (combination of architecture and device parameters) improves leakage and timing combined yield by 9.2%. In addition, LUT size 4 has the highest leakage yield, 7 has the highest timing yield, but LUT size 5 achieves the maximum combined leakage and timing yield.

CHAPTER 4

FPGA Concurrent Development of Process and Architecture Considering Process Variation and Reliability

In Chapter 2 and Chapter 3, we develop a trace-based framework (*Ptrace*) to perform device and architecture co-optimization. In this chapter, we further improve the trace-based framework (*Ptrace2*) to enable concurrent process and FPGA architecture co-development. Applying the new trace-based framework (*Ptrace2*), the user can tune eight parameters for bulk CMOS processes and obtain the chip level performance and power distribution and soft error rate (SER) considering process variations and device aging. The *Ptrace2* framework is efficient as it is based on closed-form formulas. It is also flexible as process parameters can be customized for different FPGA elements and no SPICE models and simulations are needed for these elements. Therefore, this framework is suitable for early stage process and FPGA architecture co-development. The chapter further presents a few examples to utilize the framework. We show that applying heterogeneous gate lengths to logic and interconnect may lead to 1.3X delay difference, 3.1X energy difference, and reduce standard deviation of leakage variation by 87%. This offers a large room for power and delay tradeoff. We further show that the device aging has a knee point over time, and device burn-in to reach the point could reduce the performance change over 10 years from 8.5% to 5.5% and reduce die to die leakage significantly. In addition, we also study the interaction between process

variation, device aging and SER. We observe that device aging reduces standard deviation of leakage by 65% over 10 years while it has relatively small impact on delay variation. Moreover, we also find that neither device aging due to NBTI and HCI nor process variation have significant impact on SER.

4.1 Introduction

In Chapter 2 and Chapter 3, we proposed a time efficient FPGA power and delay evaluator *Ptrace*. However, *Ptrace* assumes that the processes are mature and stable device models are available so that SPICE simulations can be carried out to obtain circuit level power and delay. The assumption that FPGA architecture development starts only after the process technology is stable may be valid in the past, but it no longer holds as we begin to develop process and architecture concurrently in order to shorten the time to market.

In this chapter we further extend the trace-based architecture framework (*Ptrace*) discussed in the previous section to consider process parameters directly, therefore we can conduct FPGA circuit and architecture evaluation when only the first order process parameters are available. Such evaluation may be used to select circuits and architectures less sensitive to process changes or process variations. It may further provide inputs for process tuning, given that the FPGA is a large volume product and an FPGA company may convince a foundry to tune process when there are large enough benefits. We call the resulting framework as *ptrace2*. As illustrated in Figure 4.1, for performance and power, ptrace2 calculates first electrical characteristics of advanced CMOS transistors, then delay, leakage power, input/output capacitance for FPGA basic circuit elements, and finally the chip level performance and power based on trace similar to that in Chapter 2 and Chapter 3. The new process variation analysis can handle non-Gaussian variation sources which is ignored by *Ptrace*.

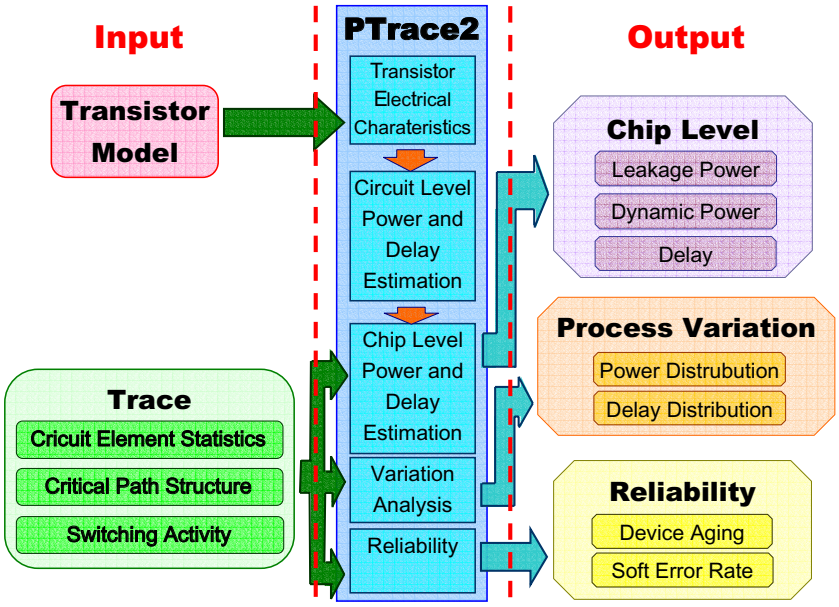


Figure 4.1: Trace-based estimation flow.

With *ptrace2*, we incorporate analytical calculations for two types of FPGA reliability, device aging (*N*egative-*B*ias-*T*emperature-*I*nstability, *NBTI* [11, 149] [160, 23] and *H*ot-*C*arrier-*I*njection, *HCI* [34, 149, 166]) and permanent soft error rate (*SER*) [60], again in the from device to chip fashion. Furthermore, we illustrate how to use this framework to improve power and performance by process and FPGA concurrent development, and to study the interaction between process variation, device aging and SER.

The rest of this chapter is organized as follows: Section 4.2 presents our implementation of ITRS device model. Sections 4.3, 4.4 and 4.5 introduce the circuit- and chip-level power and delay models and chip-level variation models, respectively. Section 4.6 presents device tuning for power and delay optimization, and Section 4.7 analyzes the reliability for FPGA. Finally, Section 4.8 studies the interaction between reliability and process variation, and Section 4.9 concludes this chapter.

4.2 Device Models

In this chapter, we implement the bulk transistor device model from ITRS 2005 MASTAR4 (*M*odel for *A*ssessment of cmo*S* *T*echnologies *A*nd *R*oadmaps) tool [67, 68, 148]. MASTAR4 is a computing tool which calculates the electrical characteristics of advance CMOS transistors [1]. It can handle different technologies including planar bulk, double gate and silicon on isolator (SOI) while we only consider traditional bulk transistor in this work. We briefly review the calculation flow in MASTAR4 below.

The calculation in MASTAR4 is based on analytical equations, which directly depends on various major technological parameters including gate length (L_{gate}), gate oxide thickness (T_{ox}), channel doping density (N_{bulk}), channel width (W), extension depth (X_{jext}) and the total series resistance for source and drain (R_{acc}). The output electrical characteristics include the on current (I_{on}), the sub-threshold leakage current (off current, I_{off}), the gate leakage current when the channel is on/off (I_{gon}/I_{goff}), the gate capacitance (C_g) and the drain/source diffusion capacitance (C_{diff}). Temperature (T) and the supply voltage (V_{dd}) also have a significant impact on these output characteristics. There are also other inputs related to mobility, velocity and gate stack etc., as well as some intermediate outputs such as effective mobility u_{eff} which are used for the final output calculation. We follow MASTAR4 tool and only tune the major process inputs while all the other inputs can be tuned as well if needed.

Figure 4.2 shows the calculation flow for the on current I_{on}. I_{on} depends on all the main inputs, i.e. L_{gate}, T_{ox}, N_{bulk}, X_{jext}, W, R_{acc}, T and V_{dd}. The feature intermediate parameters, the electrical (or effective) gate length (L_{elec}) and the electrical gate oxide thickness (T_{ox_elec}), are first calculated. The saturated threshold voltage (V_{th}) is then calculated considering the short channel effect (SCE) and drain induced bar-

[1] The device model in MASTAR4 tool is a predicted model for advanced processes and there is no correspondent SPICE models for these technologies.

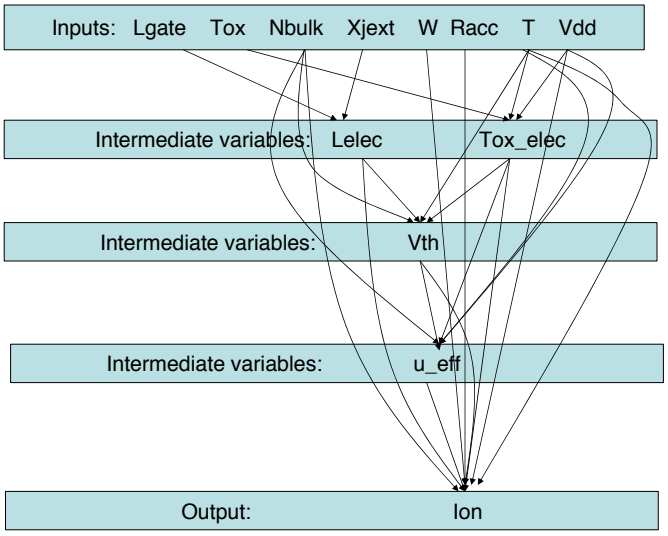


Figure 4.2: The flow for on current, I_{on}, calculation.

rier lowering (DIBL). The effective mobility (u_{eff}) is also calculated. With the main input parameters, main intermediate parameters and all other parameters, I_{on} is then calculated as,

$$V_{gt} = V_{gs} - V_{th} \tag{4.1}$$

$$I_{dsat0} = 0.5 u_{eff} \cdot C_{ox\_elec} \cdot \frac{W}{L_{elec}} \cdot V_{gt} \cdot V_{dsat} \tag{4.2}$$

$$I_{on} = \frac{I_{dsat0}}{1 + \frac{2R_{acc}I_{dsat0}}{V_{gt}} - \frac{R_{acc}I_{dsat0}}{V_{gt}+L_{elec}E_c(1+d)}} \tag{4.3}$$

where I_{dsat0} is the on current without considering the series resistance (R_{acc}), V_{gs} is the difference between the gate and source voltage levels, V_{dsat} is the velocity saturation voltage, E_c is the critical electrical field and d is an intermediate parameter. The derivation of these intermediate parameters are provided in the ITRS device model [68].

Figure 4.3 shows the calculation flow for the sub-threshold leakage current I_{off}. I_{off} depends on all the main inputs except for R_{acc}. Similar to I_{on} calculation, the

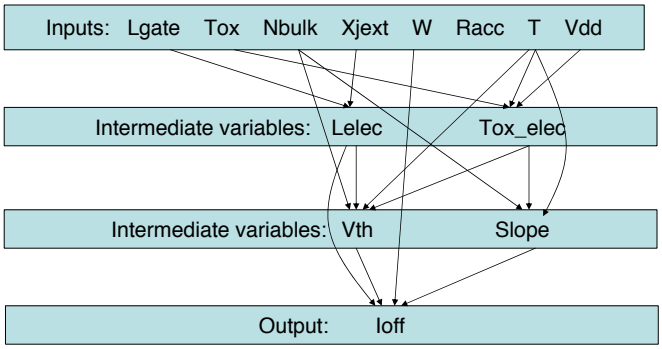


Figure 4.3: The flow for sub-threshold leakage current, I_{off}, calculation.

intermediate feature parameters L_{elec} and T_{ox_elec}, and the saturated threshold voltage V_{th} are first calculated. In order to calculate I_{off}, the sub-threshold slope ($Slope$) is also calculated as,

$$Slope = \frac{kT}{q} ln_1 0(1 + \frac{\varepsilon_s \cdot T_{ox\_elec}}{\varepsilon_{ox} \cdot T_{dep}}) \tag{4.4}$$

where k is the Boltzmann constant, T is the temperature, q is electric unit, ε_s and ε_{ox} are the dielectric permittivities of silicon and oxide, respectively. With $Slope$, V_{th} and other parameters, we can then calculate I_{off} as,

$$I_{off} = I_{th} \frac{W}{L_{elec}} e^{-V_{th}/Slope} \tag{4.5}$$

where $I_{th} = 0.5uA$.

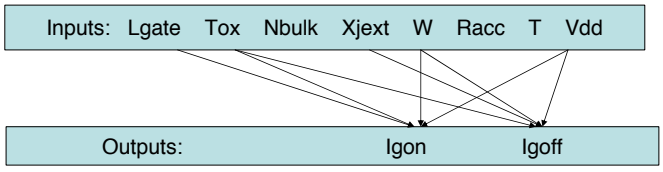


Figure 4.4: The flow for gate leakage currents I_{gon} and I_{goff} calculation.

Figure 4.4 shows the calculation flow for the gate leakage currents when the channel is on (I_{gon}) and off (I_{goff}), respectively. I_{gon} and I_{goff} depend on L_{gate}, T_{ox}, X_{jext}, W

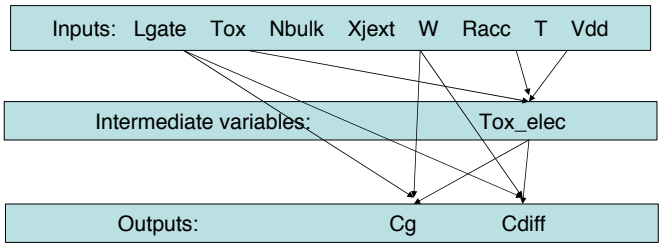


Figure 4.5: The flow for transistor gate and diffusion capacitances C_g and C_{diff} calculation.

and V_{dd}. In order to calculate I_{gon} and I_{goff}, we first calculate the gate leakage current density J_g as,

$$J_g = a_1 e^{a_2 V_g^2 + a_3 V g} e^{-a_4 T_{ox}} \tag{4.6}$$

where $a_1 = 1.44E5A/cm^2$, $a_2 = -4.02V^{-2}$, $a_3 = 13.05V^{-1}$ and $a_4 = 1/(1.17E-10m)$. With J_g, I_{gon} and I_{goff} are then calculated as,

$$I_{gon} = L_{gate} \cdot W \cdot J_g \tag{4.7}$$

$$\Delta L = L_{gate} - L_{elec} \tag{4.8}$$

$$I_{goff} = 0.5\Delta L \cdot W \cdot J_g \tag{4.9}$$

where ΔL is the difference between L_{gate} and L_{elec}.

Figure 4.5 shows the calculation flow for transistor gate and drain/source capacitances, C_g and C_{diff}, respectively. The capacitances depend on L_{gate}, T_{ox}, W, T and V_{dd}. The intermediate feature parameter T_{ox_elec} is first calculated for capacitance calculation. C_g and C_{diff} are calculated as,

$$C_g = (\frac{\varepsilon_{ox}}{T_{ox\_elec}} L_{gate} + C_{total\_fringing} + C_{overlap}) W \tag{4.10}$$

$$C_{diff} = C_{overlap} + C_{junc} \tag{4.11}$$

where the C_g calculation considers gate oxide capacitance, fringing capacitance $C_{total\_fringing}$

and overlap capacitance $C_{overlap}$, and C_{diff} calculation considers $C_{overlap}$ and junction capacitance C_{junc}.

4.3 Circuit-Level Delay and Power

In this section, we present basic circuit models for delay and power characteristics using the device model in Section 4.2. We consider buffers, LUT, SRAM, pass transistor gate, flip-flop (FF) and multiplexer [18] for the FPGA circuits, where the multiplexer is implemented as an NMOS pass transistor tree. Essentially, these FPGA circuits can be further decomposed into net-lists containing the most basic circuit elements, i.e. inverters and pass transistors. We therefore mainly discuss the power and delay for these basic circuit elements as below.

4.3.1 Delay Model

The pass transistor and multiplexer tree are modeled as a lumped capacitance, which is treated as part of the loading capacitance of an inverter. We calculate the inverter delay based on numerical integration through the transistor IV curves. In MASTAR4 tool, only the calculation for the maximum on current I_{on}, i.e. the current in velocity saturation region, is provided. We use the equations from [74] to calculate the drain-source current I_{ds} in different working regions, i.e. sub-threshold, linear, and saturation regions. I_{ds} is calculated as a function of drain, source, gate and body voltage levels as following. In the sub-threshold region, i.e. $V_{gs} < V_{th}$, I_{ds} is calculated as,

$$I_{ds} = I_{th} \cdot (W/L_{elec}) \cdot e^{((V_{gs}-V_{th})/Slope)} \tag{4.12}$$

where I_{th} is $0.5uA$, V_{gs} is the gate source voltage difference. In linear region, i.e. $V_{gs} > V_{th}$ and $V_{ds} < V_{gs} - V_{th}$, I_{ds} is calculated as,

$$I_{ds} = \frac{W}{L_{elec}} \cdot u_{eff} \cdot \frac{C_{ox\_elec}}{1 + V_{ds}/(E_c \cdot L_{elec})} \cdot (V_{gs} - V_{th} - V_{ds}/2) \cdot V_{ds} \tag{4.13}$$

where C_{ox_elec} is the gate oxide capacitance, V_{ds} is the drain source voltage difference and E_c is the critical electrical field. C_{ox_elec} and E_c are both calculated using equations in the ITRS MARSTAR4 model. In the saturation region, i.e. $V_{gs} > V_{th}$ and $V_{ds} > V_{gs} - V_{th}$, I_{ds} is calculated as,

$$I_{ds} = 0.5 \cdot \frac{W}{L_{elec}} \cdot u_{eff} \cdot \frac{C_{ox\_elec}}{1 + V_{ds}/(E_c \cdot L_{elec})} \cdot V_{ds}^2 \tag{4.14}$$

In velocity saturation region, i.e. $V_{gs} > V_{th}$ and $V_{ds} > V_{dsat}$, I_{ds} is equal to the on current I_{on} calculated by (4.3), where the velocity saturation voltage V_{dsat} is calculated using equations in the MASTAR4 tool. V_{th} in the above equations is calculated considering body-bias, short channel effect (SCE) and drain-induced barrier lowering (DIBL). Using these equations, Figure 4.7(c) shows the IV curves for an NMOS transistor under ITRS 2005 HP 32nm technology node.

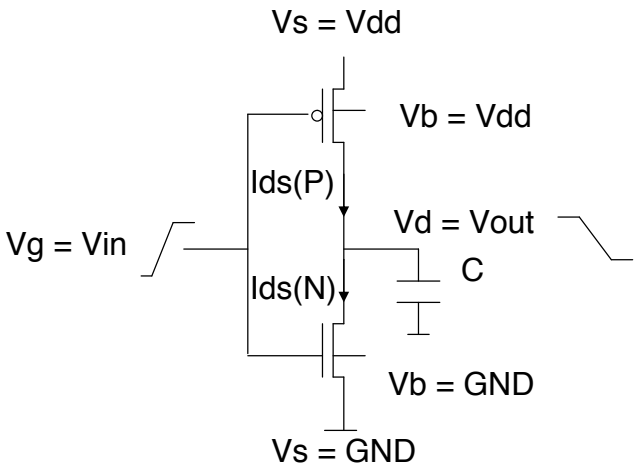


Figure 4.6: Voltage and current in an inverter during transition.

Given an inverter (see Figure 4.6), its loading capacitance C and the input voltage waveform, we can then calculate the inverter delay. At time t, we can obtain the transient PMOS and NMOS drain-source current $i_{ds}(P)$ and $i_{ds}(N)$, respectively. We can then perform numerical integration to obtain the output voltage waveform based on the following equation,

$$dV(out) = (i_{ds}(P) - i_{ds}(N)) \cdot dt/C \tag{4.15}$$

The delay is the time difference between when the output and input voltages reach $0.5V_{dd}$. We calculate the pull-down and pull-up delay for an inverter and then obtain the worst case delay as the inverter delay. Note that the input slew rate is automatically considered in this delay model. The output voltage waveform can be propagated for delay calculation of the next stage.

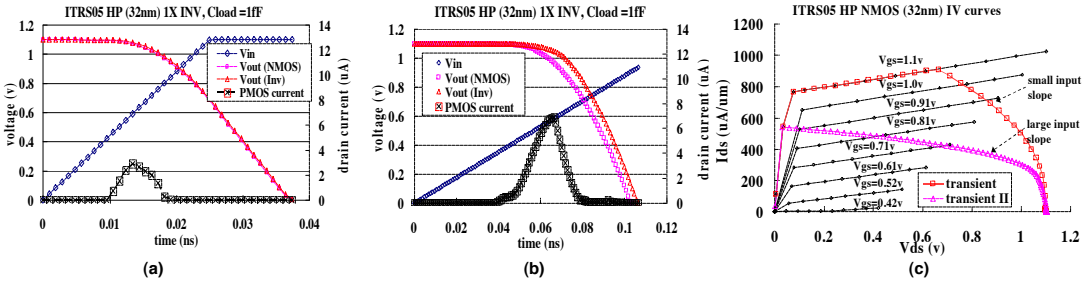


Figure 4.7: The pull-down delay of a 1X inverter at ITRS 2005 HP 32nm technology nodes. (a) Input and output voltages, and short circuit current with a small input slope; (b) Input and output voltages, and short circuit current with a large input slope; (c) The NMOS IV curves and the transition of transient current I_{ds} with small and large input slopes.

Figure 4.7 (a) shows the transient voltage transition of a 1X inverter with $1fF$ as loading capacitance C at ITRS 2005 HP 32nm technology node. The input voltage transition is from 0 to V_{dd}. The input slew rate is defined as the transition time from

$0.1V_{dd}$ (or $0.9V_{dd}$) to $0.9V_{dd}$ (or $0.1V_{dd}$). The short circuit current, i.e. the PMOS drain-source transient current is also shown in this figure. If this short circuit current is ignored, the output voltage waveform ($V_{out}(NMOS)$ in the figure) is almost overlapped the the waveform ($V_{out}(Inv)$ in the figure) considering the short circuit current. Figure 4.7 (b) shows the transient voltage transition of this inverter with a larger input slope, i.e. 5X large as the that in Figure 4.7 (a). The short circuit current becomes more significant due to a larger input slope and can no longer be ignored, i.e. the output voltage waveform without considering short circuit current differs the waveform considering short circuit current which results in a 20% delay difference. In FPGAs, the input voltage slope may be large due to a large loading capacitance. Therefore, the short circuit current is necessary to be considered for delay calculation. Figure 4.7(c) shows the NMOS transient drain-source current i_{ds} transitions with the two input slopes. With a larger input slope, the transition is slower with a larger delay. While not presented here, a similar trend for pull-up transition, i.e. input voltage transition is from V_{dd} down to 0, is observed.

ITRS MASTAR4 tool uses CV_{dd}/I_{on} to predict transistor delay. As shown in Figure 4.7, the input slope has a significant impact on the inverter delay. Since the FPGA circuit element usually has a large loading capacitance and a large input slope, our delay model is more accurate than that in MASTAR4. In addition, our delay model is flexible and can be extended to other complex gates easily.

For other FPGA circuit elements with loading capacitance, e.g. LUTs, FFs and buffers, we first breakdown them into the netlist of inverters and pass transistors. The delay of each circuit element is then calculated using the above method, e.g. the LUT delay is decomposed into the delay from LUT input passing through input buffers to the multiplexer control inputs and the delay from the SRAM passing through the multiplexer and the output buffer.

4.3.2 Power Model

In this section, we first discuss leakage power including sub-threshold and gate leakage power and then discuss dynamic power including switching power and short circuit power for basic circuit elements.

An inverter consumes both sub-threshold and gate leakage power, which depends on the input logic value. We calculate the average leakage power for an inverter, $P_{leak}(inv)$, as,

$$P_{leak}(inv) = V_{dd} \cdot (I_{off}(inv) + I_{gate}(inv)) \tag{4.16}$$

$$I_{off}(inv) = (I_{off}(P) + I_{off}(N))/2 \tag{4.17}$$

$$I_{gate}(inv) = \frac{I_{gon}(P) + I_{goff}(P) + I_{gon}(N) + I_{goff}(N)}{2} \tag{4.18}$$

where $I_off(P)$ and $I_{off}(N)$ are the PMOS and NMOS sub-threshold leakage currents, respectively, $I_{gon}(P)$, $I_{goff}(P)$, $I_{gon}(N)$ and $I_{goff}(N)$ are the PMOS and NMOS gate leakage currents when the channel is on and off, respectively. The two inverters in an SRAM cell are identical with one input as V_{dd} and one input as 0. Therefore, the average leakage calculation of an inverter can be applied to SRAM leakage calculation.

For an NMOS pass transistor, only gate leakage power is consumed, which can be either $V_{dd} \cdot I_{gon}(N)$ or $V_{dd} \cdot I_{goff}(N)$ depending on if the channel of this pass transistor is on or off. The pass transistor in a used/unused routing switch implemented by a tri-state buffer is on/off. For a multiplexer containing N NMOS pass transistors, $N/2$ of them are on while the other half are off. Based on the leakage model for inverters and pass transistors/muxes, we can calculate the leakage power for other FPGA circuit elements.

We consider switching and short circuit power for inverter dynamic power consumption. For switching power, we calculate the gate ($C_g(inv)$) and self-loading ca-

pacitances (C_{diff}) for an inverter as,

$$C_g(inv) = C_g(P) + C_g(N) \quad (4.19)$$

$$C_{diff}(inv) = C_{diff}(P) + C_{diff}(N) \quad (4.20)$$

where $C_g(P)$ and $C_g(N)$ are the PMOS and NMOS gate capacitances, respectively, $C_{diff}(P)$ and $C_{diff}(N)$ are the PMOS and NMOS diffusion capacitances, respectively. The input capacitance, internal capacitance and self-loading capacitance can then be easily extracted for each FPGA circuit element for switching power calculation purpose.

As shown in Figure 4.7 (a) and (b), short circuit current depends on input slew rate. The transient short circuit current has been calculated during delay calculation. We can simply perform a numerical integration on the transient short circuit current and obtain the short circuit energy per switch $SC(Sl)$, where Sl is the input slew rate.

4.4 Chip-level Delay and Power

With the delay and power model for basic circuits discussed in Section 4.3, we introduce the chip level delay and power estimation model in this section. In order to perform chip level estimation, we apply the similar idea as the trace-based estimation in Chapter 2. We collect the tract information in the same way as in Section 2.3, then calculate the chip level power and delay.

4.4.1 Delay Model

Similar to Section 2.3.2.3, we compute the critical path delay by adding the delay of all circuit elements in the critical path, i.e., LUT, wire segment, interconnect switch

buffer, and MUX:

$$D_{crit} = \sum_{i \in P} D_i \tag{4.21}$$

where D_i is the delay of the i_{th} circuit element in the critical path, which can be calculated by the circuit level delay model as discussed in Section 4.3.1. For each circuit element, the loading capacitance can be calculated from the in/out capacitance of the basic circuit elements as introduced in Section 4.3.2. For an interconnect wire segment, we use the Elmore delay model with resistance and capacitance suggested by ITRS [67].

4.4.2 Leakage Power Model

In the same way as in Section 2.3.2.2, the chip leakage power is modeled as the sum of the leakage power of all circuit elements:

$$P_{leak} = \sum_{i \in I} N_i^t P_i^s \tag{4.22}$$

where P_i^s is the leakage power for type i circuit element, which can be computed by circuit level leakage power model as discussed in Section 4.3.2.

4.4.3 Dynamic Power Model

Dynamic power includes switch power and short-circuit power. A circuit implemented on an FPGA cannot utilize all circuit elements. Dynamic power is only consumed by the used FPGA resources. Our switch power model distinguishes different types of used FPGA circuit elements and applies the following expression:

$$P_{sw} = \frac{1}{2} \cdot f \cdot V_{dd}^2 \cdot SW \cdot \sum_{i \in I} N_i^u \cdot C_i \tag{4.23}$$

where f is the operating frequency, V_{dd} is the supply voltage, SW is the average switching activity, and C_i is the total capacitance per switch of type i circuit elements. The

total capacitance per switch C_i can be computed from the in/out capacitance of a basic circuit element as discussed in Section 4.3.2. The switching activity is related to the input vector and logic, which is difficult to obtained without detailed simulation using a large set of input vectors. In our model, the average switching activity SW can be set by the users. We also assume that the operating frequency to be $5/6$ of the maximum frequency, that is $f = 1/(1.2 \cdot D_{crit})$. We do not set the operating frequency to maximum frequency since the actual critical path delay of some chips may increase due to process variation.

The short circuit power is related to the input slew rate. Here we model the chip short circuit power as:

$$P_{sc} = f \cdot SW \cdot \sum_{i \in T} \sum_{1 \leq j \leq N_i^u} SC_i(Sl_{i,j}) \tag{4.24}$$

where $SC_i(\cdot)$ is the function to compute short circuit energy per switch for type i circuit element and $Sl_{i,j}$ is the input slew rate of the j_{th} type i circuit element. The short circuit energy function $SC_i(\cdot)$ can be obtained from the short circuit power model of basic circuit elements as discussed in Section 4.3.2. Because of the regularity of FPGAs, the input slew rate for the same type of circuit elements is very close. Therefore, the chip short circuit power can be further simplified as:

$$P_{sc} = f \cdot SW \cdot \sum_{i \in T} SC_i(Sl_i) \tag{4.25}$$

where Sl_i is the input slew rate of type i circuit element, which is computed as the output slew rate of the element driving type i element. For example, a connection box buffer is driven by a global interconnect buffer, then the input slew rate of a connection box buffer is the output slew rate of the global interconnect buffer. Such output slew rate can be computed from the basic circuit element delay model discussed in Section 4.3.1.

With the switch power and short circuit power, the total dynamic power is computed as:

$$P_{dynamic} = P_{sw} + P_{sc} \tag{4.26}$$

Notice that dynamic power model in this chapter is different from that in Section 2.3.2.1. In Section 2.3.2.1, short circuit power is calculated as a ratio to switching power, while in this chapter, we calculate the circuit level short circuit power directly from transistor model. Therefore, the short circuit power model introduced in this section is more accurate than that in Section 2.3.2.1.

4.5 Chip Level Delay and Power Variation

Based on the chip-level power and delay model introduced in Section 4.4, we study the impact of process variation on power and delay in this section. First we introduce the variation model as below.

4.5.1 Variation Models

In general, delay and leakage power of a circuit element are functions of the underlying process parameters and they can be described as:

$$D_t = F_t^D(X_1, X_2, ...) \tag{4.27}$$

$$P_t = F_t^L(X_1, X_2, ...) \tag{4.28}$$

where F_t^L and F_t^D are functions of process parameters to calculate leakage power and delay for type t circuit element, respectively, and the process variation sources (such as gate change length and dopant density) are modeled as a random variable X. To evaluate the chip-level leakage power and delay considering process variation, we first

apply randomly generated process parameters to the device model as in Section 4.2 and then calculate the leakage power and delay for basic circuit elements as in Section 4.3. The sensitivity functions F_i^L and F_i^D can be obtained from the leakage power and delay model of basic circuit elements discussed in Section 4.3.

For each variation source X, all inter-die, intra-die spatial, and intra-die random variation is considered. That is:

$$X = X_g + X_s + X_r \tag{4.29}$$

where X_g, X_s and X_r are the inter-die, intra-die spatial, and intra-die random variation for variation source X. We assume that X_g, X_s, and X_r are independent from each other. Each circuit element has its own random variation X_r, and all circuit element of the whole chip share the same inter-die variation X_g. We use the grid based model in [173] to model the spatial variation. A chip is divided into several grids, all circuit elements in the same grid share the same spatial variation X_s and the spatial variation between different grids are correlated. The correlation coefficient decreases when the distance between two grids increases.

4.5.2 Delay Variation

With process variation, some close-to-be critical path may become critical path. Therefore, in order to analyze delay variation, circuit delay is calculated as the maximum of a set of near critical paths:

$$D_{var} = \max_{j \in C} D_j \tag{4.30}$$

where C is the set of close-to-be critical paths and D_j is the delay of the j_{th} close-to-be critical path, which is calculated as:

$$D_j = \sum_{i \in P_j} F_{ji}^D(X_1^{ji}, X_2^{ji}, ...) \tag{4.31}$$

where P_j is the set of path elements of the j_{th} path and $F_{ji}^D(\cdot)$ is the delay variation function for the i_{th} circuit element in P_j. Because there is no closed-form formula for the delay variation function $F_{ji}^D(\cdot)$, we do not have closed-form formula to compute the chip delay distribution. In this chapter, we use the Monte-Carlo simulation to obtain the chip delay distribution. We first generate M samples of variation sources $(X_1^{ji}, X_2^{ji}, ...)$ for all i, and then compute the the path delay under such samples to obtain M samples of path delay. Finally we can get the delay distribution from those M samples of path delay. The Monte-Carlo simulation allows us to consider non-Gaussian variation sources and spatial correlation, which is ignored in the previous Ptrace [167].

4.5.3 Chip Leakage Power Variation

The chip leakage power with process variation is modeled as the sum of leakage power of all circuit elements:

$$P_{leak} = \sum_{i \in T} \sum_{j=1}^{N_i^t} P_i^j = \sum_{i \in T} \sum_{j=1}^{N_i^t} F_i^L(X_1^{ij}, X_2^{ij}, ...) \tag{4.32}$$

where P_i^j is the leakage power of the j_{th} type i circuit element, and X^{ij} is the variation of the j_{th} type i circuit element.

Similar to the chip delay variation model, we use Monte-Carlo simulation to compute chip leakage distribution. We first generate M samples of variation sources and then compute the M samples of chip leakage power. However, because the random variation is unique for each circuit element, when the circuit size becomes large, a large number of random variation samples need to be computed. This makes the simulation very inefficient. In order to solve such problem, we simplify the chip leakage power variation model by applying central limit theorem similar to [25, 168].

Given the inter-die and within-die spatial variation, the total leakage power for all

type i circuit elements in the l_{th} grid can be calculated as:

$$P^i_{leak} = \sum_{j=1}^{N^t_{li}} F^L_i(X_{g1} + X^l_{s1} + X^{ilj}_{r1}, X_{g2} + X^{ilj}_{r2}, ...) \tag{4.33}$$

where X_g is the inter-die variation, X^l_s is the spatial variation in the l_{th} grid, X^{ilj}_r is the random variation for the j_{th} type i circuit elements in the l_{th} grid, N^t_{li} is the number of type i circuit element in the l_{th} grid. Usually, N^t_{li} is a very large number. Therefore, by applying the central limit theorem, we have:

$$\sum_{j=1}^{N^t_{li}} F^L_i(X_{g1} + X^l_{s1} + X^{ilj}_{r1}, X_{g2} + X^{ilj}_{r2}, ...) \approx N^t_i \cdot \mu_i(X_{g1} + X^l_{s1}, X_{g2} + X^l_{s2}, ...) \tag{4.34}$$

where

$$\begin{aligned} & \mu_{il}(X_{g1} + X^l_{s1}, X_{g2} + X^l_{s2}, ...) \\ = \;& E[F^L_i(X^i_{g1} + X^{ij}_{r1}, X^i_{g2} + X^{ij}_{r2}, ...) | X_{g1}, X^l_{s1}, X_{g2}, X^l_{s2}, ...] \end{aligned} \tag{4.35}$$

In order to compute $\mu_{il}(\cdot)$, we first generate M_r samples of random variations, and then compute the mean of leakage power under such samples. That is:

$$\begin{aligned} & \mu_{il}(X_{g1} + X^l_{s1}, X_{g2} + X^l_{s2}, ...) \\ = \;& \frac{1}{M_r} \sum_{k=1}^{M_r} F^L_j(X_{g1} + X^l_{s1} + X^k_{r1}, X_{g2} + X^l_{s1} + X^k_{r2}, ...) \end{aligned} \tag{4.36}$$

where x^k_r is the k_{th} sample of random variation. Here, notice that usually M_r is much smaller than N^t_{li} and does not depend on circuit size. Therefore such method is scalable to large circuit size. Then the chip leakage power can be modeled as:

$$P_{leak} = \sum_{i \in T} \sum_{l=1}^{G} N^t_{li} \cdot \mu_{il}(X_{g1} + X^l_{s1}, X_{g2} + X^l_{s2}, ...) \tag{4.37}$$

where G is the number of grids. With the above equation, we may generate M samples of inter-die and within-die spatial variations to compute the distribution of leakage power.

Notice that the variation model introduced in this section may handle non-Gaussian variation sources and spatial correlation, which are ignored in the variation model in Section 3.2.

4.6 Process Evaluation

Based on the chip level power and delay model presented in Section 4.4 and 4.5, we perform device tuning to minimize power and delay.

