CAD Techniques for Robust FPGA Design Under Variability

by

Akhilesh Kumar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2010

© Akhilesh Kumar 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The imperfections in the semiconductor fabrication process and uncertainty in operating environment of VLSI circuits have emerged as critical challenges for the semiconductor industry. These are generally termed as process and environment variations, which lead to uncertainty in performance and unreliable operation of the circuits. These problems have been further aggravated in scaled nanometer technologies due to increased process variations and reduced operating voltage.

Several techniques have been proposed recently for designing digital VLSI circuits under variability. However, most of them have targeted ASICs and custom designs. The flexibility of reconfiguration and unknown end application in FPGAs make design under variability different for FPGAs compared to ASICs and custom designs, and the techniques proposed for ASICs and custom designs cannot be directly applied to FPGAs. Very few techniques have been proposed for FPGA design under variability, with varying degrees of improvement in timing/power variability. However, these have not dealt with leveraging CAD, architecture and circuits co-design methodologies for FPGA design under variability, and further, have not accounted for the impact of the variability in V_{dd} arising due to IR drops which is important because the performance of a circuit becomes more sensitive to process parameters as V_{dd} is reduced.

An important design consideration is to minimize the modifications in architecture and circuit to reduce the cost of changing the existing FPGA architecture and circuit. The work in this thesis develops CAD and architecture/circuit design techniques for FPGAs to improve the timing and power yield of FPGA designs under process variations. In the case of environment variations this work focuses on developing design techniques for reducing IR-drops. The focus of this work can be divided into three principal categories, which are, improving timing yield under process variations, improving power yield under process variations and improving the voltage profile in the FPGA power grid.

The work on timing yield improvement implements a Statistical Static Timing Analysis (SSTA) framework to analyze the circuit delay under process variations, such that the statistical distribution of the critical delay can be computed. In this work, the structure of the interconnect is analyzed and it is shown that an optimum number of buffers can be inserted in the interconnect to reduce the variation in circuit delay. Several interconnect architectures are analyzed, under the constraints of the FPGA structure, to find the best architecture which leads to smallest $(\mu + 3\sigma)$ of the critical delay. The placement and routing tools are then enhanced such that the delay variability is accounted for when optimizing the critical delay of the circuit. Results indicate that up to 28% improvement in $(\mu + 3\sigma)$ of the critical delay can be obtained from the proposed methodology.

The work on power yield improvement for FPGAs selects a low power dual-V_{dd} FPGA design as the baseline FPGA architecture for developing power yield enhancement tech-

niques. A low power FPGA architecture is selected because, before applying power yield enhancement techniques to a design, it is necessary that a low power design technique is implemented. The power yield enhancement technique proposed in this work is essentially a CAD technique for placement and dual-V_{dd} assignment. The proposed CAD techniques reduce the correlation between leaking FPGA elements such that the total variability of leakage is reduced and power yield is improved. Results indicate that an average reduction of 15% in leakage variability can be obtained from the proposed methodology, with an average of 7.8% power yield improvement. A mathematical programming technique is also proposed to determine the parameters of the buffers in the interconnect such as the sizes of the transistors and threshold voltage of the transistors, all within constraints, such that the leakage variability is minimized under delay constraints. Results show a reduction of 26% in leakage variability without any delay penalty.

The variability in supply voltage in the power grid occurs due to currents being drawn by the underlying devices. The IR-drops in the power grid leads to reduced speed of the circuit and may also affect the functionality of the design. To reduce IR-drops in the power grid of FPGAs two CAD techniques are proposed in this work. The first technique is an IR-drop aware place and route technique which reduces the high currents drawn in local regions of the chip to reduce the IR-drops. The placement and routing routines are enhanced to incorporate the information about the current distribution profile in the power grid. This is done by redistributing the blocks and nets in such a way so that the spatial distribution of the switching activity profile is more smooth. The CAD techniques result in maximum IR-drop reduction of up to 53% and a reduction in standard deviation of spatial supply voltage distribution by up to 66%.

The second technique is also a CAD technique applied at the clustering stage, where the LUTs are clustered into fixed size logic blocks. The idea here again is to reduce the currents being drawn in a local region. This is achieved by carefully selecting the LUTs to be added to form a cluster. This is because if a cluster has many LUTs with high switching activity nets, then that cluster will experience a large IR-drop. The clustering technique is enhanced such that the new IR-drop aware clustering technique takes into account the switching activities of the nets in the current cluster and the switching activities of the nets connected to potential LUTs that can be added to the current cluster. Results indicate that up to 36% reduction in maximum IR-drop and 27% reduction in standard deviation of the spatial distribution of V_{dd} can be achieved from the proposed techniques.

Acknowledgements

This thesis is dedicated to my parents. My mother, who is my first teacher, and ever full of her untiring support and encouragement, gave me the strength of character for all my pursuits. My father taught me the importance of discipline and hard work with single minded devotion. Amidst all the different pursuits, they taught me how to lead a life of principles and righteousness. And many more things that I have imbibed and learned from them have shaped me as I am today.

Without the invaluable guidance, encouragement and support from my supervisor Prof. Mohab Anis, this thesis would not have been possible. It was his constant guidance that has seen me through this work. Mohab has not only been my supervisor but also a mentor and a friend as well and I have sought his opinion and guidance on many things, which have always been honest, helpful and trustful. He gave me the freedom and flexibility in my research and provided continuous guidance and support in my research pursuits. I am truly grateful and thankful to him for all his help and support and I am glad to say that I have really learned a lot from him during the last six and half years, first as a Master's student and then as a PhD student.

The committee members have also helped a lot in bringing my thesis in the current shape and I am thankful to all of them. Prof. Mark Aagaard, who was also my MASc thesis reader, gave very valuable comments on this thesis which have helped in improving this thesis tremendously. I also had an opportunity to work as a teaching assistant for Prof. Mark Aagaard and learned a lot while working with him. Prof. Manoj Sachdev's comments helped me gain a better insight and improve this thesis. Prof. Yehia Massoud, who was my external examiner, provided me with good comments on my thesis and gave some constructive suggestions for this work. I am also thankful to Prof. Eihab Abdel-Rahman and my co-supervisor Prof Karim Karim for reviewing my thesis.

I am thankful to Prof. Andrew Kennings who has always been very helpful and with whom I have had many discussions on research and teaching. I am thankful to my colleagues at the VLSI research lab with whom I have had not only technical discussions but also shared lighter moments. Javid, Vasudha, Hassan, Ahmed and Mohamed Abu-Rahma have all been a part of my stay at Waterloo. The administrative and technical staff at the Department of Electrical and Computer Engineering have always been very helpful and prompt with their support. Wendy Boles has helped me by going out of the way so many times that I cannot thank her enough. Phil Regier has always promptly helped me with all the technical issues relating to either hardware or software. I would also like thank Nizar Abdallah and Georges Nabaa at Actel Corp., Mountain View, California, for their support during my visit to the company in Fall 2008.

My stay at Waterloo as a graduate student, of almost six and half years, first as a Master's student and then as a PhD student has been made memorable due to so many

people that it is impossible to name all of them. My roommates were always there with all their support and I have shared so much with them. Aashish, Sachin and Sarvagya, with whom I have spent such a wonderful time that I cannot imagine my stay at Waterloo without them. Sachin, always ready with his comments on anything, Sarvagya, ever eager to carry out his pranks and Aashish, quietly observant, are some of the cherished memories. We have together celebrated many occasions and talked about so many things that they have become great friends and some of the closest friends for life. Niraj has been such a close and helpful friend with whom I have shared many ups and downs and had long discussions on many things. Srinath has always been very helpful and a close friend with whom I have spent a lot of wonderful time. Adi has been a great friend with whom I have been part of so many activities. Nikita has been very nice and has cooked wonderful dishes on many occasions. I will also remember the fun times with Shaweta during my earlier part of stay at Waterloo. Guru, Neeraj, Jyotsna, Navneet, Prashant, Abir, Aniket, Bala and many others have also been a part of my experiences at Waterloo. My friends from undergraduate days, Gunjan, Roopesh, Santosh, Amitesh and Nagendra have always supported me throughout my graduate studies.

During the later part of my PhD I met Jalaj, and then Darya, Prachi and Shubham. Jalaj has been my roommate and is someone very close to me and he has always been full of support. Darya is not only a wonderful friend but also became so close to me that she is like a sister to me and has always encouraged me and wished the best for all my efforts. Shubham has been such a nice and close friend through all my joys and sorrows. Prachi became a very good friend during the short time that I knew her at Waterloo. I have had a great time with them during the last year of my PhD. Thanks are due to Anupam and Surbhi and their son Anav who have treated me very warmly and friendly during all my visits to California.

Finally, I cannot appreciate enough the great emotional support that I received from my sisters and cousins. My sisters, Kanchan and Shashi, have always been a source of solace for me. My cousin, Sharad, has always supported, encouraged and helped me in my endeavors. We have discussed so many things in such wonderful terms that talking to him always made me happy. I also remember how my cousins Shailly, Rashmi, Pallavi, Himanshu and Sameer were full of good wishes for my work and eagerly waited for my visits to India. Special word of appreciation is also due to my uncles Indra Mohan and Shekhar from whom I have learned so many things in life, right from my childhood days, and who have always been a source of constant support. I would also like to thank all my family members for their constant support and encouragement in all my endeavors.

Contents

List of Tables **xi**

List of Figures **xiv**

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis Organization 2

2 FPGA Architecture and CAD Overview **4**

2.1 Introduction 4

2.2 FPGA Architecture 4

2.2.1 Logic Block 4

2.2.2 Routing Resources 6

2.2.3 I/O Blocks 8

2.3 CAD Tools 8

2.3.1 Synthesis 9

2.3.2 Placement 10

2.3.3 Routing 12

2.4 VPR and T-VPack 13

3 Background and Related Work **16**

3.1 Introduction 16

3.2 Classification of parameter variations 16

3.2.1 Process variations 17
3.2.2 Voltage Variations 19
3.2.3 Temperature Variations 19
3.3 Modeling of process variations 20
3.3.1 Principal Components Analysis (PCA) Model 20
3.3.2 Quad-Tree Model 21
3.4 Yield of a design 23
3.5 Managing Variability in ASICs 25
3.5.1 Process Variations 25
3.5.2 Supply Voltage Variations 26
3.6 FPGA Design under Variations 27
3.7 Proposed Techniques 29

4 Design for Timing Yield **32**
4.1 Introduction 32
4.2 Statistical Static Timing Analysis 33
4.3 Proposed Technique 36
4.3.1 Impact of Segment Length on Variability 36
4.3.2 Routing Architecture Evaluation 40
4.3.3 Variability-Aware Placement and Routing 43
4.4 Evaluation, Results and Discussions 45
4.4.1 Experimental Details 45
4.4.2 Results and Discussions 45
4.5 Conclusions 51

5 Design for Power Yield **53**
5.1 Introduction 53
5.2 Targeted FPGA Architecture 54
5.2.1 Statistical Power Model 56
5.3 Proposed Methodology 59

5.3.1 Preliminaries 59
5.3.2 Placement Methodology 60
5.3.3 Dual-Vdd Assignment 65
5.4 Evaluation, Results and Discussions 67
5.4.1 Experimental Details 67
5.4.2 Estimating leakage distribution and yield 67
5.4.3 Results and Discussions 69
5.5 Conclusions 73

6 Interconnect Design under Process Variations 74
6.1 Introduction 74
6.2 Impact of Process Variations on Leakage and Delay 75
6.2.1 Process Parameters and Variations 75
6.2.2 Leakage Modeling 76
6.2.3 Delay Modeling 76
6.2.4 Variation Modeling 77
6.3 Proposed Methodology 79
6.3.1 Deterministic Optimization 80
6.3.2 FOSM Based Model: Accounting for Variability 80
6.4 Evaluation, Results and Discussions 81
6.5 Conclusion 83

7 IR-Drop Aware Place and Route 84
7.1 Introduction 84
7.2 Power Grid Model 85
7.3 Proposed Methodology 87
7.3.1 IR-Drop Aware Placement 88
7.3.2 IR-Drop Aware Routing 91
7.4 Experimental Details, Results and Discussions 92
7.4.1 Experimental Details 92
7.4.2 Results and Discussions 93
7.5 Conclusions 99

8 IR-Drop Aware Clustering **100**

8.1 Introduction . . . 100

8.2 Proposed CAD Flow . . . 100

8.3 Proposed Clustering Technique . . . 102

8.4 Results and Discussions . . . 110

8.4.1 Trade-offs and Advantages . . . 112

8.5 Conclusions . . . 116

9 Conclusions and Future Work **117**

9.1 Conclusions . . . 117

9.2 Future Work . . . 118

References **127**

List of Tables

4.1 Routing architecture evaluation 41

4.2 Routing architecture evaluation: % Improvement 42

4.3 Benchmark sizes 46

4.4 Results of Variability-Aware Design for Timing Yield 47

5.1 Results of variability aware placement 70

6.1 Results of variability aware and deterministic optimizations 82

7.1 Results of IR-Drop Aware Design 93

8.1 Results of IR-Drop Aware Clustering 111

8.2 Power savings and runtime for IR-drop aware clustering 115

9.1 Summary of Proposed Techniques 118

List of Figures

2.1 A basic FPGA 5
2.2 Programmable switches used in SRAM-based FPGAs 5
2.3 A 2-input LUT 5
2.4 Basic Logic Element [1] 6
2.5 Cluster based logic block [1] 7
2.6 Island style routing architecture [1] 7
2.7 Basic CAD flow for FPGAs 8
2.8 Synthesis procedure for FPGAs 9
2.9 VPR CAD flow 14

3.1 Variation in Timing and Leakage [2] 17
3.2 A general classification of the parameter variations 18
3.3 Grid Model for PCA 21
3.4 Layer model for representing the spatially correlated process parameters . . 22
3.5 CDF of a circuit delay to determine the yield 24
3.6 PDF of a circuit delay and speed binning applied to improve the yield . . . 24
3.7 Interaction between process variations, environments variations and their impact on power and delay 31

4.1 Merging arrival times 34
4.2 Impact of buffers on the delay variability 38
4.3 Delay variability reduction using shorter segments 39
4.4 Extra wire segments in routing 39

4.5 A section of routing fabric showing different segment lengths 41

4.6 The PDFs for the baseline and variability aware implementations for the benchmark apex4 51

4.7 The CDFs for the baseline and variability aware implementations for the benchmark apex4 52

5.1 Dual-Vdd logic block implementation for power reduction 56

5.2 Example illustrating the impact of placement on leakage pdf. Spatial correlation causes the variance of leakage to increase. 61

5.3 (a) Placement is fairly spread out throughout the FPGA, which leads to reduced leakage variance. (b) Placement more concentrated, higher leakage variance. 63