4.6.1 Power and Delay Optimization

First we discuss the optimization for nominal value of power and delay. In this chapter, we consider two device parameters, supply voltage Vdd and the physical gate length L_{gate}. In our experiment, we start from the device setting of ITRS High Performance 32nm technology (*HP32*) [67] (with Vdd=1.1V, L_{gate}=32nm), and then change Vdd and L_{gate} around such setting. Moreover, we assume that the FPGA has cluster size N=6, LUT size K=7, and the global interconnect wire length W=4, which is optimized for high performance [91]. For dynamic power estimation, we assume that the switching activity SW=0.5. In addition, we also consider heterogeneous L_{gate} for logic blocks (L_c) and interconnects (L_i). Because SRAMs have little impact on FPGA delay, we assume that all SRAMs use high V_{th} (1.5X dopant density compared to the other circuit elements) and fix L_{gate}=32nm. During our evaluation, Vdd is tuned from 1.0V to 1.1V, L_{gate} (both L_c and L_c) is tuned from 31nm to 33nm. The experimental setting is summarized in Table 5.1. In our experiment, we assumed that 20 MCNC benchmarks [175] are put in one FPGA chip. Therefore, the power is computed as the sum of all 20 benchmarks and delay is computed as the longest critical path delay of 20 benchmarks.

N	K	W	SW	Vdd (V)	L_c (nm)	L_i (nm)
6	7	4	0.5	1.0, 1.05, 1.1	31, 32, 33	31, 32, 33

Table 4.1: Experimental setting.

Figure 4.8 shows the energy per clock cycle and delay tradeoff between different device settings. From the figure, we find that there is a 3X energy span and a 1.3X delay span within our search space. We also find that that the knee point in the energy delay tradeoff figure is exactly *HP32*, which has high performance without paying too much power penalty. On the other hand, we also optimize device setting by minimizing energy and delay product (ED). We show the top 2 minimum ED device settings in the figure, which we refer to as *min-ED1* and *min-ED2*.

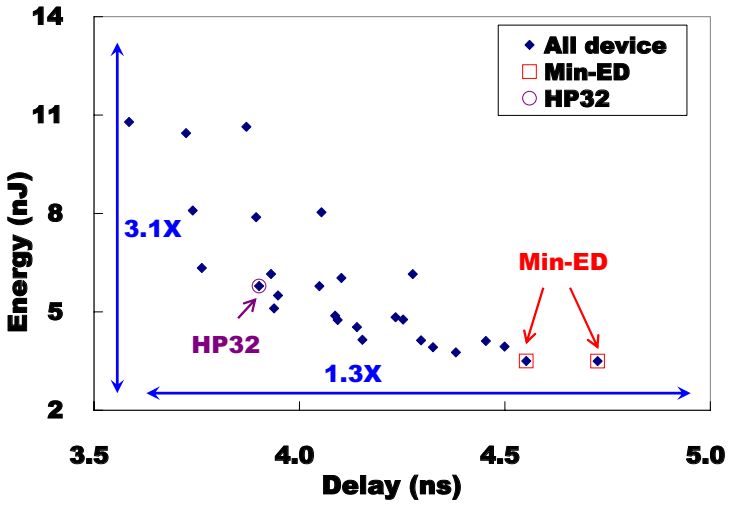


Figure 4.8: Energy and delay tradeoff.

Table 4.2 shows the power (P), delay (D), energy per clock cycle (E) , and energy delay product (ED) for *HP32* and min-ED device settings. From the table, we see that by applying device tuning, ED can be reduced by up to 29.4% (*min-ED1*) compared to *HP32*. We also find that it is worthwhile to apply heterogeneous L_{gate}. The optimum heterogeneous L_{gate} device setting (*min-ED1*) has lower ED (0.7nJ·ns lower) than the

optimum homogeneous L_{gate} device setting (*min-ED2*).

Device	Vdd (V)	L_c (nm)	L_i (nm)	P (W)	D (ns)	E (nJ)	ED (nJ·ns)
HP32	1.1	32	32	1.19	3.90	1.88	22.6
min-ED1	1.0	32	33	0.77	4.55	3.50	15.9(-29.4%)
min-ED2	1.0	33	33	0.74	4.73	3.50	16.5(-26.8%)

Table 4.2: Power, delay, energy, and ED comparison for different device settings.

4.6.2 Variation Analysis

In this section, we apply the chip level variation model as introduced in Section 4.5 to analyze the chip level leakage and delay variation. In our experiment, we consider four types of variation sources, dopant density (N_{bulk}), channel gate length (L_{gate}), oxide thickness (T_{ox}), and mobility (μ_{eff}) [190]. Moreover, because for technology under 65nm, the spatially correlated intra-die variation is small compared to the intra-die random variation [190], therefore, we ignore spatial variation in this chapter. For all variation sources, we assume that they follow normal distribution. Notice that our process variation model is based on Monte-Carlo simulation, therefore it can handle any non-Gaussian variation sources. The 3-sigma value of inter-die ($3\sigma_g$), intra-die spatial $3\sigma_s$, and intra-die random ($3\sigma_r$) variation for all types of variation sources are summarized in Table 4.3. In the table, the 3-sigma values are the represented as the percentage of nominal value. Moreover, for spatial variation, we assume that the correlation coefficient decreases to 1% when distance is 1cm ($D_{corr} = 1cm$). In our experiment, we consider two different variation settings, low variation (*LV*) and high variation (*HV*). We generate $M = 10,000$ samples for Monte-Carlo simulation and use $M_r = 100$ samples to compute the leakage mean $\mu_i(\cdot)$.

Figure 4.9 illustrates the probability density function (PDF) of delay and leakage power for *min-ED1* and *HP32* under variation setting *LV*. From the figure, we see that

$M = 10,000$		$M_r = 100$			$D_{corr} = 1cm$		
Source	Distribution	*LV*			*HV*		
		$3\sigma_g$	$3\sigma_s$	$3\sigma_r$	$3\sigma_g$	$3\sigma_s$	$3\sigma_r$
N_{bulk}	Normal	4.0%	4.0%	2.0%	6.0%	6.0%	4.0%
L_{gate}	Normal	2.0%	2.0%	1.5	3.0%	3.0%	2.2%
T_{xo}	Normal	2.0%	2.0%	1.5	3.0%	3.0%	2.2%
μ_{eff}	Normal	4.0%	4.0%	2.0%	6.0%	6.0%	4.0%

Table 4.3: Experimental setting for process variation.

min-ED1 has much less leakage variation compared to *HP32* while its delay variation is only a little bit larger than *HP32*. We also observe that the leakage roughly has a log-normal distribution and delay roughly has a normal distribution. This is because leakage power is almost exponential to the variation sources while delay is almost linear to the variation sources.

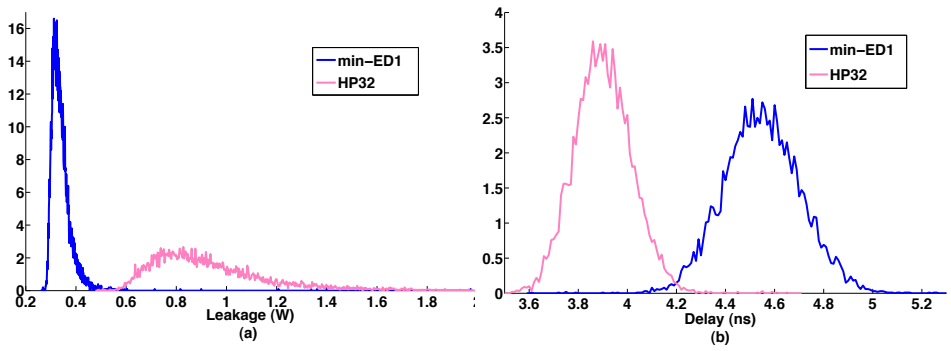


Figure 4.9: Delay and leakage distribution. (a) Leakage PDF; (b) Delay PDF.

Table 4.4 summarizes the nominal value (nom), mean (μ), and standard deviation (σ) of leakage power and delay for the min-ED device settings and *HP32*. From the table, we observe that compared to *HP32* the min-ED device settings greatly reduce leakage variation (reduce standard deviation by up to 92%) with only a small increase of delay variation (increase standard deviation by up to 37%). From the table, we also see that *HP32* is more sensitive to process variation than the min-ED device settings.

Device	Leakage (mW)					Delay (ns)				
	nom	*LV*		*HV*		nom	*LV*		*HV*	
		μ	σ	μ	σ		μ	σ	μ	σ
HP32	854	982	348	1120	575	3.90	3.95	0.122	3.99	0.185
min-ED1	328	359	51 (-86%)	398.2	68 (-88%)	4.55	4.55	0.162(+33%)	4.58	0.244(+32%)
min-ED2	315	328	32 (-89%)	375.2	48 (-92%)	4.73	4.779	0.164(+37%)	4.782	0.245(+32%)

Table 4.4: Nominal value, mean, and standard deviation comparison for leakage power and delay.

4.7 FPGA Reliability

As technology scales down to nanometer, reliability issues such as device aging with V_{th} degradation and soft error rate (*SER*) due to high-energy particle or cosmic rays become more and more significant. In this section we study these two types of reliability for FPGA, i.e. device aging and permanent soft error rate (*SER*).

4.7.1 Impact of Device Aging

First, we introduce the impact of device aging effect on FPGA power and delay. It has been shown that negative-bias-temperature-instability (*NBTI*) effect increases the threshold voltage of PMOS transistor [11][160][149][23], and hot-carrier-injection (*HCI*) increases the threshold voltage (V_{th})of NMOS transistor [34][149][166]. Due to the effect of NBTI, PMOS V_{th} increases during stress mode, i.e. with input 0, and recovers during recovery mode, i.e. with input V_{dd}. (4.38) and (4.39) are the equations for PMOS ΔV_{th} over time in the stress and recovery modes, respectively.

$$\Delta V_{th} = (K_v(t-t_0)^{0.5} + (\Delta V_{th0})^{(1/2n)})^{2n} \tag{4.38}$$

$$\Delta V_{th} = -\Delta V_{th0}(1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C(t-t_0)}}{2t_{ox} + \sqrt{Ct}}) \tag{4.39}$$

(4.40) is the equation for NMOS ΔV_{th} over time due to HCI.

$$\Delta V_{th} = \frac{q}{C_{ox}} K_2 \sqrt{Q_i} e^{\frac{E_{ox}}{E_{o2}}} e^{-\frac{\phi_{it}}{q\lambda E_m}} t^{n'} \tag{4.40}$$

The key parameters that determine the degradation rate include the inversion charge, Q_i, electrical field, E_{ox}, temperature, supply voltage V_{dd} and nominal threshold voltage V_{th}. Due to the space limit, we do not include detailed explanation for each parameters. Interested readers please refer to [23] for more details. Figure 4.10 shows the calculation flow for ΔV_{th} due to NBTI and HCI. We first calculate the effective gate length L_{elec} and oxide thickness T_{ox_elec} and then calculate V_{th}. With these intermediate parameters, we can obtain ΔV_{th} due to device aging over time. A higher V_{dd} or lower V_{th} leads to larger V_{th} increase. The V_{th} degradation due to device aging results in increase of critical path delay and decrease of leakage power.

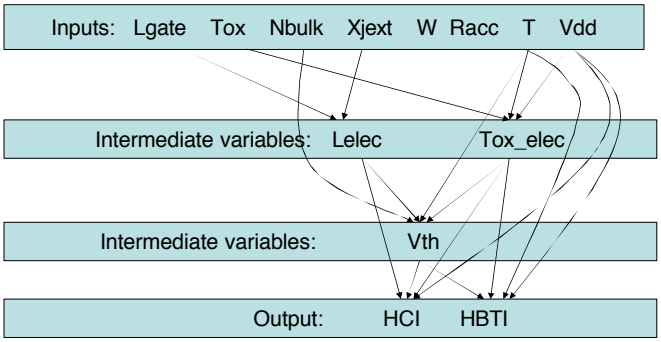


Figure 4.10: The calculation flow for ΔV_{th} due to NBTI and HCI.

Figure 4.11 illustrates the V_{th} increase (ΔV) for PMOS and NMOS transistors of both *min-ED1* and *HP32* within 10 years. From the figure, we see that *HP32* is much more sensitive to NBTI and HCI than *min-ED1*. This is due to the fact that *HP32* uses higher V_{dd} and lower V_{th} (due to smaller L_{gate}) than *min-ED1*. Hence *HP32* has larger ΔV_{th} for both NMOS and PMOS than *min-ED1*.

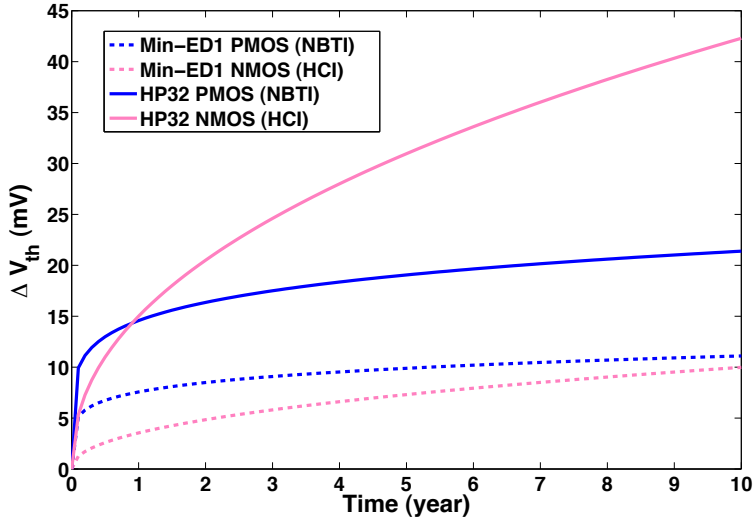


Figure 4.11: V_{th} increase caused by NBTI and HCI.

Figure 4.12 illustrates the critical path delay increase and chip leakage power decrease within 10 years for *min-ED1* and *HP32*. It is interesting to see in the figure that the leakage and delay change is most significant at the first one year. Therefore, device burn-in [96] that has been used for microprocessors but not FPGAs yet could be used to reduce leakage and delay change over time after product shipment.

Table 4.5 illustrates the current value, value after 1 year, and value after 10 years of leakage power (L) and delay (D). In the table, we also compare the change percentage with and without burn-in (burn to 1 year). The change percentage without burn-in is computed as the percentage of the difference between the 10 year value and current value, and that with burn-in is computed as the percentage of the difference between the 10 year value and the 1 year value. From the figure and table, we find that the min-ED device settings is less sensitive to device aging compared to *HP32*. This is because *HP32* has higher V_{dd} and smaller L_{gate} compared to the min-ED device settings. Therefore it has larger V_{th} degradation, hence more sensitive to device aging. We may see that compare to *HP32*, min-ED device settings can not only reduce ED but also

Device	Current		1 Year		10 Year		Change			
	L	D	L	D	L	D	w/o burn-in		w/ burn-in	
	(mW)	(ns)	(mW)	(ns)	(mW)	(ns)	L	D	L	D
HP32	854	3.90	711	4.01	640	4.23	-25.1%	+8.5%	-10.0%	+5.5%
min-ED1	328	4.55	317	4.59	311	4.64	-5.2%	+2.0%	-1.9%	+1.1%
min-ED2	315	4.73	307	4.76	302	4.82	-2.5%	+1.9%	-1.6%	+1.3%

Table 4.5: Delay and leakage change due to device aging.

increase aging reliability of FPGAs. Moreover, by applying burn-in, we can greatly reduce the delay degradation. For example, for *HP32*, burn-in reduces 10 year delay degradation from 8.5% to 5.5%.

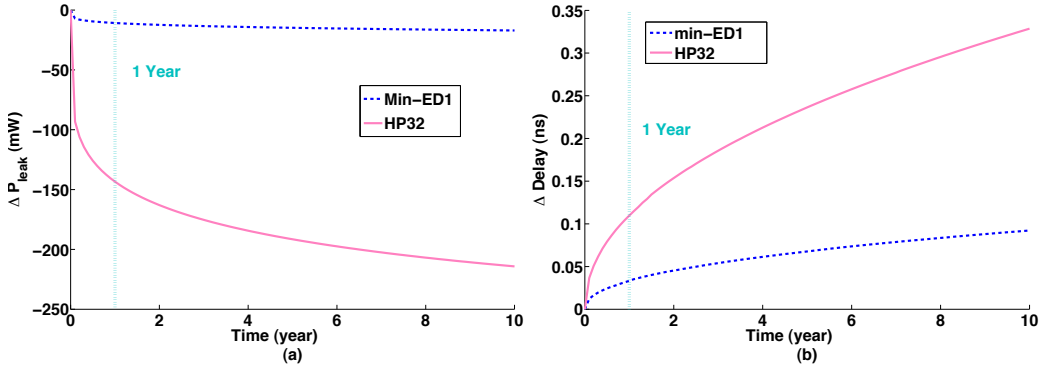


Figure 4.12: Impact of device aging. (a) Leakage change; (b) Delay change.

4.7.2 Permanent Soft Error Rate (SER)

We use the model in [60] to calculate the soft error rate (SER) for configuration SRAMs in our study. The SER in SRAMs is permanent in FPGAs, which cannot be recovered unless re-writing those affected configuration bits [14][17]. The SER for one SRAM cell is shown in (4.41),

$$SER(SRAM) = F \cdot A \cdot K \cdot e^{-Q_{crit}/Q_s} \tag{4.41}$$

where F is the neutron flux, A is the transistor drain area, K is a technology independent constant and is same for all device settings, Q_{crit} is the critical charge to incur an error, and Q_s is the charge collection slope. Figure 4.13 shows the flow to calculate the SRAM SER. We first calculate intermediate parameters including L_{elec}, T_{ox_elec} and V_{th}. Q_s, Q_{crit} and A are then calculated. With all these intermediate parameters, we can obtain the SRAM SER. A transistor with a larger Q_{crit} needs more energy to incur an error and has a smaller SER. Q_{crit} mainly depends on supply voltage V_{dd} and loading capacitance C, and slightly depends on V_{th}. On the other hand, a transistor with a larger Q_s is more effective in collecting charge and has a larger SER. Q_s depends on channel doping density N_{bulk} and V_{dd}. We use the models in [60] with linear interpolation to calculate Q_{crit} and Q_s under different device settings, i.e. different V_{dd} and V_{th}, and then calculate the SER of an SRAM cell correspondingly using (4.41). The chip-level permanent SER can be calculated as,

$$SER = Num\_SRAM \cdot SER(SRAM) \tag{4.42}$$

where Num_SRAM is the total number of SRAM cells.

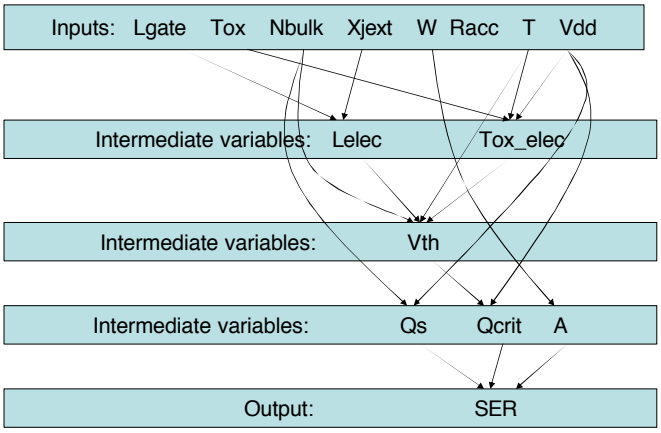


Figure 4.13: The flow for SRAM SER calculation.

Table 4.6 compares the SER for *HP32*, and the two min-ED device settings. For

the min-ED device settings, L_{gate} for SRAM cells is still 32nm. Therefor only V_{dd} may affect SER. In the table, SER is measured as number of failures in billion hours (*FIT*). The supply voltage V_{dd} in *HP32* is higher, hence the critical charge, Q_{crit}, is larger and may result in a smaller SER. For *min-ED1/2* with V_{dd} of 1.0v, SER is 1.6% higher than that of *HP32*. It is worthwhile to use Min-ED device settings to obtain a significant ED reduction with virtually no impact on SER.

Device	V_{dd} (V)	# SRAMs	SER (FIT)
HP32	1.1	12438596	362.45
min-ED1/2	1.0	12438596	368.25 (+1.6%)

Table 4.6: Soft error rate comparison.

4.8 Interaction between Process Variation and Reliability

In this section, we study the impact of device aging on process variation and the impact of device aging and process variation on SER.

4.8.1 Impact of Device Aging on Power and Delay Variation

First, we discuss the impact of device aging on process variation induced leakage and delay variation. Figure 4.14 compares the PDF of delay and leakage between current value and the value after 10 years for the device setting *min-ED1*. From the figure, we find that device aging actually reduce the leakage variation. But device aging only increase delay but has no significant impact on delay variation.

Table 4.7 summarizes the mean and standard deviation of leakage and delay for different device settings. The table compares both the current value and the value after 10 years in both low and high variation cases. From the table, we find that device aging has great impact of leakage power variation. For *HP32* device setting under low

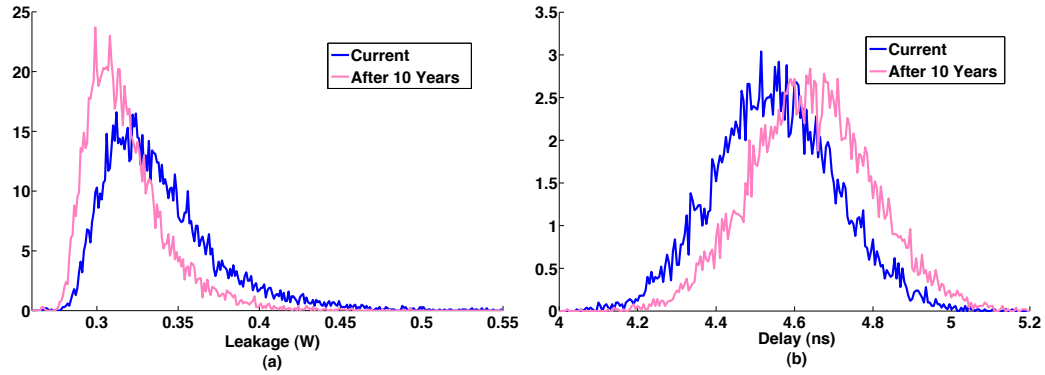


Figure 4.14: Impact of device aging on delay and leakage PDF. (a) Leakage PDF comparison; (b) Delay PDF comparison.

variation case (*LV*), device aging reduces mean by 28.8% and standard deviation by 65.2%. However, the impact of device aging on delay variation is relatively small. For *HP32* device setting, device aging only increases standard deviation by 1.7%. This is because leakage power is rough exponential to V_{th} while delay is roughly linear to V_{th}. Moreover, we observe that compared to *HP32* device setting, device aging has less impact on power and delay variation for min-ED settings. This is because *HP32* is more sensitive to device aging as discussed in Section 4.7. From the result, we also find that in high variation case, the impact of device aging on delay variation is similar to that of the regular variation case (*min-ED1*). However, when variation is high, the impact of device aging on leakage variation is much larger than that in the regular variation case. That is, when variation scale becomes larger, there is a larger impact of device aging on leakage variation.

4.8.2 Impacts of Device Aging and Process Variation on SER

As shown in Figure 4.13, the SRAM SER depends on V_{dd}, loading capacitance, V_{th} and channel doping density N_{bulk} while V_{th} and N_{bulk} may be affected by device aging and

Device	Leakage (mW)				Delay (ns)			
	current		10 years		current		10 years	
	μ	σ	μ	σ	μ	σ	μ	σ
HP32 LV	982	348	678 (-28.8%)	118 (-65.2%)	3.95	0.122	4.23 (+8.4%)	0.121 (+1.67%)
min-ED1 LV	359	51.0	321 (-6.2%)	31.2 (-32.7%)	4.55	0.162	4.65 (+2.0%)	0.162 (+0.16%)
min-ED2 LV	328	32.0	309 (-4.6%)	22.4 (-30.4%)	4.78	0.164	4.83 (+1.9%)	0.164 (+0.25%)
HP32 HV	1120	575	712 (-36.4%)	158 (-72.5%)	3.99	1.85	4.09 (+2.6%)	1.86 (+0.54%)
min-ED1 HV	398.2	68	345.1 (-13.3%)	38.1 (-44.0%)	4.55	0.318	4.66 (+2.1%)	0.319 (+0.28%)
min-ED2 HV	375.2	48	332 (-12.4%)	25 (-47.9%)	4.78	0.245	4.79 (+0.21%)	0.245 (+0.21%)

Table 4.7: Impact of device aging on process variation.

process variation. We perform the study on the impacts of device aging and process variation on SER as below.

V_{th} may increase with device aging due to NBTI and HCI. For ITRS HP 32nm technology (see Figure 4.11, V_{th} may degrade (or increase) by around 10% (or 20mV) for PMOS due to NBTI and by around 25% (or 50mV) for NMOS due to HCI after 10 years. A larger V_{th} may result in a slightly smaller critical charge Q_{crit} for an SRAM and therefore a larger SER. Table 4.8 presents the impact of device aging on SER. After 10 years, the SRAM SER may increase by 0.3% due to degraded V_{th}. Even if V_{th} degrades by 100mv for both NMOS and PMOS, the SRAM SER only increases by 0.9%.

	0 year or nominal	10 years	3σ=6% variation in N_{bulk}
SRAM SER (FIT)	2.914E-5	+0.3%	-0.18% to +0.17%

Table 4.8: The impact of device aging and process variation on SER of one SRAM under ITRS HP 32nm technology.

With process variation, N_{bulk} and channel gate length L_{gate} become random variables. N_{bulk} variation directly affects charge collection slope Q_s and therefore affects SER. L_{gate} variation may affect V_{th} and therefore implicitly affect SER. For ITRS HP 32nm technology, a variation in L_{gate} from 31nm to 33 nm may result in a variation

in V_{th} from -25% to 21% [68]. It has been shown that V_{th} variation does not have a significant impact on SER. Table 4.8 presents the impact of N_{bulk} variation on SER. A variation of 3σ as 6% in N_{bulk} only results in a variation in SER from -0.18% to 0.17%. A larger N_{bulk} may result in a larger charge collection slope Q_s and therefor a smaller SER. On the other hand, a larger N_{bulk} may result in a higher V_{th} and a smaller critical charge Q_{crit}, therefore a larger SER. These two effects compensate with each other and therefore N_{bulk} does not have a significant impact on SER either. It is clear that neither device aging due to NBTI and HCI nor process variation have any significant impact on SER.

4.9 Conclusions

In this chapter, we have developed a trace-based framework to enable concurrent process and FPGA architecture co-development. Same as the MASTAR4 model used by the 2005 ITRS [67][68][148], the user can tune eight parameters for bulk CMOS processes and obtain the chip level performance and power distribution and soft error rate (SER) considering process variations and device aging. The framework is efficient as it is based on closed-form formulas. It is also flexible as process parameters can be customized for different FPGA elements and no SPICE models and simulation are needed for these elements. Therefore, this framework is suitable for early stage process and FPGA architecture co-development.

A few examples of using this framework have been presented. We show that applying heterogeneous gate lengths to logic and interconnect may lead to 1.3X delay difference, 3.1X energy difference, and reduce standard deviation of leakage variation by 87%. This offers a large room for power and delay tradeoff. We further show that the device aging has a knee point over time, and device burn-in to reach the point could reduce the performance change over 10 years from 8.5% to 5.5% and reduce die

to die leakage significantly. In addition, we also study the interaction between process variation, device aging and SER. We observe that device aging reduces standard deviation of leakage by 65% over 10 years while it has relatively small impact on delay variation. Moreover, we also find that neither device aging due to NBTI and HCI nor process variation have significant impact on SER.

Part II

Statistical Timing Modeling and Analysis

CHAPTER 5

Non-Gaussian Statistical Timing Analysis Using Second-Order Polynomial Fitting

In Part I, we have discussed statistical modeling and optimization for FPGAs. In Part II of this dissertation, we study statistical modeling and analysis for ASICs. In this chapter, we focus on statistical static timing analysis (SSTA). Lots of research works on SSTA have been published in recent years. However, most of the existing SSTA techniques have difficulty in handling the non-Gaussian variation distribution and non-linear dependency of delay on variation sources. To solve this problem, we first propose a new method to approximate the max operation of two non-Gaussian random variables through second-order polynomial fitting. Applying the approximation, we present present new non-Gaussian SSTA algorithms for three delay models: quadratic model, quadratic model without crossing terms (semi-quadratic model), and linear model. All the atomic operations (max and sum) of our algorithms are performed by closed-form formulas, hence they scale well for large designs. Experimental results show that compared to the Monte-Carlo simulation, our approach predicts the mean, standard deviation, skewness, and 95-percentile point within 1%, 1%, 6%, and 1% error, respectively.

5.1 Introduction

Process variation introduces significant uncertainty to circuit delay as discussed in Chapter 1. In Part I, we have studied statistical optimization for FPGA circuits. Compared to FPGA, ASIC has higher performance, lower power, and smaller area. Therefore, ASICs are widely used in high performance or low power applications, where the impact of process variation is significant.

To analyze the impact of process variation on circuit delay, statistical static timing analysis (SSTA) is developed for full chip timing analysis under process variation. By performing SSTA, designers can obtain the timing distribution and its sensitivity to various process parameters.

In recent years, two types of SSTA techniques are proposed, the path-based [103, 72, 7, 120, 157, 128] and the block-based [32, 5, 49, 9, 22, 8, 163, 86, 53, 182, 13, 181, 113, 75, 180, 178, 76, 3, 183, 2, 142, 19, 179, 40] SSTA. Since the number of paths is exponential with respect to the circuit size, the path-based SSTA is not scalable to large circuits. The block-based SSTA was proposed to solve this problem. The goal of block-based SSTA is to parameterize timing characteristics of the timing graph as a function of the underlying sources of process parameters which are modeled as random variables.

The early SSTA methods [32, 163] modeled the gate delay as linear functions of variation sources and assumed all the variation sources are mutually independent Gaussian random variables. Based on such assumption, [163] presented time efficient methods with closed-form formulas [46] for all atomic operations (max and sum). However, when the amount of variation becomes larger, the linear delay model is no longer accurate [94]. In order to capture the non-linear dependence of delay on the variation sources, a higher-order delay model is thus needed [180, 178].

As more complicated or large-scale variation sources are taken into account, the assumption of Gaussian variation sources is also not valid. For example, the via resistance has an asymmetric distribution [13], while dopant concentration is more suitably modeled as a Poisson distribution [142] than Gaussian. Some of the most recent works on SSTA [76, 142, 13, 19, 179, 40] started to take non-Gaussian variation sources into account. For example, [142] applied independent component analysis to de-correlate the non-Gaussian random variables, but it was still based on a linear delay model. [76, 13, 19, 179] considered both non-linear delay model and non-Gaussian variation sources, but the computation cost of these techniques is too high to be applicable for large designs. For example, [76] proposed to compute the max operation by a regression based on Monte-Carlo simulation, which is slow. [13] computed the max operation through tightness probability while [179] applied moment matching to reconstruct the max. However, to do so, both had to resort to expensive multi-dimension numerical integration techniques. [19] handled the atomic operations by approximating the gate delay using a set of orthogonal polynomials, which needs to be constructed for different variation distributions. [40] proposed to approximate the probability density function (PDF) of max results as a Fourier Series, but it lacks the capability to handle the crossing term effects on timing.

In this chapter, we introduce a time efficient non-linear SSTA for arbitrary non-Gaussian variation sources. The major contribution of this work is two-fold. (1) We propose a new method to approximate the max of two non-Gaussian random variables as a second-order polynomial function based on least-square-error curve fitting. Experimental results shows that such approximation is much more accurate than the linear approximation based on tightness probability [163, 180, 13]. (2) Based on the new approximation of the max operation, we present our SSTA technique for three different delay models: quadratic delay model, quadratic delay model without crossing terms (*semi-quadratic model*), and linear delay model. In our method, only the

first few moments are required for different distributions and no extra effort is needed, while [19] needs to obtain different sets of orthogonal polynomials for different variation distributions. Moreover, all the atomic operations are performed by closed-form formulas, hence they are very time efficient. For the linear and semi-quadratic delay model, the computational complexity of our method is linear in both the number of variation sources and circuit size. For the quadratic delay model, the computational complexity is cubic (third-order) to the number of variation sources and linear to the circuit size. Experimental results show that for the semi-quadratic delay model, our approach is 70 times faster than the non-linear SSTA in [40], and indicates less than 1% error in mean, 1% error in standard deviation, 30% error in skewness, and 1% error in 95-percentile point. Moreover, for the more accurate quadratic delay model, our approach incurs less than 1% error in mean, 1% error in standard deviation, 6% error in skewness, and 1% error in 95-percentile point.

The rest of the chapter is organized as follows: Section 5.2 introduces the approximation of the max operation using second-order polynomial fitting; with the approximation of max, Section 5.3 presents a novel SSTA algorithm for quadratic delay model with non-Gaussian variation sources; we further apply this technique to handle both semi-quadratic and linear delay model in Section 5.4 and Section 5.5, respectively; experimental results are presented in Section 5.6; Section 5.7 concludes this chapter.

5.2 Second-Order Polynomial Fitting of Max Operation

5.2.1 Review and Preliminary

According to [163], given two random variables, A and B, the tightness probability is defined as the probability of A greater than B, i.e., $T_A = P\{A > B\} = P\{A - B > 0\}$.

Then the max operation is approximated as:

$$\max(A,B) \approx T_A \cdot A + (1 - T_A) \cdot B + c, \tag{5.1}$$

where c is a term used to match the mean and variance of $\max(A,B)$. Because (5.1) can be further written as $\max(A,B) = \max(A-B,0) + B$, we arrive at:

$$\max(A-B,0) \quad \approx \quad T_A \cdot (A-B) + c. \tag{5.2}$$

According to (5.2), we can see that the max operation in [163] is in fact approximated by a linear function subject to certain conditions (such as matching the exact mean and variance). Such a linear approximation is efficient and reasonably accurate when both A and B are Gaussian. As shown in [163, 164], the PDF predicted by such linear approximation is very close the Monte Carlo simulation. Moreover, in this case the coefficients can be computed easily, as both T_A and $E[\max(A,B)]$ can be obtained by closed-form formulas when A and B are Gaussian [163]. However, when A and B are non-Gaussian random variables, the tightness probability T_A and $E[\max(A,B)]$ are hard to obtain. For example, T_A in [13] has to be computed via expensive multi-dimensional numerical integration, thus preventing its scalability to large designs. Moreover, because the max operation is an inherently non-linear function, linear approximation would become less and less accurate, particularly when the amount of variation increases and the number of non-Gaussian variation sources increases. To overcome these difficulties, we develop a more efficient and accurate approximation method in the next section.

5.2.2 New Fitting Method for Max Operation

In this section, we introduce a new fitting method to approximate the max operation. Instead of using the linear function, we propose to use a second-order polynomial

function to approximate the max operation, i.e.,

$$\max(V,0) \approx h(V,\Theta) = \theta_2 V^2 + \theta_1 V + \theta_0, \tag{5.3}$$

where $\Theta = (\theta_0, \theta_1, \theta_2)^T$ are three coefficients of the second-order polynomial $h(v,\Theta)$. The problem thus becomes how to obtain the fitting parameters of Θ. Different from the linear fitting method through tightness probability, we compute Θ by ***matching the mean of the max operation while minimizing the square error (SE) between*** $h(V,P)$ ***and*** $max(V,0)$ ***within the*** $\pm\varepsilon$ ***range of*** V. Mathematically, this problem can be formulated as the following optimization problem:

$$\Theta = \arg \min_{E[h(V,\Theta)]=\mu_m} \int_{\mu_v-\varepsilon}^{\mu_v+\varepsilon} \big(h(v,\Theta) - \max(v,0)\big)^2 dv \tag{5.4}$$

where μ_v, σ_v^2, and γ_v are the mean, variance, and skewness of random variable of V, respectively; while μ_m and $E[h(V,\Theta)]$ are the exact and approximated mean of $\max(V,0)$, respectively. In other words,

$$\mu_m = E[\max(V,0)] \tag{5.5}$$

$$E[h(V,\Theta)] = \theta_2(\sigma_v^2 + \mu_v^2) + \theta_1\mu_v + \theta_0. \tag{5.6}$$

In (5.4), ε is the approximation range. The value of ε may affect the accuracy of the second order approximation. Intuitively, when ε is large, the probability that V lies outside the $\pm\varepsilon$ range is low. Then the impact of ignoring the difference in the low probability region is justified. However, when ε is too large, we may unnecessarily fitting the curve in a wide range without focusing on the high probability region. In this chapter, we find that assuming:

$$\varepsilon = 3\sigma_v \tag{5.7}$$

provides a good approximation of max. This can be explained by the fact that, although the circuit delay is not Gaussian, it would still be more or less Gaussian like. That is,

its PDF would be most likely centering around the $\pm 3\sigma$ range of the mean. In practice, the users can choose the range ε according to the delay distributions.

In order to solve (5.4), we first need to compute μ_m. When V is a non-Gaussian random variable, exact computation of μ_m is difficult in general. Therefore, we propose to use the following two-step procedure to approximately compute μ_m. In the first step, we approximate the non-Gaussian random variable V as a quadratic function of a standard Gaussian random variable W similar to [184], i.e.,

$$V \approx P(W) = c_2 \cdot W^2 + c_1 \cdot W + c_0. \tag{5.8}$$

The coefficients c_2, c_1, and c_0 can be obtained by matching $P(W)$ and V's mean, variance and skewness simultaneously, i.e.,

$$E[c_2 \cdot W^2 + c_1 \cdot W + c_0] = \mu_v \tag{5.9}$$

$$E[(c_2 \cdot W^2 + c_1 \cdot W + c_0 - \mu_v)^2] = \sigma_v^2 \tag{5.10}$$

$$E[(c_2 \cdot W^2 + c_1 \cdot W + c_0 - \mu_v)^3] = \gamma_v \cdot \sigma_v^3 \tag{5.11}$$

As shown in [184], solving the above equations, c_2 will be one of the following values:

$$c_{2,1} = -\frac{2\sigma_v^2 + \Delta^{2/3}}{2\Delta^{1/3}} \tag{5.12}$$

$$c_{2,2} = \frac{2(1+j\sqrt{3})\sigma_v^2 + (1-j\sqrt{3})\Delta^{2/3}}{4\Delta^{1/3}} \tag{5.13}$$

$$c_{2,3} = \frac{2(1-j\sqrt{3})\sigma_v^2 + (1+j\sqrt{3})\Delta^{2/3}}{4\Delta^{1/3}} \tag{5.14}$$

where $\Delta = \gamma_v \cdot \sigma_v^3 + j\sqrt{8\sigma_v^6 - \gamma_v^2 \cdot \sigma_v^6}$, with $j = \sqrt{-1}$. It is proved in [184] that, when $|\gamma_v| < 2\sqrt{2}$ there exists one and only one of the above three values in the range $|c_2| < \sigma_v/\sqrt{2}$. We will pick such value for c_2. When $|\gamma_v| > 2\sqrt{2}$, we may let:

$$c_2 = \begin{cases} \sigma_v/\sqrt{2} & \gamma_v > 2\sqrt{2} \\ -\sigma_v/\sqrt{2} & \gamma_v < -2\sqrt{2} \end{cases} \tag{5.15}$$

It is proved in [19] that the above equations gives $P(W)$ which matches the mean and variance of V and has the skewness closest to γ_v. With c_2, c_1 and c_0 can be computed as:

$$c_1 = \sqrt{\sigma_v^2 - 2c_2^2} \tag{5.16}$$

$$c_0 = \mu_v - c_2 \tag{5.17}$$

After obtaining the coefficients c_2, c_1, and c_0 in the second step, we approximate the exact mean of $\max(V,0)$ by the exact mean of $\max(P(W),0)$, i.e.,

$$\mu_m \approx E[\max(P(W),0)] = \int_{P(w)>0} P(w)\phi(w)dw \tag{5.18}$$

where $\phi(\cdot)$ is the PDF of the standard normal distribution. In the above equation, the integration range $P(w) > 0$ can be computed in four different cases:

Case 1: $c_2 > 0$

$$P(w) > 0 \Leftrightarrow w < t_1 \vee w > t_2 \tag{5.19}$$

where

$$t_1 = (-c_1 - \sqrt{c_1^2 - 4c_2c_0})/2c_2 \tag{5.20}$$

$$t_2 = (-c_1 + \sqrt{c_1^2 - 4c_2c_0})/2c_2 \tag{5.21}$$

Case 2: $c_2 < 0$

$$P(w) > 0 \Leftrightarrow t_2 < w < t_1 \tag{5.22}$$

Case 3: $c_2 = 0 \wedge c_1 > 0$

$$P(w) > 0 \Leftrightarrow w > -c_0/c_1 \tag{5.23}$$

Case 4: $c_2 = 0 \wedge c_1 < 0$

$$P(w) > 0 \Leftrightarrow w < -c_0/c_1 \tag{5.24}$$

Knowing the integration range, we can compute $E[\max(P(W),0)]$ under such four cases:

$$E[\max(P(W),0)] = \tag{5.25}$$
$$\begin{cases} (c_2+c_0)\big(1+\Phi(t_1)-\Phi(t_2)\big)+(c_1+t_1)\big(\phi(t_2)-\phi(t_1)\big) & c_2>0 \\ (c_2+c_0)\big(\Phi(t_1)-\Phi(t_2)\big)+(c_1+t_1)\big(\phi(t_2)-\phi(t_1)\big) & c_2<0 \\ c_0\cdot\Phi(c_0/c_1)+c_1\cdot\phi(c_0/c_1) & c_2=0\wedge c_1>0 \\ c_0\cdot\big(1-\Phi(c_0/c_1)\big)-c_1\cdot\phi(c_0/c_1) & c_2=0\wedge c_1<0 \end{cases}$$

where $\Phi(\cdot)$ is the cumulative density function (CDF) of the standard normal distribution. According to (5.25), we can compute μ_m easily through analytical formulas.

In practice, the above approximation of μ_m is accurate only when the distribution of V is close to the normal distribution. When V is not close to the normal distribution, for example, V is uniform distribution, the above approximation of μ_m is no longer accurate. Fortunately, in practice, the circuit delay has close-to-normal distribution as discussed above. Therefore, such approximation is accurate for SSTA.

After obtaining μ_m, we need to find Θ in (5.3) by solving the constrained optimization problem of (5.4). In the following, we show that (5.4) can be solved analytically as well. We first write the constraint in (5.4) as follows:

$$\theta_0 = \mu_m - \theta_2(\mu_v^2+\sigma_v^2) - \theta_1\mu_v. \tag{5.26}$$

Replacing θ_0 in (5.4) by (5.26), the square error in (5.4) can be written as:

$$\begin{aligned} SE &= \int_{\mu_v-3\sigma_v}^{0} h^2(v,\Theta)dv + \int_{0}^{\mu_v+3\sigma_v} \big(h(v,\Theta)dv-v\big)^2 dv \\ &= \int_{l_1}^{0} \big(\theta_2(v^2-\mu_v^2-\sigma_v^2)+\theta_1(v-\mu_v)+\mu_m\big)^2 dv + \\ &\quad \int_{0}^{l_2} \big(\theta_2(v^2-\mu_v^2-\sigma_v^2)+\theta_1(v-\mu_v)+\mu_m-v\big)^2 dv, \end{aligned} \tag{5.27}$$

where $l_1 = \mu_v - 3\sigma_v$ and $l_2 = \mu_v + 3\sigma_v$. By expanding the square and integral, we can transform the constrained optimization of (5.4) to the following unconstrained optimization problem, which is a quadratic form of $\Theta' = (\theta_1, \theta_2)^T$:

$$\Theta' = \arg\min SE \tag{5.28}$$

$$SE = \Theta'^T S \Theta' + Q\Theta' + t, \tag{5.29}$$

where $S = (s_{ij})$ is a 2×2 matrix, $Q = (q_i)$ is a 1×2 vector, and t is a constant. The parameters of S, Q, and t can be computed as:

$$s_{11} = (l_2^3 - l_1^3)/3 + (l_2 - l_1)\mu_v^2 - (l_2^2 - l_1^2)\mu_v \tag{5.30}$$

$$s_{22} = (l_2^5 - l_1^5)/5 + (l_2^2 - l_1^2)(\mu_v^2 + \sigma_v^2)^2 - 2(l_2^3 - l_1^3)(\mu_v^2 + \sigma_v^2)/3 \tag{5.31}$$

$$s_{12} = s_{21} = (l_2^4 - l_1^4)/4 + (l_2 - l_1)\mu_v(\mu_v^2 + \sigma_v^2) + (l_2^3 - l_1^3)\mu_v/3 - (l_2^2 - l_1^2)(\mu_v^2 + \sigma_v^2)/2 \tag{5.32}$$

$$q_1 = l_2^2\mu_v + (l_2^2 - l_1^2)\mu_m - 2l_2^3/3 - 2(l_2 - l_1)\mu_v\mu_m \tag{5.33}$$

$$q_2 = l_2^2(\mu_v^2 + \sigma_v^2) + 2(l_2^3 - l_1^3)\mu_m/3 - 2l_2^3/3 - 2(l_2 - l_1)(\mu_v^2 + \sigma_v^2)\mu_v \tag{5.34}$$

$$t = l_2^3/3 + (l_2 - l_1)\mu_m^2 - l_2^2\mu_m \tag{5.35}$$

Because the square error is always positive no matter what value the Θ' is, S is a positive definite matrix. Therefore, (5.29) is to minimize a second-order convex function of Θ' without constraints. Then the optimum of Θ' can be obtained by setting the derivative of (5.29) to zero, resulting a 2×2 system of linear equations:

$$\frac{\partial SE}{\partial \theta_1} = 2s_{11} \cdot \theta_1 + 2s_{12} \cdot \theta_2 + q_1 = 0 \tag{5.36}$$

$$\frac{\partial SE}{\partial \theta_2} = 2s_{22} \cdot \theta_2 + 2s_{12} \cdot \theta_1 + q_2 = 0 \tag{5.37}$$

Such a system of linear equations can be solved efficiently:

$$\theta_1 = \frac{s_{12} \cdot q_2 - s_{22} \cdot q_1}{2s_{11} \cdot s_{22} - 2s_{12}^2} \tag{5.38}$$

$$\theta_2 = \frac{s_{12} \cdot q_1 - s_{11} \cdot q_2}{2s_{11} \cdot s_{22} - 2s_{12}^2} \tag{5.39}$$

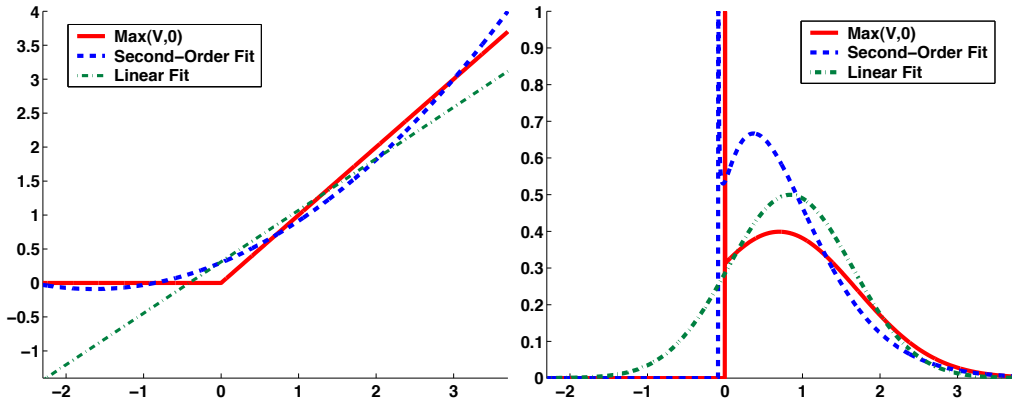
With θ_1 and θ_2, we can compute θ_0 from (5.26).

From the above discussion, we see that for a random variable V with any distribution, if we know its mean μ_v, variance σ_v^2, and skewness γ_v, we can obtain the fitting parameters Θ for $\max(V,0)$ through closed-form formulas.

Notice that in (5.4) we try to minimize the mean square error within the $\pm 3\sigma$ range. If $\mu_v > 3\sigma_v$, V is always larger than zero within the $\pm 3\sigma$ range. That is: $\max(V,0) = V$ when $V \in (\mu_v - 3\sigma_v, \mu_v + 3\sigma_v)$. From (5.4), it is easy to find that in this case, $\Theta = (0,1,0)$. That means:

$$\max(V,0) \approx V \ \ when \ \ \mu_v > 3\sigma_v \tag{5.40}$$

To show the accuracy of our second-order fitting approach to the max approximation, we compare the results obtain from our approach, the linear fitting method, and the exact (or Monte Carlo) computation. For example, by assuming $V \sim N(0.7,1)$, the exact max operation $\max(V,0)$, linear fitting through through tightness probability, and our second-order fitting can be obtained as shown in Fig. 5.2.2, where the x-axis is V, and y-axis is the results of $\max(V,0)$. The corresponding PDFs of the three approaches are shown in Fig. 5.2.2. From the figures, we see that our proposed second-order fitting method is more accurate than the linear fitting method. In particular, the PDF of our second-order fitting method has a sharp peak which approximates the impulse of exact PDF well as shown in Fig. 5.2.2. In contrast, the linear fitting method can only give a smooth PDF, which is far away from the exact max result.



(a) (b)

Figure 5.1: (a) Comparison of exact computation, linear fitting, and second-order fitting for $\max(V,0)$; (b) PDF comparison of exact computation, linear fitting, and second-order fitting for $\max(V,0)$.

5.3 Quadratic SSTA

5.3.1 Quadratic Delay Model

In Section 5.2, we introduce the second-order fitting of max operation. Here we will apply such fitting in SSTA.

We first discuss the delay model. In practice, the circuit delay is a complicate function of variation sources:

$$D = f(X_1, X_2, \ldots X_n, R) \tag{5.41}$$

where $X = (X_1, X_2, \ldots X_n)^T$ are global variation sources, R is local random variation, n is the number of global variation sources. Here, we assume that X_i's and R are mutually independent. Without loss of generality, we assume all X_i's and R have zero mean and unit variance. In order to simplify the above delay model, we apply the Taylor expansion to approximate it. Considering that the scale of local random variation is

usually smaller than that of global variation, we use second order Taylor expansion to approximate X_i's and use first order expansion to approximate R:

$$\begin{aligned} D &\approx d_0 + \sum_{i,j=1}^{n} b_{ij} X_i X_j + \sum_{i=1}^{n} a_i X_i + rR \\ &= d_0 + AX + X^T BX + rR \end{aligned} \tag{5.42}$$

where d_0 is the nominal delay, $A = (a_1, a_2, \ldots a_n)$ are the linear delay sensitivity coefficients of the global variation sources, $B = (b_{ij})$ are the second-order sensitivity coefficients, which is an $n \times n$ matrix, and r is the linear delay sensitivity coefficient of the local random variation. A, B, and r are calculated in the similar way as [180]:

$$a_i = \frac{\partial f}{\partial X_i} \tag{5.43}$$

$$b_{ij} = \frac{\partial^2 f}{\partial X_i \partial X_j} \tag{5.44}$$

$$r = \frac{\partial f}{\partial R} \tag{5.45}$$

In practice, f is a complex function and can not be expressed as closed-form formula. In this case, the coefficients A, B, and r can be obtained by measurement or SPICE simulation. Moreover, in practice, the local random variation is caused by several independent factors; therefore, the local random variation R is the sum of several independent random variables. According to the central limit theorem, R will be close to a Gaussian random variable. In this chapter, for simplicity, we assume that R is a random variable with standard normal distribution. Finally, we obtain our quadratic delay model as follows:

$$D = d_0 + AX + X^T BX + rR \tag{5.46}$$

In the rest of this chapter, we use $m_{i,k}$ and $m_{r,k}$ to represent the kth central moment for X_i and the kth moment for R, respectively. Notice that all X_i's and R are with zero mean, that means their central moments and raw moments are the same. Moreover,

we also assume that the moments of the variation sources, $m_{i,k}$ and $m_{r,k}$, are known. In practice, such moments can be computed from the samples of variation sources.

To compute the arrival time in a block-based SSTA framework, two atomic operations, max and sum, are needed. That is, given D_1 and D_2 with the quadratic form of (5.46):

$$D_1 = d_{01} + A_1 X + X^T B_1 X + r_1 R_1, \tag{5.47}$$

$$D_2 = d_{02} + A_2 X + X^T B_2 X + r_2 R_2, \tag{5.48}$$

we want to compute

$$\begin{aligned} D_m &= \max(D_1, D_2) \\ &= d_{m0} + A_m X + X^T B_m X + r_m R_m, \end{aligned} \tag{5.49}$$

$$D_s = D_1 + D_2 = d_{s0} + A_s X + X^T B_s X + r_s R_s. \tag{5.50}$$

We will present how these two operations are handled in the rest of this section.

5.3.2 Max Operation for Quadratic Delay Model

The max operation is the most difficult operation for block-based SSTA. Based on the second-order polynomial fitting method as discussed in Section 5.2.2, we propose a novel technique to compute the max of two random variables. The overall flow of the max operation is illustrated in Figure 5.2. Considering

$$D_m = \max(D_1, D_2) = \max(D_1 - D_2, 0) + D_2, \tag{5.51}$$

let $D_p = D_1 - D_2$. Without loss of generality, we assume $E[D_p] > 0$. We first compute the mean and variance of D_p, if $\mu_{D_p} > 3\sigma_{D_p}$, then $D_m = D_1$, otherwise, we compute the joint moments between D_p and X_i's and the skewness of D_p. Knowing the mean, variance, and skewness of D_p, we apply the method as shown in Section 5.2.2 to find

the fitting coefficients $\Theta = (\theta_0, \theta_1, \theta_2)$ for $\max(D_p, 0)$. Finally, we use the moment matching method to reconstruct the quadratic form of D_m.

Input: Quadratic form of D_1 and D_2
Output: Quadratic form of D_m
1. Let $D_p = D_1 - D_2$, compute μ_{D_p} and σ_{D_p}
if $\mu_{D_p} \geq 3\sigma_{D_p}$ {
    2. $Dm = D_1$
  }
else {
    3. Compute joint moments between D_p^2 and X_i
    4. Compute γ_{D_p}
    5. Get fitting coefficients Θ
    6. Compute joint moments between D_m and X_i
    7. Reconstruct the quadratic form of D_m
  }

Figure 5.2: Algorithm for computing $max(D_1,D_2)$.

5.3.2.1 Mean and Variance of D_p

In order to compute the mean and variance, we first obtain the quadratic form of D_p as follows:

$$\begin{aligned} D_p &= D_1 - D_2 \\ &= d_{p0} + A_p \cdot X + X^T B_p X + r_1 \cdot R_1 - r_2 \cdot R_2 \qquad (5.52) \\ d_{p0} &= d_{01} - d_{02} \qquad (5.53) \\ A_p &= A_1 - A_2 \qquad (5.54) \\ B_p &= B_1 - B_2 \qquad (5.55) \end{aligned}$$

Because X_i's, R_1, and R_2 are mutually independent random variables with mean 0

and variance 1, the mean and variance of D_p can be computed as:

$$\begin{aligned}
\mu_{D_p} &= E[D_p] = d_{p0} + \sum_{i=1}^{n} b_{pii} && (5.56)\\
\sigma^2_{D_p} &= E[(D_p - \mu_{D_p})^2] \\
&= r_1^2 + r_2^2 + \sum_{i=1}^{n} (a_{pi}^2 + b_{pii} m_{i,4} + 2a_{pi} b_{pii} m_{i,3}) + \\
&\quad \sum_{1 \le i < j \le n} 2(b_{pij}^2 + b_{pii} b_{pjj}) - (\sum_{i=1}^{n} b_{pii})^2 && (5.57)
\end{aligned}$$

5.3.2.2 Joint Moments and Skewness of D_p

From the definition of D_p as shown in (5.52), the joint moments between D_p^2 and X_i, X_i^2, and R can be computed as:

$$\begin{aligned}
E[X_i D_p^2] &= 2d_{p0}a_{pi} + (a_{pi}^2 + 2d_{p0}b_{pii}) m_{i,3} + 2a_{pi}b_{pii} m_{i,4} + b_{pii}^2 m_{i,5} + \\
&\quad 2 \cdot \sum_{\substack{1 \le j \le n \\ j \ne i}} \big(a_{pi} b_{pjj} + 2a_{pj} b_p ij + \\
&\quad (b_{pii} b_{pjj} + 2b_{pij}^2) m_{i,3} + 2b_{pij} b_{pjj} m_{j,3} \big) && (5.58)\\
E[R_1 D_p^2] &= r_1^2 m_{r,3} + 2r_1 \mu_{D_p} && (5.59)\\
E[R_2 D_p^2] &= r_2^2 m_{r,3} - 2r_1 \mu_{D_p} && (5.60)\\
E[X_i^2 D_p^2] &= d_{p0}^2 + r_1^2 + r_2^2 + 2d_{p0}a_{pi} m_{i,3} + 2a_{pi} b_{pii} m_{i,5} + \\
&\quad (a_{pi}^2 + 2d_{p0} b_{pii}) \cdot m_{i,4} + b_{pii}^2 m_{i,6} + \\
&\quad \sum_{\substack{1 \le j \le n \\ j \ne i}} \big(a_{pj}^2 + 2d_{p0} b_{pjj} + 2(a_{pi} b_{pjj} + 2a_{pj} b_{pij}) m_{i,3} + 2a_{pj} b_{pjj} m_{j,3} + \\
&\quad 2(b_{pii} b_{pjj} + 4b_{pij}^2) \cdot m_{i,4} + b_{pjj}^2 m_{j,4} + 4b_{pjj} b_{pij} m_{i,3} m_{j,3} \big) + \\
&\quad 2 \cdot \sum_{\substack{1 \le j < k \le n \\ j,k \ne i}} (b_{pjj} b_p kk + 2b_{pjk}^2) && (5.61)
\end{aligned}$$

$$
\begin{aligned}
E[X_iX_jD_p^2] &= 2a_{pi}a_{pj} + 4d_{p0}b_{pij} + 2\cdot(2a_{pi}b_{pij} + a_{pj}b_{pii})m_{i,3} + \\
&\quad 4b_{pii}b_{pij}m_{i,4} + 2\cdot(2a_{pj}b_{pij} + a_{pi}b_{pjj})m_{j,3} + 4b_{pjj}b_{pij}m_{j,4} + \\
&\quad 2\cdot(2b_{pij}^2 + b_{pii}b_pjj)m_{i,3}m_{j,3} + 4\cdot\sum_{\substack{1\le k\le n\\ k\neq i,j}} b_{pij}b_{pkk} \qquad (5.62)
\end{aligned}
$$

With the joint moments computed above, the raw moments, central moments, and skewness of D_p is computed as:

$$
\begin{aligned}
E[D_p^2] &= \mu_{D_p}^2 + \sigma_{D_p}^2 \qquad (5.63)\\
E[D_p^3] &= E[D_p\cdot D_p^2] \\
&= E[(d_{p0} + A_pX + X^TB_pX + r_1R_1 - r_2R_2)D_p^2] \\
&= d_{p0}\cdot E[D_p^2] + r_1\cdot E[R_1D_p^2] - r_2\cdot E[R_2D_p^2] + \\
&\quad \sum_{i=1}^{n}\left(a_{pi}\cdot E[X_iD_p^2] + b_{pii}\cdot E[X_i^2D_p^2]\right) + \\
&\quad \sum_{0\le i<j\le n} b_{pij}E[X_iX_j\cdot D_p^2] \qquad (5.64)\\
m_{D_p}(3) &= E[(D-\mu_D)^3] \\
&= E[D_p^3] - 3\mu_{D_p}\cdot E[D_p^2] + 2\mu_{D_p}^3 \qquad (5.65)\\
\gamma_{D_p} &= m_{D_p}(3)/\sigma_{D_p}^3 \qquad (5.66)
\end{aligned}
$$

5.3.2.3 Reconstruct the Quadratic Form of Dm

Knowing μ_{D_p}, σ_{D_p}, and γ_{D_p}, we apply the second-order fitting method to compute the fitting parameters $\Theta = (\theta_0, \theta_1, \theta_2)$ for $\max(D_p, 0)$. Then D_m can be represented as:

$$
\begin{aligned}
D_m &= \max(D_1, D_2) = \max(D_p, 0) + D_2 \\
&\approx \theta_2\cdot D_p^2 + \theta_1\cdot D_p + \theta_0 + D_2 \\
&= \theta_2\cdot D_p^2 + D_q + \theta_0 \qquad (5.67)\\
D_q &= \theta_1\cdot D_p + D_2 \qquad (5.68)
\end{aligned}
$$

The above equation gives the closed- form formula of D_m. However, because D_p and D_q are in quadratic form as (5.46), the representation of D_m in (5.67) is a 4^{th} order polynomial of X_i's. In order to reconstruct the quadratic form for D_m, we first compute the the mean of D_m, and joint moments between D_m and variation sources:

$$E[D_m] = \theta_2 \cdot E[D_p^2] + E[D_q] + \theta_0 \tag{5.69}$$

$$E[X_i D_m] = \theta_2 \cdot E[X_i D_p^2] + E[X_i D_q] \tag{5.70}$$

$$E[X_i^2 D_m] = \theta_2 \cdot E[X_i^2 D_p^2] + E[X_i^2 D_q] + \theta_0 \tag{5.71}$$

$$E[X_i X_j D_m] = \theta_2 \cdot E[X_i X_j D_p^2] + E[X_i X_j D_q] \tag{5.72}$$

$$E[R_1 D_m] = \theta_2 \cdot E[R_1 D_p^2] + E[R_1 D_q] \tag{5.73}$$

$$E[R_2 D_m] = \theta_2 \cdot E[R_2 D_p^2] + E[R_2 D_q] \tag{5.74}$$

The joint moments between D_p^2 and X_i's are computed in (5.58)-(5.62). The joint moments between D_q and variation sources are:

$$E[D_q] = d_{q0} + \sum_{i=1}^{n} b_{qii} \tag{5.75}$$

$$E[X_i D_q] = a_{qi} m_{i,2} + b_{qii} m_{i,3} \tag{5.76}$$

$$E[X_i^2 D_q] = d_{q0} + a_{qi} m_{i,3} + b_{qii} m_{i,4} + \sum_{\substack{1 \le j \le n \\ j \ne i}} b_{qjj} \tag{5.77}$$

$$E[X_i X_j D_q] = 2 b_{qij} \tag{5.78}$$

$$E[R_1 D_q] = \theta_1 \cdot r_1 \tag{5.79}$$

$$E[R_2 D_q] = (1 - \theta_1) \cdot r_2 \tag{5.80}$$

Because we want to reconstruct D_m in the quadratic form, as shown in (5.49), by applying the moment matching technique similar to [178], we have:

$$E[D_m] = \sum_{i=1}^{n} b_{mii} + d_{m0} \tag{5.81}$$

$$E[X_i D_m] = a_{mi} + b_{mii} \cdot m_{i,3} \tag{5.82}$$

$$E[X_i^2 D_m] = a_{mi} \cdot m_{i,3} + b_{nii} \cdot m_{i,4} + \sum_{\substack{1 \le j \le n \\ j \ne i}} b_{mjj} \tag{5.83}$$

$$E[X_i X_j D_m] = 2b_{mij} \tag{5.84}$$

With $E[D_m]$, $E[X_i D_m]$, $E[X_i^2 D_m]$, and $E[X_i X_j D_m]$ computed in (5.69)-(5.72), the sensitivity coefficients $A_m = (a_{mi})$ and $B_m = (b_{mij})$ can be obtained by solving the linear equations above:

$$b_{mii} = \frac{E[X_i^2 D_m] - E[D_m] - E[X_i D_m] m_{i,3}}{m_{i,4} - m_{i,3}^2 - 1} \tag{5.85}$$

$$b_{mij} = E[X_i X_j D_m] \tag{5.86}$$

$$a_{mi} = E[X_i D_m] - b_{mii} \cdot m_{i,3} \tag{5.87}$$

$$d_{m0} = E[D_m] - \sum_{i=1}^{n} b_{mii} \tag{5.88}$$

Finally, we consider the random term of D_m. Because the random term in D_m comes from the random terms in D_1 and D_2, we assume that $r_m R_m = r_{m1} R_1 + r_{m2} R_2$. Because the random variation sources R_m, R_1, and R_2 are Gaussian random variables, by applying the moment matching technique similar to (5.84), we have:

$$r_{m1} = E[R_1 \cdot D_m] \tag{5.89}$$

$$r_{m2} = E[R_2 \cdot D_m] \tag{5.90}$$

$$r_m = \sqrt{r_{m1}^2 + r_{m2}^2} \tag{5.91}$$

Where $E[R_1 \cdot D_m]$ and $E[R_2 \cdot D_m]$ are computed in (5.73) and (5.74).

5.3.3 Sum Operation for Quadratic Delay Model

The sum operation is straight forward. The coefficients of D_s can be computed by adding the correspondent coefficients of D_1 and D_2:

$$d_{s0} = d_{01} + d_{02} \tag{5.92}$$

$$A_s = A_1 + A_2 \tag{5.93}$$

$$B_s = B_1 + B_2 \tag{5.94}$$

$$r_s = \sqrt{r_1^2 + r_2^2} \tag{5.95}$$

5.3.4 Computational Complexity of Quadratic SSTA

For each max operation in SSTA based on a quadratic delay model, we need to calculate n^2 joint moments between D_p and X_iX_js; for each joint moment, we need to compute the sum of n numbers; hence the computational complexity is $O\{n^3\}$, where n is the number of variation sources. For the sum operation, we need to compute the sum of two $n \times n$ metric; hence the computational complexity is $O\{n^2\}$. Because the total number of max and sum operations is linear with respect to circuit sizes N, the total complexity is $O\{n^3N\}$.

5.4 Semi-Quadratic SSTA

5.4.1 Semi-Quadratic Delay Model

In Section 5.3, we introduce the SSTA for quadratic delay model. However, in practice, the impact of the crossing terms are usually very weak [178, 19]. Therefore, if we ignore the crossing terms, we may speed up the SSTA process without affecting the accuracy too much. Ignoring the crossing terms, the quadratic delay model in (5.46) is

re-written as:

$$D = d_0 + AX + BX^2 + rR \tag{5.96}$$

where d_0, A, r, and R are defined in the similar way as the quadratic delay model in (5.46), $X^2 = (X_1^2, X_2^2, \ldots, X_n^2)^T$ are the square of variation sources, and $B = (b_1, b_2, \ldots, b_n)$ are the second order sensitivity coefficients for the square terms. We defined such quadratic model without crossing terms as *Semi-Quadratic Delay Model*. We will introduce the atomic operations, max and sum, for the semi-quadratic delay model in the rest of this section.

5.4.2 Max Operation for Semi-Quadratic Delay Model

The overall flow of the max operation of semi-quadratic delay model is similar to that of quadratic delay model as illustrated in Figure 5.2. The only difference is that we do not need to compute the joint moments between the crossing terms and D_m.

5.4.2.1 Moments of D_p

We defined $D_p = D_1 - D_2$ in a similar way as (5.52). In order to compute the central moments of D_p, we first rewrite D_p to the following form:

$$D_p = d'_{p0} + \sum_{i=1}^{n} Y_{pi} + r_1 R_1 - r_2 R_2 \tag{5.97}$$

$$d'_{p0} = d_{p0} + \sum_{i=1}^{n} b_{pi} \tag{5.98}$$

$$Y_{pi} = a_{pi} X_i + b_{pi} X_i^2 - b_{pi} \tag{5.99}$$

with $a_{pi} = a_{1i} - a_{2i}$ and $b_{pi} = b_{1i} - b_{2i}$. Because X_is are mutually independent random variables with mean 0 and variance 1, Y_{pi}s are independent random variables with zero

mean. Therefore, the first 3 central moments of D_p are:

$$\mu_{D_p} = d'_{p0} \tag{5.100}$$

$$\sigma^2_{D_p} = r_1^2 + r_2^2 + \sum_{i=1}^{n} E[Y_{pi}^2] \tag{5.101}$$

$$m_{D_p}(3) = E[(D_p - \mu_{D_p})^3] = r_1^3 + r_2^3 + \sum_{i=1}^{n} E[Y_{pi}^3] \tag{5.102}$$

where

$$E[Y_{pi}^2] = b_{pi}^2(m_{i,4} - 1) + 2a_{pi}b_{pi}m_{i,3} + a_{pi}^2 \tag{5.103}$$

$$E[Y_{pi}^3] = 3a_{pi}b_{pi}^2\big(m_{i,5} - 2m_{i,3}\big) + 3a_{pi}^2 b_{pi}\big(m_{i,4} - 1\big) + a_{pi}^3 m_{i,3} + b_{pi}^3\big(m_{i,6} - 3m_{i,4} + 2\big) \tag{5.104}$$

With the central moments of D_p, the raw moments and skewness can be computed easily.

5.4.2.2 Reconstruct the Semi-Quadratic Form of D_m

Similar to the quadratic SSTA, by knowing the mean, variance and skewness of D_p, the fitting coefficients Θ can be obtained. Then the joint moments between X_is and D_m can be computed in the similar ways as (5.69)-(5.74). Here the joint moments between X_is and D_p^2, D_q are:

$$E[X_i D_p^2] = E[X_i Y_{pi}^2] + 2d'_0 E[X_i Y_{pi}] \tag{5.105}$$

$$E[X_i^2 D_p^2] = E[X_i^2 Y_{pi}^2] + 2d'_0 E[X_i^2 Y_{pi}] + E[X_i^2]E[(D_p - Y_{pi})^2] \tag{5.106}$$

$$E[X_i D_q] = E[X_i Y_{qi}] \tag{5.107}$$

$$E[X_i^2 D_q] = E[X^2 Y_{qi}] \tag{5.108}$$

where $E[Y_{pi}^2]$ are computed in (5.102), $E[(D_p - Y_{pi})^2] = \sigma^2_{D_p} - E[Y_{pi}^2]$, and Y_{qi}s are defined in the similar way as Y_{pi}s:

$$Y_{qi} = a_{qi}X_i + b_{qi}X_i^2 - b_{qi} \tag{5.109}$$

with $a_{qi} = \theta_1(a_1 - a_2) + a_2$ and $b_{qi} = \theta_1(b_1 - b_2) + b_2$. The joint moments between X_is and Y_{pi}s are:

$$E[X_i^2 Y_{pi}^2] = \left(a_{pi}^2 - 2_{pi}^2\right) m_{i,4} + b_{pi}^2 \left(m_{i,6} - 1\right) + 2a_{pi} b_{pi} \left(m_{i,5} - m_{i,3}\right) \quad (5.110)$$

$$E[X_i^2 Y_{pi}] = a_{pi} m_{i,3} + b_{pi} \left(m_{i,4} - 1\right) \quad (5.111)$$

$$E[X_i Y_{pi}^2] = b_{pi}^2 m_{i,5} + 2a_{pi} b_{pi} \left(m_{i,4} - 1\right) + \left(a_{pi}^2 - 2b_{pi}^2\right) m_{i,3} \quad (5.112)$$

$$E[X_i Y_{pi}] = a_{pi} + b_{pi} m_{i,3} \quad (5.113)$$

The joint moments between X_is and Y_{qi}s can be computed in the same way.

Knowing the joint moments between X_is and D_m, by applying the moment matching technique, we will have similar equations as (5.84). And then A_m and B_m can be obtained by solving such equations. Finally, we can compute the random term of D_m in the same way as the quadratic model, as shown in (5.91).

The sum operation of the semi-quadratic delay model is similar to that of the quadratic delay model. We can compute $D_s = D_1 + D_2$ by summing up the correspondent coefficients.

From the discussion above, it is easy to see that for the semi-quadratic SSTA, the computational complexity for both max and sum operation is $O\{n\}$, where n is the number of variation sources. The number of max and sum operations is linear to the circuit size.

5.5 Linear SSTA

5.5.1 Linear Delay Model

In the previous sections, we introduce the SSTA for non-linear delay model. In practice, when the variation scale is small, the circuit delay can be approximated as a linear

function of variation sources:

$$D = d_0 + A \cdot X + r \cdot R \tag{5.114}$$

where X, d_0, A, r, and R are defined in the similar way as the quadratic delay model in (5.46). The difference is that there are no second order terms. Hence the atomic operations of the linear delay model are much simpler than those of quadratic delay model. We will introduce such operations in the rest of this section.