5.4 Variability-Aware Dual-Vdd assignment technique 65

5.5 Power distribution without and with variability aware placement for alu4 . 71

5.6 Power distribution without and with variability aware placement for seq . . 72

5.7 CDF of power distributions for alu4 for the baseline implementation and variability aware placement 72

6.1 Interconnect in FPGAs having buffered switches evenly spaced. 74

6.2 Schematic of a buffered switch. The SRAM cell controls the pass transistor. 75

6.3 CDFs for the deterministic and the variability-aware optimizations. 83

7.1 Mesh style power grid model 85

7.2 Proposed IR-drop aware Place and Route CAD flow 87

7.3 Current distribution for baseline implementation: *des* 94

7.4 Voltage distribution for baseline implementation: *des* 94

7.5 Current distribution for IR-drop aware implementation: *des* 95

7.6 Voltage distribution for IR-drop aware implementation: *des* 95

7.7 Current distribution for baseline implementation: *s38417* 96

7.8 Current distribution for IR-drop aware implementation: *s38417* 96

7.9 Ratio of the circuit delay for the IR-drop aware and baseline implementation 98

8.1 Proposed IR-drop aware CAD flow 101

8.2 Logic cluster with input and output nets. 103
8.3 Criticality tie breaking technique [3] 105
8.4 Computing the transition density cost during clustering. 106
8.5 Current and voltage distribution for the baseline implementation: *alu4* . . 112
8.6 Current and voltage distribution for the IR-drop aware implementation: *alu4* 112
8.7 Current distributions for the baseline and IR-drop aware implementations: *ex1010* 113
8.8 Ratio of the circuit delay for the IR-drop aware and baseline implementations.114

Chapter 1

Introduction

1.1 Motivation

Integrated circuits are now virtually present in all high-performance computing, communications, and consumer electronics applications. With the increasing complexity of these applications, there has been a growing need to integrate the functions of these applications in smaller packages. To enable this integration, the semiconductor technology is continuously being scaled in the nanometer regime. The high level and complexity of integration along with scaled nanometer technologies present many enormous and critical challenges which must be effectively managed by the designers. In the nanometer technologies, the two most important design challenges cited by the semiconductor industry are the increasing leakage power and the process variations in device characteristics. These two serious issues threaten the life time of silicon technology, and will hinder the development of the microelectronics industry if not addressed. Standby leakage power has been growing at an alarming rate, and constitutes a larger fraction of the total chip power in current and future technology generations. Moreover, the manufacturing process of nanometer transistors and structures has introduced several new sources of variation that have made the control of process (device dimensions) variations more difficult. Additionally, environmental variations are caused by uncertainty in the environmental conditions during the operation of a chip, namely, power supply and temperature fluctuations. Both the process and the environmental variations significantly impact the chips' performance, power dissipation and reliability, and thereby reduce the yield of a design. In recent years several techniques have been proposed for addressing the standby leakage power. The leakage power problem is further aggravated by its strong dependence on process and environmental variations, leading to variation in leakage power which can be as high as 20X [2]. This makes it more difficult for designs to meet a power budget resulting in yield loss. Technology scaling has resulted in circuits which can operate at higher speeds, but this has also made the timing

optimization more complex. Traditionally, timing optimization has been done at all levels, where maximum savings in the clock cycles are obtained at the architecture level design optimization. However, circuit level techniques try to further push this limit to increase the operating clock frequency. The delays of the logic gates and interconnects are strongly dependent on the process and environmental parameters, which makes the clock frequency to have significant variation due to variation in these parameters. Meeting the target clock frequency with these variations is a challenge and results in timing yield loss.

Field Programmable Gate Arrays have emerged as a competitive alternative to Application Specific Integrated Circuits (ASICs) to implement designs and their popularity have grown in recent years. FPGAs are preferred means to implement a design for low to medium volume productions because of significant cost reduction and time-to-market advantages. Hence, FPGAs are now utilized extensively in various communication systems/devices. The number of design starts based on FPGAs is continuously increasing because of advances in FPGA technology and newer architectures with improvement in speed and area. Over the past decade, the management of leakage power in FPGA designs has always been overshadowed by performance improvement and dynamic power minimization techniques. However, with contemporary and future FPGAs being built in nanometer technologies, leakage power cannot be ignored. This is aggravated by the very nature of FPGAs, where typical block utilization is around 60% [4], such that 40% of the FPGA is dissipating standby leakage power! Only recently have FPGA designers started to tackle leakage power [5, 6, 7, 8]. The leakage power problem in FPGAs is further compounded by the fact that FPGAs need more number of transistors per logic gate as compared to custom VLSI designs and ASICs. In addition, process and environmental variations impact FPGAs in these principal areas: timing analysis, leakage power prediction, leakage tolerant design, and reliability. The focus of this research is to develop innovative architectures/circuit/CAD co-design for optimization of timing and leakage yield of FPGAs under process variations and improve the robustness of FPGAs under supply voltage variations due to IR-drops. This would enable FPGA designers to answer the following question: "How to utilize novel FPGA architecture, circuit and design automation techniques cooperatively to maximize the FPGA design yield and improve robustness under power, timing and area constraints?"

1.2 Thesis Organization

This thesis is organized as follows:

Chapter 2 gives an overview of a typical *SRAM-based* FPGA architecture which is targeted in this work and is widely used in industry.

Chapter 3 describes the process and environment variations and its impact on VLSI

circuits. This section also explains the various modeling techniques for analyzing the impacts of the variations and the modeling approach adopted in this work. This is then followed by a discussion of the related work done for FPGAs.

Chapter 4 proposes a CAD and architecture co-design technique for improving the timing yield of FPGA designs under process variations. Results are presented to show the improvement in the timing yield.

Chapter 5 proposes a CAD methodology for improving the power yield of FPGA designs under process variations. The CAD methodology is explained and the results are presented to show the power yield improvement.

Chapter 6 proposes a mathematical programming technique for determining the parameters of the transistors of the buffers, such as the sizes and the threshold voltages, in the FPGA interconnects for reducing leakage variability under delay constraints.

Chapter 7 proposes placement and routing techniques for improving the supply voltage profile in the power grid of FPGAs. The proposed placement and routing techniques are explained along with power grid modeling and the results for the improvement of voltage profile in the power grid are discussed.

Chapter 8 proposes logic clustering technique to improve the supply voltage profile in the power grid of FPGAs. The novel clustering technique is discussed along with the trade-offs and supply voltage profile improvement.

Chapter 9 concludes the work in the thesis and outlines future work.

Chapter 2

FPGA Architecture and CAD Overview

2.1 Introduction

This chapter describes the FPGA architecture that has been adopted for this research. The various elements of the FPGA is described and the CAD tools associated for implementing an application on the FPGA has been discussed. The CAD flow and each of the stages in the CAD flow is explained along with their algorithms.

2.2 FPGA Architecture

A basic FPGA is shown in Fig. 2.1. The FPGA architecture is very regular in structure. It is made up of two main components - logic blocks (CLBs) and routing resources. The logic blocks implement the functionality of the given circuit while the routing resources provide the connectivity for implementing the logic. The logic blocks have the flexibility to connect to the routing resources surrounding them. The logic blocks and the routing resources are configurable, so that they can be programmed to implement any logic. Though many types of architectures have been experimented with, the most popular one is the SRAM based architecture which is described below and has been used in this work [1].

2.2.1 Logic Block

The logic block of the SRAM based FPGA is LUT (look-up-table) based and are composed of basic logic elements (BLE). LUT is an array of SRAM cells to implement a truth table.

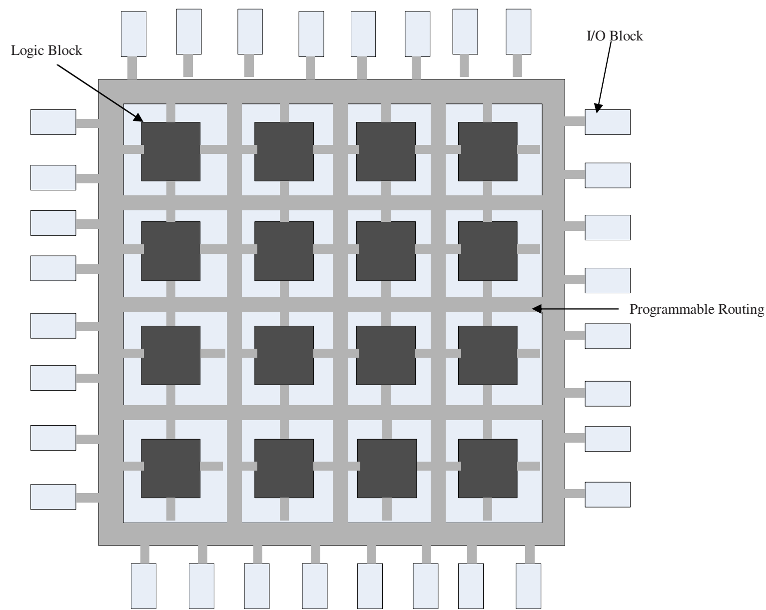


Figure 2.1: A basic FPGA

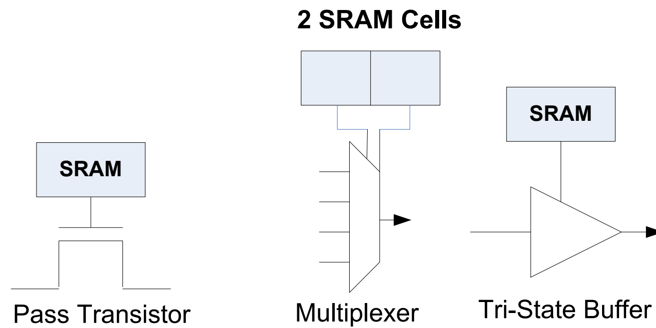


Figure 2.2: Programmable switches used in SRAM-based FPGAs

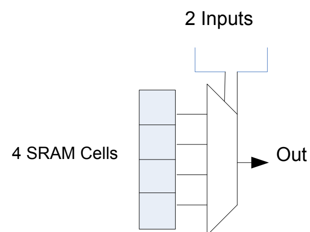


Figure 2.3: A 2-input LUT

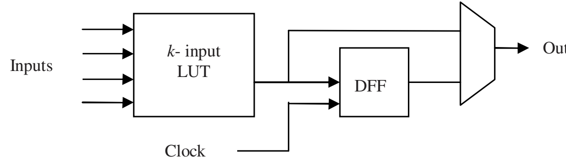


Figure 2.4: Basic Logic Element [1]

Fig. 2.3 shows a two input LUT. It has 4 SRAM cells and a multiplexer to select one of the SRAM cells. The selection is done by the two select signals to the multiplexer, which serve as inputs to the truth-table. Each BLE consists of a k-input LUT, flip-flop and a multiplexer for selecting the output either directly from the output of LUT or the registered output value of the LUT stored in the flip-flop. Fig. 2.4 shows the basic logic element. Previous works have shown that the 4-input LUT is the most optimum size as far as logic density, and utilization of resources are concerned, and this has been widely used. Cluster based logic blocks were investigated in [1] and it was shown that the cluster based logic blocks are better in speed and area. The structure of a cluster based logic block is shown in Fig. 2.5. In the cluster based logic block, the logic block is made up of N BLEs. There are I inputs to the logic block such that each input can connect to all the BLEs. Also the output of each BLE can drive one of the inputs of each of the BLEs. The clock feeds all the BLEs. The work in [1] showed that the logic clusters containing 4 to 10 BLEs achieve good performance. Each subblock is made up of a BLE and the corresponding LUT input multiplexers.

2.2.2 Routing Resources

The routing resources are of various types, but the one used in this work is the island-based architecture. In the island based architecture, the routing resources form a mesh like structure with the horizontal and vertical routing channels. The horizontal and vertical routing channels are connected by switch boxes which are programmable and thus provide the flexibility in making the connections. The logic blocks are connected to the routing channels through the connection boxes which are also programmable. Fig. 2.6 shows the island style routing architecture [1]. The programmable switches used for implementing the interconnections are shown in Fig. 2.2. These programmable switches have SRAM cells which can be programmed to either turn on or turn off the switch. Apart from the logic blocks and the routing resources, the clock distribution is assumed to have a dedicated network. Most of the commercial FPGAs have a structure similar to the one described above or some variant of the above architecture.

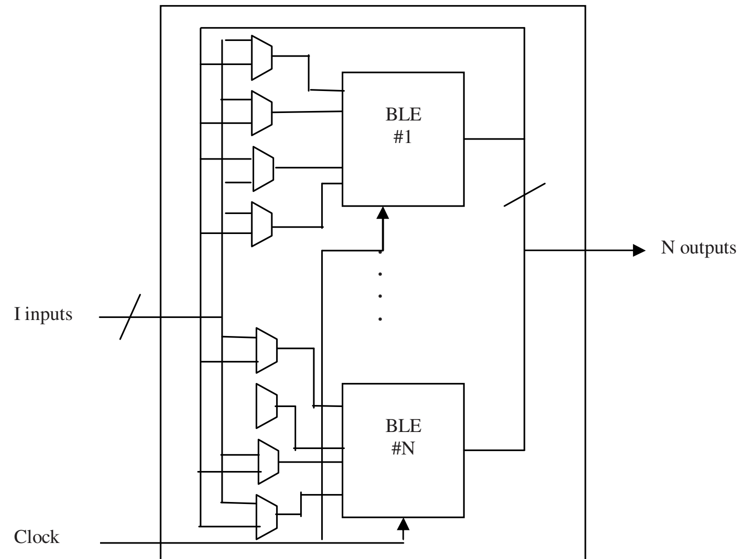


Figure 2.5: Cluster based logic block [1]

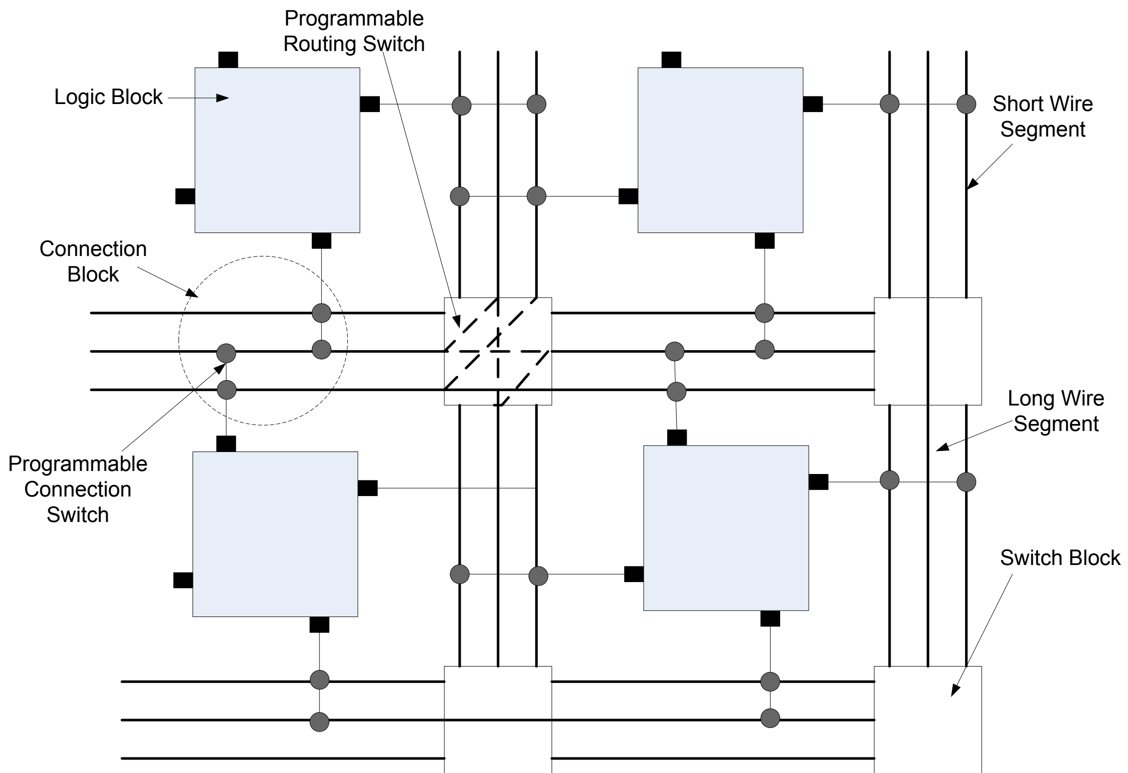


Figure 2.6: Island style routing architecture [1]

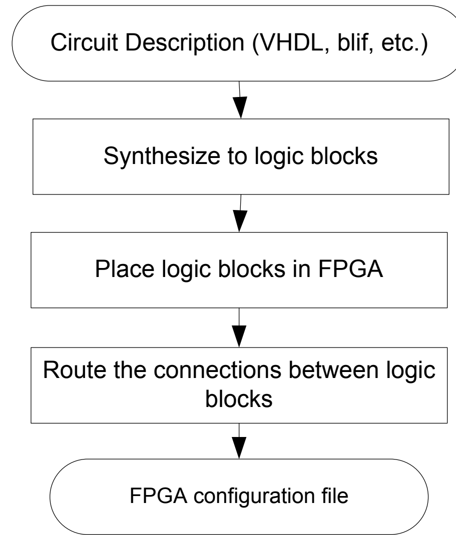


Figure 2.7: Basic CAD flow for FPGAs

2.2.3 I/O Blocks

The I/O blocks are also programmable so that they can be configured either as input or as output, or can be tri-stated.