5.5.2 Max Operation for Linear Delay Model

The overall flow of the max operation of linear delay model is similar to that of quadratic delay model as illustrated in Figure 5.2 except that we do not need to compute the high order joint moments between D_m and X_is.

5.5.2.1 Mean and Variance of D_p

The linear form of $D_p = D_1 - D_2$ can be computed in the similar way as (5.52). Then the mean and variance of D_p can be computed as:

$$\mu_{D_p} = E[D_p] = d_{p0}; \tag{5.115}$$

$$\sigma^2_{D_p} = E[(D-\mu_D)^2] = \sum_{i=1}^{n} a^2_{pi} + r^2_1 + r^2_2 \tag{5.116}$$

5.5.2.2 Joint Moments and Skewness of D_p

The joint moments between D^2_p and the variation sources and random variation can be computed as:

$$E[X_i \cdot D^2_p] = a^2_{pi} \cdot m_{i,3} + 2a_{pi} \cdot d_{p0} \tag{5.117}$$

With the joint moments computed above, the third order raw moments D_p is computed as:

$$E[D_p^3] = \sum_{i=1}^{n} a_i E[X_i \cdot D_p^2] + r_1 E[R_1 \cdot D_p^2] + r_2 E[R_2 \cdot D_p^2] \tag{5.118}$$

The second order raw moment, central moments, and skewness of D_p can be computed in the same way as the quadratic SSTA in (5.66).

5.5.2.3 Reconstruct the Linear Canonical Form of *Dm*

Knowing the mean, variance, and skewness of D_p, we may apply the method in Section 5.2.2 to compute the fitting parameters Θ for $\max(D_p, 0)$, and then represent D_m in the form of:

$$D_m \approx \theta_2 \cdot D_p^2 + D_q + p_0 \tag{5.119}$$

where D_q is defined in the similar way as (5.67). The above equation gives the closed-form formula of D_m, however, such formula is not in the linear canonical form. We still apply moment matching technique to reconstruct D_m to the linear form. First, the mean of D_m and the joint moments between Dm and variation sources are:

$$E[D_m] = \theta_2 \cdot E[D_p^2] + E[D_q] + p_0 \tag{5.120}$$

$$E[X_i \cdot D_m] = \theta_2 \cdot E[X_i \cdot D_p^2] + E[X_i \cdot D_q] \tag{5.121}$$

Where $E[D_p^2]$ is computed in (5.118). The joint moments between D_q and variation sources are computed as:

$$E[D_q] = d_{s0} \tag{5.122}$$

$$E[X_i \cdot D_q] = a_{qi} \tag{5.123}$$

With the joint moments, by applying the moment matching technique in [178], we have:

$$a_{mi} = E[X_i \cdot D_m] \tag{5.124}$$

$$d_{m0} = E[D_m] \tag{5.125}$$

Finally, the random term r_m can be computed in the same way as the quadratic model in (5.91).

For the linear SSTA we discussed above, the computational complexity for both max and sum operation is $O\{n\}$. Similar to the quadratic SSTA, the number of max and sum operations is linear to the circuit size.

5.6 Experimental Result

We have implemented our SSTA algorithm in C for all three delay models we discussed before: quadratic delay model (*Quad SSTA*), semi-quadratic delay model (*Semi-Quad SSTA*), and linear delay model (*Lin SSTA*). We also define three comparison cases: (1) our implementation of the linear SSTA for Gaussian variation sources in [163], which we refer to as *Lin Gau*; (2) the non-linear SSTA using Fourier Series approximation for non-Gaussian variation sources in [40] (*Fourier SSTA*); (3) 100,000-sample Monte-Carlo simulation (*MC*). We apply all the above methods to the ISCAS89 suite of benchmarks in TSMC 90nm technology.

In our experiment, we consider two types of variation sources gate channel length L_{gate} and threshold voltage V_{th}. For each type of variation source, inter-die, intra-die spatial, and intra-die random variation are considered. We use the grid-based model in [172, 173] to model the spatial variation. The number of grids (the number of spatial variation sources) is determined by the circuit size and larger circuits have more variation sources. We also assume that the 3σ value of the inter-die (σ_g), intra-die

spatial (σ_s), and intra-die random (σ_r) variation are 10%, 10%, and 5% of the nominal value, respectively. In the following, we perform the experiments for two variation setting: (1) both L_{gate} and V_{th} have skew-normal distributions [16]; and (2) L_{gate} has a normal distribution and V_{th} has a Poisson distribution. The experimental setting is shown in Table 5.1.

Setting	L_{gate} dist	V_{th} dist	$3\sigma_g$	$3\sigma_s$	$3\sigma_r$
(1)	Skewnormal	Skewnormal	10%	10%	5%
(2)	Normal	Poisson	10%	10%	5%

Table 5.1: Experiment setting. The 3σ value is the normalized with respect to the nominal value.

Fig. 5.3 illustrates the PDF comparison for circuit s15850 under variation setting (1). In the figure, delay is normalized with respect to the nominal value. From the figure, we find that, compared to the Monte-Carlo simulation, the *Quad SSTA* is the most accurate, the next is *Semi-Quad SSTA*, and *Lin SSTA* is least accurate. Such result is expected, because the *Quad SSTA* captures all the second-order effects; the *Semi-Quad SSTA* captures only partial second order effects; while the *Lin SSTA* captures only linear effects and ignores all non-linear effects. Moreover, *Fourier SSTA* [40] has similar accuracy as *Semi-Quad SSTA* because they both apply semi-quadratic delay model. In addition, we also find that all these three non-Gaussian SSTA is more accurate than *Lin Gau* [163].

Table 5.2 compares the run time in second (T), and the error percentage of mean (μ), standard deviation (σ), and skewness (γ) under variation setting (1). In the table, the error percentage of mean (μ), standard deviation (σ), and 95-percentile point (95%) is computed as $100 \times (MC\_value - SSTA\_value)/\sigma_{MC}$, and the error percentage of skewness γ is computed as $100 \times (\gamma_{MC} - \gamma_{SSTA})/\gamma_{MC}$. Moreover, the average error in the table is average of the absolute value; and the average runtime is the average runtime ratio between SSTA and Monte-Carlo simulation. For each benchmark, G

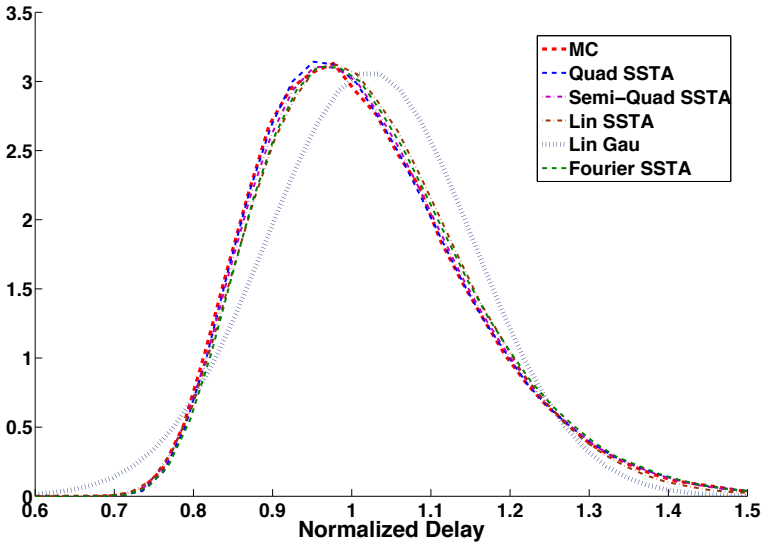


Figure 5.3: PDF comparison for circuit s15850.

refers to the number of gates; and N refers to the total number of variation sources. For fair comparison, when we perform *Lin Gau* on non-Gaussian variation sources, we first approximate the non-Gaussian random variables to the Gaussian random variables by matching the mean and variance, then use *Lin Gau* to obtain the linear canonical form of the circuit delay, finally, we still use the non-Gaussian variation sources to reconstruct the PDF of the circuit delay. From the table, we see that for the *Quad SSTA*, the error of mean, standard deviation, and 95-percentile point is within 0.3%, and the error of skewness is within 5%. *Semi-Quad SSTA*, *Lin SSTA* result similar mean and standard deviation error, but the error of skewness and 95-percentile point is much larger, especially for the *Lin SSTA*, the error of skewness is up to 30% and the error of 95-percentile point is up to 2%. This is because the *Lin SSTA* ignores all non-linear effects which significantly affect the skewness, and the inaccurate skewness results larger error of 95-percentile point. Compared to *Lin Gau*, all the three non-Gaussian SSTA methods predict more accurate mean values and 95-percentile points, which is the most important timing characteristic, and the non-linear SSTA methods,

Quad SSTA and *Semi-Quad SSTA* also give much more accurate skewness. Moreover, we also find that both *Semi-Quad SSTA* and *Lin SSTA* have similar run time as *Lin Gau*, but the run time of *Quad SSTA* is longer especially when the number of variation sources is large. This is because the computational complexity of *Semi-Quad SSTA*, *Lin SSTA*, and *Lin Gau* is the same, but *Quad SSTA* has higher complexity than the other two SSTA methods. We also observe that compared to *Fourier SSTA*, *Semi-Quad SSTA* has similar accuracy with almost 70X speed up. This is due to the fact that *Semi-Quad SSTA* uses the same semi-quadratic delay model as *Fourier SSTA*, while the computational complexity of *Semi-Quad SSTA* ($O\{n\}$) is lower than that of *Fourier SSTA* ($O\{nk^2\}$), where n is the number of variation sources, and k is the maximum number of orders of Fourier Series [40]. After all, the run time of all the SSTA methods is significantly shorter than the Monte-Carlo simulation.

Table 5.3 illustrates the results under variation setting (2). From this table, we can find the similar trend as Table 5.2. Moreover, we also find that our methods are still highly accurate under such variation setting. For *Quad SSTA*, the error of mean, standard deviation, and 95-percentile point is within 1%, and the error of skewness is within 6%. This shows that our approach works well for different distributions.

bench	G	N	*Quad SSTA*					*Semi-Quad SSTA*					*Lin SSTA*					*Lin Gau*					*Fourier SSTA*					MC
name			μ	σ	γ	95%	T	μ	σ	γ	95%	T	μ	σ	γ	95%	T	μ	σ	γ	95%	T	μ	σ	γ	95%	T	T
s208	61	12	-0.02	0.04	1.03	-0.03	0.01	-0.02	0.04	-10.7	-0.23	0.01	-0.02	-0.32	-22.8	-1.1	0.01	-0.02	-0.32	-22.8	-2.99	0.01	-0.04	-0.07	-9.5	-0.4	1.01	15.3
s386	118	18	-0.08	0.01	2.52	-0.03	0.01	-0.05	-0.05	-9.45	-0.36	0.01	-0.06	-0.4	-22.5	-1.23	0.01	-0.04	-0.36	-23	-2.96	0.01	0.02	-0.02	-10.9	0.36	1.17	33.6
s400	106	25	0.23	-0.06	-0.88	0.13	0.03	0.3	-0.12	-15.2	-0.21	0.01	0.29	-0.65	-31.4	-1.51	0.01	0.35	-0.66	-31.7	-3.56	0.01	-0.17	-0.24	-10	-0.2	1.28	38.7
s444	119	25	-0.1	-0.12	2.92	-0.17	0.04	-0.09	-0.15	-12.2	-0.51	0.01	-0.09	-0.56	-26.8	-1.56	0.01	-0.07	-0.5	-27.1	-3.37	0.01	-0.02	-0.22	-9.95	-0.57	1.38	41.9
s832	262	33	-0.06	-0.04	-0.11	-0.17	0.16	-0.06	-0.05	-12.1	-0.4	0.01	-0.06	-0.5	-27.6	-1.53	0.01	-0.06	-0.5	-27.6	-3.5	0.01	-0.1	-0.25	-14	-0.32	1.91	103
s953	311	43	-0.09	-0.11	2.03	-0.32	0.17	-0.15	-0.24	-9.87	-0.86	0.01	-0.17	-0.68	-27.7	-1.98	0.01	-0.01	-0.67	-28.1	-3.7	0.01	-0.02	-0.14	-7.23	-0.59	1.99	122
s1238	428	55	0.12	0.01	0.59	0.15	0.43	0.22	-0.04	-10.4	-0.04	0.01	0.21	-0.47	-26.5	-1.12	0.01	0.28	-0.45	-26.6	-2.98	0.01	-0.16	-0.04	-6.59	-0.63	2.17	201
s1423	490	68	-0.01	-0.08	-1.25	0.06	0.75	-0.01	-0.08	-10.2	-0.14	0.01	-0.01	-0.49	-25.2	-1.18	0.01	-0.01	-0.47	-25.3	-3.12	0.01	-0.21	-0.01	-4.15	-0.3	2.18	288
s1494	588	68	-0.08	-0.08	3.63	0.08	1.06	-0.07	-0.16	-6.02	-0.24	0.01	-0.08	-0.52	-20.6	-1.14	0.01	-0.01	-0.44	-21.4	-2.85	0.01	-0.2	-0.12	-4.41	-0.22	2.28	320
s5378	1004	82	0.06	-0.12	4.71	-0.16	4.95	0.09	-0.2	-6.72	-0.49	0.05	0.08	-0.66	-26.7	-1.65	0.02	0.18	-0.56	-27.4	-3.31	0.02	-0.07	-0.11	-5.68	-0.6	5.42	1238
s9234	2027	99	0.01	-0.11	0.98	-0.24	10.4	0.04	-0.23	-8.74	-0.61	0.09	0.02	-0.71	-28.4	-1.82	0.04	0.29	-0.66	-28.7	-3.43	0.03	0.04	-0.07	-13.7	0.61	6.89	2801
s13207	2573	115	-0.01	-0.01	3.04	-0.06	25.9	0.02	-0.09	-4.4	-0.33	0.15	0.01	-0.49	-22.2	-1.34	0.07	0.16	-0.49	-22.2	-3.09	0.06	0.09	-0.19	-13.3	0.54	9.04	4717
s15850	3448	135	0.18	-0.02	2.66	0.27	38.9	0.23	-0.07	-4.85	0.06	0.19	0.22	-0.52	-24.6	-1.11	0.08	0.51	-0.53	-24.7	-2.92	0.08	0.18	-0.09	-4.38	0.64	11.5	6484
s38417	8709	176	0.11	-0.01	2.22	0.07	202	0.13	-0.04	-3.8	-0.07	0.62	0.13	-0.49	-23.1	-1.21	0.26	0.24	-0.47	-23.2	-3.16	0.24	0.08	-0.19	-10.5	0.79	23.2	19116
s38584	11448	176	-0.05	-0.02	0.1	0.01	218	-0.04	-0.12	-6.37	-0.29	0.68	-0.06	-0.57	-25.1	-1.44	0.29	0.11	-0.57	-25.2	-3.31	0.25	-0.11	-0.19	-14.4	-0.53	25.1	19459
Ave	-	-	0.08	0.06	1.91	0.13	1/720	0.1	0.11	8.74	0.32	1/19814	0.1	0.54	25.4	1.39	1/35819	0.15	0.51	25.7	3.22	1/39248	0.1	0.13	9.25	0.49	1/260	-

Table 5.2: Error percentage of mean, standard deviation, skewness, and 95-percentile point for variation setting (1). Note:the error percentage of mean (μ), standard deviation (σ), and 95-percentile point (95%) is computed as $100 \times (MC\_value - SSTA\_value)/\sigma_{MC}$, and the error percentage of skewness γ is computed as $100 \times (\gamma_{MC} - \gamma_{SSTA})/\gamma_{MC}$.

bench	G	N	*Quad SSTA*					*Semi-Quad SSTA*					*Lin SSTA*					*Lin Gau*					*Fourier SSTA*					MC
name			μ	σ	γ	95%	T	μ	σ	γ	95%	T	μ	σ	γ	95%	T	μ	σ	γ	95%	T	μ	σ	γ	95%	T	T
s208	61	12	-0.03	0.05	1.64	-0.05	0.01	-0.03	0.04	-20	-0.35	0.01	-0.05	-0.07	-58.9	-0.81	0.01	-0.05	-0.07	-58.9	-1.17	0.01	0.02	-0.25	-15	0.63	0.99	15.4
s386	118	18	-0.07	0.01	5.54	-0.05	0.02	-0.05	-0.04	-16.5	-0.4	0.01	-0.08	-0.16	-58.2	-0.91	0.01	-0.08	-0.11	-58.8	-1.15	0.01	-0.01	-0.02	-16.5	-0.34	1.2	33.9
s400	106	25	0.25	-0.07	-2.48	0.25	0.02	0.32	-0.13	-25.7	-0.18	0.01	0.29	-0.3	-71.3	-0.93	0.01	0.35	-0.3	-71.9	-1.21	0.01	-0.15	-0.27	-20.8	-0.36	1.27	39
s444	119	25	-0.09	-0.12	3.27	-0.23	0.03	-0.08	-0.14	-22.4	-0.62	0.01	-0.1	-0.28	-65	-1.22	0.01	-0.09	-0.19	-65.1	-1.43	0.01	0.01	-0.09	-13.6	0.3	1.38	42.1
s832	262	33	-0.05	-0.04	-3.26	-0.14	0.14	-0.05	-0.05	-23.5	-0.48	0.01	-0.08	-0.19	-66.9	-1.1	0.01	-0.08	-0.19	-66.9	-1.45	0.01	-0.01	-0.18	-13.4	-0.38	1.93	103
s953	311	43	-0.09	-0.11	3.96	-0.44	0.18	-0.15	-0.23	-16.6	-1	0.01	-0.19	-0.38	-66.9	-1.67	0.01	-0.02	-0.35	-67.4	-1.75	0.01	-0.12	-0.22	-16.5	-0.38	1.95	122
s1238	428	55	0.11	-0.02	-3.92	0.09	0.43	0.21	-0.05	-22	-0.13	0.01	0.18	-0.19	-69.8	-0.76	0.01	0.24	-0.16	-70.2	-1	0.01	-0.17	-0.05	-16.1	-0.58	2.15	201
s1423	490	68	-0.01	-0.08	-4.96	0.01	0.75	-0.01	-0.09	-20.2	-0.22	0.01	-0.03	-0.22	-64.1	-0.82	0.01	-0.02	-0.2	-64	-1.2	0.01	0.16	-0.26	-16.9	0.59	2.19	288
s1494	588	68	-0.08	-0.11	1.71	0.03	1.05	-0.07	-0.18	-16.9	-0.32	0.01	-0.1	-0.3	-59	-0.85	0.01	-0.04	-0.21	-58.3	-1.05	0.01	0.17	-0.15	-16.7	0.6	2.29	322
s5378	1004	82	0.05	-0.17	1.26	-0.14	4.95	0.08	-0.25	-16.1	-0.54	0.04	0.04	-0.41	-68.3	-1.23	0.02	0.12	-0.27	-68.5	-1.24	0.01	-0.02	-0.1	-19.4	-0.39	5.44	1238
s9234	2027	99	0.01	-0.13	2.52	-0.22	10.4	0.04	-0.25	-12.5	-0.65	0.08	0.01	-0.42	-68.6	-1.39	0.03	0.25	-0.34	-69	-1.27	0.04	-0.1	-0.2	-15.3	-0.41	6.76	2795
s13207	2573	115	0.01	-0.02	5.21	0.01	25.7	0.04	-0.09	-7.95	-0.27	0.15	0.01	-0.23	-62.3	-0.89	0.06	0.14	-0.21	-62.4	-1.01	0.06	-0.11	-0.04	-13.2	-0.28	9.11	4720
s15850	3448	135	0.18	-0.07	0.4	0.18	38.8	0.22	-0.12	-12.1	-0.02	0.19	0.18	-0.27	-66.4	-0.77	0.08	0.47	-0.25	-66.5	-0.84	0.07	0.18	-0.16	-19.9	0.4	11.3	6477
s38417	8709	176	0.1	-0.03	4.59	0.05	202	0.12	-0.06	-6.09	-0.14	0.62	0.09	-0.2	-63.8	-0.84	0.27	0.2	-0.17	-63.9	-1.07	0.24	0.13	-0.27	-17.4	0.6	23.6	19097
s38584	11448	176	-0.06	-0.08	-3.38	-0.04	218	-0.05	-0.16	-14.7	-0.32	0.68	-0.09	-0.31	-65.2	-1.03	0.29	0.08	-0.3	-65.3	-1.22	0.26	-0.07	-0.07	-19.8	-0.6	25.2	19460
Ave	-	-	0.08	0.07	3.21	0.13	1/681	0.1	0.13	16.9	0.38	1/20497	0.1	0.26	65	1.01	1/37942	0.15	0.22	65.1	1.2	1/42392	0.09	0.16	16.7	0.46	1/260	-

Table 5.3: Mean, standard deviation, and skewness comparison for variation setting (2).

5.7 Conclusions

In this chapter, we have proposed a new method to approximate the max operation of two non-Gaussian random variables using second-order polynomial fitting. It has been shown that such approximation is more accurate than the approximation using linear fitting through tightness probability. By applying such approximation, we present new SSTA algorithms for three different delay models, i.e., quadratic model, quadratic model without crossing terms (semi-quadratic model), and linear model. All atomic operations of these algorithms are performed by closed-form formulas, hence they are very time efficient. The computational complexity of both the linear delay model and the semi-quadratic delay model is linear to the number of variation sources, and that of the quadratic delay model is cubic (third-order) to the number of variation sources. Moreover, the computational complexity is linear to the circuit size for all three delay models. The SSTA with semi-quadratic delay model results similar error as the SSTA using Fourier Series [40] with almost 70X speed up. Moreover, compared to Monte-Carlo simulation for non-Gaussian variation sources, the SSTA with quadratic delay model results the error of mean, standard deviation, skewness, and 95-percentile point is within 1%, 1%, 6%, and 1%, respectively.

CHAPTER 6

Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across-Wafer Variability

Modeling spatial variation is important for statistical analysis. Most existing works model spatial variation as spatially correlated random variables. We discuss process origins of spatial variability, all of which indicate that spatial variation comes from deterministic across-wafer variation, and purely random spatial variation is not significant. We analytically study the impact of across-wafer variation and show how it gives an appearance of correlation. We have developed a new die-level variation model considering deterministic across-wafer variation and derived the range of conditions under which ignoring spatial variation altogether may be acceptable. Experimental results show that for statistical timing and leakage analysis, our model is within 2% and 5% error from exact simulation result, respectively, while the error of the existing distance-based spatial variation model is up to 6.5% and 17%, respectively. Moreover, our new model is also 6X faster than the spatial variation model for statistical timing analysis and 7X faster for statistical leakage analysis.

6.1 Introduction

In Chapter 5, we developed an efficient and accurate SSTA flow in which process variation is modeled as inter-die, within-die spatial, and within-die random variations. However, new measurement results show that the variation model in Chapter 5 is not accurate. Therefore, we developed a more accurate and efficient variation model and applied it on the SSTA flow presented in Chapter 5.

There are several existing works that focus on analyzing and modeling process variation [6, 32, 172, 106, 108, 134, 24, 25, 173]. The simplest method models process variation as the sum of inter-die (global) variation and independent within-die (local random) variation [25]. Later, it was observed that within-die variation is spatially correlated and the correlation depends on the distance between two within-die locations. [6, 32] model spatial variation as correlated random variables, and principle component analysis is applied to perform statistical timing analysis. In this model, a chip is divided into several grids and each grid has its own spatial variation. The spatial variations of different grids are correlated and the correlation coefficient depends on the distance between two grids. [172, 173] focuses on the extraction of spatial correlation and it models the correlation coefficient as a function of distance. Several more complex spatial correlation models have been proposed in [45, 105, 189, 55, 64].

In contrast to the spatial correlation models, process oriented modeling has concluded that within-die spatial variation is caused by deterministic across wafer and across-field variation while purely random within-die spatial variation is not significant [190, 54, 50]. However, in practical design flow, designers do not know the within-wafer location or within-field location of each die; therefore, we need to analyze the impact of across-wafer variation and across-field variation on die-scale. Since silicon measurements cited in this chapter indicate that across-wafer variation is much more significant than the across-field variation, we consider only across-wafer varia-

tion in this chapter, but the approach can be easily extended to account for across-field variation.

In this chapter, we first analyze the impact of deterministic across-wafer variation on spatial correlation. We observe that when quadratic across-wafer variation model is used as in [54, 186, 125]:

1. Different locations of the chip may have different mean and variance. Such differences increase when the chip size increases.

2. When chip size is small, the correlation coefficients for a certain Euclidean distance are within a narrow range. This explains why most existing works find that spatial correlation is a function of distance.

3. Within-die spatial variation is *NOT* spatially correlated when across-wafer systematic variation is removed.

4. Within-die spatial variation is *NOT* independent from inter-die variation.

5. If chip size is not large, the two-level inter-/within-die decomposition of process variation is still very accurate.

Based on our analysis, we propose three accurate and efficient spatial variation models considering across-wafer variation. We then apply our new model to the SSTA flow introduced in Chapter 5. Experimental results show that our model is more accurate and efficient compared to the distance-based spatial variation model in [172, 173]. Compared to the exact simulation, the error of our model for statistical timing analysis is within 2% and the error for statistical leakage analysis is within 5%. On the other hand, the error of the distance-based spatial correlation model is up to 6.5% for statistical timing analysis and up to 17% for statistical leakage analysis. Moreover,

our model is 6X faster than the distance-based spatial correlation model for statistical timing analysis and 7X faster for statistical leakage analysis.

The rest of this chapter is organized as follows: Section 6.2 discusses the physical causes for across-wafer variation; Section 6.3 analyzes the impact of across-wafer variation on die-scale; Section 6.4 discusses the case when the across-wafer variation is not a perfect parabola; Section 6.5 introduces the new variation models; the new models are applied to statistical timing analysis in Section 6.6 and statistical leakage analysis in Section 6.7; Section 6.8 summarizes the advantages and disadvantages of different variation models; and finally Section 6.9 concludes this chapter.

6.2 Physical Origins of Spatial Variation

In silicon manufacturing, there are many steps that cause non-uniformity in devices across the wafer. Interestingly, most of these processes by the very nature of the equipment follow a radially varying trend across the wafer. Most processes are "center-fed" or "edge-fed" with the boundary conditions at the edge of wafer being substantially different. Moreover, wafers are often rotated to increase process uniformity across them which further leads to radial behavior of non-uniformity. This is further exacerbated by advent of single-wafer processing for 300mm wafers.

For example, overlay error includes errors in the position and rotation of the wafer stage during exposure, wafer stage vibration, and the distortion of the wafer with respect to the exposure pattern [139]. Magnification and rotation components of overlay error increase from center of the wafer outwards.[1] During chemical vapor deposition (CVD) step, species depletion and temperature non-uniformity on the wafer at lower temperatures may cause thickness non-uniformity [159, 132]. Redeposition effect in

[1] Overlay error can directly impact critical dimension in double patterning.

physical vapor deposition (PVD) [27] may cause non-uniformity. Moreover, center peak shape of the RF electric field distribution [77] also leads to a center peak shape of etch rate, and chamber wall conditions [78] also cause etch rate non-uniformity. In real processes, the wafers are rotated to improve uniformity. [78, 27] show that the etch rate varies radially across the wafer: the etch rate is high at the center of the wafer and decreases toward the edges. Post-exposure bake (PEB) temperatures are higher at the center of the wafer and decrease outwards [185]. Similarly, other processes ranging from resist coat to wafer deformation due to vacuum chuck holding it follow a bowl-shaped trend across the wafer. All these processes cause a systematic across-wafer variation in physical dimensions.

Across-wafer variation of gate length observed in several recent silicon measurements [153, 54, 186, 125] validates our arguments. [126] also shows that ring oscillator frequency and leakage current decrease from the center to the edge of the wafer. Figure 6.1 shows industry data of ring oscillator frequency for wafers from two different industry processes. Process 1 is with 45nm technology and process 2 is with 65nm technology. From the figure, we see that for both process, ring oscillator frequency decreases from the center to the edge of the wafer. Moreover, it has also been shown that there is no spatial correlation for threshold voltage variation [190]. Therefore, the across-wafer frequency and leakage variation is mainly caused by gate length variation.

It has been shown that for process 1, the across-wafer frequency variation can be approximated as a quadratic function (a parabola) [126]. For process 2, the across-wafer variation is not a perfect parabola as process 1. However, it follows a systematic trend and the ring oscillator frequency decreases from the center to the edge of the wafer. Since the amount measurement data of for process 2 (more than 300 wafers) is much more than process 1, in the rest of this chapter, all of our simulation and

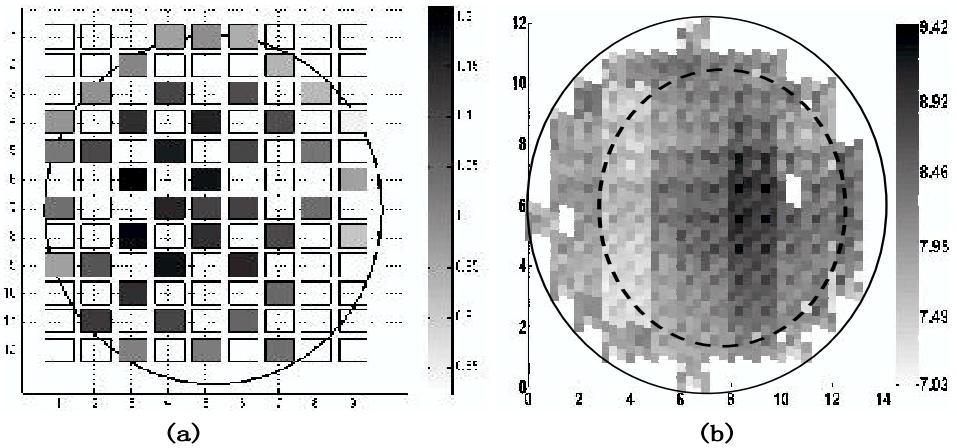


Figure 6.1: Ring oscillator frequency within a wafer. (a) Process 1; (b) Process 2.

experiments are based on the measurement result of process 2.

Besides across-wafer variation, lithography-induced effects such as lens aberrations can lead to systematic across-field variation and across-die variation. Across-die variation can be modeled as within-die deterministic mean shift and will not cause within-die spatial correlation. Moreover, silicon measurements cited in this chapter indicate that across-wafer variation is much more significant (probably due to advancements in resolution enhancement and lithographic equipment) than across-field and across-die variation. Hence, for simplicity, we consider only across-wafer variation in this chapter.

6.3 Analysis of Wafer Level Variation and Spatial Correlation

To analyze the impact of across-wafer variation on die-scale, we assume the across-wafer variation to be a quadratic function as in[24, 54, 186, 125]:

$$v_p = ax_w^2 + by_w^2 + cx_w + dy_w + m_w + m_f + m_d + m_r \tag{6.1}$$

where v_p is a variation source, such as L_{eff}, a, b, c, and d are function coefficients, which are obtained from fitting the measurement data from industry process as shown in Figure 6.1, m_w comprises inter-die random, inter-wafer, inter-lot variation and the fitting error of quadratic fitting as in (6.1); m_f and m_d are the across-field and across-die variation, respectively, as discussed in Section 6.2, we consider only across-wafer variation and ignore these two types of variations (m_f and m_d) in this chapter; m_r is the random noise, and (x_w, y_w) is across-wafer location. In the rest of this section, we will analyze the spatial variation based on the above model. Table 7.3 summarizes the mathematical notations used in this section.

We obtain the coefficients of the above across-wafer variation model by fitting the industrial 65nm process measured ring oscillator delay with 300 wafers from 23 lots. In the rest of this section, all simulations are based-on this extracted model.

6.3.1 Variation of Mean and Variance with Location

According to (6.1), it is easy to find that for a die, whose center lies on (x_c, y_c) wafer coordinates, the variation at location (x, y) in a die (assuming the coordinate of the center of the die to be $(0,0)$) is:

$$v_p(x,y) = a(x_c+x')^2 + b(y_c+y')^2 + c(x_c+x') + d(y_c+y') + m_w + m_r(x,y) \quad (6.2)$$

The definitions of (x_c, y_c) and (x', y') are in Table 7.3. In this section, we assume that a, b, c, and d are fixed for a process. In practice, these coefficients may vary slightly for wafer-to-wafer or lot-to-lot. We will further discuss this in Section 6.4.

In real design flow, the die location in the wafer (x_c, y_c) is not known to designers. We can convert the wafer-level systematic variation model to a die-level model by noting that the dies are always distributed evenly on the wafer. Therefore, we may model x_c and y_c as random variables which are evenly distributed in the center at $(0,0)$

Symbols	Description
(x_c, y_c)	Location of the center of the die in the wafer
r_w	wafer radius
m_w	Inter-die variation
m_r	Within-die random variation
σ_m^2	Variance of m_w
σ_r^2	Variance of m_r
a, b, c, d	Across-wafer variation coefficients
(l_x, l_y)	x and y dimension die size
(x_w, y_w)	Within-wafer location
(x, y)	Within-die location
ω	Angle between the die and wafer coordinate
$v_p(x, y)$	Variation of within-die location (x, y)
x'	$x' = x\cos\omega + y\sin\omega$
y'	$y' = x\sin\omega - y\cos\omega$
l'_x	$l'_x = l_x\cos\omega + l_y\sin\omega$
l'_y	$l'_y = l_x\sin\omega - l_y\cos\omega$
x''	$x'' = ax' + c/2$
y''	$y'' = by' + d/2$
l''_x	$l''_x = ax' + c/2$
l''_y	$l''_y = by' + d/2$
$r_{d\mu}$	$r_{d\mu} = \sqrt{x''^2/a + y''^2/b}$
$r_{d\sigma}$	$r_{d\sigma} = \sqrt{x''^2 + y''^2}$
δ	Euclidean distance between (x''_1, y''_1) and (x''_2, y''_2)
	$\delta = \sqrt{(x''_1 - x''_2)^2 + (y''_1 - y''_2)^2}$
r_m	$r_m = \sqrt{l_x^2 + l_y^2}$
r'_m	$r'_m = \sqrt{l_x'^2 + l_y'^2}$
r''_m	$r''_m = \sqrt{l_x''^2 + l_y''^2}$
k_0	$k_0 = r_w^2(a+b)/4 - c^2/4a - d^2/4b$
k_1	$k_1 = r_w^4(a^2+b^2)/16 - r_w^4 ab/24 + \sigma_m^2$
k_2	$k_2 = k_1/r_w^2$
α	$\alpha = x''_1 x''_2 + y''_1 y''_2$
β	$\beta = \sigma_r^2/r_w^2$
in the wafer s_0	$s_0 = \cos^2\omega(al_x^2 + bl_y^2) + \sin^2\omega(bl_x^2 + al_y^2)$
v_g	Inter-die variation
v_s	Within-die spatial variation
v_l	Within-die random variation

Table 6.1: Notations.

with radius r_w (radius of the wafer). It is easy to see that x_c and y_c are identical and their PDF is [2]:

$$PDF(x_c) = 2\sqrt{r_w^2 - x_c^2}/(\pi r_w) \qquad -r_w < x_c < r_w \tag{6.3}$$

In this case, the variation at location (x,y), $v_p(x,y)$, is expressed as a function of four random variables: x_c, y_c, m_w, and $m_r(x,y)$. We may easily calculate the mean and variance of $v_p(x,y)$:

$$\mu_{v_p(x,y)} = k_0 + x''^2/a + y''^2/b = k_0 + r_{d\mu}^2 \tag{6.4}$$

$$\sigma^2_{v_p(x,y)} = k_1 + r_w^2(x''^2 + y''^2) = k_1 + \sigma_r^2 + r_w^2 r_{d\sigma}^2 \tag{6.5}$$

where x'', y'', $r_{d\mu}$, $r_{d\sigma}$, k_0, and k_1 are defined in Table 7.3.

From (6.4) and (6.5), it is interesting to note that different die locations may have different means and variances[3]. The location (x_0, y_0) having the smallest mean and variance is given by:

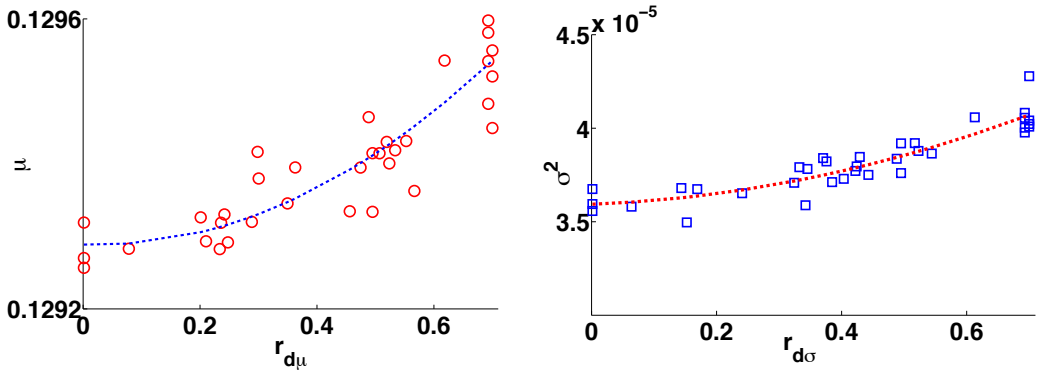
$$x_0 = -c\cos\omega/2a - d\sin\omega/2b \tag{6.6}$$

$$y_0 = d\cos\omega/2b - c\sin\omega/2a \tag{6.7}$$

The locations farther off from (x_0, y_0) will have larger mean and variance. Figures 6.2(a) and 6.2(b) illustrate the mean and variance of ring oscillator delay for different $r_{d\mu}$ (or $r_{d\sigma}$).

[2] x_c and y_c are not independent. In order to generate samples of x_c and y_c, we may assume $x_c = r_s\cos\theta$ and $y_c = r_s\sin\theta$, where r_s and θ are independent. r_s follows triangle distribution ranging from 0 to r_w, θ follows uniform distribution ranging from 0 to 2π

[3] Such difference is caused by the nonlinearity of the across-wafer variation function (we assume quadratic function as in Equation(6.1)). If the across wafer variation function is a piecewise linear function, the mean and variance will be the same for all locations on a die.



(a) μ V.S. $r_{d\mu}$ (b) σ^2 V.S. $r_{d\sigma}$

Figure 6.2: Mean and variance for different r_d.

6.3.2 Appearance of Spatial Correlation

Besides mean and variance, we are also interested in the covariance between two locations (x_1, y_1) and (x_2, y_2). From (6.2), we may calculate the covariance as:

$$Cov = k_1 + r_w^2(x_1'' x_2'' + y_1'' y_2'') \tag{6.8}$$

With covariance and variance calculated as above, we may obtain the correlation coefficient as:

$$\rho = \sqrt{\frac{k_2^2 + 2k_2\alpha + \alpha^2}{(k_2+\beta)^2 + (r_{d\sigma 1}^2 + r_{d\sigma 2}^2)(k_2+\beta) + r_{d\sigma 1}^2 r_{d\sigma 2}^2}} \tag{6.9}$$

where α, β, and k_2 are defined in Table 7.3. From (6.9), we obtain the upper bound and lower bound of the correlation coefficient for a certain Euclidean distance:

$$\rho \leq \rho_u = \sqrt{1 - \frac{\delta^2 k_2 + \delta\beta/2 + 2\beta k_2 + \beta^2}{(k_2+\beta)^2 + 2r_m''^2(k_2+\beta) + r_m''^4}} \tag{6.10}$$

$$\rho \geq \rho_l = \sqrt{1 - \frac{\delta^2(k_2 + r_m''^2 - \delta^2/4 + \beta) + \beta(2k_2 + r_m''^2)}{(k_2+\beta)^2 + \delta^2(k_2+\beta)/2 + \delta^4/16}} \tag{6.11}$$

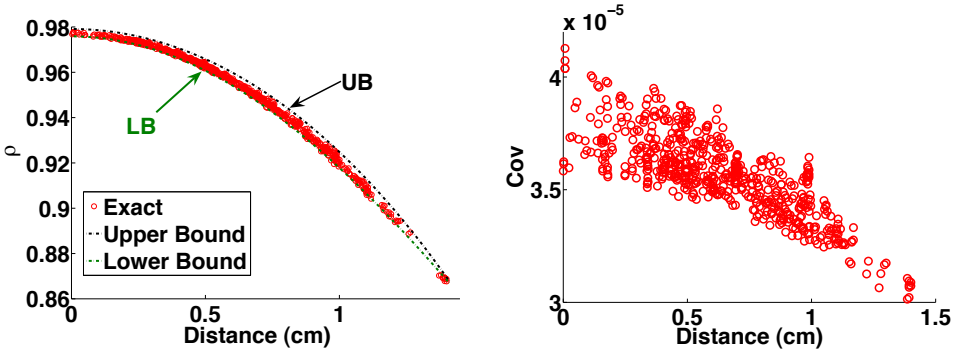
where δ is the Euclidean distance between (x_1'', y_1'') and (x_2'', y_2''), l_x'' , l_y'', and r_m'' are defined in Table 7.3. From the upper bound and lower bound, we may also calculate

the range of correlation coefficient:

$$\rho_u - \rho_l \leq \sqrt{r_m^2/(r_m^2 + k_2 + \beta)} \tag{6.12}$$

Notice that usually the wafer size is much larger than the die size, that is $k_2 \gg r_m^2$, therefore, the range of correlation coefficient for a certain distance is very small. Moreover, from the above equation, we also find that when the variances of the inter-die residual and within-die random variation increase, the range decreases. This explains why most existing works [172, 173, 45] find that spatial correlation is a function of distance.

Figure 6.3(a) illustrates the exact data for 40 locations, the upper bound and the lower bound. From the figure, we find that the range of ρ for a certain distance is very small. Although the correlation coefficient is within a narrow range, covariance is not, as shown in Figure 6.3(b). This is because of the differences of variance across the die.



(a) ρ (b) Covariance

Figure 6.3: Apparent spatial correlation and covariance as a function of distance.

Figure 6.4 shows the correlation coefficient for within-die variation after subtracting the mean variation of the die (mainly caused by across wafer variation) [4]. We

[4]In the figure, the correlation coefficient can be a negative number when distance is large. This is because after subtracting the mean, when the within-die variation of one corner increases, the within-die variation of the opposite corner must decrease. That means, the within-die variations of opposite corners are negative correlated. Moreover even when two locations are very close, if they lie on the opposite side of the center, their correlation is still near zero.

observe that the within-die spatial variation is almost *NOT* spatially correlated, as empirically observed in [172, 173]. This further validates that the spatial variation is caused by systematic across-wafer variation.

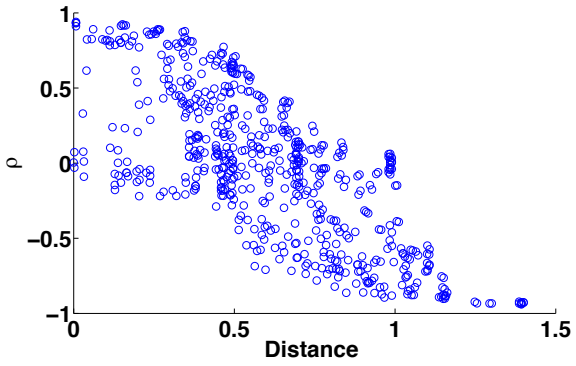


Figure 6.4: Correlation coefficient for within-die spatial variation after inter-die variation is removed.

6.3.3 Dependence between Inter-Die and Within-Die Variation

In most existing variation models, process variation is decomposed into inter-die, within-die spatial, and within-die random variation:

$$v_p = v_g + v_s + v_l \tag{6.13}$$

where v_g is the inter-die variation, v_s is the within-die spatial variation, and v_l is the within-die variation. Usually v_g is modeled as the variation of the chip mean, v_l is the pure random local variation, and v_s is the residual. v_g, v_s, and v_l are assumed to be independent.

With the variation model in (6.2), we may also calculate the inter-die, within-die spatial, and within-die random variation. Inter die variation is calculated as the varia-

tion of the chip mean:

$$\begin{aligned} v_g &= \frac{1}{l_x l_y} \iint\limits_{\substack{|x|<l_x/2 \\ |y|<l_y/2}} v_p(x,y)dxdy \\ &= ax_c^2 + by_c^2 + cx_c + dy_c + m_w + s_0 \end{aligned} \tag{6.14}$$

where s_0 is defined in Table 7.3.

Within-die random variation is the local random variation: $v_l = R$, and within-die spatial variation is calculated as the remaining variation:

$$\begin{aligned} v_s(x,y) &= v_p(x,y) - v_g - v_l \\ &= r_{d\mu}^2 + 2ax'x_c + 2by'y_c - s_1 \end{aligned} \tag{6.15}$$

where s_1 is defined in Table 7.3. From the above equations, we find that both inter-die and within-die spatial variations are functions of random variables x_c and y_c. Hence, we may not decompose process variation into independent inter-die and within-die spatial variation.

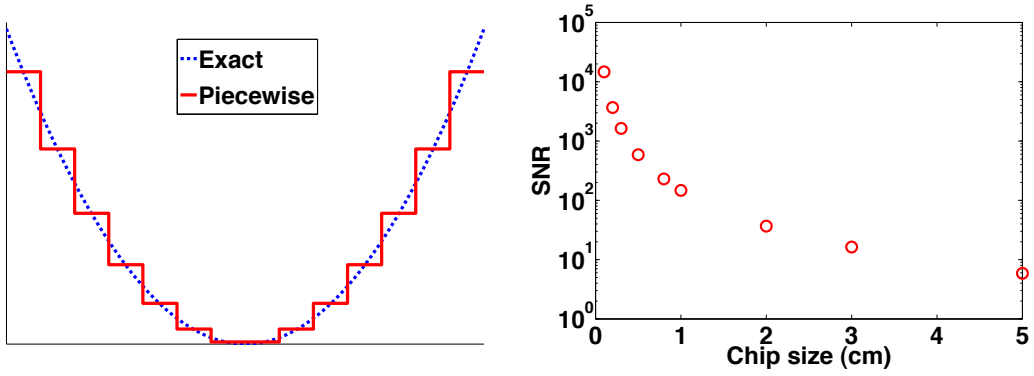
6.3.4 When can Spatial Variation be Ignored?

In this section, we analyze the accuracy of the simple two-level inter-/within-die variation model for different chip sizes. If we only consider inter-/within-die variation, we may lump the across-wafer variation into inter-die variation, that is, approximate the across-wafer variation as a piecewise constant function, as shown in Figure 6.5(a). To evaluate the impact of the approximation error, we may treat such approximation error as noise and the process variation as signal; and then evaluate the signal to noise ratio. In order to do this, we calculate the mean square approximation error and the total variance of variation. The signal to noise ratio when ignoring the spatial variation

is given as:

$$\begin{aligned} SNR &= \sigma^2_{total}/MSE \\ &\approx \frac{6abr_w^4 + 6(c+d)r_w^2 + \sigma_M^2 + \sigma_R^2}{abr_w^2(l_x^2 + l_y^2) + 2(c+d)l_x l_y} \end{aligned} \tag{6.16}$$

It can be seen that MSE depends on chip size. When chip size is small, MSE is small. This is because we approximate the across-wafer variation as a piecewise constant function with small steps, hence such approximation is accurate. Figure 6.5(b) illustrates the SNR for different die sizes. It can be seen that the SNR decreases when die size increases as expected. We also observe that when chip size (l_x and l_y) is smaller than 1cm, the SNR is up to 100. That means, two-level inter-/within-die variation model is accurate.



(a) Piecewise constant

(b) SNR V.S. chip size.

Figure 6.5: Approximating across-wafer variation.

6.4 General Across-Wafer Variation Model

In the previous section, we assumed that the across-wafer variation is a quadratic function as shown in Equation 6.2. In practice, across-wafer variation may not be an exact

parabola. Moreover, the across-wafer variation may be slightly different for different wafers. Therefore, there will be some fitting residual after subtracting the across-wafer parabola:

$$v(x,y) = v_p + v_r \tag{6.17}$$

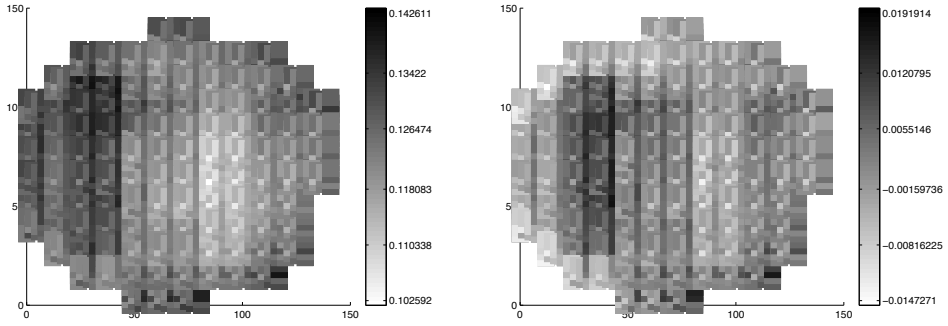
where v_p is the quadratic across-wafer variation model as shown in (6.2) and V_r is fitting residual. In the previous section, we assume that the fitting residual is lumped into inter-die random variation. However, the fitting residual contains not only inter-die random variation but a systematic trend of within-die variation. Figure 6.6(a) illustrates the original delay variation across the wafer, and Figure 6.6(b) illustrates the fitting residual of a wafer delay variation after subtracting the quadratic across-wafer variation function. From the figure, we observe that for each die, there is a systematic trend: If one corner of the die is faster, then the opposite corner is slower. From the die point of view, such trend will also introduce some spatial correlation. In order to model this trend, we approximate the fitting residual as a linear function of within-die location:

$$v_r(x',y') = s_x x + s_y y + m'_w \tag{6.18}$$

where s_x and s_y are x-dimension and y-dimension slope of within-die trend, respectively, and m'_w is inter-die part of the residual, which can be lumped into inter-die random variation. We also find that the trends of within-die variation for each die are different for different dies. In this case, we may model s_x and s_y as random variables. Figure 6.7 illustrates the distribution of s_x and s_y obtained from measurement data of process 2. From the figure, we find that both s_x and s_y are almost uncorrelated and they both follow Gaussian distribution.

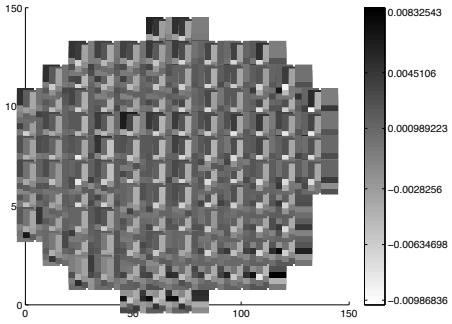
Notice that when we model the fitting residual as a linear within-die variation trend, two more random variables s_x and s_y are introduced. This makes the variation model

more complicated. When the across-wafer variation is a perfect parabola and the fitting residual is not significant, we may just lump the fitting residual in to inter-die and random variation.



(a) Original delay variation.

(b) Residual after subtracting Equation(6.2).

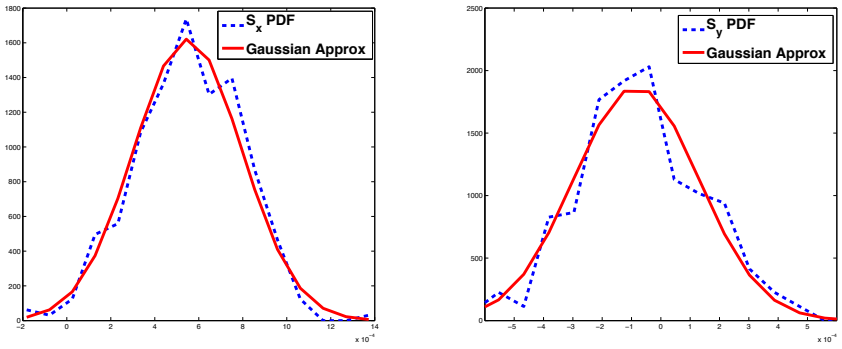


(c) Residual after subtracting (6.18).

Figure 6.6: Ring oscillator frequency within a wafer.

In addition, we also observe that after subtracting the model of fitting residual in (6.18), the remaining variation is almost uncorrelated, as illustrated in Figure 6.6(b) [5].

[5]It seems that there is a systematic within-die variation pattern. In this chapter, we assume that all this within-die pattern to be independent random variation. In practice, such within-die variation pattern can be modeled as mean shift.



(a) PDF of S_x (b) PDF of S_y

Figure 6.7: PDF of S_x and S_y.

Combining (6.18) and (6.2) we obtain a general die-level across-wafer variation model:

$$\begin{aligned} v_p(x,y) &= a(x_c+x')^2+b(y_c+y')^2+c(x_c+x')+ \\ &\quad d(y_c+y')+s_x x+s_y y+m_w+m_r(x,y) \end{aligned} \tag{6.19}$$

6.5 Modeling Spatial Variability

As discussed in Section 6.1, spatial variation largely comes from the deterministic across-wafer variation. Hence, modeling the within-die variation as spatial-correlated random variable is not accurate as discussed in Section 6.3.

In this section, we introduce three new spatial variation models considering across-wafer variation:

- *S*lop *A*ugmented *A*cross-*W*afer variation model (*SAAW*).
- *Q*uadratic *A*cross-*W*afer variation model (*QAW*).

- *L*ocation *D*ependent *A*cross-*W*afer variation model (*LDAW*).

In the following of this section, we will discuss these models in detail:

6.5.1 Slope Augmented Across-Wafer Model

(6.19) calculates the variation for a given location (x, y). In the equation, the die location (x_c, y_c) are modeled as random variables and their PDF is shown in Equation (6.3). (6.19) provides a new spatial variation model. We refer to this new model as *S*lope *A*ugmented *A*cross-*W*afer variation model (*SAAW*) .

Notice that in *SAAW* model, there are only six random variables, inter-die random variation m_w, within-die random variation m_r, die location within the wafer x_c and y_c, and slope of fitting residual s_x and s_y. However, for the traditional distance-based spatial variation model, the number of spatial variation sources depends on the number of grids. Larger chip needs more variables. Therefore, our new model not only models the across-wafer variation accurately but also is more efficient than the traditional spatial correlation model.

6.5.2 Quadratic Across-Wafer Model

As discussed in Section 6.4, when the across-wafer variation is a perfect parabola, the fitting residual is not significant[6], we may just lump the fitting residual into inter-die random variation and simplify (6.19) to (6.2). In this case, there are only four random variables, x_c and y_c, m_w, and m_r. We refer to this model as *Q*uadratic *A*cross-*W*afer variation model (*QAW*). Since *QAW* does not consider fitting residual, it is not as accurate as *SAAW*.

[6]For example, process 1 as discussed in Section 6.2.

6.5.3 Location Dependent Across-Wafer Model

As discussed in Section 6.3.4, when die size is small enough, applying the two-level inter-/within-die variation model does not introduce much error. However, inter-/within-die variation model still does not consider the mean and variance difference at different locations of a chip, as discussed in Section 6.3.1. To further improve the accuracy of inter-/within-die variation model, we account for this:

$$v(x,y) \approx m'_d + \mu_{v_p}(x,y) + \sigma_{v_p}(x,y)m'_r(x,y) \tag{6.20}$$

where m'_d is inter-die variation including inter-lot random, inter-wafer random, inter-die random, and across-wafer variation; $m'_r(x,y)$ is within-die variation including within-die random variation and residual of across-wafer variation; $\mu_{v_p}(x,y)$ and σ_{v_p} are mean and variance difference at different locations of a chip, which can be calculated from (6.4) and (6.5). We refer to the above model as *L*ocation *D*ependent *A*cross-*W*afer variation model (*LDAW*). Notice that in Equation 6.20, $\mu_{v_p}(x,y)$ and $\sigma_{v_p}(x,y)$ are deterministic value for a certain within-die location (x,y). Therefore, *LDAW* model has only two random variables: m'_d and m'_r which is the same as two level inter-/within-die model. Hence compared to inter-/within-die model, *LDAW* has similar efficiency but higher accuracy because *LDAW* considers mean and variance difference across the chip while inter-/within-die model does not.

6.6 Application of Spatial Variation Models on Statistical Timing Analysis

In this section, we apply our across-wafer variation models to statistical static timing analysis.

6.6.1 Delay Model

As discussed in Chapter 5, cell delay can be modeled as a quadratic function of variation sources, as shown in (5.46). For *SAAW* in (6.19) and *QAW* (6.2), each variation source is a quadratic function of random variables. Therefore, by applying *SAAW* or *QAW* on quadratic cell delay model, the cell delay becomes a 4^{th} order function of random variables. However, our SSTA flow in Chapter 5 can only handle a quadratic function of random variables. Therefore, we apply the moment matching technique in the way as Section 5.3 to match the 4^{th} function to a quadratic function of random variables by matching the mean and first two order joint moments. And then, we may apply the quadratic SSTA flow in Section 5.3 to estimate the chip delay variation. Notice that moment matching approximation is performed only once for all cells and does not increase the run time of SSTA. *LDAW* is a linear function of variation sources. Therefore, when applying *LDAW*, cell delay is a quadratic function of random variables, hence quadratic-SSTA can be applied. Moreover, as discussed in Section 5.6, Semi-quadratic SSTA greatly improves the speed with only a little accuracy loss compared to quadratic SSTA. Therefore, in this application, we may also ignore the crossing terms and apply semi-quadratic SSTA to improve efficiency.

Sometimes, cell delay model can be simplified as a linear function of variation sources, as shown in (5.114). In this case, applying *SAAW* or *QAW* results in a quadratic function of random variable and hence quadratic SSTA or semi-quadratic SSTA can be applied; and applying *LDAW* results in a linear function of random variables where linear SSTA can be applied.

6.6.2 Experimental Result

We have applied our new model to the SSTA flow introduced in Chapter 5. In order to verify the efficiency and accuracy, three comparison cases are defined: 1) Monte-Carlo simulation with the exact deterministic across-wafer variation model [7], which is the golden case for comparison; 2) distance-based spatial correlation model from [172, 173], which is referred to as *SP*atial *C*orrelation model (*SPC*); 3) two-level inter-/within-die variation model, which is referred to as *I*nter-/*W*ithin-die variation model (*IW*).

We apply all the above methods to the ISCAS85 suite of benchmarks in PTM 45nm technology[124]. We assume random placement for ISCAS85 circuits. Since process variation has smaller impact on interconnect delay than on logic cell delay, we only consider logic cell delay when calculating the full chip delay variation. In the experiment, we consider the gate length variation obtained from minimum square error fitting on the ring oscillator delay from industrial 65nm process (Process 2 as discussed in Section 6.2) measurement from the model as shown in (6.19). We obtain the across-wafer coefficients a, b, c, and d [8], fitting residual coefficients s_x and s_y [9], standard deviation of random inter-wafer, inter-die, and within-die variation as percentage with respect to the nominal value. Then we assume that the percentages of all the above coefficients to nominal value are the same at 45nm technology node and 65nm technology node.

[7] In the simulation, each wafer may have different across-wafer variation which is obtained from measurement data of process 2. We have simulated 318 wafers correspondent to 318 measured wafers.

[8] We apply quadratic function to fit the across-wafer variation for each wafer to obtain the fitting coefficients for each wafer, then use average coefficients of all wafers for our experiment.

[9] We obtain the slope of fitting residual s_x and s_y for each chip, and then calculate the mean and variance of s_x and s_y for all chips. In the experiment, we assume s_x and s_y to be Gaussian random variables with mean and variance obtained from the measurement data.

6.6.2.1 Full Chip Delay

In the experiment, we assume that the chip size is 2cm×2cm and the wafer radius is 15cm. Since ISCAS85 benchmarks are very small, the impact of spatial variation on delay is not significant within the circuit. In order to show such impact,we assume the benchmarks are stretched on a 2cm×2cm chip. In our experiment, for the *SPC* model, we divide the chip to $10 \times 10 = 100$ grids. Table 6.2 illustrates the percentage error of mean (μ), standard deviation (σ), and 95-percentile point (95%) and run time (T) of different variation models. In the table, we also compared the result of using quadratic cell delay model (Quad) and linear cell delay model (Lin). We only use quadratic cell delay model for golden case simulation (exact). The error is calculated as error of different variation models compared to the golden case simulation. For *SAAW* and *QAW*, we also compare the results of applying quadratic SSTA with crossing terms (*SAAW* Quad and *QAW* Quad) and applying SSTA without crossing terms (*SAAW* S-Quad and *QAW* S-Quad). From the table, we have the following observations:

- Compared to full quadratic SSTA, *semi-quadratic* SSTA (SSTA without crossing terms) achieves up to 8X speed up with less than 1% accuracy loss.
- *SAAW* is more accurate than *QAW*. This is because the fitting residual is significant for the measurement data, *QAW* ignores fitting residual and hence introduces more error.
- The error of *SAAW* using *semi-quadratic* SSTA is within 2% while the error of spatial correlation model is up to 6.5%.
- Compared to quadratic cell delay model, linear cell delay has less than 2% accuracy loss. This is because in our experiment, the cell delay variation is well approximated by a linear function.

- For linear cell delay model, *SAAW* achieves about 6X speed up compared to *SPC*. This is because there are 100 grids in the spatial correlation model, resulting in 37 spatial random variables[10], while *SAAW* has only 6 random variables.

- *LDAW* and *IW* are very efficient. However, both model has much larger error than others. This is because both model ignore correlation. *LDAW* is significantly more accurate than *IW* with no runtime penalty.

Since linear cell delay model and semi-quadratic SSTA are accurate. In the rest of this subsection, we assume linear cell delay and apply semi-quadratic SSTA for all experiments. Moreover, since *SAAW* is more accurate than *QAW* with only a small run time overhead, in the following experiment, we do not consider the *QAW* model in the following experiments.

[10]There are 100 correlated spatial random variables, we apply PCA to truncate some insignificant principle components and there remains 37 significant principle components.

In the above experiment, we only considered big chips. As discussed in Section 6.3.4, when the chip size is small, the impact on across-wafer variation at die level is not significant. In order to verify this, we perform delay estimation of ISCAS85 benchmarks stretching on different size chips. Table 6.3 shows the percentage error for different models under different chip size. From the table, we find that when chip size is small, *LDAW* and *IW* are accurate. Considering that *LDAW* has similar run time but is more accurate (although when chip size is small, the accuracy improvement is limited) compared to *IW*, *LDAW* is always better than *IW*.

6.6.2.2 Delay of Blocks on Different Locations on a Chip

The above experiment assumes that the benchmarks are stretched on a chip. However, in real design, especially for big chips, the design is separated into several blocks and each block only occupies a small region of a chip. In this case, the critical path is within a small region instead of spanning all over the chip. As discussed in Section 6.3.1, different chip locations may have different mean and variance. Therefore, when a block is placed at different locations on a chip, its delay variation may be different. In order to show such effect, we assume that the ISCAS85 benchmark circuit is placed (no stretched) in different locations on a chip: center (C), lower left corner (LL), lower right corner (LR), upper left corner (UL), and upper right corner (UR), and then calculate the delay variation with location. Since ISCAS85 benchmarks are very small, the impact of spatial variation on delay is not significant within the circuit. Therefore, in this experiment, we only compare two models *LDAW* and *IW*[11].

Table 6.4 compares the percentage error of *LDAW* and *IW* for ISCAS85 benchmarks placing on different locations on a 2cm$\times$2cm chip. From the table, we find that

[11]When the circuit is in a small region, *SAAW* and *QAW* will give similar result as *LDAW*, and *SPC* will give similar result as *IW*.

Bench-	delay	Exact			*SAAW* Quad				*SAAW* S-Quad				*QAW* Quad				*QAW* S-Quad				*LDAW**				*SPC**				*IW**			
mark	model	μ	σ	95%	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T
c1908	Quad	17.6	2.27	24.5	-0.4	-0.9	-1.0	146	-0.9	-1.7	-1.7	27	-0.8	-1.9	-2.3	54	-1.3	-2.5	-3.2	19	-2.1	-7.5	-6.9	9	-1.5	-4.2	-3.8	1450	-2.6	-10.2	-8.9	8
	Lin	-	-	-	-0.8	-1.5	-1.4	150	-1.4	-1.8	-2.0	26	-0.9	-3.6	-3.1	53	-1.2	-3.4	-3.9	18	-2.6	-7.5	-8.1	10	-2.1	-4.4	-4.0	135	-3.0	-11.5	-10.3	10
c3540	Quad	25.7	3.43	34.5	+0.4	+0.9	+0.7	212	-0.4	-1.3	-1.1	36	+0.4	-1.8	-1.2	76	-0.9	-2.1	-1.9	25	+0.4	-5.8	-4.6	13	-1.2	-4.8	-4.0	4210	-1.4	-7.3	-6.5	12
	Lin	-	-	-	-0.6	-1.2	-1.2	209	-1.1	-1.9	-1.6	35	-0.9	-3.6	-3.1	77	-1.2	-5.1	-3.9	27	-1.8	-6.5	-6.0	9	-2.0	-6.5	-5.7	202	-2.9	-9.3	-8.8	10
c7552	Quad	48.9	6.47	64.7	-0.6	+0.3	+0.2	435	-0.8	-0.2	-0.9	67	-0.8	-1.6	-1.4	115	-1.6	-1.5	-1.7	48	-2.7	-3.6	-4.0	20	-1.0	-2.3	-2.9	8182	-2.1	-6.7	-6.5	22
	Lin	-	-	-	-0.6	-0.5	-0.6	430	-1.5	-1.4	-1.6	101	-1.1	-1.3	-1.4	109	-1.9	-2.2	-2.8	79	-3.3	-4.6	-4.9	16	-3.3	-3.5	-4.3	433	-3.9	-8.9	-8.2	15

Table 6.2: Delay percentage error for different variation models. Note: the μ, σ, and 95-percentile point for exact simulation is in *ns*. Run time (T) is in *ms*. * for *LDAW*, *SPC*, and *IW*, linear SSTA is applied when assuming linear cell delay model.

Bench-	Chip	Exact			*SAAW*			*LDAW*			*SPC*			*IW*		
mark	size	μ	σ	95%	μ	σ	95%	μ	σ	95%	μ	σ	95%	μ	σ	95%
c1908	10	17.4	2.24	24.2	-1.2	-1.8	-1.9	-2.2	-5.5	-5.9	-1.7	-3.5	-3.3	-2.5	-6.8	-7.1
	6	17.5	2.23	24.3	-1.1	-1.5	-1.6	-2.2	-4.1	-4.2	-1.2	-2.6	-2.4	-2.2	-4.5	-4.3
	3	17.6	2.22	24.4	-0.9	-1.2	-1.1	-1.1	-1.8	-1.6	-1.0	-1.5	-1.5	-1.9	-2.1	-2.5
c3540	10	25.6	3.42	34.6	-1.0	-2.2	-1.6	-1.4	-6.2	-4.9	-1.8	-5.4	-4.8	-2.2	-7.5	-6.9
	6	25.8	3.45	34.3	-0.8	-1.2	-1.0	-1.2	-3.8	-3.3	-1.5	-3.4	-3.0	-1.3	-4.2	-3.7
	3	25.8	3.44	34.4	-0.5	-1.0	-1.0	-1.1	-2.1	-2.0	-1.0	-1.9	-1.9	-1.1	-2.2	-2.3
c7552	10	48.9	6.47	64.7	-1.2	-1.1	-1.4	-2.8	-3.5	-3.7	-2.5	-2.2	-2.7	-3.2	-5.6	-6.3
	6	48.9	6.47	64.7	-1.0	-1.1	-1.3	-1.4	-2.5	-2.3	-2.2	-2.1	-2.5	-2.9	-3.1	-3.3
	3	48.9	6.47	64.7	-0.6	-0.9	-1.0	-1.0	-1.1	-1.2	-0.9	-1.3	-1.4	-1.3	-1.4	-1.6

Table 6.3: percentage error for ISCAS85 benchmark stretching on a chip with different chip size. Note: We assume square chips and chip size means edge length in *mm*. The values of exact simulation are in *ns*.

the estimation of *LDAW* is within 1% error from the exact simulation and the error of *IW* is up to 8%. This is because *LDAW* predicts different mean and variance for different location correctly, as discussed in Section 6.5 while *IW* can only give the same mean and variance for all locations.

Table 6.5, 6.6, and 6.7 shows percentage error of *LDAW* and *IW* for ISCAS85 benchmarks placing on different locations on a 1cm$\times$1cm, 6mm$\times$6mm, and 3mm$\times$3mm chip, respectively. From the tables, we find that the error of *IW* becomes smaller when chip size is small.

bench	loca-	Exact			*LDAW*			*IW*		
mark	tion	μ	σ	95%	μ	σ	95%	μ	σ	95%
c3540	C	25.4	3.29	32.2	+0.4	+0.3	+0.3	+1.1	+2.4	+2.8
	LL	24.8	3.22	31.9	+0.8	+0.6	+0.3	+3.5	+1.4	+1.9
	LR	26.2	3.35	33.1	-0.7	+0.6	-0.6	-1.9	-2.4	-1.8
	UL	26.5	3.36	33.3	-0.2	+0.3	+0.3	-3.3	-3.0	-4.4
	UR	27.1	3.41	34.1	-0.3	-0.3	-0.6	-6.1	-4.1	-5.2
c7552	C	48.2	6.37	60.2	+0.8	+0.3	+0.2	+1.0	+0.9	+0.9
	LL	47.2	6.11	58.5	+0.4	+0.3	+0.7	+3.6	+4.6	+3.1
	LR	49.4	6.51	62.3	-0.2	+0.3	+0.3	-0.8	-1.9	-3.0
	UL	49.5	6.65	63.1	-0.2	+0.1	+0.1	-1.0	-4.0	-4.1
	UR	50.1	6.91	65.3	-0.4	+0.3	+0.1	-1.0	-7.4	-7.4

Table 6.4: Delay percentage error at different locations in a $2cm \times 2cm$ chip. Note: The values of exact simulation are in *ns*.

bench	loca-	Exact			*LDAW*			*IW*		
mark	tion	μ	σ	95%	μ	σ	95%	μ	σ	95%
c3540	C	25.4	3.26	31.9	+0.5	+0.4	+0.2	+1.3	+1.6	+1.9
	LL	25.0	3.25	31.6	+0.5	+0.3	+0.3	+2.4	+1.3	+2.4
	LR	25.8	3.30	32.4	-0.4	-0.4	-0.4	-1.1	-0.9	-0.8
	UL	26.1	3.32	32.6	-0.4	+0.3	-0.2	-2.0	-1.5	-1.4
	UR	26.2	3.34	33.0	-0.3	-0.3	-0.6	-2.6	-1.8	-2.2
c7552	C	48.6	6.38	60.4	+0.2	-0.1	-0.2	+1.0	+0.5	+0.6
	LL	48.2	6.35	60.3	-0.2	-0.2	+0.2	+1.7	+1.0	+1.1
	LR	48.9	6.42	60.4	+0.2	-0.2	+0.2	-0.2	-0.7	-0.5
	UL	49.1	6.43	61.0	-0.2	-0.2	-0.2	-0.9	+1.0	-1.3
	UR	49.2	6.45	61.2	-0.3	-0.4	-0.5	-1.1	-1.4	-1.8

Table 6.5: Delay comparison for ISCAS85 benchmark in $1cm \times 1cm$ chip. Note: The values of exact simulation are in *ns*.

bench	loca-	Exact			*LDAW*			*IW*		
mark	tion	μ	σ	95%	μ	σ	95%	μ	σ	95%
c3540	C	25.5	3.27	32.0	+0.4	+0.2	+0.3	+0.9	+0.7	+1.4
	LL	25.1	3.25	31.7	+0.5	+0.3	+0.3	+2.4	+1.3	+2.4
	LR	26.0	3.32	32.6	-0.4	-0.4	-0.4	-1.1	-0.9	-0.8
	UL	26.2	3.34	32.8	-0.4	+0.3	-0.2	-2.0	-1.5	-1.4
	UR	26.3	3.35	33.1	-0.3	-0.3	-0.6	-2.6	-1.8	-2.2
c7552	C	48.4	6.37	60.1	+0.2	-0.1	-0.2	+1.0	+0.5	+0.6
	LL	48.1	6.34	59.9	-0.2	-0.2	+0.2	+1.7	+1.0	+1.1
	LR	49.0	6.43	60.7	+0.2	-0.2	+0.2	-0.2	-0.7	-0.5
	UL	49.3	6.46	61.3	-0.2	-0.2	-0.2	-0.9	+1.0	-1.3
	UR	49.4	6.48	61.5	-0.3	-0.4	-0.5	-1.1	-1.4	-1.8

Table 6.6: Delay comparison for ISCAS85 benchmark in $6mm \times 6mm$ chip. Note: The values of exact simulation are in *ns*.

bench	loca-	Exact			*LDAW*			*IW*		
mark	tion	μ	σ	95%	μ	σ	95%	μ	σ	95%
c3540	C	25.6	3.28	32.2	+0.4	+0.2	+0.3	+0.5	+0.3	+0.8
	LL	25.4	3.26	320	+0.5	+0.6	+0.3	+1.2	+1.0	+1.2
	LR	25.8	3.31	32.4	-0.2	-0.4	-0.6	-0.5	-0.7	-0.7
	UL	25.9	3.22	32.5	-0.2	+0.3	-0.2	-1.0	-1.1	-0.9
	UR	26.0	3.31	32.7	-0.2	-0.2	-0.3	-0.7	-0.8	-1.1
c7552	C	48.6	6.38	60.3	+0.2	-0.3	+0.7	+0.2	+0.5	+1.1
	LL	48.4	6.37	60.1	-0.2	0.1	+0.2	+0.4	+0.3	+0.7
	LR	48.8	6.42	60.6	+0.2	-0.3	+0.3	-0.6	-0.9	-0.5
	UL	48.9	6.43	60.9	-0.2	+0.2	-0.2	-0.4	+0.0	-0.6
	UR	49.2	6.45	61.2	-0.2	-0.2	-0.3	-0.7	-0.8	-1.1

Table 6.7: Delay comparison for ISCAS85 benchmark in $3mm \times 3mm$ chip. Note: The values of exact simulation are in *ns*.