2.3 CAD Tools

To implement a circuit on the current generation FPGAs, CAD tools are needed which can generate the configuration bits for the SRAM cells of the FPGAs. Usually the circuit description is provided using Verilog, VHDL, SystemC, or other higher level descriptions. The CAD tools for the FPGAs read this input and output a configuration file for programming the FPGA. Fig. 2.7 shows the basic CAD flow for implementing a digital circuit/system on FPGAs [1]. The CAD flow has three main tasks: Synthesis, placement and routing. In the following sections synthesis, placement and routing for FPGAs are explained. Since VPR and T-Vpack have been used in this work, the discussion will be kept in context of these CAD tools. Almost all of the commercial FPGA CAD flows perform the same basic functions of synthesis, placement and routing.

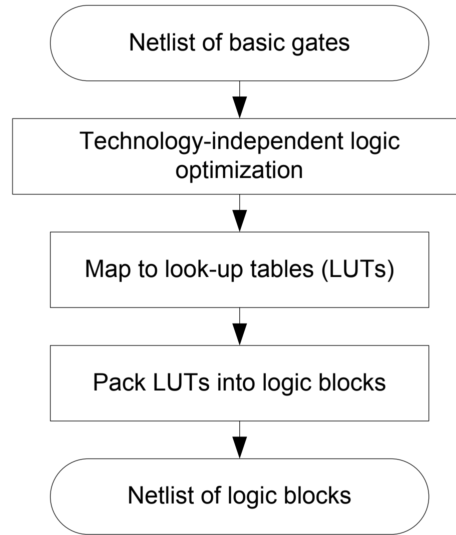


Figure 2.8: Synthesis procedure for FPGAs

2.3.1 Synthesis

The synthesis of a netlist involves conversion of a circuit description, usually in hardware description language (HDL), into a netlist of basic gates. This netlist of basic gates is then converted into a netlist of FPGA logic blocks. Fig. 2.8 shows the steps involved in the synthesis of a circuit description into a netlist of logic blocks.

Technology independent logic optimization involves the removal of redundant logic and simplification of the logic [9, 10]. The optimized netlist is then mapped to look-up tables, which is a process of identifying the logic gates that would go into a LUT [11]. The final step of the synthesis procedure is the clustering of the LUTs and flip-flops (for sequential logic) into logic blocks. The goal here is usually to minimize the number of logic blocks and/or minimize the delay. The work in [12] used a measure of closeness of LUTs to pack them into a cluster to form a logic block.

The work in [1] uses a timing driven logic block packing tool, called T-VPack. The tool targets packing the BLEs into a cluster shown in Fig. 2.5. It needs the parameters such as number of BLEs per cluster, number of inputs per cluster, size of the LUTs, and number of clocks per cluster. The first stage of the packing procedure simply forms the BLEs by packing a register and a LUT together. Initially the packing procedure packs the BLEs greedily into a cluster, followed by a hill climbing phase if the greedy phase is not able to fill the cluster completely.

To enable a timing driven packing, it is necessary to get an estimate of delays through various paths of the netlist. To enable this computation three types of delays are modeled:

delay through a BLE, *LogicDelay*, delay between blocks in the same cluster, *IntraClusterConnectionDelay*, and the delay between blocks in different clusters, *InterClusterConnectionDelay*. The values for these are set as 0.1, 0.1 and 1.0 for *LogicDelay*, *IntraClusterConnectionDelay* and *InterClusterConnectionDelay*, respectively. The *InterClusterConnectionDelay* cannot be determined until the circuit has been implemented on the FPGA. However, these values represent the correct trend of values, and the performance of T-Vpack is not very sensitive to the exact values. The criticality of a connection is defined as

$$ConnectionCriticality(i) = 1 - \frac{slack(i)}{MaxSlack} \tag{2.1}$$

where $MaxSlack$ is the largest slack amongst all the connections in the circuit. A new cluster is created by selecting a seed BLE having the highest criticality amongst the un-clustered BLEs. After the seed BLE has been selected, an attraction function is used to determine the next un-clustered BLE, B, to be added to the current cluster C. The attraction function is given by:

$$\mathit{Attraction}(B) = \alpha.\mathit{Criticality}(B) + (1-\alpha)\left[\frac{Nets(B) \cap Nets(C)}{MaxNets}\right] \tag{2.2}$$

where the first term represents the timing part, and the second term represents the cost of nets shared between the current cluster and the BLE under consideration. Any value of α between 0.4 and 0.8 gives good results. The computation of *Criticality* of a BLE is explained in [1] and also the tie-breaker mechanism used for the case when two or more BLEs have the same criticality. Essentially, the tie-breaker mechanism selects that BLE which reduces the length of the largest number of critical paths.

The hill-climbing phase tries to add more BLEs to the cluster in case it is not full. In this phase adding a BLE to a cluster is allowed even if it leads to more inputs required for the cluster than the maximum allowable. This is done because in some cases the BLE being added might have all its inputs from the BLEs in the current cluster and also might drive the inputs of some of the BLEs in the current cluster. In this case the number of inputs required for the cluster decreases by one. However, this hill climbing phase increases the logic utilization only by 1 - 2% in some of the circuits.

2.3.2 Placement

The work in [1] developed the tool VPR for placement and routing. For placement the FPGA is considered as a set of legal discrete positions at which the logic blocks of the

synthesized netlist can be placed. For placement, the architecture descriptions needed by VPR are:

1. The number of logic block input and output pins.
2. The number of I/O pads that fit into one row or column of the FPGA.
3. The routing channel width (number of tracks in a routing channel).

The placement technique used in VPR is based on simulated annealing [13], which imitates the annealing process used to gradually cool a molten metal to produce high quality metal objects. The simulated annealing works by first starting with an initial random placement by placing the logic blocks randomly on the available locations in the FPGA. The placement then proceeds by making a large number of moves to improve the placement. This is done by selecting a logic block randomly and its new location also randomly. This would produce a change in the cost function, and if the cost function improves, the move is always accepted. However, if the cost function worsens, there is still some probability of acceptance of the move. The probability of acceptance is given by $e^{-\triangle C/T}$, where $\triangle C$ is the change in the cost function and the goal is to decrease the cost function. The T is the temperature parameter and controls the probability of acceptance of the moves which worsen the placement. The temperature is initially set to a high value so that at the beginning of the annealing, virtually all the moves are accepted. The temperature is gradually decreased as the placement improves, such that finally the probability of accepting a bad move is almost negligible. The flexibility of accepting bad moves allows the simulated annealing schedule to overcome the local minima in the cost function.

The VPR sets the initial temperature in the same way as in [14]. The number of moves attempted at each temperature is done as in [15]. It is set to

$$MovesPerTemperature = InnerNum.(N_{blocks})^{4/3} \tag{2.3}$$

where the default value of $InnerNum$ is 10, and N_{blocks} is the number of logic blocks in the netlist. The fraction of moves accepted is kept close to 0.44 for as long as possible, as it yields better results [15]. However, VPR uses a new method of updating the temperature. The VPR computes the new temperature as $T_{new} = \gamma.T_{old}$, where the value of γ depends on the fraction of moves accepted at T_{old}. The idea is to spend maximum time near the temperatures at which large improvements in placement occur. The annealing procedure is not very sensitive to the exact value of γ, if it has the right form, γ is close to 1 if the fraction of moves accepted is close to 0.44, whereas γ is small if the fraction of moves accepted is near 1 or 0. VPR has a timing driven placement and uses a cost function to optimize both the timing and the delay. The complete timing driven placement algorithm

is explained in detail in [16]. The cost function for the timing driven placement developed in [16] is given by

$$\triangle C = \lambda.\frac{\triangle TimingCost}{PreviousTimingCost} + (1-\lambda).\frac{\triangle WiringCost}{PreviousWiringCost} \tag{2.4}$$

where $\triangle TimingCost$ and $\triangle WiringCost$ represent the change in the timing cost and the change in the wiring cost, respectively, due to a move. The simulated annealing procedure is terminated when

$$T < \epsilon.\frac{Cost}{N_{nets}} \tag{2.5}$$

where N_{nets} is the total number of nets in the circuit and the value of ϵ is set as 0.005.

2.3.3 Routing

The routing of the placed netlist, essentially, determines the switches that need to be turned on in the routing resources of the FPGA. The routing algorithm in VPR is based on the Pathfinder algorithm proposed in [17]. The Pathfinder repeatedly rips-up and re-routes every net in the circuit until all congestion is resolved. One routing iteration involves ripping-up and re-routing every net in the circuit. The first routing iteration routes for minimum delay, even if it leads to congestion, or overuse of routing resources. To remove this overuse another routing iteration is performed. The cost of overusing a routing resource is increased after every iteration. This improves the chance of resolving the congestion. At the end of each routing iteration all the nets are completely routed, although with some congestion. Based on this routing, timing analysis can be done to compute the critical path and also the slack of each source sink connection. The timing driven router uses an Elmore delay model to compute the delays of all the connections. The criticality of a connection beteen source of net i and one of its sink j is computed as follows:

$$Crit(i,j) = max\left(\left[MaxCrit - \frac{slack(i,j)}{D_{max}}\right]^{\eta}, 0\right) \tag{2.6}$$

where $slack(i,j)$ is the slack available to the connection and D_{max} is the delay of the critical path. $MaxCrit$ and η are the parameters which determine how the slack impacts the congestion delay trade-off in the cost function. In VPR η is set to 1 and $MaxCrit$ is set to 0.99.

The VPR creates a routing resource graph to describe the FPGA architecture and connectivity information. The wire and the logic block pins of the FPGA are represented

as nodes in the routing resource graph, and the switches are represented as directed edges in the graph. This routing resource graph is used to perform the routing.

The routing of a net is done by starting with a single node in the routing resource graph, corresponding to the source of the net. A wave expansion algorithm is invoked (k-1) times to connect the source to each of the net's (k-1) sinks, in order of the criticality of the sinks, the most critical sink being the first. The cost for using a node n during this expansion is given by:

$$Cost(n) = Crit(i,j).delay(n, topology) + [1 - Crit(i,j)].b(n).h(n).p(n) \qquad (2.7)$$

where $b(n)$, $h(n)$ and $p(n)$ are the base cost, historical congestion, and present congestion as explained in [1]. This procedure is repeated for each of the nets to get the complete routing.

2.4 VPR and T-VPack

The CAD tools used in this work are VPR, for placement and routing, and T-VPack for clustering of the BLEs [1]. VPR is invoked on the command line as follows [18]

$$vpr\ netlist.net\ architecture.arch\ placement.p\ routing.r\ [-options] \qquad (2.8)$$

where *netlist.net* is the circuit description providing the information about the connectivity of the logic blocks, *architecture.arch* is the architecture file which describes the architectural parameters of the FPGA. The output of the final placement is written in *placement.p*, or, if the circuit is only being routed, the placement information is read from the file *placement.p*. The final routing information is written in *routing.p*. VPR has two basic modes of operation. In the first mode, VPR places a circuit on the FPGA and routes it for minimum routing channel width. In the other mode, when the user specifies the routing channel width, VPR attempts to route the circuit only once and if it is un-routable it simply aborts, reporting that the circuit is un-routable. The VPR also provides graphics which shows the actual placement and routing of the logic blocks, along with the routing switches.

T-VPack reads a netlist in the *blif* (Berkeley Logic Interchange Format) format having look-up tables (LUTs) and flip-flops (FFs) and packs these into logic blocks. The output of the T-Vpack is in the *.net* format, which is a netlist of logic blocks. T-VPack is invoked on the command line by

$$t - vpack\ input.blif\ output.net\ [-options] \qquad (2.9)$$

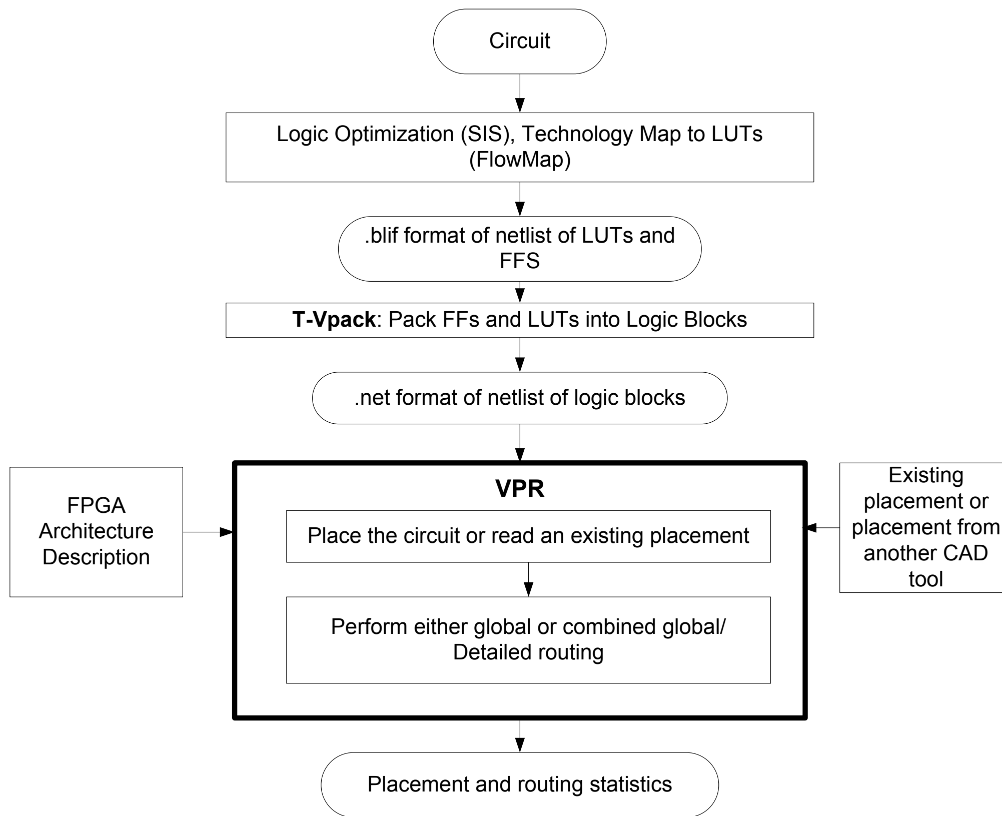


Figure 2.9: VPR CAD flow

where options are used to specify the size of the LUTs, cluster size, inputs per cluster and various optimization options.