6.7 Application of Spatial Variation Models on Statistical Leakage Analysis

Besides SSTA, we also apply our variation model to statistical leakage power analysis. Usually, cell leakage power variation is modeled as exponential function of variation sources:

$$P_{leak} = P_0 \cdot e^{\sum c_i V_i} \tag{6.21}$$

where P_0 is the nominal leakage power and c_i's are sensitivity coefficients. The full chip leakage power is calculated as the sum of leakage power of all cells:

$$P_{chip} = \sum_{i \in Cell} P_{i,leak} \tag{6.22}$$

where $Cell$ is the set of all cells in the chip and $P_{i,leak}$ is leakage power of the i^{th} cell. Since each variation source is a quadratic function as in (6.19), the cell leakage power is an exponential of a quadratic function of random variables. Considering that the random variables may be non-Gaussian, there is no closed-form equation to calculate the full chip leakage power. Therefore, in this chapter, we apply Monte-Carlo simulation to obtain the full chip leakage power variation.

We have implemented leakage variation analysis with different models in Matlab. In the experiment, we use the same setting and comparison cases as the SSTA experiment in Section 6.6. For each variation model, we use 100,000 sample Monte-Carlo simulation to obtain the full chip leakage power for all variation models. For the leakage analysis, we assume that 900 copies of ISCAS benchmark circuits are placed in a 30×30 array on a 2cm$\times$2cm chip. Table 6.8 compares the leakage variation for ISCAS85 benchmarks. From the table, we observe that:

- Error of *SAAW* is within 5% while error of *SPC* is up to 17%. Moreover, *SAAW* is 7X faster than *SPC* because there are fewer random variables for *SAAW*.

Bench-	Exact			*SAAW*				*QAW*				*LDAW*				*SPC*				*IW*			
mark	μ	σ	95%	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T	μ	σ	95%	T
c1355	62.1	14.5	92.5	+1.5	+3.3	+3.2	16.3	+3.4	+6.5	+5.4	9.8	-5.5	-15.6	-16.9	2.5	+5.3	+10.4	+12.2	123	-7.9	-19.6	-20.9	2.7
c1908	95.6	20.3	144	+0.9	+4.4	+3.7	15.5	+2.3	+7.8	+6.3	9.7	-6.5	-17.8	-19.6	2.6	+5.7	+14.8	+14.3	122	-8.6	-19.2	-23.5	2.9
c2670	131	22.9	181	+1.4	+2.7	+1.7	15.7	+2.9	+8.5	+4.7	9.5	-7.9	-16.9	-22.0	2.8	+6.8	+12.2	+9.4	122	-9.2	-20.3	-25.5	2.4
c3540	201	37.4	282	+1.5	+2.3	+1.8	15.4	+3.1	+5.8	+4.4	10.4	-5.6	-16.5	-20.2	2.6	+4.9	+11.2	+8.2	123	-8.3	-18.2	-23.5	2.7
c7552	403	73.2	562	+1.6	+2.7	+1.9	15.3	+3.7	+6.0	+5.0	10.1	-7.3	-12.6	-16.9	2.6	+7.1	+13.8	+10.7	122	-9.2	-20.3	-24.5	2.5

Table 6.8: Leakage error percentage for different models in $2cm \times 2cm$ chip. Note: The exact values are in mW. Run time (T) is in s.

- *SAAW* is more accurate than *QAW*, but is about 50% slower.
- Both *LDAW* and *IW* are not accurate. This is because both these models do not consider correlation and hence under estimate the leakage power variation.

Similar to SSTA, for leakage variation analysis, we also perform leakage estimation in different size chips: 1cm×1cm, 6mm×6mm, and 3mm×3mm. For the 1cm×1cm chip, we assume that 225 copies of ISCAS85 benchmark circuits are placed in a 15×15 array, for the 6mm×6mm chip, we assume 100 copies of ISCAS85 benchmark circuits are placed in a 10×10 array, and for the 3mm×3mm chip, we assume 25 copies of ISCAS benchmark circuits are placed in a 5×5 array. Table 6.9 shows the error percentage for different model on different size chips. From the table, we find that the error of *LDAW*, *SPC*, and *IW* reduces when chip size becomes smaller as expected.

Bench-	Chip	Exact			*SAAW*			*LDAW*			*SPC*			*IW*		
mark	size	μ	σ	95%	μ	σ	95%	μ	σ	95%	μ	σ	95%	μ	σ	95%
c1355	10	15.4	3.52	23.2	+1.2	+3.1	+3.0	-4.7	-10.6	-11.2	+4.8	+7.5	+9.2	-5.4	-12.3	-13.8
	6	5.92	1.62	10.3	+1.0	+2.7	+2.9	-3.5	-7.5	-8.3	+2.7	+6.2	+6.7	-3.9	-8.1	-9.2
	3	1.48	0.40	2.58	+0.6	+1.8	+2.0	-1.8	-3.5	-3.7	+1.6	+2.9	+3.1	-2.0	-3.5	-4.0
c1908	10	23.9	5.7	36.1	+1.0	+2.7	+2.9	-4.2	-12.3	-14.1	+3.7	+8.5	+9.2	-5.5	-13.9	-15.1
	6	10.6	2.25	16.1	+0.9	+2.0	+2.1	-3.4	-7.3	-8.2	+1.9	+5.6	+6.8	-3.5	-7.3	-9.0
	3	2.65	0.57	4.03	+1.0	+2.2	+2.2	-1.3	-3.5	-4.0	+1.2	+2.9	+3.1	-1.9	-4.0	-4.4
c2670	10	32.8	5.72	45.2	+1.2	+2.8	+2.9	-5.3	-11.1	-14.0	+4.2	+8.2	+9.1	-6.5	-12.3	-17.1
	6	14.6	2.55	20.1	+1.0	+1.8	+1.9	-3.5	-6.2	-7.1	+2.4	+4.3	+6.0	-4.3	-7.1	-8.3
	3	3.65	0.65	5.03	+0.8	+1.4	+1.7	-1.9	-3.2	-3.7	+2.0	+3.6	+3.5	-2.2	-4.5	-5.1
c3540	10	50.5	9.37	71.1	+1.2	+1.8	+2.1	-3.9	-7.8	-10.5	+2.9	+5.3	+6.2	-5.0	-9.6	-14.5
	6	22.3	4.16	30.2	+1.0	+1.4	+1.7	-2.3	-4.5	-5.5	+1.8	+3.5	+3.6	-3.4	-6.1	-8.5
	3	5.58	1.05	7.55	+0.9	+1.2	+1.4	-1.2	-3.1	-3.0	+1.0	+1.4	+2.0	-1.4	-3.4	-3.9
c7552	10	102	18.5	141	+1.4	+2.3	+2.4	-4.6	-7.3	-9.9	+4.0	+6.2	+8.6	-6.3	-8.2	-12.5
	6	45.0	8.19	6.26	+1.2	+1.9	+1.9	-3.0	-5.2	-7.3	+2.7	+4.2	+5.1	-3.5	-6.0	-8.2
	3	11.4	2.06	1.58	+0.9	+0.7	+1.3	-1.2	-1.8	-2.0	+1.5	+1.6	+1.6	-2.3	-2.9	-3.4

Table 6.9: Leakage error for different variation model on different size chips. Note: exact values are in mW.

6.8 Summary of Different Models

In previous sections, we compared the accuracy and efficiency of different models. Table 6.10 summarizes the advantages and disadvantages of our proposed spatial variation models (*SAAW*, *QAW*, and *LDAW*), and the traditional variation models (*SPC* and *IW*). Our proposed across-wafer variation models exactly model the across-wafer variation and the number of random variables does not depend on chip size. Therefore they are accurate and efficient. *SAAW* has six random variables and it can be applied to any across-wafer variation models. *QAW* has four random variables, hence it is more efficient than *SAAW*. However, it can be applied only when the across-wafer variation is a perfect parabola. *LDAW* is the most efficient, ignores correlation and only works for small chips. Moreover, *SAAW* and *QAW* need to know the across-wafer variation, therefore, one needs to track the die locations within the wafer to build up the model.

On the other hand, the traditional variation models (as well as *LDAW*) only require measurement on a die without tracking die locations. Therefore, they are somewhat easier to build. However, such models are not accurate compared to our proposed models. Moreover, for *SPC*, since the number of random variables depends on number of grids, it is not as efficient as our proposed models.

Model Type	Advantages	Disadvantages	Models	# of RVs	Case to Apply
Across-wafer models	Accurate Efficient	Need die tracking to extract	*SAAW* Equ(6.19)	6	large chip, non-parabola across-wafer variation
			QAW Equ (6.2)	4	large chip, parabola across-wafer variation
			LDAW Equ (6.20)	2	small chip
Traditional models	Easy to extract	Not accurate	*SPC*	Depend on # of grids	large chip
			IW	2	small chip

Table 6.10: Summary of different variation models.

6.9 Conclusions

In this chapter, we analytically study the impact of systematic across-wafer variation on within-die spatial variation. For simplicity, we assume that across-wafer variation is a quadratic function. We first observe that different locations of a chip may have different means and variances and such difference becomes more significant when chip size increases. Secondly, we find that spatial correlation is visible only when the across wafer systematic is not taken into account. When it is taken into account, we show that within-die random variability does not exhibit a strong or useful pattern of spatial correlation. We exploited these observations in order to create a much more accurate and efficient model for performance variability prediction. Thirdly, we find that the within-die spatial variation is *NOT* independent of the inter-die variation. However, when chip size is small enough, such dependence is weak and the across-wafer variation can be lumped in to inter-die variation. In this case, the two level inter-/within-die variation model is still accurate. Later, we also analyze the case when the across-wafer is not a perfect quadratic function. Based on the above analysis, we have proposed an accurate and efficient variation model for deterministic across wafer variation. We further apply our new variation model to two applications: statistical static timing analysis and statistical leakage analysis. Experimental result shows that compared to the distance-based spatial variation model, our new model reduces the error from 6.5% to 2% for statistical timing analysis and reduces error from 17% to 5% for statistical leakage analysis. Our model also improves the run time by 6X for statistical timing analysis and by 7X for statistical leakage analysis.

CHAPTER 7

On Confidence in Characterization and Application of Variation Models

In this chapter we study statistics of statistics. Statistical modeling and analysis have become the mainstay of modern design-manufacturing flows. Most analysis techniques assume that the statistical variation models are reliable. However, due to limited number of samples (especially in the case of lot-to-lot variation), calibrated models have low degree of confidence. The problem is further exacerbated when production volumes are low ($\leq$ 65 lots) causing additional loss of confidence in statistical analysis (since production only sees a small snapshot of the entire distribution). The problem of confidence in statistical analysis is going to be further worsened with advent of 450mm wafers. We mathematically derive confidence intervals for commonly used statistical measures (mean, variance, percentile corner) and analysis (SPICE corner extraction, statistical timing). Our estimates are within 2% of simulated confidence values. Our experiments (with variability assumptions derived from test silicon data from a 45nm industrial process) indicate that for moderate characterization volumes (10 lots) and low-to-medium production volumes (15 lots), a significant guardband (e.g., 34.7% of standard deviation for single parameter corner, 38.7% of standard deviation for SPICE corner, and 52% of standard deviation for 95-percentile point of circuit delay are needed) to ensure 95% confidence in the results. The guardbands are non-negligible for all cases when either production or characterization volume is not

large. We also study the interesting one production lot case which may be common for prototyping as well as for academic designs. The proposed methods are not runtime-intensive (always within 10s) as they require Monte-Carlo simulations on closed form expressions.

7.1 Introduction

In process variation modeling, analysis or optimization, statistical characteristics of variation sources, such as mean, variance, and skewness, are often assumed to be known and reliable.

In practice, the statistical characteristics of the variation sources are obtained from measurement of a set of samples. Due to limited number of measurements, the measured statistical characteristics may be unreliable. This is especially true for lot-to-lot variation, since the number of lots measured for characterization is usually small.

Moreover, the existing statistical analysis, implicitly assume that the production chips have the same statistical characteristics as the corresponding population values[1]. When the number of production samples is not small, the production statistical characteristics may significantly deviate from their population values. Therefore, the uncertainty in statistics of measured data as well as production data should be considered in statistical analysis. [177] models the uncertainty of mean, variance, and correlation coefficients as an interval, and then estimates the range of mean and variance of circuit performance. However, this work only models the uncertainty as an interval, ignoring their true distributions. It also ignores the uncertainty introduced by limited number of production lots.

Before we move further, it is important to briefly review the concept of confidence

[1]Here by "population", we mean the statistical values in the case of infinite measured as well as production samples.

intervals in statistical analysis. Consider a standard normal random variable X, we choose n samples of it $(X_1, X_2, \ldots, X_n)$. Once the samples are chosen, the sample mean $\hat{\mu}$ and variance $\hat{\sigma}^2$ are fixed. However, if we repeatedly choose n samples for 1,000 times, and calculate the sample mean and variance for each time, the results may differ. When the sample mean is smaller than a certain value μ_{conf} in 900 out of the 1000 runs, we say that we have 90% confidence that the sample mean is smaller than μ_{conf}. Notice that the confidence is not yield but the probability that certain statistical characteristic is within a given interval. Once chips are produced, the distribution of (say) circuit delay is fixed. But due to the uncertainty of the statistical model, we do not know exactly the production mean and variance prior. For a given confidence, one can estimate the distribution of mean and variance (of course the actual mean and variance are going to be just one sample out of these distributions).

In this chapter, we study the uncertainty of statistical models and analyze the impact of such uncertainty on statistical analysis. The contributions of this chapter are:

1. Given the number of measured lots and the number of production lots,we estimate the distribution of the production mean and variance of variation sources.

2. For a given confidence level, we estimate the worst case fast/slow corner ($\mu \pm k\sigma$ corner) for variation sources.

3. We extend the ideas to extract SPICE corners depending on desired confidence level as well as number of measured/produced silicon lots.

4. We estimate the confidence interval of mean, variance and quantile for statistical timing analysis.

Experimental results show that the required guardband (to reach desired confidence) estimated by our method is very close to the exact simulation value. We also observe

that the guardband value increases dramatically when confidence level increases. We need to introduce up to 30% guardband value to achieve 95% confidence.

The rest of this chapter is organized as follows: Section 7.2 gives the problem formulation; then Section 7.3 studies the confidence interval estimation for variation sources; Section 7.4 presents the estimation of worst case delay for SPICE corner estimation and Section 7.5 analyzes the impact of mean and variance uncertainty on statistical timing analysis; Section 7.6 discusses how to choose confidence level in real circuit design; and finally Section 7.7 concludes this chapter.

7.2 Problem Formulation

Process variation is decomposed into inter-lot (V_l), inter-wafer(V_w), inter-die (V_d), and within-die (V_r) variation:

$$V = V_l + V_w + V_d + V_r \tag{7.1}$$

$$\mu_V = \mu_l + \mu_w + \mu_d + \mu_r \tag{7.2}$$

$$\sigma_V^2 = \sigma_l^2 + \sigma_w^2 + \sigma_d^2 + \sigma_r^2 \tag{7.3}$$

where μ_l (σ_l^2), μ_w (σ_w^2), μ_d (σ_d^2), and μ_r (σ_r^2) are means (variances) of inter-lot, inter-wafer, inter-die, and within-die variations, respectively. It has been shown that in case of finite number of samples the sample mean follows Gaussian distribution and the sample variance follow χ^2 distribution [4]. The variances of the means and variances are:

$$\sigma_{\mu l}^2 = \sigma_l^2 / N_l \tag{7.4}$$

$$\sigma_{\sigma_l^2}^2 = \sigma_l^2 / N_l \tag{7.5}$$

$$\sigma_{\mu w}^2 = \sigma_w^2 / N_w \tag{7.6}$$

$$\sigma_{\sigma_w^2}^2 = \sigma_w^2 / N_w \tag{7.7}$$

$$\sigma^2_{\mu d} = \sigma^2_d / N_d \tag{7.8}$$

$$\sigma^2_{\sigma^2_d} = \sigma^2_d / N_d \tag{7.9}$$

$$\sigma^2_{\mu r} = \sigma^2_r / N_r \tag{7.10}$$

$$\sigma^2_{\sigma^2_r} = \sigma^2_r / N_r \tag{7.11}$$

where N_l, N_w, N_d, and N_r are total number of lots, wafers, dies, and circuit elements. There are usually 20 to 50 wafers per lot, more than one hundred dies per wafer, and tens of measured circuit elements within each die[2]. Therefore, $N_r \gg N_d \gg N_w \gg N_l$. Hence $\sigma^2_{\mu l} \gg \sigma^2_{\mu w} \gg \sigma^2_{\mu d} \gg \sigma^2_{\mu r}$, $\sigma^2_{\sigma^2_l} \gg \sigma^2_{\sigma^2_w} \gg \sigma^2_{\sigma^2_d} \gg \sigma^2_{\sigma^2_r}$. Therefore, the uncertainty of mean and variance comes largely from lot-to-lot variation. Hence in this chapter, we focus our attention on lot-to-lot variation.

Table 7.1 illustrates five cases of number of measured lots and number of production lots.When the number of measured lots or production lots is small, the confidence interval is large and hence statistical analysis is not reliable. In this case, we may want to guardband statistical analysis to ensure a certain degree of confidence. Example use models for these different scenarios can be: commodity process/high volume part (*L-L*); niche process/high column part (*S-L*); commodity process/niche part (*L-S*); niche process/niche part (*S-S*); and commodity (or niche) process/prototyping (*L-1*).

In practice, only the measured as opposed to the population value is known. Therefore, our the problem is:

Given the number of measured lots and production lots and the measured values of mean and variance of variation sources, estimate the distribution of statistical measures of production lots.

In rest of the chapter, we assume that the lot-to-lot variation of all the variation sources follows Gaussian distribution.

[2]Note that 85 lots of 25 300mm wafers each with die-size of 100mm^2 amounts to production volume of 1.5 million chips.

Case	# Measured lots	# Production lots	Confidence interval	Reliability of Statistical analysis
L-L	Large	Large	Small	High
S-L	Small	Large	Large	Low
L-S	Large	Small	Large	Low
S-S	Small	Small	Large	Low
L-1	Any	1	Large	Low

Table 7.1: Reliability of statistical analysis for different number of measured and production lots.

7.3 Confidence Interval for Variation Sources

In this section, we will discuss confidence interval estimation for a single variation source. Table 7.3 summarizes the notation used in the rest of the chapter.

7.3.1 Mean and Variance

Let's first discuss the mean and variance. From (7.1), we calculate the total mean and variance of measured value as:

$$\hat{\mu}_t = \hat{\mu}_l + \mu_o \tag{7.12}$$

$$\hat{\sigma}_t^2 = \hat{\sigma}_l^2 + \sigma_o^2 \tag{7.13}$$

As discussed in Section 7.2, the uncertainty of mean and variance comes largely from lot-to-lot variation. Therefore, we assume the mean μ_o and variance σ_o^2 of all other variation are reliable. In the same way as above, we calculate the total mean and variance of the production as:

$$\tilde{\mu}_t = \tilde{\mu}_l + \mu_o \tag{7.14}$$

$$\tilde{\sigma}_t^2 = \tilde{\sigma}_l^2 + \sigma_o^2 \tag{7.15}$$

In the rest of this subsection, we will focus on analyzing confidence interval for lot-to-lot variation.

n	number of lots
μ, σ^2	mean and variance
$cn_{f/s}$	fast/slow corner
no hat	population value (μ, σ^2 and so on)
$\hat{\_}$	value obtain from measured lots ($\hat{n}$, $\hat{\mu}$, and so on)
$\tilde{\_}$	value of production lots ($\tilde{n}$, $\tilde{\mu}$, and so on)
$\square_t$	total value including inter-lot, inter-wafer, inter-die and within-die variation (μ_t, σ_t^2, and so on)
$\square_l$	inter-lot variation value (μ_l, σ_l^2, and so on)
$\square_w$	inter-wafer variation value (μ_w, σ_w^2, and so on)
$\square_d$	inter-die variation value (μ_d, σ_d^2, and so on)
$\square_r$	within-die variation value (μ_r, σ_r^2, and so on)
$\square_o$	total value except inter-lot variation (μ_o, σ_o^2, and so on) $\mu_o = \mu_w + \mu_d + \mu_r$, $\sigma_o^2 = \sigma_w^2 + \sigma_d^2 + \sigma_r^2$

Table 7.2: Notations.

In order to estimate the confidence interval, we first estimate the population values of mean and variance from measurement data [4]:

$$\mu_l = \hat{\mu}_l + \hat{\sigma}_l \cdot M_1 \cdot \sqrt{\frac{\hat{n}-1}{\hat{n}Q_1}} \tag{7.16}$$

$$\sigma_l^2 = (\hat{n}-1)\hat{\sigma}_l^2/Q_1 \tag{7.17}$$

where $M_1 \sim N(0,1)$ is a random variable with standard normal distribution, and $Q_1 \sim \chi^2_{\hat{n}-1}$ is a random variable with χ^2 distribution with $\hat{n}-1$ degrees of freedom [4]. Then, we estimate the production mean and variance with respect to population mean and variance:

$$\tilde{\mu}_l = \mu_l + \sigma_l \cdot M_2/\sqrt{\tilde{n}} \tag{7.18}$$

$$\tilde{\sigma}_l^2 = \sigma_l^2 \cdot Q_2/(\tilde{n}-1) \tag{7.19}$$

where $M_2 \sim N(0,1)$ is a random variable with standard normal distribution and $Q_2 \sim \chi^2_{\tilde{n}-1}$ is a random variable with χ^2 distribution with $\tilde{n}-1$ degrees of freedom. Finally,

we obtain the production mean and variance as:

$$\tilde{\mu}_l = \hat{\mu}_l + \hat{\sigma}_l \cdot \sqrt{\frac{\hat{n}-1}{Q_1}}\left(\frac{M_1}{\sqrt{\hat{n}}} + \frac{M_2}{\sqrt{\tilde{n}}}\right) = \hat{\mu}_l + \hat{\sigma}_l \eta T \tag{7.20}$$

$$\tilde{\sigma}_l^2 = \hat{\sigma}_l^2 \cdot \frac{Q_2(\hat{n}-1)}{Q_1(\tilde{n}-1)} = \hat{\sigma}_l^2 \cdot \zeta^2 \cdot U \tag{7.21}$$

where

$$\eta = \sqrt{\frac{\hat{n}+\tilde{n}}{\hat{n}\tilde{n}}} \tag{7.22}$$

$$\zeta = \sqrt{\frac{\hat{n}-1}{\tilde{n}-1}} \tag{7.23}$$

$$T = \frac{\sqrt{\hat{n}-1}(\sqrt{\tilde{n}}M_1 + \sqrt{\hat{n}}M_2)}{\sqrt{(\hat{n}+\tilde{n})Q_1}} \tag{7.24}$$

$$U = \frac{Q_2}{Q_1} \tag{7.25}$$

It is easy to find that: $\frac{\sqrt{\tilde{n}}M_1+\sqrt{\hat{n}}M_2}{\sqrt{\hat{n}+\tilde{n}}} \sim N(0,1)$, hence, T is the ratio between a standard normal random variable and square root of a χ^2 random variable. Therefore, T is a random variable with *t-distribution* with $\hat{n}$ degrees of freedom [4]. Moreover, U is the ratio of two χ^2 random variables. The PDF of U is [4]:

$$PDF_U(u) = \frac{\Gamma(\frac{\hat{n}+\tilde{n}}{2}-1)u^{\frac{\hat{n}}{2}-1}}{\Gamma(\frac{\hat{n}}{2})\Gamma(\frac{\tilde{n}}{2})(u+1)^{\frac{\hat{n}+\tilde{n}}{2}}} \tag{7.26}$$

where $\Gamma(\cdot)$ is Gamma function [4] which is calculated as:

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt \tag{7.27}$$

7.3.2 Simplifying the Large Lot Case

In the previous subsection, we analyzed the case when both $\hat{n}$ and $\tilde{n}$ are small (*S-S* case). We may simplify our model when either $\hat{n}$ or $\tilde{n}$ is large.

When the number of measured lots $\hat{n}$ is large (*L-S* case), we can approximate the

population mean and variance as measured mean and variance:

$$\mu_l \approx \hat{\mu}_l \tag{7.28}$$

$$\sigma_l^2 \approx \hat{\sigma}_l^2 \tag{7.29}$$

In this case, (7.20) and (7.21) can be simplified to:

$$\tilde{\mu}_l = \hat{\mu}_l + \hat{\sigma}_l \cdot M_2/\sqrt{\tilde{n}} \tag{7.30}$$

$$\tilde{\sigma}_l^2 = \hat{\sigma}_l^2 \cdot Q_2/(\tilde{n}-1) \tag{7.31}$$

In the other case when the number of production lots $\tilde{n}$ is large (*L-S* case), the production values of mean and variance are very close to the population value, i.e.:

$$\tilde{\mu}_l \approx \mu_l \tag{7.32}$$

$$\tilde{\sigma}_l^2 \approx \sigma_l^2 \tag{7.33}$$

Therefore, (7.20) and (7.21) can be simplified to:

$$\tilde{\mu}_l = \hat{\mu}_l + \hat{\sigma}_l M_1 \cdot \sqrt{\frac{\hat{n}-1}{\hat{n}Q_1}} \tag{7.34}$$

$$\tilde{\sigma}_l^2 = (\hat{n}-1)\hat{\sigma}_l^2/Q_1 \tag{7.35}$$

7.3.3 One Production Lot Case

Besides *S-S*, *L-S*, and *S-L* cases, we also consider the special case of one production lot (*L-1* case), that is $\tilde{n} = 1$. In this case, production mean still follows the same distribution as shown in (7.20) and (7.21). However, since there is only one lot, lot-to-lot variation has no impact on variance. Uncertainty of variance estimation is mainly caused by wafer-to-wafer variation. Hence, in this case, the production variance is calculated as:

$$\tilde{\sigma}_t^2 = \hat{\sigma}_w^2 \cdot Q_2'/(\tilde{n}'-1) + \sigma_d^2 + \sigma_r^2 \tag{7.36}$$

Targeted conf.%	50	60	70	80	90
Measured conf.%	51.72	60.34	72.41	81.03	93.10

Table 7.3: Experimental validation of estimation of confidence interval for mean for $\hat{n} = 5$ and $\tilde{n} = 1$.

where $\tilde{n}'$ is number of production wafers, Q'_2 is a χ^2 random variable with $\tilde{n}' - 1$ freedom, and $\hat{\sigma}^2_w$ is the measured variance of wafer-to-wafer variation.

7.3.4 Experimental Validation for One Lot Case

The problems in validation of the confidence-based guardband models proposed in this work should be obvious by now. We need exceedingly large amounts of data spread out over hundreds of lots preferably over multiple independent designs. Fortunately, the mathematical derivations in this work make very few approximations if any. Nevertheless, in this section we try to validate the theory above for the case with small number of measured lots and one production lot. To get over the hurdle of huge data requirement, we use wafer-to-wafer variation instead of lot-to-lot variation.

We obtain wafer-to-wafer ring oscillator delay variation data from an industrial 65nm process with 348 wafers spread over 23 lots. We randomly divide these wafers into 58 sets with 6 wafers per set. We model each set as a design and assume that 5 of the wafers are used to generate the statistical model ($\hat{n} = 5$) and the other wafer is the production wafer ($\tilde{n} = 1$). For each set we apply (7.20) to estimate the distribution of production mean and calculate the confidence interval for a targeted confidence level. We obtain the measured confidence by counting the number of sets that production value is within the confidence interval: Measured Conf = Number of match sets/ Total number of sets.

Table 7.3 compares the targeted confidence level and the measured confidence level. The small error ($\leq 4\%$) is due to limited data.

7.3.5 Fast/Slow Corner Estimation

The next statistical characteristics we consider are fast and slow corners $\widetilde{cn}_{f/s}$ for variation sources.

Usually, the fast/slow corner is expressed as:

$$\widetilde{cn}_{f/s} = \tilde{\mu}_t \pm k_{f/s}\tilde{\sigma}_t \tag{7.37}$$

From (7.20) and (7.21), we have:

$$\widetilde{cn}_f = \hat{\mu}_l + \mu_o + \hat{\sigma}_l \eta T - k_f \sqrt{\hat{\sigma}^2 \zeta^2 U + \sigma_o^2} \tag{7.38}$$

$$\widetilde{cn}_s = \hat{\mu}_l + \mu_o + \hat{\sigma}_l \eta T + k_s \sqrt{\hat{\sigma}^2 \zeta^2 U + \sigma_o^2} \tag{7.39}$$

For a given confidence level, we want to estimate the worst case fast/slow corner, $cn^w_{f/s}$. However, fast corner and slow corners are dependent on each other. Therefore, we need to handle them together. Assuming that hold time violation is harder to fix, higher confidence may be needed on fast corner. We estimate the worst case fast/slow corner such that there is cf_f confidence that the fast corner is larger than cn^w_f and there is cf_t confidence that the fast corner is larger than cn^w_f and the slow corner is smaller than cn^w_s, that is: [3]

$$P\{\widetilde{cn}_f > cn^w_f\} = cf_f \tag{7.40}$$

$$P\{\widetilde{cn}_f > cn^w_f, \widetilde{cn}_s < cn^w_s\} = cf_t \tag{7.41}$$

Due to the complexity of (7.38), we use Monte-Carlo simulations to obtain the distributions of the fast/slow corners. Then the worst case fast corner is calculated as:

$$cn^w_f = CDF^{-1}_{\widetilde{cn}_f}(cf_f) \tag{7.42}$$

With the worst case fast corner value, from the samples, we may also obtain the conditional CDF of $\widetilde{cn}_s$ given $\widetilde{cn}_f > cn^w_f$, $CDF_{\widetilde{cn}_s|\widetilde{cn}_f<cn^w_f}$. Then the worst case slow corner

[3] Without loss of generality, we assume fast implies smaller parameter value (e.g. channel width).

is calculated as:

$$cn_s^w = CDF^{-1}_{\widetilde{cn}_s|\widetilde{cn}_f<cn_f^w}(cf_t/cf_f) \tag{7.43}$$

We then express the worst case corner value as:

$$cn_{f/s}^w = \hat{\mu} \pm k'_{f/s}\hat{\sigma} \tag{7.44}$$

When the worst case corner values are known, we can easily obtain the value of $k'_{f/s}$.

Table 7.4 shows the value of $k'_{f/s}$ under different confidence levels. In the table, the guardband percentage is calculated as: $100 \times (k'_{f/s} - k_{f/s})/k_{f/s}$. In the experiment, we let $\hat{n} = 10$, $\tilde{n} = 15$, $k_f = k_s = 3$ and we assume that the variance of inter-lot variation is 18% of the total variance (derived from the data described in section 7.3.4). We also assume that the variation source is with standard normal distribution [4]. In the experiment, we perform the estimation for 1000 runs. For each run, we generate $\hat{n}$ measured samples to obtain the measured mean $\hat{\mu}$ and variance $\hat{\sigma}^2$, and then apply the method above to calculate the worst case value of $k'_{f/s}$. Notice that the $k'_{f/s}$ depends on the measured values $\hat{\mu}$ and $\hat{\sigma}^2$, hence each run may result in a different $k'_{f/s}$ value. In the table, the $k'_{f/s}$ values are calculated as the average of 1000 runs. From the table, we can find that to ensure high confidence, $k'_{f/s}$ needs to be 30% over $k_{f/s}$, i.e., we need a large guardband.

As discussed in Section 7.3.2, when the number of measured lots is large (*L-S*), the mean and variance can be simplified as (7.30). Hence the corner model can be also simplified as:

$$\widetilde{cn}_f = \hat{\mu}_l + \hat{\sigma}_l \frac{M_2}{\sqrt{\tilde{n}}} - k_f \sqrt{\frac{\hat{\sigma}_l^2 \cdot Q_2}{(\tilde{n}-1)} + \sigma_o^2} \tag{7.45}$$

$$\widetilde{cn}_s = \hat{\mu}_l + \hat{\sigma}_l \frac{M_2}{\sqrt{\tilde{n}}} + k_s \sqrt{\frac{\hat{\sigma}_l^2 \cdot Q_2}{(\tilde{n}-1)} + \sigma_o^2} \tag{7.46}$$

[4] The nominal mean and variance of variation sources does not affect the value of k'_f/s.

cf_t %	cf_f %	k'_f	k'_s
50	60	3.080 (2.67%)	3.246 (8.20%)
60	70	3.155 (5.13%)	3.275 (9.13%)
70	80	3.255 (8.50%)	3.307 (10.2%)
80	90	3.422 (14.7%)	3.345 (11.5%)
90	95	3.594 (19.8%)	3.518 (17.2%)
95	99	4.042 (34.7%)	3.619 (20.6%)

Table 7.4: Guardband of fast/slow corner for different degrees of confidence assuming $\hat{n} = 10$, $\tilde{n} = 15$.

cf_t %	cf_f %	k'_f	k'_s
50	60	3.035 (1.17%)	3.103 (3.43%)
60	70	3.112 (3.73%)	3.121 (4.03%)
70	80	3.124 (4.14%)	3.138 (4.60%)
80	90	3.236 (7.87%)	3.215 (7.17%)
90	95	3.401 (13.4%)	3.342 (11.4%)
95	99	3.625 (20.8%)	3.502 (16.7%)

Table 7.5: Guardband of fast/slow corner for different degrees of confidence assuming large $\hat{n}$ and $\tilde{n} = 15$.

Table 7.5 shows the average $k'_{f/s}$ value for the *L*-*S* case. As expected, the guardband in this case is smaller than the *S*-*S* case.

Similarly, when the number of production lots is large (*S*-*L*), corner model can be simplified to:

$$\widetilde{cn}_f = \hat{\mu}_l + \hat{\sigma}_l \cdot M_1 \cdot \sqrt{\frac{\hat{n}-1}{\hat{n}Q1}} - k_f \sqrt{\frac{(\hat{n}-1)\hat{\sigma}_l^2}{Q_1} + \sigma_o^2} \tag{7.47}$$

$$\widetilde{cn}_s = \hat{\mu}_l + \hat{\sigma}_l \cdot M_1 \cdot \sqrt{\frac{\hat{n}-1}{\hat{n}Q1}} + k_s \sqrt{\frac{(\hat{n}-1)\hat{\sigma}_l^2}{Q_1} + \sigma_o^2} \tag{7.48}$$

Table 7.6 shows the average $k'_{f/s}$ value for the *S*-*L* case. From the table, it is evident that the guardband is smaller than that of the *S*-*S* case.

Figure 7.1 compares the 90% confidence value of k'_f between the *L*-*S* (or *S*-*L*) and *L*-*L*. We observe that when $\tilde{n} \geq 60$ (or $\hat{n} \geq 85$), *L*-*S* (or *S*-*L*) can be treated as *L*-*L*.

cf_t %	cf_f %	k'_f	k'_s
50	60	3.052 (1.73%)	3.132 (4.40%)
60	70	3.134 (4.47%)	3.155 (5.17%)
70	80	3.165 (5.50%)	3.193 (6.43%)
80	90	3.272 (9.07%)	3.267 (8.90%)
90	95	3.431 (14.4%)	3.370 (12.3%)
95	99	3.690 (23.0%)	3.523 (17.4%)

Table 7.6: Guardband of fast/slow corner for different confidence levels assuming $\hat{n} = 10$ and large $\tilde{n}$.

That is, the statistical analysis is reliable and we do not need to perform guardband estimation. In the experiment, when error of k'_f value between *L-L* and *L-S* (or *S-L*) cases is within 3%, we consider $\hat{n}$ (or $\tilde{n}$) value is large.

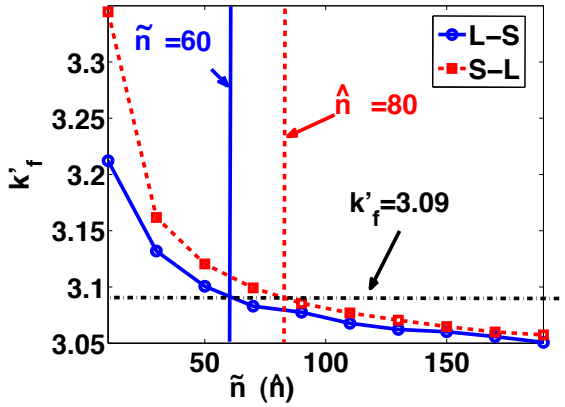


Figure 7.1: Comparison of *S-L*, *L-S*, and *L-L* case.

Note that increasing lot-to-lot variation will increase the value of $\hat{n}$ and $\tilde{n}$ that can be considered large as shown in Figure 7.2. This is because when the lot-to-lot variation increase, the uncertainty of mean and variance increases. Therefore, we need a larger number of measured lots (or production lots) to be considered as large.

As discussed in Section 7.3.2, in the special case when there is only one production lot (*L-1*), lot-to-lot variation has no impact on the variance. Instead, the uncertainty of

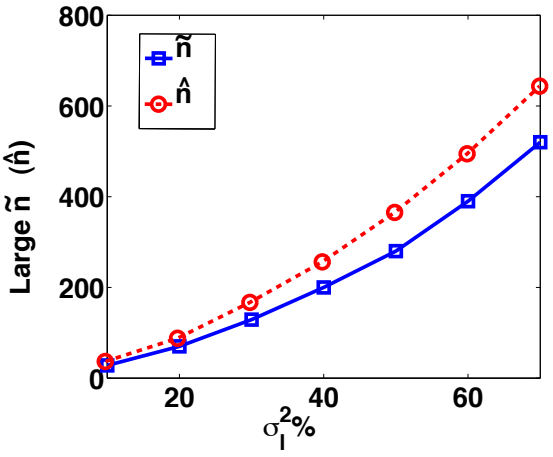


Figure 7.2: Number of $\hat{n}$ or $\tilde{n}$ which can be considered as large for different fraction of lot-to-lot variation assuming maximum 3% error at 90% confidence.

the variance comes from wafer-to-wafer variation. From (7.36), we can calculate the fast/slow corners as:

$$\tilde{cn}_f = \hat{\mu}_l + \mu_o + \hat{\sigma}_l \eta T - k_f \sqrt{\hat{\sigma}_w^2 Q_2'/(\tilde{n}'-1) + \sigma_o^2} \tag{7.49}$$

$$\tilde{cn}_s = \hat{\mu}_l + \mu_o + \hat{\sigma}_l \eta T + k_s \sqrt{\hat{\sigma}_w^2 Q_2'/(\tilde{n}'-1) + \sigma_o^2} \tag{7.50}$$

Table 7.7 shows the $k'_{f/s}$ for the *L-1* case. In the experiment, we assume that there are 25 wafers per lot, $\tilde{n}' = 25$. From the table, we see that the guardband is smaller than that of the *L-S* and *S-L* cases. This somewhat surprising result is explained by the fact that lot-to-lot variation has no impact on variance (but still affects the mean), hence the uncertainty of the corner is smaller.

cf_t %	cf_f %	k'_f	k'_s
50	60	3.051 (1.70%)	3.261 (8.70%)
60	70	3.116 (3.87%)	3.278 (9.27%)
70	80	3.197 (6.57%)	3.293 (9.77%)
80	90	3.320 (10.7%)	3.307 (10.2%)
90	95	3.438 (14.6%)	3.432 (14.4%)
95	99	3.574 (19.1%)	3.475 (15.8%)

Table 7.7: Guardband of fast/slow corner for different confidence levels assuming $\hat{n} = 10$, $\tilde{n} = 1$, and $\tilde{n}' = 25$.

7.4 Confidence in SPICE Fast/Slow Corner

Fast/slow corners for circuit simulation is the major interface between design and process. A common approach to derive the corners is to use measured values from canonical circuits such as an inverter chain. Variation source values corresponding to the corner delay $D^w_{f/s}$ are then calculated.

We assume that the inverter chain delay is a linear function of variation sources:

$$D = d_0 + \sum_{i=1}^{m} c_i X_i \tag{7.51}$$

where m is the number of variation sources, d_0 is the nominal delay, X_i's are variation sources, and c_i's are sensitivity coefficients of inverter chain delay to variation sources. Considering all variation sources to be independent, the mean and variance of the product inverter chain delay can be calculated as:

$$\tilde{\mu}_D = d_0 + \sum_{i=1}^{m} c_i(\mu_{oi} + \tilde{\mu}_{li}) \tag{7.52}$$

$$\tilde{\sigma}^2_D = \sum_{i=1}^{m} c_i^2(\sigma^2_{oi} + \tilde{\sigma}^2_{li}) \tag{7.53}$$

where μ_{oi} and σ^2_{oi} are the mean and variance of other variation for the i^{th} variation source, respectively, $\tilde{\mu}_{li}$ and $\tilde{\sigma}^2_{li}$ are the mean and variance of lot-to-lot variation for the i^{th} variation source, respectively. $\tilde{\mu}_{li}$ and $\tilde{\sigma}^2_{li}$ can be calculated as shown in Section 7.3.1.

Usually, the delay corner is expressed as

$$\tilde{D}_{f/s} = \tilde{\mu}_D \pm k_{f/s}\tilde{\sigma}_D \tag{7.54}$$

As in Section 7.3.5 we use Monte-Carlo simulation to obtain the PDF of the fast and slow delay corners. One sided confidence interval is then used to obtain the worst case fast and slow delay corners D^w_f/s. Corresponding corner values of variation sources are calculated by solving the following [121]:

$$D^w_{f/s} = d_0 + \sum_{i=1}^{m} c_i X^w_i \tag{7.55}$$

$$\frac{X^w_{1f/s} - \hat{\mu}_1}{c_1\hat{\sigma}_1} = \frac{X^w_{2f/s} - \hat{\mu}_2}{c_2\hat{\sigma}_2} \ldots = \frac{X^w_{mf/s} - \hat{\mu}_m}{c_m\hat{\sigma}_m} \tag{7.56}$$

Table 7.8 illustrates the guardband percentage for the worst case delay and the corresponding variation source values under different confidence levels. In the table, the guardband percentage is calculated as: $100\times|$worst_case_value - nominal_value$|$/nominal_value. In the experiment, we use PTM bulk low power SPICE model for 45nm technology [1]. We assume three variation sources, gate length L, threshold voltage for NMOS V_{tn} and PMOS V_{tp}. For gate length variation, we assume $3\sigma = 3nm$, for threshold voltage variation, we assume that the 3σ value is 20% of the nominal value. We also assume that the inter-lot variation is 18% as in previous sections. In the experiment, we let $\hat{n} = 10$ and $\tilde{n} = 15$. Since measured corner values affect production corners,we show the average of 1000 runs.

From the table, we observe that under the same confidence level, the guardband percentage of delay corner is less than that of k'_f value of variation sources as shown in Table 7.4. This is because the k'_f value of variation sources does not depend on the nominal mean and variance. However, for the worst case delay, the guard band percentage does depend on nominal mean. When the nominal mean increases, the guardband percentage decreases.

cf_t	cf_f	D_f^w	L_f^w	V_{tnf}^w	V_{tpf}^w	D_s^w	L_s^w	V_{tns}^w	V_{tps}^w
50	60	0.29	0 07	0.21	0.20	0.52	0.13	0.39	0.36
60	70	0.57	0 14	0.43	0.40	0.77	0.19	0.58	0.55
70	80	1.15	0 29	0.86	0.81	1.29	0.32	0.97	0.91
80	90	2.01	0 50	1.50	1.41	1.80	0.45	1.35	1.27
90	95	3.15	0 79	2.36	2.22	2.84	0.71	2.13	2.00
95	99	4.58	1 15	3.44	3.23	3.61	0.90	2.71	2.54
Nominal		349	42.62	0.558	0.603	388	47.41	0.612	0.643

Table 7.8: Percentage guardband of worst case delay corner and corresponding variation source corners under different confidence levels assuming $\hat{n} = 10$, $\tilde{n} = 15$. Note: delay value is in *ps*, L_{gate} value is in *nm*, and V_{th} value is in V.

Targeted $Conf$		Simulated $Conf$	
cf_t	cf_f	cf_t	cf_f
50	60	51.9	61.6
60	70	61.7	72.4
70	80	71.6	81.4
80	90	81.1	91.3
90	95	90.8	95.7
95	99	95.5	99.2

Table 7.9: Confidence comparison for inverter chain based corner assuming $\hat{n} = 10$, $\tilde{n} = 15$.

In Table 7.9, we also compare the targeted confidence and the exact confidence of the estimated worst case corner value. The flow to obtain the simulated confidence is shown in Figure 7.3. In the experiment, we let $n_{run} = 1,000$. From the table, we find that the exact simulated confidence is a little bit higher than the target value. Such error largely comes from the fitting of the linear delay model to SPICE.

Table 7.10 and 7.11 show the average guardband percentage for the *L-S* and *S-L* cases, respectively. Note that the guardband, though appears to be small, is significant when compared to variation itself (e.g. σ for V_{tp} variation is only 6.7% of nominal value, while the guardband of fast corner is up to 2.58% of nominal value. That is, the

1. $n_f = 0, n_t = 0$
2. for i=1 to n_{run}
   3. Generate $\hat{n}$ samples of measured lots /* calculate estimated value */
   4. Calculate $\hat{\mu}$ and $\hat{\sigma}^2$ for each variation sources from samples
   5. Perform 10,000-sample MC simulation on (7.54)
      according to $\hat{\mu}$ and $\hat{\sigma}^2$.
   6. Obtain worst case corners: D_f^w, D_s^w.
   7. Generate $\tilde{n}$ samples of production lots /* calculate simulated value */
   8. Calculate $\tilde{\mu}$ and $\tilde{\sigma}^2$ for each variation source from samples.
   9. Perform 10,000-sample SPICE MC simulation on inverter
      chain delay based on $\tilde{\mu}$ and $\tilde{\sigma}^2$.
   10. Calculate mean μ_D and variance σ_D^2 of SPICE MC samples.
   11. Calculate simulated corners: $D_f = \mu_D - k_f\sigma_D, D_s = \mu_D + k_s\sigma_D$.
   12. if $D_f > D_f^w$ /* compare simulated value and estimated value */
      13. n_f=$n_f + 1$.
      14. if $D_s < D_s^w$
         15. $n_t = n_t + 1$
16. $cf_f = n_f/n_{run}, cf_t = n_t/n_{run}$

Figure 7.3: Flow to simulate confidence.

guardband value is up to 38.7% of standard deviation).

Figure 7.1 compares the 90% confidence value of D_f^w between the *L-S* (or *S-L*) and *L-L*. In the figure, D_f^w is normalized with respect to the nominal value. From the figure, We observe that $\tilde{n} \geq 30$ (or $\hat{n} \geq 45$), can be considered as "large". Similar to the single source case, when error of k_f' value between *L-L* and *L-S* (or *S-L*) case are within 3%, we consider $\hat{n}$ (or $\tilde{n}$) value is large. We also find that for the worst case delay corner, the value of $\hat{n}$ (or $\tilde{n}$) to be considered as large is smaller than that of the single variation source as in Figure 7.1. This is because the guardband of worst case delay is smaller than that of single variation source as discussed above. Hence, we need fewer lots to achieve the same model reliability.

Table 7.12 shows the average guardband percentage for the *L-1* case. The guardband percentage of this case is smaller that of *S-S*, *L-S*, or *S-L* cases. As discussed in Section 7.3.5, this is because lot-to-lot variation has no impact on variance. Therefore,

cf_t	cf_f	D_f^w	L_f^w	V_{tnf}^w	V_{tpf}^w	D_s^w	L_s^w	V_{tns}^w	V_{tps}^w
50	60	0.21	0.05	0.16	0.15	0.39	0.10	0.29	0.27
60	70	0.43	0.11	0.32	0.30	0.58	0.14	0.43	0.41
70	80	0.86	0.21	0.64	0.61	0.97	0.24	0.72	0.68
80	90	1.50	0.38	1.13	1.06	1.35	0.34	1.01	0.95
90	95	2.36	0.59	1.77	1.67	2.13	0.53	1.59	1.50
95	99	3.44	0.86	2.58	2.42	2.71	0.68	2.03	1.91

Table 7.10: Percentage guardband of worst case delay corner and corresponding variation sources corners under different confidence levels assuming large and $\tilde{n} = 15$.

cf_t	cf_f	D_f^w	L_f^w	V_{tnf}^w	V_{tpf}^w	D_s^w	L_s^w	V_{tns}^w	V_{tps}^w
50	60	0.24	0.06	0.18	0.17	0.44	0.11	0.33	0.31
60	70	0.49	0.12	0.37	0.34	0.66	0.16	0.49	0.46
70	80	0.97	0.24	0.73	0.69	1.10	0.27	0.82	0.77
80	90	1.70	0.43	1.28	1.20	1.53	0.38	1.15	1.08
90	95	2.68	0.67	2.01	1.89	2.41	0.60	1.81	1.70
95	99	3.90	0.97	2.92	2.75	3.07	0.77	2.30	2.16

Table 7.11: Percentage guardband of worst case delay corner and corresponding variation source corners under different confidence levels assuming $\hat{n} = 10$ and large $\tilde{n}$.

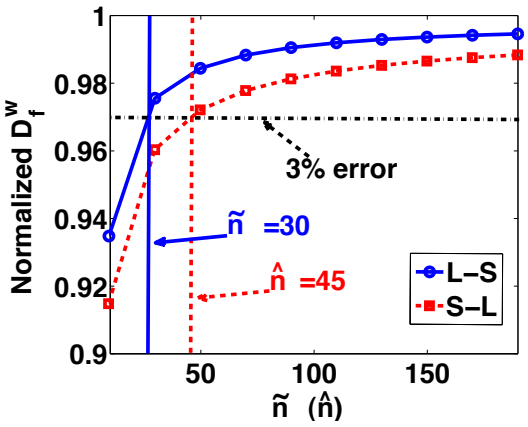


Figure 7.4: 90% confident D_f^w with varying $\hat{n}$ and $\tilde{n}$.

cf_t	cf_f	D_f^w	L_f^w	V_{tnf}^w	V_{tpf}^w	D_s^w	L_s^w	V_{tns}^w	V_{tps}^w
50	60	0.23	0.06	0.17	0.16	0.42	0.11	0.32	0.30
60	70	0.38	0.10	0.29	0.27	0.53	0.13	0.40	0.37
70	80	0.69	0.17	0.52	0.49	0.71	0.18	0.53	0.50
80	90	0.95	0.24	0.71	0.67	0.90	0.23	0.68	0.63
90	95	1.46	0.37	1.10	1.03	1.21	0.30	0.91	0.85
95	99	2.05	0.51	1.54	1.45	1.87	0.47	1.40	1.32

Table 7.12: Percentage guardband of worst case delay corner and corresponding variation source corners under different confidence levels assuming $\hat{n} = 10$, $\tilde{n} = 1$, and $\tilde{n}' = 25$.

we need lower guardband to achieve the same confidence.

In this section, we illustrated the extra guardband required in corners (which ironically are themselves guardbands by definition) to ensure confidence in statistical analysis when the number of lots used to characterize variation or number of production silicon lots are small. Next we discuss the impact of limited characterization or production data on one of the most talked about chip-level statistical analysis methods, namely, statistical timing analysis.

7.5 Confidence in Statistical Timing Analysis

In this section, we will discuss about confidence estimation for statistical static timing analysis (SSTA). Since quadratic SSTA with non-Gaussian variation sources is very complicated, we only analyze linear SSTA with Gaussian variation sources in this section.

By performing linear SSTA in Section 5.5, the chip delay is expressed as a linear canonical form of variation sources:

$$D = d_0 + \sum_{i=1}^{m} c_i X_i \tag{7.57}$$

Since circuit delay has a linear canonical form, its mean and variance can be calculated in the same way as shown in (7.52). We again use Monte-Carlo simulation on the linear equation to obtain the CDF of the output mean ($CDF_{\mu_D}(\cdot)$) and variance ($CDF_{\sigma_D^2}(\cdot)$). Then for a given confidence level $Conf$, it is easy to obtain the worst case mean and variance: [5]

$$\mu^w = CDF_{\mu_D}^{-1}(Conf) \tag{7.58}$$

$$\sigma^{w2} = CDF_{\sigma_D^2}^{-1}(Conf) \quad \sigma^w = \sqrt{\sigma^{w2}} \tag{7.59}$$

Another quantity of interest is a certain percentile point. Since we assume Gaussian variation sources and apply linear canonical form to approximate circuit delay, the circuit delay also follows Gaussian distribution. Therefore, the percentile point $C(p\%)$ can be calculated as:

$$C(p\%) = \mu + k\sigma \tag{7.60}$$

$$k = \Phi^{-1}(p\%) \tag{7.61}$$

[5] Notice that we have known the distribution of mean and variance, it is easy for us to obtain any interval of mean and variance under a certain confidence.

where $\Phi(\cdot)$ is the CDF of standard normal distribution. In this way, the percentile point is converted to a $\mu + k\sigma$ corner. We may apply the same method as in Section 7.4 to obtain the CDF of the percentile point, and then calculate its worst case value.

Figure 7.5 illustrates the worst case mean, standard deviation, and 95% percentile point for ISCAS85 benchmarks. In the experiment, we apply the same experimental setting as that for the inverter chain as in Section 7.4. As discussed before, the worst case value depends on the measured values of mean and variance. Hence, different runs may result in different worst case values. In the figure, the worst case value is averaged over 1,000 runs and normalized with respect to the nominal value. From the figure, we observe that the guardband of chip delay is lower than that of single variation source and inverter chain delay. This is because chip delay has a larger nominal mean to variance ratio than inverter chain. However, although the guardband value seems not large, it is very significant compared to the scale of variation. For example, for the 95-percentile point, the nominal value is 6.9X of standard deviation and the guardband value is up to 7.6% of nominal value to ensure 95% confidence. That means the guardband value is up to 52% of standard deviation.

In Table 7.13, we compare the targeted confidence and the simulated confidence of the estimated confidence interval. The simulated confidence is obtained in Figure 7.3. The only difference is that instead of running SPICE Monte-Carlo simulation, we apply SSTA [41] to obtain the chip delay distribution. Due to our simplifying assumption that the linear canonical form is independent of small changes in mean and variance, there is a small error between targeted and simulated confidence.

Figure 7.6 illustrates the mean, standard deviation, and 95-percentile point of ISCAS85 benchmarks under different confidence level for *S-L* case. We ignore *L-S* and *L-1* cases where the need of SSTA is not well motivated.

Figure 7.7 compares the 90% confidence value of 95-percentile point for C7552

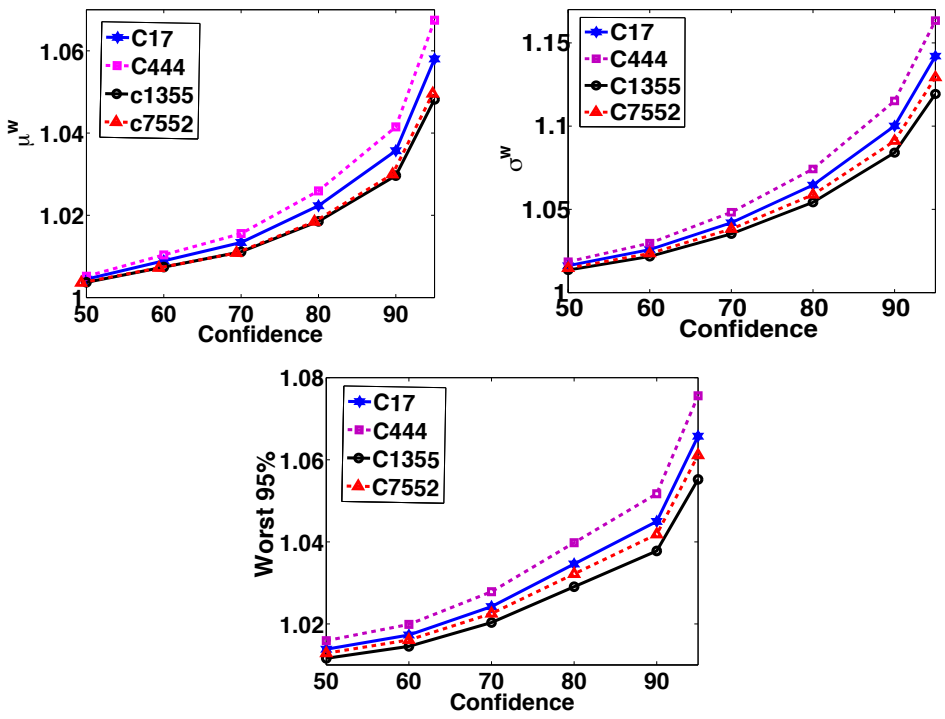


Figure 7.5: Worst case mean, standard deviation, and 95-percentile point normalized with respect to the nominal value for ISCAS85 benchmarks, assuming $\hat{n} = 10$ and $\tilde{n} = 15$.

Targeted	Simulated *conf*		
conf	μ	σ^2	95%
50	51.3	51.4	51.3
60	60.8	60.9	61.2
70	70.1	70.7	70.8
80	80.9	81.0	80.3
90	90.5	90.4	90.2
95	95.5	95.2	95.3

Table 7.13: Confidence comparison for ISCAS85 benchmark 95-percentile point delay assuming $\hat{n} = 10$, $\tilde{n} = 15$.

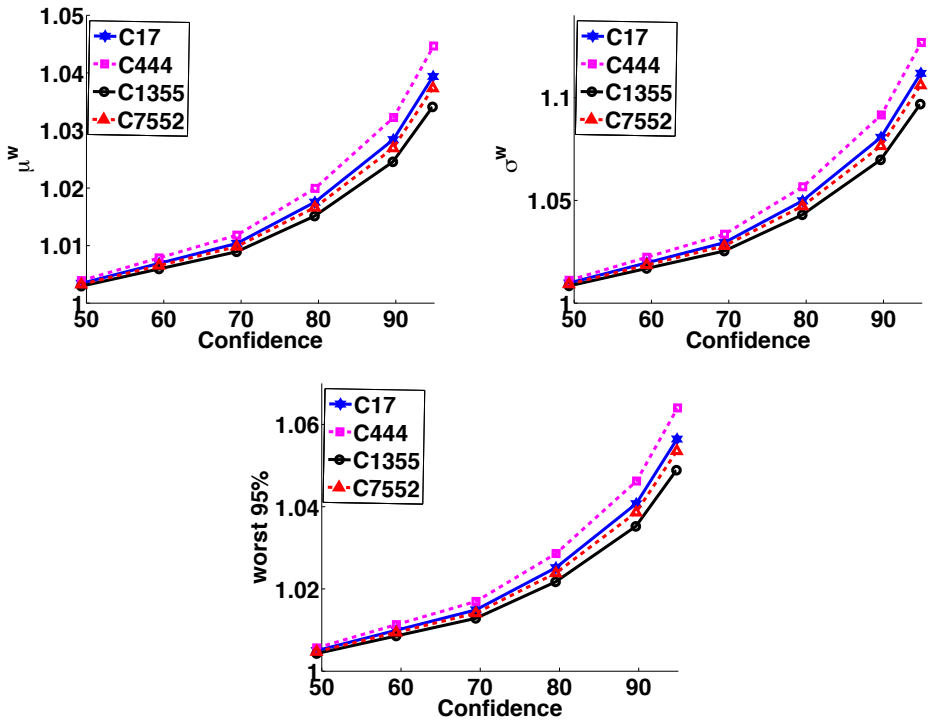


Figure 7.6: Normalized worst case mean, standard deviation, and 95-percentile point for ISCAS85 benchmarks, assuming $\hat{n} = 10$ and $\tilde{n}$ is large.

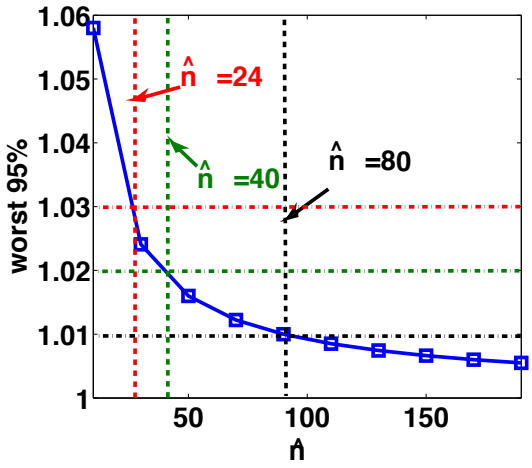


Figure 7.7: 90% confidence worst case 95-percentile point with varying $\hat{n}$ for C7552. Percentile point is normalized with respect to the nominal value.

between *S-L* and *L-L*. In the figure, percentile point is normalized with respect to the nominal value. From the figure, we observe that the guardband of chip delay is lower than that of single variation source and inverter chain delay. We find that we only need a small number of lots ($\hat{n} \geq 24$) to bound the error between *S-L* and *L-L* under 3%. If we want to bound the error to 2% or 1%, we need $\hat{n} \geq 40$ and $\hat{n} \geq 80$, respectively.

Again, we see that limited data can lead to significantly large guardbanding to ensure good confidence in analysis results. The method discussed in this section is fairly straightforward, fast (essentially Monte Carlo on a linear expression, runtime is within 10s) and accurate to estimate confidence in results of statistical timing analysis especially when the variability models are known to suffer from limited lot-to-lot variation data.

7.6 Practical Questions: Determining Confidence Induced Guardband for Real Designs

So far in the chapter, we have proposed techniques to quantify the confidence in statistical circuit analysis and estimate guardband required to attain a desired level of confidence. The answer to "what is the desired level of confidence" is not an easy one especially given the steep increase in guardband to ensure higher levels of confidence. Besides quality of models ($\hat{n}$) and production volume ($\tilde{n}$), two issues influence the choice of confidence level:

1. *Risk (in terms of schedule, performance, power, etc) tolerable for the design.* A lower confidence does *not* necessarily translate to lower yield but it implies that the probability of the statistical analysis actually matching real silicon is smaller. Therefore, confidence level is directly related to risk. For designs where power/performance constraints are flexible or time to volume is not critical (i.e., if yield is lower than expectation, running more lots is acceptable), a lower confidence is acceptable. Confidence levels provide a tradeoff between risk of higher manufacturing cost (due to low yield) and higher upfront design cost (due to guardbanding).

2. *Extent of post-silicon tuning options available.* Ability to change design characteristics during or post-manufacturing mitigates the need for high confidence in models. If the silicon foundry can extensively adjust the process (which essentially shifts the mean) for the design[6], the silicon can be *made* to match the analysis and ensuring high confidence in design-side estimates is not essential. Similarly, tunability using knobs such as adaptive body biasing, adaptive voltage scaling can also alleviate downside of low confidence.

[6]This is usually possible only for high-volume, high-value designs.

In the end, the choice of confidence will be as much a business decision as a technical decision as it quantifies the desire for predictability of statistics of manufacturing.

7.7 Conclusions

In this chapter, we have studied impact of characterization and production silicon volumes on believability of statistical variation models and analysis. We have developed methods to estimate the distribution of production silicon statistical characteristics (e.g., mean, variance, percentile corners, etc) in terms of (unreliable) measured statistical characteristics, measured data volume, and intended production volume. The loss of reliability largely stems from lot-to-lot variation. Our experiments and results indicate the following.

1. For small characterization or production volumes, best/worst corners for a variation source may need as much as 30 % guardband to attain 95% confidence. Our estimates show strong agreement with confidence measured from a set of real silicon data (maximum error $\leq 4\%$).

2. For a typical SPICE corner extraction methodology, the guardband needed for 95% confidence is about 14% of the nominal uncertainty (difference between best/worst corners) for low-to-medium volume production ($\hat{n} = 10, \tilde{n} = 15$).

3. Finally, the 95^{th} percentile delay as estimated using statistical timing analysis needs a guardband of 5%-7%.

Predicted statistics are reliable for number of measured lots larger than 85 and number of production lots larger than 60. We assumed lot-to-lot variation to be 18% of total variance for our experiments. The required guardband will increase if lot-to-lot variation increases as a fraction of total variation. We believe that confidence-

level induced guardband should be considered for all low-to-medium volume designs. Moreover, the problem is likely to become more severe when 450mm wafer sizes are adopted by the industry (since less number of lots would be required to meet the same production volume).

CHAPTER 8

Conclusions

In modern *V*ery *L*arge *S*cale *I*ntegration (*VLSI*) circuit industry, the scaling of *C*omplementary *M*etal *O*xide *S*emiconductor (*CMOS*) technology enables the ever growing number of transistors on a chip, hence improves the chip functionality and reduces the manufacturing cost. However, the process variation caused by inevitable manufacturing fluctuations and imperfections also increases when technology scales down. Process variation introduces significant power and performance uncertainty to VLSI circuits and becomes a potential stopper for further technology scaling.

In this dissertation, we consider two types of the most common VLSI circuits: *F*ield-*P*rogrammable *G*ate *A*rray (*FPGA*) and *Application S*pecific *I*ntegrated *C*ircuit (*ASIC*). We have modeled, analyzed, and optimized power and delay for both FPGAs and ASICs.

In Part I, including Chapter 2, 3, 4, we have studied modeling, analysis, and optimization of FPGA power and performance with existence of process variation.

In Chapter 2, we have developed a trace-based power and performance evaluation framework(*Ptrace*) for FPGA. Then we performed device and architecture co-evaluation on hundreds of device and architecture combinations to optimize power, delay, and area. Compared to the baseline, our optimization reduces the energy-delay product by 18.4% and area by 23.3%.

In Chapter 3, we have extended Ptrace to consider process variation. Then we per-

formed device and architecture co-optimization to maximize power and delay yield. Compared to the baseline, our optimization improves leakage power and timing combined yield by 9.4%.

In Chapter 4, we have developed transistor level voltage and current model based on ITRS *MASTAR4* (*M*odel for *A*ssessment of cmo*S* *T*echnologies *A*nd *R*oadmaps), and then developed circuit level power, delay, and reliability model. Combining the circuit level model with Ptrace, we have developed a framework to evaluate FPGA power, delay, and reliability simultaneously. We have shown that applying heterogeneous gate lengths to logic and interconnect may lead to 1.3X delay difference, 3.1X energy difference. This offers a large room for power and delay tradeoff. We have further shown that the device aging has a knee point over time and device burn-in to reach the point could reduce the performance change over 10 years from 8.5% to 5.5% and significantly reduce leakage variation. In addition, we have also studied the interaction between process variation, device aging, and SER. We observe that device aging reduces standard deviation of leakage by 65% over 10 years while it has relatively small impact on delay variation. Moreover, we also find that neither device aging due to NBTI and HCI nor process variation have significant impact on SER.

In Part II, including Chapter 5, Chapter 6, and Chapter 7, we have studied statistical timing modeling and analysis for ASICs.

In Chapter 5, we have proposed a new method to approximate the max operation of two non-Gaussian random variables using second-order polynomial fitting. By applying such approximation, we have presented new SSTA algorithms for three different delay models: quadratic model, quadratic model without crossing terms (semi-quadratic model), and linear model. All atomic operations of these algorithms are performed by closed-form formulas, hence they are very time efficient. Experimental results show that compared to Monte-Carlo simulation, our SSTA flow predicts the

mean, standard deviation, skewness, and 95-percentile point within 1%, 1%, 6%, and 1% error, respectively.

In Chapter 6, we have studied the impact of systematic across-wafer variation on within-die variation. Based on the analysis, we have developed an efficient and accurate spatial variation model to model across-wafer variation at die level. We then implemented the new model to the SSTA flow in Chapter 5. Experimental results show that our new model reduces error from 6.5% to 2% and improve run time by 6X compared to the traditional spatial variation model.

In Chapter 7, we have studied impact of characterization and production silicon volumes on believability of statistical variation models and analysis. We mathematically derived the confidence intervals for commonly used statistical measures (mean, variance, percentile corner) and analysis (SPICE corner extraction, statistical timing). Our estimates are within 2% of simulated confidence values. Our experiments indicate that for moderate characterization volumes (10 lots) and low-to-medium production volumes (15 lots), a significant guardband (e.g., 34.7% of standard deviation for single parameter corner, 38.7% of standard deviation for SPICE corner, and 52% of standard deviation for 95%-tile point of circuit delay are needed) to ensure 95% confidence in the results. The guardbands are non-negligible for all cases when either production or characterization volume is not large.

This dissertation makes contributions to statistical modeling, analysis, and optimization for FPGAs and ASICs. However, there are still lots of research works need to be done in this field. In practice, the statistical modeling and analysis techniques presented in Part II can not only be applied to ASICs, but also to FPGAs. The SSTA method introduced in Chapter 5 can be applied to FPGA delay variation for more accurate and efficient delay variation estimation; the model of spatial correlation in Chapter 6 can also be applied to FPGA power and delay variation estimation to im-

prove the accuracy and efficiency; and we may also apply the method in Chapter 7 to estimate the confidence for FPGA power and delay variation analysis. In the future, applying the statistical analysis techniques for ASICs to FPGAs is one of our research topics.

CHAPTER 9

Appendix

In the appendix, I briefly introduce the works I have finished in my Ph.D. program and are not introduced in this dissertation:

- **FPGA Performance Optimization via Chipwise Placement Considering Process Variations**. We develop a statistical chipwise placement flow to improve FPGA performance. First, we propose an efficient high-level trace-based estimation method, similar to *Ptrace*, to evaluate the potential performance gain achievable through chipwise FPGA placement without detailed placement. Second, we develop a variation-aware detailed placement algorithm *vaPL* within the VPR framework [18] to leverage process variation and optimize performance for each chip. Chipwise placement *vaPL* is deterministic for each chip when the chip's variation map is known, and leads to different placements for different chips of the same application. Our experimental results show that, compared to the existing FPGA placement, variation aware chipwise placement improves circuit performance by up to 12.1% for the tested variation maps. This work was published in *International Conference on Field Programmable Logic and Applications (FPL)*, August 2006.

- **Fourier Series Approximation for Max Operation in Non-Gaussian and Quadratic Statistical Static Timing Analysis**. We propose efficient algorithms to handle the max operation in SSTA with both quadratic delay dependency and

non-Gaussian variation sources simultaneously. Based on such algorithms, we develop an SSTA flow with quadratic delay model and non-Gaussian variation sources. All the atomic operations, max and add, are calculated efficiently via either closed-form formulas or low dimension (at most 2D) lookup tables. We prove that the complexity of our algorithm is linear in both variation sources and circuit sizes, hence our algorithm scales well for large designs. Compared to Monte Carlo simulation for non-Gaussian variation sources and nonlinear delay models, our approach predicts the mean, standard deviation and 95% percentile point with less than 2% error, and the skewness with less than 10% error. This work was published in *Design Automation Conference (DAC)*, June 2007.