The complete VPR CAD flow is shown in Fig. 2.9. SIS [19] is used for logic optimization of the circuit. FlowMap [11] is used for technology-mapping to 4-LUTs and flip-flops. FlowMap produces an output in the *.blif* format. T-VPack packs the netlist into logic blocks and produces an output in the *.net* format. VPR is then used for the placement and routing of the netlist. Other logic optimizers and technology mappers, instead of SIS and FlowMap can also be used in this CAD flow.

Chapter 3

Background and Related Work

3.1 Introduction

The parameter variations affect the performance and the reliability of a circuit, and traditionally guard-banding has been used by providing an excess of safety margin for circuit delay and power. This is done to ensure that the worst case condition in the variations of process, voltage and temperature (PVT) are satisfied. However, with too many process corners it becomes extremely difficult to determine the actual worst case corner, resulting in either too pessimistic or too optimistic designs. Further, designing at worst case corner severely limits the achievable performance-cost trade-off for the circuit.

The variability also results in an increased cost of manufacturing because the chips with lower performance are discarded. The 2006 International Technology Roadmap for Semiconductor (ITRS) identifies the variability as one of the key difficult challenges in the scaled technologies. Fig. 3.1 shows the variation in leakage and frequency of microprocessors in a wafer. It shows that for a 30% variation in frequency a 20X variation in leakage is observed.

3.2 Classification of parameter variations

The parameter variations can be broadly classified into process and environment variations. The process variations relate to all the physical variations caused during the manufacturing process and/or through aging, whereas the environment variations relate to the variations in the operating environment of the chip. Fig. 3.2 shows a general classification of the parameter variations. Although the process parameter variations would in general affect the voltage and temperature variations, the figure does not show that in order to simplify the

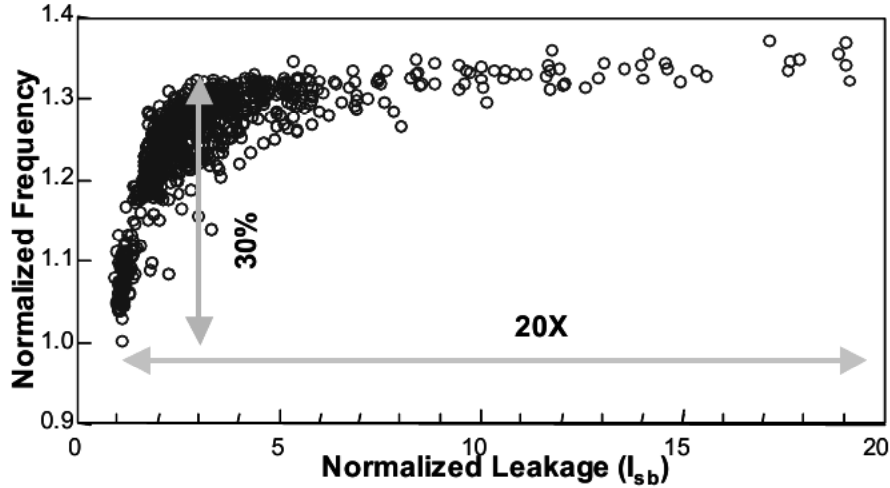


Figure 3.1: Variation in Timing and Leakage [2]

depiction. A detailed figure showing the interaction between process parameters variations and environment variations and their impact on power and timing is shown later in this chapter.

3.2.1 Process variations

The process variations can be classified as die-to-die variations and within-die variations. The die-to-die variations have their origin in lot-to-lot variations, wafer-to-wafer variations and within wafer variations. The die-to-die variations impact the parameters in such a way that the values of the parameters remain the same for all the devices on a single die, but vary across different instances of the chip. Within-die variations cause the parameters to vary across a single die. The systematic variations arise from such phenomena that has a predictable behavior. These variations arise from phenomena such as *Optical Proximity Effect* (OPC) and *Chemical Mechanical Polishing* (CMP). Therefore, theoretically these variations can be modeled and accounted for, by using deterministic analysis. However, since the layout is known at a later design stage and also the modeling is too complicated for the deterministic analysis, it is advantageous to model these variations statistically. The random variations arise from the truly random processes and for these parameters only statistical behavior can be modeled. These variations thus need to be modeled through random variables during the design phase. These random variations can be either independent for each device or can exhibit a spatial correlation. A further classification of the process variations based on their characteristics is as follows [20].

- **Source:** This relates to the variations arising from the sources such as polishing,

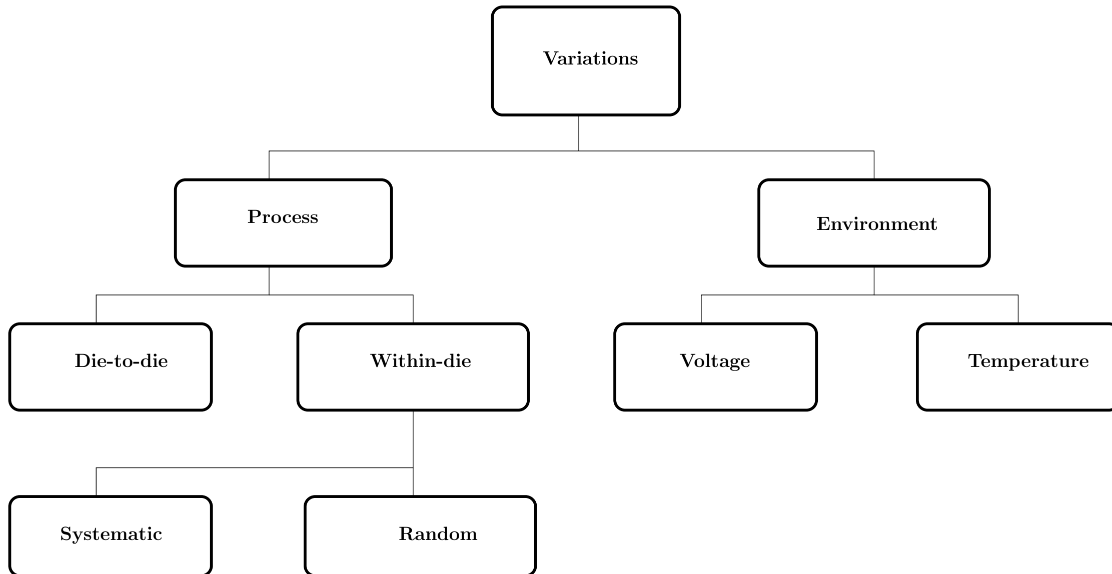


Figure 3.2: A general classification of the parameter variations

lithography, resist, etching, and doping. The non-uniform layout density results in a non-uniform dielectric thickness across the die, after the CMP process. The denser parts of the chip slows down the polishing resulting in the part getting more polished than the other parts. Smaller feature sizes have made lithography variations become more prominent. Also, the stepper lens heating, uneven lens focusing, and aberration lead to variations. The resist coating is non-uniform at the edges due to surface tension and leads to thickness variations. After resist, the etching causes variations due to uneven etching power and density. Since the number of dopant atoms have decreased with scaling, depositing these small number of dopant atoms uniformly for all the transistors is not possible and leads to variations in the dopant concentration.

- **Granularity:** This classifies the variations according to the die-to-die or the within-die variations.

- **Manifestation:** This refers to the systematic and the random variations.

- **Design impact:** The variations in the manufacturing process results in the variations in one or more design parameters such as the channel length, the threshold voltage and the device and the interconnect widths. Further, the channel length variations impact the threshold voltage of the transistors, the channel length variations caused due to factors such as wafer non-uniformity and lens focus and aberration. The width of the devices vary because of polishing or lithography issues.

- **Aging:** Most of the variations are static in nature, i.e., they do not change with the age of the die. However, some parameters might vary with age, such as the negative bias temperature instability (NBTI) effect in PMOS devices which cause the threshold voltage of the PMOS devices to increase with aging.

The process variations classified above typically manifest themselves as variations in the channel length, gate oxide thickness, and threshold voltage fluctuations. These process variations have been modeled in this work.

3.2.2 Voltage Variations

The supply voltage,V_{DD} has been scaling with technology, but a lower limit is set due to reliability concerns. The switching activities and leakage currents in the different parts of the circuit lead to a current distribution in the power supply network which is not uniform. The non-uniformity of the current distributions and its variation with time leads to voltage drops in the supply network across the chip which is both spatial and temporal and in nature. The voltage drops occur due to resistance and inductance of the power supply network. These voltage variations affect the performance of a circuit, and for example, a 10% V_{DD} variation can cause a 20% variation in the delay [20].

3.2.3 Temperature Variations

Elevated temperatures occur in chips during the operation of a chip. The temperature increase is due to the heat generation as a result of power dissipation through switching and leakage. The temperature also gets affected by the ambient temperature of the chip. A higher ambient temperature would decrease the rate of heat flow from the chip to the outside atmosphere, resulting in temperature rise of the chip. The temperature variations are spatial and temporal in nature. The spatial temperature variations are caused due to higher power dissipation in certain parts of the chip as compared to the other parts, resulting in hot spots in the areas with higher power dissipation. The temporal temperature variations are caused due to different periods of activity. During the idle period the temperature of the chip would be lower than during the period in which it is active. The temperature variations not only affect the performance of the chip but can also lead to thermal runaways.

3.3 Modeling of process variations

The modeling of the process variations for computing the delay and the power has been investigated extensively. The process variations are random in nature, so they can be mathematically modeled as random variables. However, the main complexity in their behavior is that they exhibit spatial correlation across a chip. Ignoring these spatial correlations can lead to significant errors in analysis and design. Devices which are closer exhibit stronger correlation than the devices which are far apart. Early on, the analog designers used the Pelgrom model for computing the variation between different devices [21]. The Pelgrom model states that for a group of equally designed MOSFET devices, the variance (or mismatch) can be expressed as a function of their size and the distance between them. For example, the threshold voltage variance can be written as,

$$\sigma^2(V_{T0}) = \frac{A_{VT0}^2}{W.L} + S_{VT0}^2.D^2, \tag{3.1}$$

where A_{VT0} and S_{VT0} are technology-dependent coefficients, W and L are the device dimensions, and D is the distance between the devices. Although the Pelgrom model can give a good insight into the nature of variations, it is difficult to scale it in for a design with large number of gates. In such a scenario it is important to account for the impact of multiple sources on a single location to analyze the overall effect of variations.

Two most popular methods for modeling spatially correlated process parameter variations are the *principal components based model*, and the *quad-tree model*. In the former, after obtaining the correlation information, Principal Component Analysis (PCA) is applied. The PCA is used to develop a set of uncorrelated random variables from a set of correlated random variables [22]. The quad-tree model proposed in [23], models the process variations by diving the chip into hierarchical levels and is adopted in this work. The two models are discussed in the following subsections.

3.3.1 Principal Components Analysis (PCA) Model

In the PCA model the spatial correlation is modeled by dividing the chip into n grids, such that each grid is associated with a principal component. Each of the n principal components are independent normal random variables with zero mean and unit variance. The grid model for the spatial correlation is shown in Fig. 3.3. The spatial correlation matrix is based on distance, location, orientation and other factors, and would give a correlation value for each grid with all the grids on the chip.

The value of a process parameter, for example the channel length, of the gate i, is expressed as,

$$L_{g,i} = L_{nom,i} + \sum_j \alpha_{i,j} \triangle L_j, \tag{3.2}$$

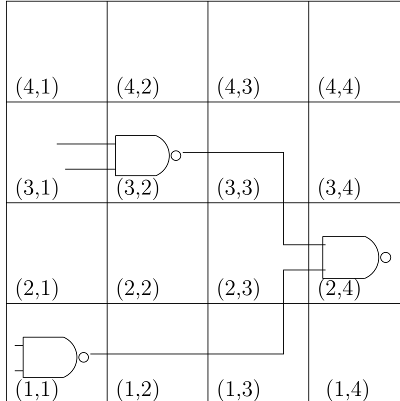


Figure 3.3: Grid Model for PCA

where $\triangle L_j$ is the jth component and $\alpha_{i,j} = \sigma_i.v_{i,j}.\sqrt{\lambda_j}$. The σ_i is the standard deviation for grid i, v_{ij} is the ith element in the jth eigenvector of the correlation matrix and λ_j is the jth eigenvalue of the correlation matrix [22]. The sensitivity matrix, P, for the PCA model can be written as,

$$P = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \dots & \alpha_{1,m} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \dots & \alpha_{2,m} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \dots & \alpha_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n,1} & \alpha_{n,2} & \alpha_{n,3} & \dots & \alpha_{n,m} \end{pmatrix} \tag{3.3}$$

where each grid of the Fig. 3.3 is associated with one column and one row. Once the spatially correlated parameters have been decomposed into a set of independent principal components, any analysis can be easily performed.

3.3.2 Quad-Tree Model

In this work, the quad-tree model is selected for modeling the process parameter variations. The process parameters, such as the gate lengths of two closely placed transistors have almost identical variation, which means that one random variable can be used for modeling the gate lengths of both transistors. However, the gate lengths of the transistors which are far apart need to be modeled with different random variables with spatial correlation. The quad-tree model accounts for spatial correlation through modeling the variations at several hierarchical levels.