- **Accounting for Non-linear Dependence Using Function Driven Component Analysis**. We first analyze the impact of non-linear dependence on statistical analysis and show that ignoring non-linear dependence may cause error in both statistical timing and power analysis. Then we introduce a function driven component analysis (*FCA*) to minimize the error introduced by non-linear dependence. Experimental results show that the proposed FCA method is more accurate compared to the traditional PCA or ICA. This work was published in *IEEE/ACM Asia South Pacific Design Automation Conference (ASPDAC)*, February 2009.

- **Efficient Additive Statistical Leakage Estimation**. We present an efficient additive leakage variation model. We use polynomial instead of exponential function to represents the leakage variation considering inter-die variation, then extend such model to consider both within-die spatial and within-die random variation. Due to the additivity, this new model is more efficient than the exponential model when calculating the full chip leakage power. Experimental results show that our method is 5X faster than the existing Wilkinson's approach

with no accuracy loss in mean estimation, about 1% accuracy loss in standard deviation estimation and 99% percentile point estimation. This work was published in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Volume 28, issue: 11, November 2009, pp:1777 - 1781.

REFERENCES

[1] PTM SPICE model. In *http://www.eas.asu.edu/ ptm/latest.html.*

[2] Soroush Abbaspour, Hanif Fatemi, and Massoud Pedram. Non-gaussian statistical interconnect timing analysis. In *Proc. Design Automation and Test Conf. in Europe*, March 2006.

[3] Soroush Abbaspour, Hanif Fatemi, and Massoud Pedram. Parameterized block-based non-gaussian statistical gate timing analysis. In *Proc. Asia South Pacific Design Automation Conf.*, January 2006.

[4] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York: Dover, 1965.

[5] Aseem Agarwal, David Blaauw, and Vladimir Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Proc. Intl. Conf. Computer-Aided Design*, pages 900 – 907, Nov. 2003.

[6] Aseem Agarwal, David Blaauw, Vladimir Zolotov, Savithri Sundareswaran, Min Zhao, Kaushik Gala, and Rajendran Panda. Statistical delay computation considering spatial correlations. In *Proc. Asia South Pacific Design Automation Conf.*, 2003.

[7] Aseem Agarwal, David Blaauw, Vladimir Zolotov, Savithri Sundareswaran, Min Zhao, Kaushik Gala, and Rajendran Panda. Statistical delay computation considering spatial correlations. In *Proc. Asia South Pacific Design Automation Conf.*, pages 271 – 276, January 2003.

[8] Aseem Agarwal, David Blaauw, Vladimir Zolotov, and Sarma Vrudhula. Computation and refinement of statistical bounds on circuit delay. In *Proc. Design Automation Conf.*, June 2003.

[9] Aseem Agarwal, Vladimir Zolotov, and David Blaauw. Statistical timing analysis using bounds and selective enumeration. In *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, volume 22, pages 1243 – 1260, Sept. 2003.

[10] Elias Ahmed and Jonathan Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, pages 3–12, February 2000.

[11] M.A. Alam and S. Mahapatra. A comprehensive model of pmos nbti degradation. In *Microeletronics Reliability*, August 2005.

[12] Jason H. Anderson, Farid N. Najm, and Tim Tuan. Active leakage power optimization for FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2004.

[13] Hongliang Chang andVladimir Zolotov, Sambasivan Narayan, and Chandu Visweswariah. Parameterized block-based statistical timing analysis with non-Gaussian and nonlinear parameters. In *Proc. Design Automation Conf.*, pages 71–76, June 2005. Anaheim, CA.

[14] Ghazanfar Asadi and Mehdi B. Tahoori. Soft error rate estimation and mitigation for SRAM-based FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2005.

[15] Maryam Ashouei, Abhijit Chatterjee, Adit D. Singh, and Vivek De. A dual-vt layout approach for statistical leakage variability minimization in nanometer CMOS. In *Proc. Int. Conf. on Computer Design*, October 2005.

[16] Adelchi Azzalini. The skew-normal distribution and related multivariate families. *Scandinavian Journal of Statistics*, 32:159–188, June 2005.

[17] M. Bellato, P. Bernardi1, D. Bortolato, A. Candelori, M. Ceschia, A. Paccagnella, M. Rebaudengo1, M. Sonza Reorda, M. Violante1, and P. Zambolin. Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA. In *Design Automation and Test in Europe*, March 2004.

[18] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, February 1999.

[19] Sarvesh Bhardwaj, Praveen Ghanta, and Sarma Vrudhula. A framework for statistical timing analysis using non-linear delay and slew models. In *Proc. Intl. Conf. Computer-Aided Design*, November 2006.

[20] Sarvesh Bhardwaj and Sarma Vrudhula. Leakage minimization of nano-scale circuits in the presence of systematic and random variations. In *Proc. Design Automation Conf.*, 2005.

[21] Sarvesh Bhardwaj and Sarma Vrudhula. Leakage minimization of digital circuits using gate sizing in the presence of process variations. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, March 2008.

[22] Sarvesh Bhardwaj, Sarma Vrudhula, and David Blaauw. τau: Timing analysis under uncertainty. In *Proc. Intl. Conf. Computer-Aided Design*, pages 615 – 620, Nov. 2003.

[23] Sarvesh Bhardwaj, Wenping Wang, Rakesh Vattikonda, Yu Cao, and S. Vrudhula. Predictive modeling of the nbti effect for reliable design. In *Proc. IEEE Custom Integrated Circuits Conf.*, September 2006.

[24] Duane Boning, James Chung, Dennis Ouma, and Rajesh Divecha. Spatial variation in semiconductor processes: Modeling for control. In *Proceedings Process Control Diagnostics and Modeling in Semiconductor Manufacturing II Electrochemical Society Meeting*, May 1997.

[25] 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.

[26] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impacts on circuits and microarchitecture. In *Proc. Design Automation Conf.*, June 2003.

[27] Jozef Brcka and Rodney L. Robison. Wafer redeposition impact on etch rate uniformity in ipvd system. *IEEE Transactions on Plasma Sciences*, February 2007.

[28] Michael Brown, Cyrus Bazeghi, Matthew Guthaus, and Jose Renau. Measuring and modeling variabilityusing low-cost FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2009.

[29] James Burnham, Chih-Kong Ken Yang, and Haitham Hindi. A stochastic jitter model for analyzing digital timing-recovery circuits. In *Proc. Design Automation Conf.*, July 2009.

[30] Steven M. Burns, Mahesh C. Ketkar, Noel Menezes, Keith A. Bowman, James W. Tschanz, and Vivek De. Comparative analysis of conventional and statistical design techniques. In *Proc. Design Automation Conf.*, June 2007.

[31] Nicola Campregher, Peter Y. K. Cheung, George A. Constantinides, and Milan Vasilko. Yield enhancements of design-specific fpgas. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2006.

[32] Hongliang Chang and Sachin S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proc. Intl. Conf. Computer-Aided Design*, pages 621 – 625, Nov. 2003.

[33] Hongliang Chang and Sachin S. Sapatnekar. Full-chip analysis of leakage power under process variations, including spatial correlations. In *Proc. Design Automation Conf.*, June 2005.

[34] Kueing-Long Chen, Stephen A. Saller, Imelda A. Groves, and David B. Scott. Reliability effects on mos transistors due to hot-carrier injection. *IEEE Journal of Solid-State Circuits*, February 1985.

[35] Ruiming Chen, Lizheng Zhang, Vladimir Zolotov, Chandu Visweswariah, and Jinjun Xiong. Static timing: Back to our roots. In *Proc. Asia South Pacific Design Automation Conf.*, February 2008.

[36] Ruiming Chen and Hai Zhou. Timing budgeting under arbitrary process variations. In *Proc. Intl. Conf. Computer-Aided Design*, November 2007.

[37] Ruiming Chen and Hai Zhou. Fast estimation of timing yield bounds for process variations. *IEEE Trans. VLSI Syst.*, March 2008.

[38] Lerong Cheng, Fei Li, Yan Lin, Phoebe Wong, and Lei He. Device and architecture cooptimization for FPGA power reduction. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 19:1211–1221, July 2007.

[39] Lerong Cheng, Yan Lin, and Lei He. Tracebased framework for concurrent development of process and FPGA architecture considering process variation and reliability. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2008.

[40] Lerong Cheng, Jinjun Xiong, and Lei He. Non-linear statistical static timing analysis for non-gaussian variation sources. In *Proc. Design Automation Conf.*, June 2007.

[41] Lerong Cheng, Jinjun Xiong, and Lei He. Non-gaussian statistical timing analysis using Second-Order polynomial fitting. In *Proc. Asia South Pacific Design Automation Conf.*, February 2008.

[42] Lerong Cheng, Jinjun Xiong, and Lei He. Non-Gaussian statistical timing analysis using Second-Order polynomial fitting. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, January 2009.

[43] Lerong Cheng, Jinjun Xiong, Lei He, and Mike Hutton. FPGA performance optimization via chipwise placement considering process variations. In *International Conference on Field-Programmable Logic and Applications*, August 2006.

[44] Kaviraj Chopra, Saumil Shah, Ashish Srivastava, David Blaauw, and Dennis Sylvester. Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation. In *Proc. Intl. Conf. Computer-Aided Design*, November 2005.

[45] Kaviraj Chopra, Narendra Shenoy, and David Blaauw. Variogram based robust extraction of process variation. In *Proc. τ, Timing and Analysis Utilities*, February 2007.

[46] Charles E. Clark. The greatest of a finite set of random variables. In *Operations Research*, pages 85 – 91, 1961.

[47] David Lewis ect. The stratix routing and logic architecture. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2003.

[48] Vijay Degalahal and Tim Tuan. Methodology for high level estimation of FPGA power consumption. In *Proc. Asia South Pacific Design Automation Conf.*, January 2005.

[49] Anirudh Devgan and Chandramouli Kashyap. Block-Based static timing analysis with uncertainty. In *Proc. Intl. Conf. Computer-Aided Design*, November 2003.

[50] Nigel Drego, Anantha Chandrakasan, and Duane Boning. Lack of spatial correlation in mosfet threshold voltage variation and implications for voltage scaling. *IEEE Trans. Semiconductor Manufacturing*, May 2009.

[51] Fei Li and Yan Lin and Lei He. Vdd programmability to reduce FPGA interconnect power. In *Proc. Intl. Conf. Computer-Aided Design*, November 2004.

[52] Zhuo Feng, Peng Li, and Yaping Zhan. Fast second-order statistical static timing analysis using parameter dimension reduction. In *Proc. Design Automation Conf.*, June 2007.

[53] F.N.Najm and N.menezes. Statistical timing analysis based on a timing yield model. In *Proc. Design Automation Conf.*, June 2004.

[54] Paul Friedberg, Willy Cheung, and Costas J. Spanos. Spatial modeling of micron-scale gate length variation. In *Data Analysis and Modeling for Process Control III*, March 2006.

[55] Qiang Fu, Wai-Shing Luk, Jun Tao, Changhao Yan, and Xuan Zeng. Characterizing intra-die spatial correlation using spectral density method. In *Proc. IEEE International Symposium on Quality Electronic Design*, March 2008.

[56] Takayuki Fukuoka, Akira Tsuchiya, and Hidetoshi Onodera. Statistical gate delay model for multiple input switching. In *Proc. Asia South Pacific Design Automation Conf.*, February 2008.

[57] 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.

[58] 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.

[59] 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.

[60] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 2000.

[61] Khaled R. Heloue, Sari Onaissi, and Farid N. Najm. Efficient block-based parameterized timing analysis covering all potentially critical paths. In *Proc. Intl. Conf. Computer-Aided Design*, November 2008.

[62] Dadgour H.F., Sheng-Chih Lin, and Banerjee K. A statistical framework for estimation of full-chip leakage-power distribution under parameter variations. *Electron Devices, IEEE Transactions on*, November 2007.

[63] Katsumi Homma, Izumi Nitta, and Toshiyuki Shibuya. Non-gaussian statistical timing models of die-to-die and within-die parameter variations for full chip analysis. In *Proc. Asia South Pacific Design Automation Conf.*, February 2008.

[64] Shin ichi OHKAWA, HirooMASUDA, and Yasuaki INOUE. A novel expression of spatial correlation by a random curved surface model and its application to LSI designs. *IEICE TRANS. FUNDAMENTALS*, 2008.

[65] Masanori Imai, Takashi Sato Noriaki Nakayama, and Kazuya Masu. Non-parametric statistical static timing analysis: An ssta framework for arbitrary distribution. In *Proc. Design Automation Conf.*, June 2008.

[66] International Technology Roadmap for Semiconductor. In *http://public.itrs.net/*, 2002.

[67] International Technology Roadmap for Semiconductor. In *http://public.itrs.net/*, 2005.

[68] International Technology Roadmap for Semiconductor. A user's guide to MASTAR4. In *http://www.itrs.net/models.html*, 2005.

[69] International Technology Roadmap for Semiconductor. Design. In *http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_Design.pdf*, 2007.

[70] Javid Jaffari and Mohab Anis. On efficient monte carlo-based statistical static timing analysis of digital circuits. In *Proc. Intl. Conf. Computer-Aided Design*, November 2008.

[71] Jason H. Anderson and Farid N. Najm. Low-power programmable routing circuitry for FPGAs. In *Proc. Intl. Conf. Computer-Aided Design*, November 2004.

[72] J. Jess, K. Kalafala, S. Naidu, R. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. In *Proc. Design Automation Conf.*, June 2003.

[73] Jochen Jess. Parametric yield estimation for deep submicron VLSI circuits. In *15th Symposium on Integrated Circuits and Systems Design*, September 2002.

[74] J.M.Rabaey. *Digital Integrated Circuits - A Design Perspective*. Prentice Hall Inc., 2003.

[75] Kunhyuk Kang, Bipul C. Paul, and Kaushik Roy. Statistical timing analysis using levelized covariance propagation. In *Proc. Design Automation and Test Conf. in Europe*, March 2005.

[76] Vishal Khandelwal and Ankur Srivastava. A general framework for accurate statistical timing analysis considering correlations. In *Proc. Design Automation Conf.*, pages 89 – 94, June 2005.

[77] Tae Won Kim and Eray S.Aydil. Investigation of etch rate uniformity of 60 mhz plasma etching equipment. *Japanese Journal of Applied Physics*, November 2001.

[78] Tae Won Kim and Eray S.Aydil. Effects of chamber wall conditions on cl concertration and si etch rate uniformity in palsma etching reactors. *Journal of The Electrochemical Society*, June 2003.

[79] Dirk Koch, Christian Haubelt, and Juergen Teich. Efficient hardware checkpointing – concepts, overhead analysis, and implementation. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[80] J. Kouloheris and A. El Gamal. FPGA area vs. cell granularity - lookup tables and PLA cells. In *1st ACM Workshop on FPGAs, berkeley, CA*, February 1992.

[81] Sanjay V. Kumar, Chandramouli V. Kashyap, and Sachin Sapatnekar. A framework for block-based timing sensitivity analysis. In *Proc. Design Automation Conf.*, June 2008.

[82] Ian Kuon and Jonathan Rose. Measuring the gap between fpgas and asics. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, February 2007.

[83] Eric Kusse and Jan M. Rabaey. Low-energy embedded FPGA structures. In *Proc. Intl. Symp. Low Power Electronics and Design*, pages 155–160, August 1998.

[84] Julien Lamoureux, Guy G. Lemieux, and Steven J.E. Wilton. Glitchless: An active glitch minimization techniques for FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[85] Julien Lamoureux and Steven J.E. Wilton. On the interaction between power-aware FPGA CAD algorithms. In *Proc. Intl. Conf. Computer-Aided Design*, pages 701–708, November 2003.

[86] Jiayong Le, Xin Li, and Lawrence T. Pileggi. STAC: Statisical timing analysis with correlation. In *Proc. Design Automation Conf.*, Jun 2004.

[87] 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.

[88] Bing Li, Ning Chen, Manuel Schmidt, Walter Schneider, and Ulf Schlichtmann. On hierarchical statistical static timing analysis. In *Proc. Design Automation and Test Conf. in Europe*, March 2009.

[89] Fei Li, Deming Chen, Lei He, and Jason Cong. Architecture evaluation for power-efficient FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2003.

[90] Fei Li, Yan Lin, and Lei He. FPGA power reduction using configurable dual-vdd. In *Proc. Design Automation Conf.*, June 2004.

[91] Fei Li, Yan Lin, and Lei He. Field programmability of supply voltages for FPGA power reduction. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 26, April 2007.

[92] Fei Li, Yan Lin, Lei He, Deming Chen, and Jason Congs. Power modeling and characteristics of field programmable gate arrays. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 24:1712–1724, November 2005.

[93] 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.

[94] Xin Li, Jiayong Le, Padmini Gopalakrishnan, and Lawrence T Pileggi. Asymptotic probability extraction for non-normal distributions of circuit performance. In *Proc. Intl. Conf. Computer-Aided Design*, November 2004.

[95] Xin Li, Jiayong Le, and L.T. Pileggi. Projection based statistical analysis of full-chip leakage power with non-log-normal distributions. In *Proc. Design Automation Conf.*, June 2006.

[96] Weiping Liao, Lei He, and Kevin Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, July 2005.

[97] Saihua Lin, Yu Wang, Rong Luo, and Huazhong Yang. A capacitive boosted buffer technique for high-speed process-variation-tolerant interconnect in udvs application. In *Proc. Asia South Pacific Design Automation Conf.*, February 2008.

[98] Yan Lin and Lei He. Dual-vdd interconnect with chip-level time slack allocation for FPGA power reduction. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 25, October 2006.

[99] Yan Lin and Lei He. Stochastic physical synthesis for FPGAs with pre-routing interconnect uncertainty and process variation. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[100] Yan Lin, Mike Hutton, and Lei He. Placement and timing for FPGAs considering variations. In *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2006.

[101] Yan Lin, Fei Li, and Lei He. Circuits and architectures for vdd programmable FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2005.

[102] 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.*, January 2005.

[103] Jing-Jia Liou, A. Krstic, L.-C. Wang, and Kwang-Ting Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *Proc. Design Automation Conf.*, June 2002.

[104] Bao Liu. Signal probability based statistical timing analysis. In *Proc. Design Automation and Test Conf. in Europe*, March 2008.

[105] Bao Liu. Spatial correlation extraction via random field simulation and production chip performance regression. In *Proc. Design Automation and Test Conf. in Europe*, March 2008.

[106] Frank Liu. A general framework for spatial correlation modeling in vlsi design. In *Proc. Design Automation Conf.*, June 2007.

[107] Jui-Hsiang Liu, Lumdo Chen, and Chen Charlie Chung-Ping. Accurate and analytical statistical spatial correlation modeling for vlsi dfm applications. In *Proc. Design Automation Conf.*, June 2008.

[108] Jui-Hsiang Liu, Ming-Feng Tsai, Lumdo Chen, and Charlie Chung-Ping Chen. Accurate and analytical statistical spatial correlation modeling for vlsi dfm applications. In *Proc. Design Automation Conf.*, June 2008.

[109] M. Liu, Wang Wei-Shen, and M. Orshanskyg. Leakage power reduction by dual-vth designs under probabilistic analysis of vth variation. In *Proc. Intl. Symp. Low Power Electronics and Design*, August 2004.

[110] Andrea Lodi, Luca Ciccarelli, and Roberto Giansante. Combining low-leakage techniques for FPGA routing design. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Februray 2005.

[111] Yuanlin Lu and V.D. Agrawal. Statistical leakage and timing optimization for submicron process variation. In *VLSI Design*, January 2007.

[112] W. Maly, A.J. Strojwas, and S.W. Director. VLSI yield prediction and estimation: A unified framework. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 5:114–130, Jan. 1986.

[113] Hratch Mangassarian and Mohab Anis. On statistical timing analysis with inter- and intra-die variations. In *Proc. Design Automation and Test Conf. in Europe*, March 2005.

[114] Murari Mani, Anirudh Devgan, and Michael Orshansky. An efficient algorithm for statistical minimization of total power under timing yield constraints. In *Proc. Design Automation Conf.*, pages 309–314, June 2005.

[115] Yohei Matsumoto, Masakazu Hioki, Takashi Kawanami, Toshiyuki Tsutsumi, Tadashi Nakagawa, Toshihiro Sekigawa, and Hanpei Koike. Performance and yield enhancement of FPGAs with with-in die variation using multiple configurations. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[116] Hushrav D. Mogal, Haifeng Qian, Sachin S. Sapatnekar, and Kia Bazargan. Clustering based pruning for statistical criticality computation under process variations. In *Proc. Intl. Conf. Computer-Aided Design*, November 2007.

[117] Rajarshi Mukherjee and Seda Ogrenci Memik. Power-driven design partitioning. In *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2004.

[118] Ayhan Mutlu, Jiayong Le, Ruben Molina, and Mustafa Celik. A parametric approach for handling local variation effects in timing analysis. In *Proc. Design Automation Conf.*, July 2009.

[119] Siva Narendra, Vivek De, Shekhar Borkar andDimitri A. Antoniadis, and Anantha P. Chandrakasan. Full-chip subthreshold leakage power prediction and reduction techniques for sub-0.18-um CMOS. *Solid-State Circuits, IEEE Journal of*, March 2004.

[120] Michael Orshansky and Arnab Bandyopadhyay. Fast statistical timing analysis handling arbitrary delay correlations. In *Proc. Design Automation Conf.*, June 2004.

[121] Michael Orshansky, Sani R. Nassif, and Duane Boning. *Design for Manufacturability and Statistical Design*. Springer, 2007.

[122] Pierrick Pedron. Tutorial on statistical static timing analysis (SSTA). In *Sophia Antipolis MicroElectronics Forum*, 2008.

[123] 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*, September 2002.

[124] Predictive Technology Model. In *http://www.eas.asu.edu/ ptm/*.

[125] Kun Qian and Costas J. Spanos. A comprehensive model of process variability for statistical timing optimization. In *Design for Manufacturability through Design-Process Integration II*, April 2008.

[126] Kun Qian and Costas J. Spanos. Hierachical modeling of spatial variability of a 45nm test chip. In *Design for Manufacturability through Design-Process Integration III*, March 2009.

[127] Sreeja Raj, Sarma B.K. Vrudhula, and Janet Wang. A methodology to improve timing yield in the presence of process variations. In *Proc. Design Automation Conf.*, June 2004.

[128] Anand Ramalingam, Ashish Kumar Singh, Sani R. Nassif, Gi-Joon Nam, Michael Orshansky, and David Z. Pan. An accurate sparse matrix based framework for statistical static timing analysis. In *Proc. Intl. Conf. Computer-Aided Design*, November 2006.

[129] Rajeev Rao, Anirudh Devgan, David Blaauw, and Dennis Sylveste. Parametric yield estimation considering leakage variability. In *Proc. Design Automation Conf.*, June 2004.

[130] Jonathan Rose, Robert J. Francis, David Lewis, and Paul Chow. Architecture of field-programmable gate arrays: The effect of logic functionality on area efficiency. *IEEE Journal of Solid-State Circuits*, 1990.

[131] Jonathan Rose, Robert J. Francis, David Lewis, and Paul Chow. Architecture of field-programmable gate arrays: The effect of logic functionality on area efficiency. *Proc. IEEE Int. Solid-State Circuits Conf.*, 1990.

[132] Shigeru Sakai, Masaaki Ogino, Ryosuke Shimizu, and Yukihiro Shimogaki. Deposition uniformity control in a commercial scale HTO-CVD reactor. *MATERIALS RESEARCH SOCIETY SYMPOSIUM PROCEEDINGS*, 2007.

[133] Ashoka Sathanur, Antonio Pullini, Giovanni De Micheli, Luca Benini, and Enrico Macii. Physically clustered forward body biasing for variability compensation in nano-meter CMOS design. In *Proc. Design Automation and Test Conf. in Europe*, March 2009.

[134] Takashi Sato, Hiroyuki Ueyama, Noriaki Nakayama, and Kazuya Masu. Determination of optimal polynomial regression function to decompose on-die systematic and random variations. In *Proc. Asia South Pacific Design Automation Conf.*, February 2008.

[135] Pete Sedcole and Peter Y. K. Cheung. Parametric yield in FPGAs due to within-die delay variations: A quantative analysis. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[136] Pete Sedcole and Peter Y. K. Cheung. Parametric yield in FPGAs due to within-die delay variations: a quantitative analysis. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[137] Pete Sedcole, Justin S. Wong, and Peter Y.K. Cheung. Measuring and modeling FPGA clock variability. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2008.

[138] Davood Shamsi, Petros Boufounos, and Farinaz Koushanfar. Post-silicon timing characterization by compressed sensing. In *Proc. Intl. Conf. Computer-Aided Design*, November 2008.

[139] James R. Sheats. *Microlithography Science and Technology*. CRC, 1998.

[140] Xukai Shen, Yu Wang, Rong Luo, and Huazhong Yang. Leakage power reduction through dual vth assignment considering threshold voltage variation. In *ASICON*, October 2007.

[141] Sean X. Shi, Anand Ramalingam, Daifeng Wang, and David Z. Pan. Latch modeling for statistical timing analysis. In *Proc. Design Automation and Test Conf. in Europe*, March 2008.

[142] Jaskirat Singh and Sachin Sapatnekar. Statistical timing analysis with correlated non-gaussian parameters using independent componenet analysis. In *ACM/IEEE International Workshop on Timing Issues*, pages 143–148, Feb. 2006.

[143] Satwant Singh, Jonathan Rose, Paul Chow, and David Lewis. The effect of logic block architecture on FPGA performance. *IEEE Journal of Solid-State Circuits*, 1992.

[144] Amith Singhee and Rob A. Rutenbar. Beyond low-order statistical response surfaces: Latent variable regression for efficient, highly nonlinear fitting. In *Proc. Design Automation Conf.*, June 2007.

[145] Amith Singhee, Sonia Singhal, and Rob A. Rutenbar. Exploiting correlation kernels for efficient handling of intra-die spatial correlation, with application to statistical timing. In *Proc. Design Automation and Test Conf. in Europe*, March 2008.

[146] Amith Singhee, Sonia Singhal, and Rob A. Rutenbar. Practical, fast monte carlo statistical static timing analysis: Why and how. In *Proc. Intl. Conf. Computer-Aided Design*, November 2008.

[147] Satish Sivaswamy and Kia Bazargan. Variation-aware routing for FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2007.

[148] Thomas Skotnicki. Heading for decananometer CMOS - Is navigation among icebergs still a viable strategy? In *Solid-State Device Research Conference*, pages 19–33, September 2000.

[149] Mahapatra Souvik, Saha Dipankar, Varghese Dhanoop, and Bharath P. Kumar. On the generation and recovery of interface traps in MOSFETs subjected to NBTI, FN, and HCI stress. *IEEE. Trans. on Electron Device*, 53, July 2006.

[150] Suresh Srinivasan, Aman Gayasen, Narayanan Vijaykrishnan, and Tim Tuan. Leakage control in FPGA routing fabric. In *Proc. Asia South Pacific Design Automation Conf.*, January 2005.

[151] Ashish Srivastava, Robert Bai, David Blaauw, and Dennis Sylvester. Modeling and analysis of leakage power considering within-die process variations. In *Proc. Intl. Symp. Low Power Electronics and Design*, 2002.

[152] Ashish Srivastava, Saumil S. Shah, Kanak B. Agarwal, Dennis M. Sylvester, David Blaauw, and Stephen Director. Accurate and efficient parametric yield estimation considering correlated variations in leakage power and performance. In *Proc. Design Automation Conf.*, June 2005.

[153] Brian E. Stine., Duane S. Boning, and James E. Chung. Analysis and decomposition of spatial variation in integrated circuit processes and devices. In *Semiconductor Manufacturing, IEEE Transactions on*, pages 24 – 41, Feb. 1997.

[154] Yuuri Sugihara, Yohei Kume, Kazutoshi Kobayashi, and Hidetoshi Onodera. Speed and yield enhancement by track swapping on critical paths utilizing random variations for FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2008.

[155] Shingo Takahashi, Yuki Yoshida, and Shuji Tsukiyama. Gaussian mixture model for statistical timing analysis. In *Proc. Design Automation Conf.*, July 2009.

[156] Li Tao and Yu Zhiping. Statistical analysis of full-chip leakage power considering junction tunneling leakage. In *Proc. Design Automation Conf.*, June 2007.

[157] Shuji Tsukiyama. Toward stochastic design for digital circuits: statistical static timing analysis. In *Proc. Asia South Pacific Design Automation Conf.*, January 2005.

[158] Tim Tuan and Bocheng Lai. Leakage power analysis of a 90nm FPGA. In *Proc. IEEE Custom Integrated Circuits Conf.*, 2003.

[159] SALI Jaydeep V., PATIL S. B., JADKAR S. R., and TAKWALE M. G. Hot-Wire CVD growth simulation for thickness uniformity. In *International Conference on Cat-CVD Process*, 2001.

[160] Rakesh Vattikonda, Wenping Wang, and Yu Cao. Modeling and minimization of pmos nbti effect for robust nanometer design. In *Proc. Design Automation Conf.*, July 2006.

[161] Vineeth Veetil, Dennis Sylvester, and David Blaauw. Efficient monte carlo based incremental statistical timing analysis. In *Proc. Design Automation Conf.*, June 2008.

[162] Chandu Visweswariah. Death, taxes and failing chips. In *Proc. Design Automation Conf.*, June 2003.

[163] Chandu Visweswariah, Kaushik Ravindran, Kerim Kalafala, Steven G. Walker, and Sambasivan Narayan. First-order incremental block-based statistical timing analysis. In *Proc. Design Automation Conf.*, June 2004.

[164] Chandu Visweswariah, Kaushik Ravindran, Kerim Kalafala, Steven G. Walker, Sambasivan Narayan, Daniel K. Beece, Jeff Piaget, Natesan Venkateswaran, and Jeffrey G. Hemmett. First-order incremental block-based statistical timing analysis. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2006.

[165] Ruilin Wang and Cheng Kok Koh. A frequency-domain technique for statistical timing analysis of clock meshes. In *Proc. Intl. Conf. Computer-Aided Design*, November 2007.

[166] Wenping Wang, Vijay Reddy, Anand T. Krishnan, Rakesh Vattikonda, Srikanth Krishnan, and Yu Cao. An integrated modeling paradigm of circuit reliability for 65nm CMOS technology. In *Proc. IEEE Custom Integrated Circuits Conf.*, September 2007.

[167] Ho-Yan Wong, Lerong Cheng, Yan Lin, and Lei He. FPGA device and architecture evaluation considering process variations. In *Proc. Intl. Conf. Computer-Aided Design*, November 2005.

[168] Phoebe Wong, Lerong Cheng, Yan Lin, and Lei He. FPGA device and arhitecture evaluation consdering process variation. In *Proc. Intl. Conf. Computer-Aided Design*, November 2005.

[169] Lin Xie, Azadeh Davoodi, Jun Zhang, and Tai-Hsuan Wu. Adjustment-based modeling for statistical static timing analysis with high dimension of variability. In *Proc. Intl. Conf. Computer-Aided Design*, November 2008.

[170] Xilinx Corporation. Virtex-II 1.5v platform FPGA complete data sheet. July 2002.

[171] Jinjun Xiong, Chandu Visweswariah, and Vladimir Zolotov. Statistical ordering of correlated timing quantities and its application for path ranking. In *Proc. Design Automation Conf.*, July 2009.

[172] Jinjun Xiong, Vladimir Zolotov, and Lei He. Robust extraction of spatial correlation. In *Proc. Intl. Symp. Physical Design*, April 2006.

[173] Jinjun Xiong, Vladimir Zolotov, and Lei He. Robust extraction of spatial correlation. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2007.

[174] A. M. Yaglom. Some classes of random fields in n-Dimensional space, related to stationary random processes. *Theory Probability Applications*, January 1957.

[175] S. Yang. Logic synthesis and optimization benchmarks, version 3.0. Technical report, Microelectronics Center of North Carolina (MCNC), 1991.

[176] Xiaoji Ye, Yaping Zhan, and Peng Li. Statistical leakage power minimization using fast equi-slack shell based optimization. In *Proc. Design Automation Conf.*, 2007.

[177] Guo Yu, Wei Dong, Zhuo Fang, and Peng Li. Statistical static timing analysis considering process variation model uncertainty. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, October 2008.

[178] Yaping Zhan, Andrzej J. Strojwas, Xin Li, and Lawrence T. Pileggi. Correlaiton-aware statistical timing analysis with non-gaussian delay distribution. In *Proc. Design Automation Conf.*, pages 77–82, June 2005. Anaheim, CA.

[179] Yaping Zhan, Andrzej J. Strojwas, David Newmark, and Mahesh Sharma. Generic statistical timing analysis with non-gaussian process parameters. In *Austin Conference on Integrated Systems and Circuits*, May 2006.

[180] Lizheng Zhang, Weijen Chen, Yuhen Hu, John A. Gubner, and Charlie Chung-Ping Chen. Correlation-preserved non-gaussian statistical timing analysis with quadratic timing model. In *Proc. Design Automation Conf.*, pages 83 – 88, June 2005.

[181] Lizheng Zhang, Weijen Chen, Yuhen Hu, John A. Gubner, and Charlie Chung-Ping Chen. Statistical timing analysis with extended pseudo-canonical timing model. In *Proc. Design Automation and Test Conf. in Europe*, pages 952 – 957, March 2005.

[182] Lizheng Zhang, Yuhen Hu, and Charlie Chung-Ping Chen. Block based statistical timing analysis with extended canonical timing model. In *Proc. Asia South Pacific Design Automation Conf.*, January 2005.

[183] Lizheng Zhang, Yuhen Hu, and Chung-Ping Chen. Statistical timing analysis with path reconvergence and spatial correlations. In *Proc. Design Automation and Test Conf. in Europe*, March 2006.

[184] Lizheng Zhang, Jun Shao, and Charlie Chung-Ping Chen. Non-gaussian statistical parameter modeling for SSTA with confidence interval analysis. In *Proc. Intl. Symp. Physical Design*, Aprial 2006.

[185] Qiaolin Zhang, Kameshwar Poolla, and Costas J. Spanos. One step forward from run-to-run critical dimension control: Across-wafer level critical dimension control through lithography and etch process. *Journal of Process Control*, 18(10):937 – 945, 2008. Advanced Process Control for Semiconductor Manufacturing, First IFAC Workshop on Advanced Process Control for Semiconductor Manufacturing.

[186] Qiaolin Zhang, Cherry Tang, Jason Cain, Angela Hui, Tony Hsieh, Nick Maccrae, Bhanwar Singh, Kameshwar Poolla, and Costas J. Spanos. Across-wafer cd uniformity control through lithography and etch process: experimental verification. In *Metrology, Inspection, and Process Control for Microlithography XXI*, April 2007.

[187] Songqing Zhang, Vineet Wason, and Kaustav Banerjee. A probabilistic framework to estimate full-chip subthreshold leakage power distribution considering within-die and die-to-die p-t-v variations. In *Proc. Intl. Symp. Low Power Electronics and Design*, Aug 2004.

[188] Songqing Zhang, Vineet Wason, and Kaustav Banerjee. A probabilistic framework to estimate full-chip subthreshold leakage power distribution considering within-die and die-to-die variations. In *Proc. Intl. Symp. Low Power Electronics and Design*, 2004.

[189] Wangyang Zhang, Wenjian Yu, Zeyi Wang, Zhiping Yu, Rong Jiang, and Jinjun Xiong. An efficient method for chip-level statistical capacitance extraction considering process variations with spatial correlation. In *Proc. Design Automation and Test Conf. in Europe*, March 2008.

[190] Wei Zhao, Yu Cao, Frank Liu, Kanak Agarwal, Dhruva Acharyya, Sani Nassif, and Kevin Nowka. Rigorous extraction of process variations for 65nm CMOS design. In *European Solid-State Circuits Conference*, September 2007.