A process parameter, such as channel length is represented as sum of its nominal value L_{nom}, inter-die variation $\triangle L_{inter}$, intra-die variation $\triangle L_{intra}$, and random variation

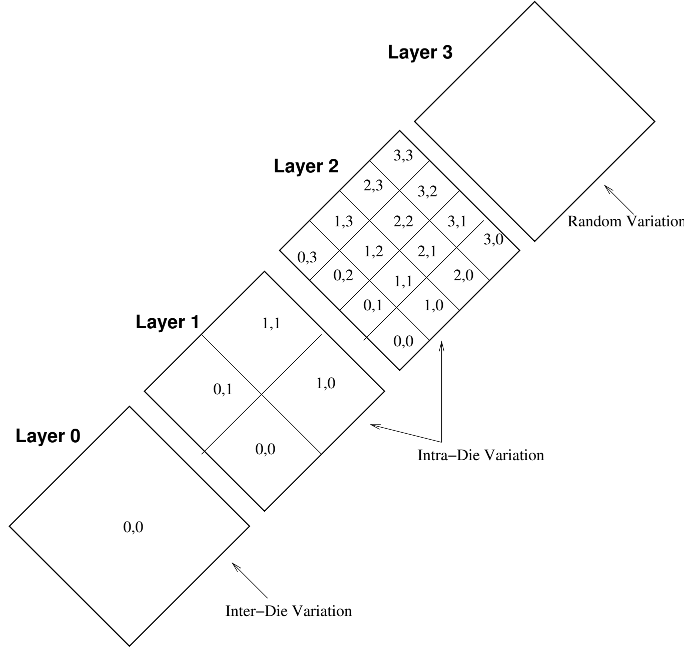


Figure 3.4: Layer model for representing the spatially correlated process parameters

$\triangle L_{random}$.

$$L_{eff} = L_{nom} + \triangle L_{inter} + \triangle L_{intra} + \triangle L_{random}. \quad (3.4)$$

In the quad-tree model a process parameter for the complete chip is modeled at several hierarchical levels. The entire chip is divided into several levels with each level modeling a component of the total variation. Starting from the 0th level, each level (i^{th} level) has 4^i equal sized partitions as denoted in Fig. 3.4. Finally, a random level with only one grid becomes the last level in the model. For each process parameter, there is a random variable associated with each of the partitions of each level. All such random variables are independent and identically distributed. To model the process variations for a logic gate, the partition in which the logic gate lies in each of the levels is determined. These variations are then added to obtain the total variation in a process parameter for a logic gate. The spatial correlation of a process parameter between two logic gates is accounted for, by the number of common partitions they share across the different levels. Fig. 3.4 illustrates the modeling of the variations for a chip with three levels. The level 0 represents the inter-die variations because it is common for all the logic gates of the chip. Levels 1 and 2 represent the intra-die variations. Consider a logic gate lying at the top right side of the die, i.e., at the grid location (3,0) in the level 2, and another logic gate lying adjacent to it at the grid location (2,0) in the level 2. The total channel lengths for these logic gates

are expressed:

$$L_{eff_{gate1}} = L_{nom} + L_{eff_{0,(0,0)}} + L_{eff_{1,(2,0)}} + L_{eff_{2,(3,0)}} + L_{random} \tag{3.5}$$

$$L_{eff_{gate2}} = L_{nom} + L_{eff_{0,(0,0)}} + L_{eff_{1,(2,0)}} + L_{eff_{2,(2,0)}} + L_{random}, \tag{3.6}$$

where $L_{eff_{i,(j,k)}}$ represents the random variable for the channel length associated with the level i and the partition (j,k). L_{random} represents the independent random variation. It can be seen that the two logic gates share the [0,(0,0)] and the [1,(2,0)] partitions. This sharing incorporates the spatial correlation in the model, implying that more the number of grids shared, higher is the corresponding spatial correlation. Increasing the number of levels for modeling can improve the accuracy of the model at the expense of the run time. Since the random variables associated with the different partitions are independent within and across the levels, the computation of the means and the variances of the leakage or delay are easier. The total variation of a process parameter is distributed across the different levels in accordance with their spatial correlation. Specifically, the total variance for a process parameter is written as $\sigma^2 = \sum_{i=0}^{n} \sigma_i^2$, where n is the total number of levels and σ_i is the standard deviation of the corresponding random variable for the level i. The quad-tree model is verified by the actual measurements in [24].

3.4 Yield of a design

The yield of a design is defined as the number of chips that meet the target performance criterion. The performance parameter is typically the circuit delay or the power dissipation (assuming that the functionality of the circuit is correct). Under parameter variations, the performance characteristics no longer remain deterministic, but are modeled as random variables. The yield of a design for a performance criterion is defined as the CDF of the random variable representing the performance characteristic. For example, given a PDF, $f(T_d)$, of the circuit delay, T_d, the yield at the target delay is calculated by computing the CDF, $F(T_d)$, and is given by the equation 3.7.

$$Yield = F(T_d < TargetDelay) = \int_0^{TargetDelay} f(T_d) dT_d \tag{3.7}$$

The yield point represents the probability of a chip meeting the target delay. Fig. 3.5 shows the yield point for a target delay of 8.3ns for a circuit implemented on an FPGA. In a manufacturing process fabricating a large volume of chips, 90% of the chips will meet the target delay.

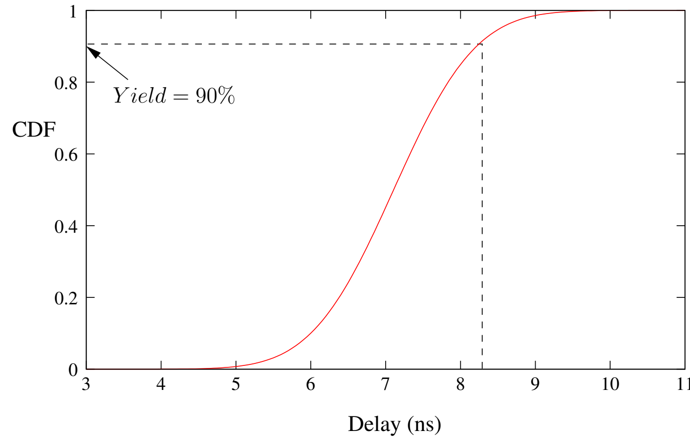


Figure 3.5: CDF of a circuit delay to determine the yield

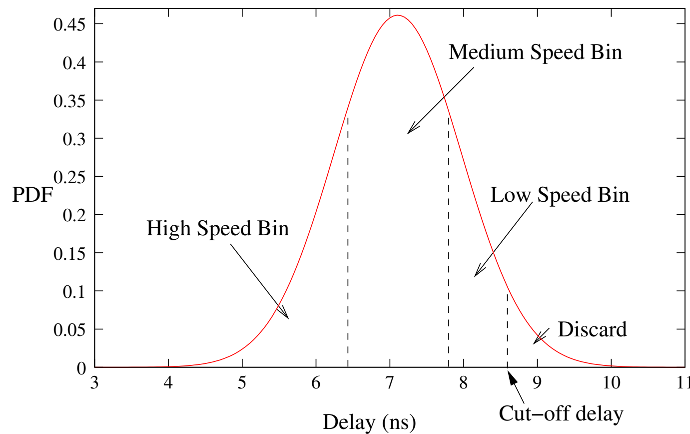


Figure 3.6: PDF of a circuit delay and speed binning applied to improve the yield

Another technique commonly used for improving the yield of a design is *binning*, which is worthwhile to point out in this discussion. An example of speed binning is shown in Fig. 3.6. The figure shows the PDF of the delay of a circuit implemented on an FPGA. The PDF is divided into three bins, the high speed bin for chips with lower circuit delays, the medium speed bin for chips with higher circuit delays, and the low speed bin for chips with highest circuit delay. Any chip having a delay larger than the cut-off delay is discarded, leading to yield loss. The speed binning is done to improve the profitability from selling the chips. This is done by selling the chips in different bins at different prices, with the chips from the lowest speed bin being sold at the least price, and the chips from the higher speed bins being sold at a higher price.

3.5 Managing Variability in ASICs

3.5.1 Process Variations

Traditionally, process corners have been used for analyzing designs to meet the targeted performance, power and other design considerations at the best, nominal and worst case process corners. However, this may lead to pessimistic or optimistic designs. Moreover, it is very difficult to determine whether a particular process corner is indeed a best, nominal or worst case corner, because of the significant increase in the number of varying process parameters and operating conditions with technology scaling. Therefore, to design VLSI circuits under process variations, statistical techniques need to be adopted.

Several research works have proposed techniques for designing and analyzing VLSI circuits and systems under process variations for timing and power optimization [25, 26, 27, 28, 29, 30, 31, 32]. These papers have proposed techniques for statistical timing analysis and/or statistical power analysis and their optimization. The work in [31] proposed a design technique to reduce the build up of a *wall* of critical paths to make the circuit more robust to variations. It argues that the deterministic optimizers builds up many critical paths to reduce the critical delay by even a small amount. Under variations these large number of critical paths will have a higher probability to exceed the critical delay and thus cause an yield loss. In [32], a dual-Vt and gate sizing algorithm was proposed which considered the delay and leakage variability. Results demonstrate that based on such a method, a leakage saving of 15%-35% can be obtained compared to the deterministic algorithms while accounting for the leakage variability. The results were reported for the 95th and 99th percentile of the leakage power distribution. The authors in [25] proposed an Adaptive Body Biasing technique for mitigating the impact of variability on performance and leakage. A statistically-aware clustering technique is proposed to cluster the logic gates such that a same body bias can be applied to a cluster. The results show a 38%-71% improvement in the leakage power compared to a dual-Vt implementation, and the delay variability was reduced by 2-9X. The work in [26] targets parametric yield improvement of the design under leakage and timing constraints. The gate sizing is performed with incremental computation of the yield gradient using a heuristic proposed in the paper. A non-linear optimizer is then used to perform the optimization. The results show that up to 40% yield improvement can be obtained compared to a deterministically optimized circuit. In [27], the authors propose a joint design-time and post-silicon optimization for improving the parametric yield for leakage power. This is achieved through a robust linear programming to obtain an optimal body-bias policy, once the uncertain variables are known. An improvement of 5%-35% in leakage power is obtained from this methodology. The authors in [28] propose a statistical gate sizing method based on the sensitivity computation. A new objective function is proposed for optimizing the circuit and an algorithm is developed for computing

the sensitivity. A pruning and statistical slack based approach is used, which shows an improvement of 16% in the 99-percentile circuit delay and a 31% improvement in the standard deviation for the same circuit area. A new approach is proposed in [30] for speed binning where instead of optimizing for yield, total profit maximization is done. This is based on the fact that chips in different bins are sold at different prices. Again, a sensitivity based gate sizing algorithm is proposed. An algorithm is proposed to determine the optimal bin boundaries. A joint sizing and optimal bin boundaries determination approach is also investigated. Results show that a 36% improvement in profit can be obtained from the proposed approach. In [29], a gate level method is proposed to estimate the parametric yield of a design under leakage and timing constraints by finding a joint PDF of leakage and timing. This is necessary because the leakage and timing are correlated and the assumption of their independence will lead to errors in joint yield computation.

However, these papers have proposed techniques for custom VLSI designs and ASICs, and cannot be directly applied to FPGAs because of the intrinsic nature of programmability of FPGAs. Another challenge in FPGAs is that the circuits which are finally mapped to the FPGAs are not known and the resources for FPGAs are fixed once they are fabricated, thus limiting the flexibility for design optimization.

3.5.2 Supply Voltage Variations

Technology scaling has led to scaled wires and increased packing density of logic gates. Scaling of wires increase their resistance proportionately, and high packing density of logic gates cause more current to be drawn in a local area, which result in increased IR-drops in the power supply network of a chip. Additionally, the currents are usually distributed non-uniformly in the chip leading to spatial non-uniformity in IR-drops. The IR-drops cause the logic gates to operate at a voltage lower than the full V_{dd}, which affects not only the switching speed of the gates, but can also affect the correct operation of logic and clock skew [33, 34]. Thus, it is very critical to develop efficient design techniques for robust power grids which minimize IR-drops in the power network.

Several techniques have been proposed for designing a reliable power grid for a chip. Most of these techniques relate to sizing the wires of the power grid [35, 36], topology optimization [37, 38], and decoupling capacitances [39, 40]. Another technique proposed in [41] involves determining the pitches and sizes of the wires in a non-uniform power grid.

The techniques proposed in these papers are for custom VLSI design and/or ASICs and cannot be directly applied to FPGAs because of the programmable nature of the FPGAs and the end application to be implemented on the FPGA is unknown. This necessitates developing CAD techniques which reduce IR-drops while mapping the application to the FPGA.

3.6 FPGA Design under Variations

The major design challenges for FPGAs have been area and power in earlier technologies. However, in scaled nanometer technologies it is critical for the FPGA industry to address the design challenges stemming from the process and environment variations. Very few published work exists targeting the design of FPGAs under variations [42, 43, 44, 45, 46, 47, 48, 49].

The work in [49] includes process variations by creating a *variation map* for each FPGA chip, and then a detailed placement is performed for optimizing the timing. The variation map describes the detailed variations in the devices and interconnects, after the fabrication of the chip. The variation map is obtained by applying test circuits for each chip before mapping an application to the FPGA. A variation aware placement is developed for considering the variation in the critical delay of a circuit. This is performed in a deterministic manner because the variation map for a chip gives the *actual* values of the process parameters on a chip. The reported results indicate a performance improvement, by using the proposed chip-wise variation aware placement, of 5.3%. However, the authors do not provide details for obtaining the variation map, and generating a variation map for each chip is expensive.

In [43], a placement algorithm is described for improving the timing yield of the FPGAs. The delay of a circuit is modeled as a first order canonical form of the process variations. The guard banding and speed binning is discussed and the reduction of yield due to within-die variation and correlation is explained (with speed binning). The authors propose a statistical placement methodology to reduce the yield loss. Versatile Place and Route (VPR) [1] was augmented with the statistical placement methodology, which performs SSTA at each placement iteration, and therefore, attempts to optimize the statistical delay instead of the deterministic delay of the FPGA. Using this methodology, the authors report a reduction in the yield loss of 5X with the guard banding and 25X with the speed binning. The authors used a 10% global and 10% local random variation in the channel length and the threshold voltage. However, it is not related how their methodology performs in the presence of within-die variations with spatial correlations which becomes important in scaled technologies.

Again, a variation aware placement technique is suggested for leakage power and timing in [48]. A Block Discarding Policy scheme is provided to optimize the placement under process variations for timing and leakage. The policy works on the principle of selecting a block on the FPGA for the placement based on leakage and delay values. Then a leakage and a delay thresholds are chosen for such a selection methodology. Although, a threshold voltage variation is considered, the spatial correlations are not accounted for. The work is based on the assumption that for an FPGA chip, the exact leakage and delay values for all the blocks are available (i.e., with variations), and therefore, these leakage and delay values

can be used for optimizing the placement. Also, the VPR is modified such that each of the blocks in the VPR placement routine has the leakage and delay values with variations. A leakage cost function is used in the placement cost function, but its mathematical form is not provided in the paper. The results show a 14% saving in leakage by using the scheme, and a 10% improvement in the clock frequency. This improvement in clock frequency is observed by simply providing the delays with the variations in the VPR framework. There is no statistical analysis of the delay and leakage, and this work depends on obtaining accurate values of the delay and leakage for each block and each FPGA chip, which is computationally very expensive.

In [45], statistical leakage and timing models for computing the leakage and timing yield for FPGAs are developed. The leakage model under variations is empirically derived, whereas the timing variability model is obtained from SPICE simulations for the basic circuit elements of the FPGA. Delay points are sampled from SPICE simulations and the delay PDF is directly constructed from the PDF of channel length. Although leakage varies with channel length, gate oxide thickness and threshold voltage, the delay is modeled to vary only with the channel length of the devices. For computing the leakage yield, a baseline architecture is assumed with the target leakage set as the nominal leakage with an offset of 30%. The authors evaluate various combinations of logic block cluster sizes and LUT sizes in the FPGAs for their yield and the simulations indicate that some combinations can result in an improved yield. However, the authors do not consider the spatial correlation of the intra-die variations of the process parameters. It is analyzed how the yield can change if the supply voltage and the threshold voltage of the devices are changed in the FPGA. Additionally, the timing yield is analyzed for different combinations of the cluster and the LUT sizes. No CAD technique is proposed to enhance the timing and leakage yield in FPGAs.

The researchers in [44] introduce an adaptive body biasing technique for the FPGAs to compensate for the process variations. Each FPGA chip is proposed to have a characterizer to measure the variations in the threshold voltage for each block by measuring the delay through the block. The body biasing can be accomplished according to the threshold voltage fluctuation for the block. Each tile has the extra configuration bits (2-3 SRAM bits) for determining the applied biasing, and a circuitry to apply the bias. The results indicate a 30.3% decrease in the standard deviation of delay for three levels of body biasing, and 3.3X reduction in the standard deviation of delay for seven levels of body biasing can be achieved. The standard deviation of the leakage is reduced by 78% for three levels of body biasing, and by 18.8X for seven levels of body biasing. However, the area overhead for each tile is 1.6%, whereas the area overhead of the characterizer is nine FPGA tiles. Further, there will be an additional cost due to the triple well process for designing the FPGAs.

The work in [47] uses the FPGA and ASIC CAD flows for a set of benchmarks to show

the impact of process variations on the FPGA and the ASIC implementations, for the same circuits. One of the conclusions is that, although the impact of the process variations is more pronounced in the case of the ASICs, its impact on the FPGAs cannot be ignored. It also proposes a variation aware routing methodology to improve the timing yield of the FPGAs, with a 3.95% improvement in the delay for the same yield, from a deterministic router with a 3σ guard banding for the delays of circuit elements. However, no architecture evaluations were done for routing resources, which is an important design criterion, given that the routing delay dominates the total delay of the circuit. Further, no statistical placement optimization was considered which is also a critical factor governing the overall circuit delay and thus the timing yield.

In [42], the authors propose statistical techniques for the major steps in the FPGA CAD flow, i.e., logic clustering, placement, and routing. The stochastic clustering technique proposed takes into account the uncertainty in the routing interconnect at the clustering level, where no information about the routing is available. The uncertainty value in the interconnect delay arising due to the uncertainty in the interconnect usage is modeled as a random variable apart from the process parameters. The actual values of these variables are determined heuristically. The statistical information is embedded in the clustering phase through the clustering cost function, by incorporating the statistical criticality of a basic logic element. Similarly, during the placement and the routing phases the statistical criticality as defined in [42], is computed for the cost functions. Again, no routing architecture evaluations are done in this work.

No known work exists for dealing with supply voltage variations in the power grid of FPGAs, apart from the work proposed in this thesis and discussed in chapters 7 and 8.

3.7 Proposed Techniques

These challenges in designing FPGAs under process variations in nanometer technologies, motivate the following, which are the contributions of this work.

- **Timing and power yield improvement:** Earlier works for FPGAs have not accounted for the intra-die variation with spatial correlations while performing the optimization. The work in this thesis is based on a model suitable for both intra and inter-die process variations, thus making it more flexible. The process parameters under consideration in this work are the channel length, substrate dopants, gate oxide thickness. However, the model developed is generic to incorporate any number of process parameter variations. The timing yield improvement technique proposed in this work develops architecture enhancements along with CAD tool improvements

to increase the timing yield of the design. The power yield enhancement design technique is essentially a CAD technique used for a low power FPGA. The power yield improvement targets reduction in leakage variability leading to improvement in total power yield.

- **CAD for IR-drop reduction in FPGA power grid:** This work proposes an enhanced IR-drop aware place and route technique to improve the voltage profile of the power grid in FPGAs by reducing IR-drops and spatial variance of the supply voltage. A faster technique at an earlier stage in design flow, at the clustering stage is also proposed. The IR-drop aware clustering also improves the voltage profile of the power grid in FPGAs in a similar way. The trade-offs associated with both these techniques are also analyzed.

Fig. 3.7 shows the interaction between the process variations and environment variations and their impact on power and delay. This work targets power and timing yield optimization under process variations and improving the supply voltage profile. It does not intend to analyze and optimize impact of supply voltage variations on delay and power, and temperature variations. The figure shows various *links* indicating how different factors affect each other and also power and delay. It also shows the *links* that this work explores.

One of the objectives of this work was to minimize the changes to the existing architectures and circuits such that most of the modifications are done to CAD tools for FPGAs. This will prevent changes to the FPGA architectures and circuits which can be costly. As will be observed in the later chapters which describe the work, the changes to circuits and architectures are minimal and most of the modifications are proposed for the CAD tools for FPGAs. Additionally, the proposed changes to the circuits and the architecture does not alter the fundamental fabric of the generic SRAM-based FPGA.

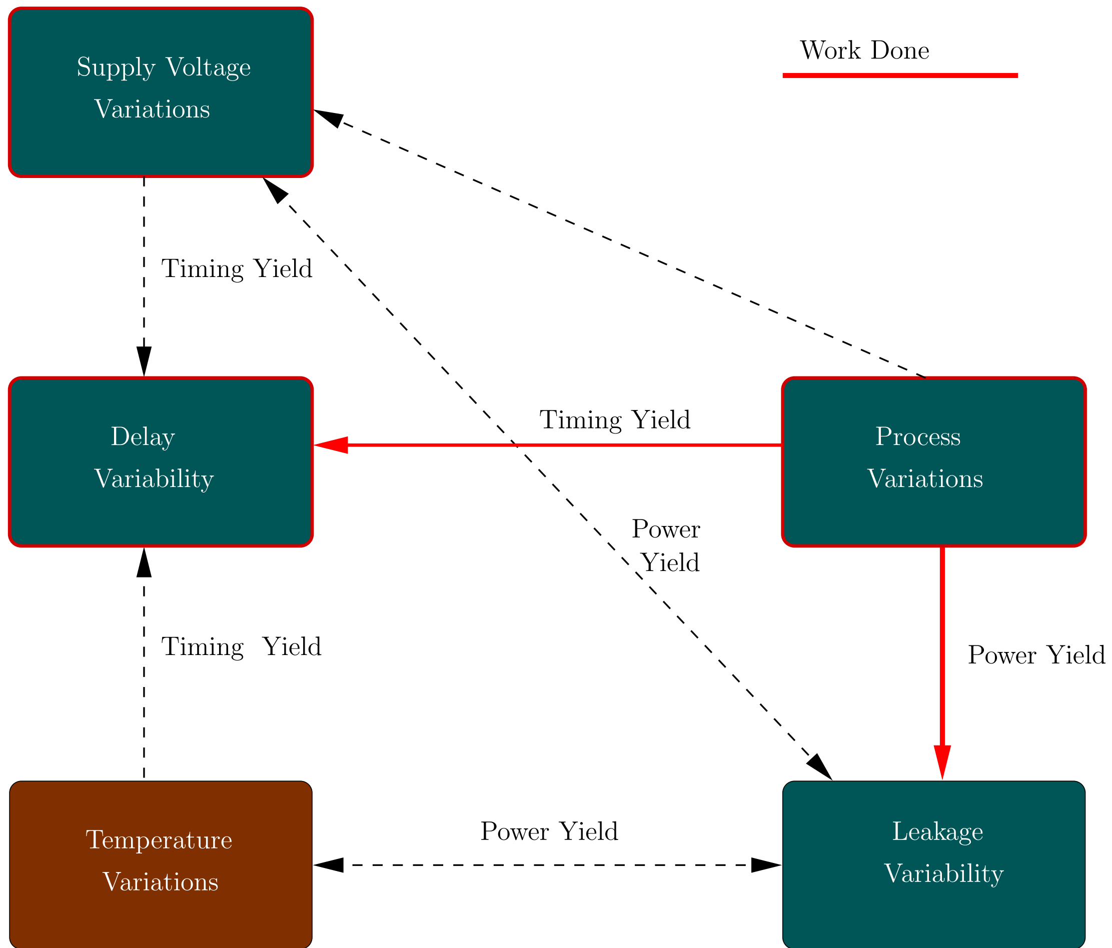


Figure 3.7: Interaction between process variations, environments variations and their impact on power and delay

Chapter 4

Design for Timing Yield

4.1 Introduction

This chapter explains the work on timing yield improvement for FPGAs. The timing yield of a VLSI design is defined as the fraction of chips which meets circuit delay target, out of all the fabricated chips. This means that only those chips which meet the target circuit delay can run at the desired frequency of operation. The rest of the chips which cannot run at the target frequency of operation are then discarded and results in yield loss. The main ideas and results proposed and discussed in this chapter are as follows:

1. **Routing architecture enhancements for timing yield with SSTA:** Earlier works on FPGAs have not accounted for the intra-die variations with spatial correlations, while the optimization is performed. This work is based on a model suitable for both the intra and the inter-die process variations, thus introducing flexibility. The process parameters under consideration in this work are the channel length, the substrate dopants, and the gate oxide. An analysis of the routing architectures and an evaluation of these routing architectures for their suitability under process variations is performed. It is imperative that the routing architecture design is considered, because a principal part of the total delay is due to the routing segments.

2. **Variability aware CAD for improved timing yield:** The placement and routing tools are enhanced to enable a statistical optimization. A variability-aware place and route technique is proposed which accounts for the timing variability through statistical delay information available during the place and route phases.

This chapter first discusses the statistical static timing analysis (SSTA) technique adopted in this work and then explains the proposed design techniques from improving the timing yield of FPGAs by reducing delay variability [50].

4.2 Statistical Static Timing Analysis

Recently, many published works have proposed techniques for the SSTA, such as [23, 51, 22, 52]. The main ideas behind any SSTA technique are, (1) modeling the process parameter variations along with their correlations, and (2) computing the statistical critical delay, which can be either the block-based approach or the path-based approach. The work in [53] is a high level model for 3-D circuits which first starts with the assumption of knowing the number of critical paths in the circuit and is based on the work in [54] for 2-D circuits. Also, it does not consider the spatial correlation of process parameters which leads to additional complexity. The SSTA technique adopted in this work takes into account the spatial correlations of the process parameters and propagates the delays without any assumption of knowing the critical paths.

In this work the SSTA technique proposed in [23] is adopted for computing the critical delay of the circuit. The inter-die and intra-die process parameter variations are modeled as discussed in the Section 3.3. The two main steps performed by the STA (and SSTA) are the propagation and the merging of the arrival times at the different circuit nodes. In the propagation step the input arrival time at a logic gate is propagated to its output by adding the delay of the gate, whereas in the merging step the maximum of all the arrival times at the output of the gate is computed. However, in the case of the SSTA, the arrival times are random variables and the Cumulative Distribution Function (CDF) or the Probability Density Function (PDF) of the arrival times are propagated. The complexity of the SSTA arises from the correlation between the arrival times at the same logic gate or at different logic gates. This correlation is due to two factors: the re-convergent fanouts and the spatial correlation of the process parameters. It has been shown in [55] that ignoring the correlation due to the re-convergent fanouts, leads to an upper bound in the statistical delay analysis, and as a result is conservative. However, the spatial correlation due to the process parameters cannot be ignored [23].

The arrival time is modeled as follows:

$$a = A_{nom} + \sum_i s_i.p_i + A_{random} \tag{4.1}$$

where A_{nom} is the arrival time at the nominal process parameter values, s_i is the sensitivity of the arrival time to the process parameter, p_i, modeled as a random variable, and A_{random} is the random independent component of the arrival time.

Similar to the arrival time, the delay of a gate is modeled as follows:

$$d = D_{nom} + \sum_i sd_i.p_i + D_{random}, \tag{4.2}$$

where sd_i is the sensitivity of the gate delay to the process parameter, p_i. The arrival time propagation is achieved by adding the individual components of the arrival time

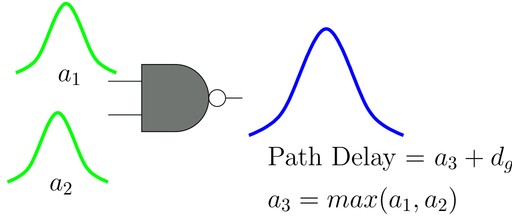


Figure 4.1: Merging arrival times

and the gate delay to obtain a canonical form of the arrival time at the gate output. This operation is an exact one and does not lead to any inaccuracy. The next step is to compute the maximum of the arrival times at the gate output.

Consider two arrival times $a_1 = A_{nom,1} + \sum_i s_{i,1}.p_i + A_{random,1}$, and $a_2 = A_{nom,2} + \sum_i s_{i,2}.p_i + A_{random,2}$ as shown in Fig. 4.1. The $max(a_1, a_2)$ is given by the arrival time $a_3 = A_{nom,3} + \sum_i s_{i,3}.p_i + A_{random,3}$ such that each of the components of the max is calculated as follows:

$$A_{nom,3} = max(A_{nom,1}, A_{nom,2}) \tag{4.3}$$

$$s_{i,3} = max(s_{i,1}, s_{i,2}) \tag{4.4}$$

$$A_{random,3} = max(A_{random,1}, A_{random,2}) \tag{4.5}$$

This process of computing the maximum of the two arrival times is not an exact computation, but gives an upper bound, resulting in a conservative estimate. In case of computing the maximum of more than two arrival times, the simple procedure outlined in (4.3) - (4.5), can result in a large error accumulation. The authors in [23] have introduced a heuristic to reduce the error. Instead of propagating one arrival time at the output of a logic gate, multiple arrival times are propagated. The larger the number of the arrival times propagated, the better the accuracy is. When no merging operation is performed at any intermediate node, the arrival time computed at the primary output is exact, but the computation complexity is large. At each node, the heuristic propagates only K arrival times by merging some of the arrival times. For all the arrival times, a_i, incident on a node the maximum arrival times, $m_{i,j}$, for each pair a_i, a_j, is calculated. All the maximum arrival times, $m_{i,j}$, are arranged in descending order and the $m_{i,j}$ with the minimum mean replaces the a_i, a_j pair in the arrival time set. Thus, the number of arrival times is reduced by one. This process is repeated, until only K arrival times remain. This procedure attempts to merge only those arrival times which are likely to have the least or no impact on the critical delay of the circuit. The arrival times are propagated in this manner, until the primary output or a sink is reached, where the maximum operation is again performed to obtain the upper

bound on the critical delay of the circuit. Some other works have instead proposed analytical techniques for computing the max operation [56]. However, the proposed technique has the flexibility of being as accurate as desired by increasing the number of arrival times to be propagated.

Besides the other parameters, the delay of a logic element is also dependent on the threshold voltage of the transistors of a logic element. The threshold voltage varies with the channel length of the transistors due to the short channel effects. In particular, a roll-off in the threshold voltage is observed as the channel length is reduced. In this work, the threshold voltage model from BSIM4 is chosen [57]. The threshold voltage of a transistor is modeled as

$$\begin{aligned} Vth &= Vth_0 + Vth_{body} - Vth_{SCE} - Vth_{DIBL} \\ &+ Vth_{halo} - Vth_{DITS}, \end{aligned} \tag{4.6}$$

where Vth_{body}, is the body effect, Vth_{SCE} is the short channel effect, Vth_{DIBL} is the drain induced barrier lowering effect, Vth_{halo} is the threshold voltage shift due to the halo pocket implants at the drain and source junctions, and Vth_{DITS} is the drain induced threshold shift due to the source and the drain pocket implants. The various components of the threshold voltage are functions of the channel length, the gate oxide thickness and the substrate doping. The BSIM4 analytical models are employed for the threshold voltage dependence on these process parameters.

The variation in the threshold voltage due to the random dopant fluctuations is modeled as follows [58]:

$$\sigma_{Vth_{rdf}} = \frac{Q.T_{ox}}{\epsilon_{ox}} \sqrt{\frac{N_{ch}.W_{dm}}{3.L.W}}, \tag{4.7}$$

where W_{dm} is the channel depletion width, N_{ch} is the channel doping concentration, and L, W are the channel length and width, respectively. To a first order approximation small variations in L will not impact the random dopant fluctuations and hence variations in L and threshold voltage variations due to random dopant fluctuations can be considered independent. For a function $y = g(x)$, where x is a random variable with variance σ_x^2, the variance of y is approximated by computing [59]

$$\sigma_y^2 = \left(\frac{dg(x)}{dx}\right)^2 \cdot \sigma_x^2. \tag{4.8}$$

The delay of a circuit element is expressed as a function, $f(L_{eff})$, and the variance in the delay, $\sigma^2_{delay_{L_{eff}}}$, is computed by (4.8). Similarly, the variance in the delay due to gate oxide thickness variation $\sigma^2_{delay_{Tox}}$ is calculated. A closed form expression for $f(L_{eff})$ and

$f(Tox)$ is unwieldy, and can be represented through a set equations presented in [57], and is omitted here for brevity. Since the channel length variations, the random dopant fluctuations and the gate oxide thickness variations are independent, the total variance of delay is calculated by:

$$\sigma^2_{delay} = \sigma^2_{delay_{rdf}} + \sigma^2_{delay_{Leff}} + \sigma^2_{delay_{Tox}}. \tag{4.9}$$

To compute the variance and the mean of the delay, it is modeled as a function of $Leff$, Tox and Vth. Using the statistical model, the mean and the variance of delay D for a logic element at location (j,k) for i^{th} level, are computed as follows:

$$\begin{aligned} E\{D\} &= D(L_{nom}, Vth_{nom}, Tox_{nom}) \\ &+ \frac{1}{2}\sum_{i=0}^{n}\left(\frac{\partial^2 D}{\partial Leff^2}.\sigma^2_{Leff_{i,(j,k)}}\right. \\ &+ \frac{\partial^2 D}{\partial Vth^2}.\sigma^2_{Vth(rdf)_{i,(j,k)}} \\ &+ \left.\sum_{i=0}^{n}\frac{\partial^2 D}{\partial Tox^2}.\sigma^2_{Tox_{i,(j,k)}}\right), \end{aligned} \tag{4.10}$$

$$\begin{aligned} \sigma^2_D &= \sum_{i=0}^{n}\left((\frac{\partial D}{\partial Leff})^2.\sigma^2_{Leff_{i,(j,k)}}\right. \\ &+ (\frac{\partial D}{\partial Vth})^2.\sigma^2_{Vth(rdf)_{i,(j,k)}} \\ &+ \left.(\frac{\partial D}{\partial Tox})^2.\sigma^2_{Tox_{i,(j,k)}}\right), \end{aligned} \tag{4.11}$$

where all the partial derivatives, which represent the sensitivities of the delay to the process parameters, are computed at the nominal values of these process parameters, at i^{th} level and location (j,k). Since the random variables, $Leff_{i,(j,k)}$, for the different values of i, j, k are independent, the delay variances are added to obtain the total delay variance.

4.3 Proposed Technique

4.3.1 Impact of Segment Length on Variability

This section proposes a theoretical basis for routing architecture enhancements in this work. Here a *single* wire segment is considered with buffers and expressions for its delay

and its variability are discussed. Consider a wire segment with m buffers which are equally placed along the length L of the wire. The propagation delay through the wire is given by (4.12), [60]. The standard deviation of the propagation delay, under variations in the channel length of the transistors, and the assumption that the variations are independent across the buffers is computed from 4.12, and is given by (4.13),

$$\begin{aligned} t_p &= m\Bigg(0.69\frac{R_d}{s}\Bigg(s\gamma C_d + \frac{cL}{m} + sC_d\Bigg) \\ &+ 0.69\Bigg(\frac{rL}{m}\Bigg)(sC_d) + 0.38rc\Bigg(\frac{L}{m}\Bigg)^2\Bigg), \end{aligned} \tag{4.12}$$

$$\sigma_{t_p} = \frac{0.69}{s}\Bigg(\frac{cL}{\sqrt{m}} + \sqrt{m}(s\gamma C_d + sC_d)\Bigg)\frac{\partial R_d}{\partial L_{\mathit{eff}}}\sigma_{L_{\mathit{eff}}}, \tag{4.13}$$

where R_d is the resistance of the buffer (minimum-size), C_d is the input capacitance of the buffer (minimum-size), γ is the ratio between the intrinsic output and input capacitances of the buffers, s is the size of the buffer, r is the resistance of the wire per unit length, and c is the capacitance of the wire per unit length. In FPGAs, the wiring capacitances dominate the total capacitance in the routing segments. This is because the routing in an FPGA requires more wiring resources than its ASIC counterpart. Also in the scaled technologies, interconnects have become the dominant source of delays. Consequently, (4.13) is simplified by ignoring the buffer capacitances such that

$$\sigma_{t_p} = \frac{0.69}{s}\Bigg(\frac{cL}{\sqrt{m}}\Bigg)\frac{\partial R_d}{\partial L_{\mathit{eff}}}\sigma_{L_{\mathit{eff}}}, \tag{4.14}$$

which shows that the standard deviation of the propagation delay varies inversely with $\sqrt{m}$. This implies that as the number of buffers increases for a given length of wire, the standard deviation of the delay decreases. However, this decrease cannot continue indefinitely, because the capacitances of the buffers will start to play a more dominant role, as the number of buffers is increased, and the terms ignored in (4.14) can no longer be ignored.

Consider two different examples of interconnect buffers in a routing segment in an FPGA. In Fig. 4.2(a), the routing segment consists of three identical buffers, distributed throughout the length of the wire such that the capacitance of each of the wire segment is C_1. The capacitance due to the buffers is ignored because it is much less than that of the wire capacitance. In the Fig. 4.2(b), the complete wire is driven by a single buffer of the same size, as the one in Fig. 4.2(a). The total capacitance of the wire is C_2 where

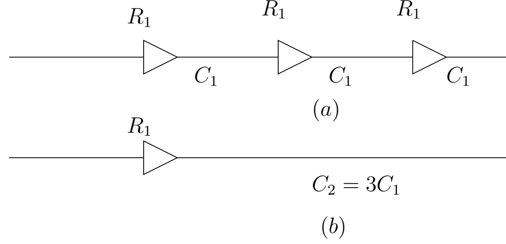


Figure 4.2: Impact of buffers on the delay variability

$C_2 = 3.C_1$. The average on-resistance of the buffers is given by R_1 and the resistance of the wires is ignored because it is small compared with the on-resistance of the buffers.

With these assumptions, the simple first order expressions for the variance of delay for the two cases are calculated, with the variation in the channel length L_{eff}. It is assumed here that the process variations of each of the buffers are independent. Here, it is worthwhile to point out that factors such as cross-talk can further complicate the scenario. However, cross-talk would lead to increased dominance of wire capacitance and hence the above discussion becomes even more relevant and important.

$$\sigma_{D_1}^2 = 3.\left(\frac{\partial R_1}{\partial L_{eff}}.C_1\sigma_{L_{eff}}\right)^2 \tag{4.15}$$

$$\begin{aligned} \sigma_{D_2}^2 &= \left(\frac{\partial R_1}{\partial L_{eff}}.3C_1.\sigma_{L_{eff}}\right)^2 \\ &= 3.\sigma_{D_1}^2 \end{aligned} \tag{4.16}$$

The equations (4.15) and (4.16) compute the variances of the delays for the two cases in Fig. 4.2(a) and 4.2(b) respectively. It is evident that the delay variance (standard deviation) in Fig. 4.2(b) is three ($\sqrt{3}$) times greater than the delay variance in Fig. 4.2(a). These expressions indicate that the number of buffers for a wire segment can be increased to reduce the delay variance, until the buffer capacitance becomes a significant part of the total capacitance. Furthermore, the approximations in (4.15) and (4.16) are valid only to a certain extent and are presented to intuitively analyze the impact of the number of buffers in the routing segments.

Fig. 4.3 shows the routing of a net and how the delay variability is affected by the number of buffers in the routing of the nets for two different cases. Fig. 4.3 (a) illustrates a case when the routing uses fewer buffers resulting in larger delay variability, whereas in Fig. 4.3 (b), more buffers are used by the net resulting in lesser delay variability.

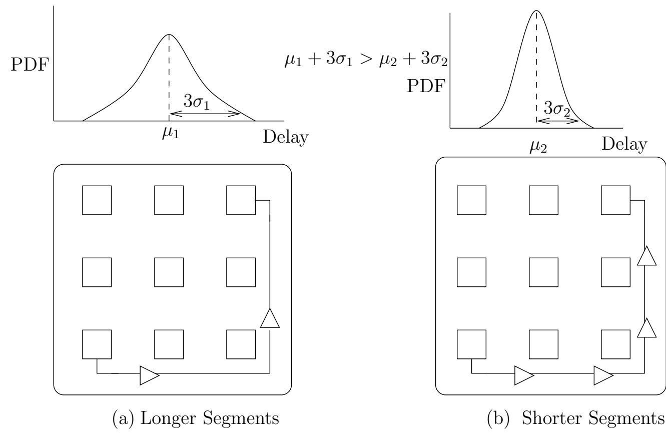


Figure 4.3: Delay variability reduction using shorter segments

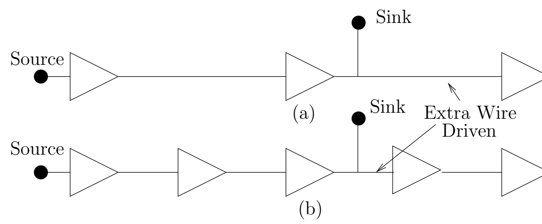


Figure 4.4: Extra wire segments in routing

Another case which typically occurs in FPGAs is that of the unused parts of the tracks being driven by the buffers. This occurs because of the location of source and sink pairs. Consider the example illustrated in Fig. 4.4, which shows the interconnect between a pair of source and sink. In Fig. 4.4 (a), the second buffer which ultimately connects to the sink has to drive a long wire as well even though that is not required in the routing, and this results in larger delay variability in accordance to (4.14), compared to Fig. 4.4 (b), where the third buffer, which ultimately connects to the sink, drives a smaller wire segment not necessary for the routing between the source and sink pair.

The above discussion shows that shorter wire segments are good for delay variability, when the wire capacitances dominate the total interconnect capacitance. However, the actual scenario in an FPGA is more complicated due to spatial correlation in the process parameters across different buffers and also the capacitance of the buffers would affect the actual mean and standard deviation of the circuit delay. Further, in case many buffers are inserted in a routing, the capacitances of the buffers will become important and therefore a simple architecture having only shorter wire segments will not suffice. An architecture which supports both longer and shorter wire segments is required and is explored in this work.

In ASICs, where there is more flexibility for buffer insertion and sizing a mathematical optimization problem can be devised with more accurate delay models to determine the location of these buffers. However, in an FPGA, since it is pre-fabricated, and the FPGA design cannot be targeted for a particular application, the design approach that is adopted in this work allows routing segments which have more buffers for a given wire length than other routing segments. The router during the routing phase then selects the most appropriate routing segments.

The timing yield of a design is defined as the number of chips meeting the target timing cutoff. Therefore, one of the optimization approaches can be chosen to reduce the delay variability such that most of the chips meet the required timing. A similar approach is to reduce the delay at a certain confidence level, which can be the 3σ point on the delay distribution curve. Since the target delay cutoff is not known for the FPGAs, the objective of this work is to reduce the $(\mu + 3\sigma)$ delay of the circuit.

4.3.2 Routing Architecture Evaluation

A typical FPGA routing architecture is composed of horizontal and vertical routing segments, connected by switch boxes. The routing fabric of an FPGA contains different lengths of wire segments as shown in Fig. 4.5. Here, there are three different routing segments, the wire that spans eight logic blocks with eight switch boxes, the wire that spans eight logic blocks with four switch boxes, and the wire that spans four logic blocks

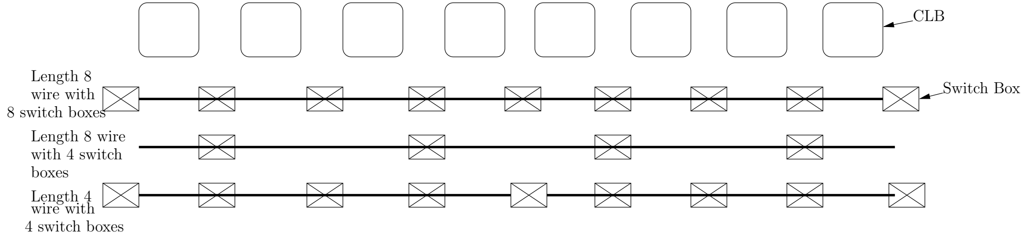


Figure 4.5: A section of routing fabric showing different segment lengths

Table 4.1: Routing architecture evaluation

Architecture	Length 2 segment	Length 4 segment	Length 8 segment
arch1	10%	45%	45%
arch2	20%	40%	40%
arch3	30%	35%	35%
arch4	40%	30%	30%
arch5	50%	25%	25%

with four switch boxes. In the first case of the wire spanning eight logic blocks with eight switch boxes, the segment has the flexibility of connecting to the vertical routing segments at all the intersections of horizontal and vertical routing tracks. In the second case of the wire spanning eight logic blocks with four switch boxes, the segment has the flexibility of connecting to the vertical routing segments at only four intersections of the horizontal and the vertical routing segments.

An FPGA routing architecture consists of different ratios of these routing segments. For instance, in an FPGA where the routing channels have 100 tracks, the distribution can be 20 tracks with segments of length 2, 40 tracks with segments of length 4 and 40 tracks with segments of length 8.

There can possibly be a very large number of combinations of segment lengths, resulting in as many number of routing architectures. Theoretically, all of these can be evaluated, however, the search space is too large such that it is computationally infeasible to evaluate all the possible combinations. Instead, this work intends to demonstrate that by providing some shorter segments in the existing routing fabric the timing variability can be reduced. To this end, this work explores five different combinations of routing segments as shown in Table 4.1. The baseline architecture consists of the architecture with 50% segments of length 4 and 50% segments of length 8. Architectures with segments of length 4 and 8 are explored in [1], in a similar manner, but not for variability aware design.

This work proposes a theoretical basis and a methodology for exploring routing architectures to reduce timing variability in FPGAs. An FPGA designer can incorporate shorter

Table 4.2: Routing architecture evaluation: % Improvement

Architectures →	arch1	arch2	arch3	arch4	arch5
Mean	3.61%	7.5%	8.79%	7.2%	5.97%
Std. Dev.	9.8%	9.1%	9.3%	8.7%	8.4%

segments in the FPGA according to the design constraints and evaluate which combination leads to best results. Indeed, the different segment length combinations explored in this work is not exhaustive and thus there is no guarantee that this set of combinations of routing segments contains the best routing architecture for reducing timing variability. The methodology allows an FPGA designer to to judiciously select and design routing fabric for FPGAs, such that the timing yield can be enhanced.

Table 4.2 shows the average improvements in the $(\mu + 3\sigma)$ point for different architectures compared to the baseline architecture for the 20 MCNC benchmarks. It can be observed that the mean improvement for all the architectures vary between 4% and 9%. This indicates that providing smaller routing segments would invariably lead to improvement in timing variability. An architecture which leads to high improvements in certain benchmarks while less improvements or degradations in other benchmarks is not desirable. In such a case just measuring the mean improvement will lead to an inappropriate architecture selection. To prevent this, the standard deviations of the improvements are also calculated to analyze the architectures. The last row in Table 4.2 lists the standard deviation of the improvements. From the table, it can be seen that the architectures $arch2$, $arch3$, and $arch4$, have mean improvements better than the other benchmarks, between 7% and 9%, and therefore these architectures are selected for further evaluation. To further narrow down the choice to one architecture, the standard deviations are also evaluated. It can be seen that although $arch3$ has the greatest mean improvement, $arch4$ is the best candidate because of good mean improvement and smallest standard deviation. Therefore, $arch4$ is selected as the candidate architecture for further evaluation. While there is no guarantee that the suggested routing architecture is *the best* routing architecture, the proposed methodology provides a framework and a theoretical basis to guide the design of FPGAs for timing variability. Also, the best combination of routing segments, from among those evaluated, suggested in this work may not be the best for all FPGAs, depending on its application area and the benchmarks on which the FPGA is evaluated before its design is finalized. Further, an FPGA designer might have to deal to additional constraints which might make some routing segment combinations infeasible. However, even in such cases the same methodology, with additional constraints, can be employed to arrive at the final architecture. More discussion on the results is presented in Section 4.4.

4.3.3 Variability-Aware Placement and Routing

The CAD approach employed to enhance the performance of the placement and routing tool under the process variations, involves incorporating the delay variability information in the placement and routing phase of the design flow. The VPR implements a timing driven placement for a netlist to map an application to an FPGA [16]. VPR uses a simulated annealing algorithm for the placement of the logic blocks on the CLBs. It is an algorithm which mimics the annealing procedure to cool molten metal slowly for producing high quality metal structures. The algorithm begins with a random placement, and repeatedly moves the logic blocks to newer locations and evaluates whether the move can be accepted or not. The acceptance or rejection depends on the placement cost, computed by the VPR. If the move results in a reduction of the placement cost, the move is accepted. If the placement cost increases as a result of the move, there is still some probability that the move will be accepted. The acceptance of some bad moves prevents the placement tool from being stuck at some local minimum.

The placement cost of the VPR is the sum of the timing cost and the wiring cost. The timing cost is computed on a source sink basis. For a source sink pair (i, j) [16],

$$Timing_Cost(i,j) = Delay(i,j).Crit(i,j)^{crit_exp}, \tag{4.17}$$

$$Crit(i,j) = 1 - \frac{Slack(i,j)}{D_{max}}, \tag{4.18}$$

where $Delay(i, j)$ is the delay between source (i) and sink (j), $Crit(i, j)$ is the criticality of the connection between them, $Slack(i, j)$ is the slack available with the source and the sink pair, D_{max} is the critical path delay, and $crit_exp$ is for assigning large weights to critical timing connections. The total timing cost is then computed as

$$Timing_Cost = \sum_{(i,j)} Timing_Cost(i,j). \tag{4.19}$$

The wiring cost is estimated by computing the *bounding box* of the placed logic blocks [16]. Essentially, the bounding box is the smallest rectangle within which all the logic blocks lie for the current placement. The wiring cost is an estimate of the wire length used by the netlist. The authors in [16] develop an *auto-normalizing* cost function for the placement. The cost function, $\triangle C$, used for the placement routine, is defined as

$$\begin{aligned} \triangle C &= \lambda.\frac{\triangle Timing_Cost}{Previous_Timing_Cost} \\ &+ (1-\lambda).\frac{\triangle Wiring_Cost}{Previous_Wiring_Cost}, \end{aligned} \tag{4.20}$$

where λ is a factor for giving different weights to the timing cost and the wiring cost, $\triangle Timing_Cost$ is the change in the $Timing_Cost$ because of the current move, and

$\triangle Wiring_Cost$ is the change in the $Wiring_Cost$ because of the current move. The timing driven placement in the VPR optimizes both the wiring cost and the timing cost depending on the value of λ. In [16], it is proposed that $\lambda = 0.5$ and $crit_exp = 8$ are the best values for the timing and wiring cost trade-off.

Since the optimization goal under process variations is to minimize the $(\mu + 3\sigma)$ point of the critical delay, the $Timing_Cost$ evaluations is performed at the $(\mu + 3\sigma)$ point as shown in (4.21),

$$Timing_Cost(i, j) = (\mu + \alpha.\sigma)_{(i,j)}.Crit(i, j)^{crit_exp}, \tag{4.21}$$

where α is the factor for choosing the point on the delay PDF to be optimized. In this work, $\alpha = 3$, since the goal is to optimize the $(\mu + 3\sigma)$ point of the delay PDF. This is a straightforward approach resulting in a direct optimization of the $(\mu + 3\sigma)$ point. The value of α can be conveniently chosen to optimize the targeted point.

The routing in the FPGA is based upon the Pathfinder algorithm [1]. The Pathnder repeatedly rips-up and re-routes each net in the circuit, until all the congestion is resolved. One routing iteration involves ripping-up and re-routing each net in the circuit. The first routing iteration routes for the minimum delay, even if the iteration leads to congestion, or the routing resources are overused. To remove this overuse another routing iteration is performed. The cost of overusing a routing resource increases for each iteration, thereby improving the chance of resolving the congestion. At the end of each routing iteration all the nets are routed, but with some congestion. Based on this routing, a timing analysis is carried out to compute the critical path and also the slack of each source sink connection. A net is routed by starting with a single node in the routing resource graph, corresponding to the source of the net. A wave expansion algorithm is invoked k times to connect the source to each of the net's k sinks, in the order of the criticality of the sinks, the most critical sink being the first. The cost for using node n during this expansion for connecting the sink j of the net i is expressed as

$$\begin{aligned} cost(n) &= crit(i, j).delay(n, topology) \\ &+ [1 - crit(i, j)].b(n).h(n).p(n), \end{aligned} \tag{4.22}$$

where $crit(i, j)$ is the criticality of the connection, $delay(n, topology)$ is the delay of the connection after including node n in the path, and $b(n), h(n), p(n)$ are the base cost, the historical congestion, and the present congestion [1]. To incorporate the variability information in the router, the routing cost function is modified and instead of the nominal value of $delay(n, topology)$, its mean and standard deviation are computed as follows:

$$\begin{aligned} delay(n, topology) &= \mu_{delay}(n, topology) \\ &+ 3\sigma_{delay}(n, topology) \end{aligned} \tag{4.23}$$

In the variability-aware router the $3\sigma_{delay}(n, topology)$ is computed at each routing iteration and used in the SSTA driven routing optimization engine.

4.4 Evaluation, Results and Discussions

4.4.1 Experimental Details

The 45nm Berkeley Predictive Model is chosen as the technology node for the simulations. It is demonstrated in [24] that three levels in the quad-tree model for the process variations lead to sufficiently accurate results for SSTA, and hence five levels are chosen in this work. Since the FPGAs have a regular structure, in which a tile is replicated across the chip, instead of using 4^i grids at level i, a scheme in which the grid size at level 0 is the size of the FPGA, the level 1 has the grid size of 8x8 FPGA tiles, the level 2 has the grid size of 4x4 FPGA tiles, and the level 2 has the grid size of 2x2 FPGA tiles, is selected. This essentially means that for any level, all the tiles within the grid has perfect correlation, for example, at level 2 all the 4 FPGA tiles in a grid will have a single random variable to represent the variations in a process parameter. Such a scheme, as opposed to the one in which each level is divided into 4^i grids, avoids the FPGA tiles from being partitioned into more than one grid. The last level represents the random independent variations. In case of availability of the fabrication data and the spatial correlation information for process parameters, the grid sizes and the number of levels can be accurately determined by using the methodology described in [24]. In the absence of actual measurement data for process variations, the channel length variations and the gate oxide thickness variations are modeled at levels 1, 2, and 3 which represent intra-die variations and models the spatial correlation in the process parameters between the different parts of the chip. A 3σ variation of 20% in L_{eff} and a 3σ variation of 15% in Tox are assumed, and distributed equally over the levels [23], in the absence of the actual fabrication data for the spatial correlation. The variation in the threshold voltage, due to random dopant fluctuations is modeled at the last level, representing an independent random variable. A set of MCNC benchmarks is selected in this work for obtaining the results.

4.4.2 Results and Discussions

To evaluate the routing architecture, several different routing architectures are simulated. The evaluation of different routing architectures are listed in Table 4.1. The idea behind exploring the routing architectures is to determine the proportion of the different routing segments required to reduce the $(\mu + 3\sigma)$ point of the critical delay.

Table 4.3: Benchmark sizes

Benchmarks	# of CLBs	Benchmarks	# of CLBs
alu4	192	ex5p	139
apex2	240	frisc	446
apex4	165	misex3	178
bigkey	214	pdc	582
clma	1054	s298	243
des	200	s38417	802
diffeq	189	s38584.1	806
dsip	172	seq	221
elliptic	454	spla	469
ex1010	599	tseng	133

The baseline architecture is the architecture with 50% wire which spans four logic blocks and 50% wire segments spanning eight logic blocks, which was explored in [1]. This baseline architecture is used to measure the improvement in the $(\mu + 3\sigma)$ of the critical delay by using the proposed design technique. The five routing architectures evaluated have different percentages of the routing segments of lengths two, four and eight. The proportion of the track segments spanning two logic blocks is increased and the remaining tracks are equally divided between the segments spanning eight and four logic blocks respectively. The sizes of different benchmarks are shown in Table 4.3 in terms of number of CLBs, where each CLB has a size of 8 BLEs.

The routing architecture does not have the flexibility to be altered once the FPGA is fabricated. Therefore, this evaluation should be performed by an FPGA designer before the architectural parameters of the FPGA are fixed and the FPGA is fabricated. Based on the simulation results, as discussed in Section 4.3.2, the best architecture determined for reducing the variability, under the given technology and constraints, is the $arch4$. The results shown in Table 4.4 use $arch4$ for the variability-aware design.

Table 4.4 offers a comparison of the $(\mu + 3\sigma)$ delays of the baseline and the variability aware designs. Column 5 lists the improvements due to architecture enhancements and column 6 lists the improvements after CAD optimization is applied to the FPGA with enhanced architecture. It can be seen that with just the architecture enhancements, improvement in $(\mu + 3\sigma)$ delay can be obtained. Further, it can be observed that the $(\mu + 3\sigma)$ delay of the variability aware design improves by up to 28%, depending on the benchmark, when variability-aware CAD optimization is applied to the enhanced FPGA architecture. These improvements result from reduction in the mean and the variance of the delays. Another observation that can be made from the table is that in some benchmarks variability-aware place and route does not lead to any improvement or slightly degrades

Table 4.4: Results of Variability-Aware Design for Timing Yield

Benchmarks	Baseline ($\mu + 3\sigma$)	Variability-Aware Architecture		Improvement ($\mu + 3\sigma$)		Std. Dev. Improvement		Yield Improvement
		Deterministic CAD ($\mu + 3\sigma$) (ns)	Variability-Aware CAD ($\mu + 3\sigma$) (ns)	Arch. Enhancement	Arch. and Variability Aware CAD	Arch. Enhancement	Arch. and Variability-Aware CAD	
alu4	8.77971	8.64609	8.32891	1.52%	5.13%	16.87%	21.75%	3.00%
apex2	10.2202	9.27657	9.03988	9.23%	11.54%	17.88%	18.61%	16.25%
apex4	8.48323	8.49214	8.02503	-0.1%	5.40%	15.71%	18.43%	3.91%
bigkey	5.73215	5.74827	6.19854	-0.3%	-8.13%	7.19%	7.02%	-0.44%
clma	19.4209	21.3906	21.1896	-10.1%	-9.1%	11.7%	15.98%	-2.17%
des	11.2742	9.97441	14.3802	11.5%	-27.5%	13.65%	0.7%	16.12%
diffeq	7.86171	5.94118	5.65697	24.4%	28.04%	18.41%	26.4%	68.23%
dsip	6.11283	5.52062	5.43229	9.7%	11.13%	4.7%	5.25%	13.58%
elliptic	11.5893	10.2334	10.0177	11.7%	13.56%	13.91%	18.88%	17.78%
ex1010	15.2968	14.989	14.5394	2%	4.95%	15.88%	18.63%	2.94%
ex5p	9.46913	8.47661	8.00807	10.5%	15.42%	15.19%	18.96%	39.91%
frisc	14.4183	12.1726	12.3823	15.6%	14.12%	9.94%	16.83%	37.61%
misex3	8.51963	8.00565	8.0134	6%	5.94%	18.00%	21.65%	4.59%
pdc	14.4815	14.4253	14.1629	0.4%	2.2%	15.90%	17.30%	0.33%
s298	12.9248	10.7352	10.8236	16.9%	16.25%	13.18%	18.62%	58.01%
s38417	12.7315	13.22	11.8097	-3.8%	7.24%	13.88%	18.64%	2.95%
s38584.1	9.71621	9.38845	9.09571	3.37	6.38%	10.71%	15.18%	2.87%
seq	9.94624	8.61802	9.23896	13.3%	7.11%	18.83%	17.16%	24.50%
spla	13.1557	13.0828	12.9597	0.6%	1.48%	14.14%	20.06%	-0.224%
tseng	6.57158	5.1804	5.42482	21.1%	17.45%	8.29%	10.80%	56.65%

the $(\mu+3\sigma)$ delay, for example, in the cases of $frisc$ and $misex3$. This is attributed to the fact that the exact circuit delay and its variance is known only after the routing is complete and during the placement stage only an estimation can be made with regards to the delay and its variance. Further, the variance estimation is complicated by the spatial correlation factor, and the topology of a path. So, even if two nets have same number of buffers and wire lengths, the arrival time delay variance can be significantly different depending on the spatial correlation of process parameters and their topology. However, it should be noted that the delay variability CAD optimization approach is applicable during the mapping of a benchmark to the FPGA, and it can be turned on only if it results in delay variability improvement.

Columns 7 and 8 show the improvements in the standard deviations from architecture enhancement and architecture enhancement with variability-aware CAD, respectively. It can be observed that reduction in standard deviation of up to 22% can be achieved using the proposed technique. The optimization approach (architecture and variability-aware CAD) targets the reduction of the $(\mu+3\sigma)$ delay and this may result in an increase in the mean of the delay delay in the optimized design, however, the $(\mu+3\sigma)$ decreases, for example, as in the case of the benchmark *alu4* where the mean of the delay increases slightly in the optimized design, but the standard deviation of the delay decreases by 21.75% leading to an overall improvement in $(\mu+3\sigma)$ delay of 5.13%.

In the cases of the benchmarks *bigkey* and *clma* the standard deviation of the critical delay decreases when the architecture is enhanced, but the mean delay increases such that the $(\mu+3\sigma)$ delay increases. In the case of the benchmark *clma* the standard deviation of the delay decreases by 16%, but the mean delay increases by 19.5%, resulting in overall degradation in the $(\mu+3\sigma)$ delay of 9.1%. In the case of benchmark *des*, it can be observed that the architecture improvement leads to an improvement of 11.5% in the $(\mu+3\sigma)$ critical delay, however, when variability-aware place and route is used for this benchmark, it leads to degradation in the $(\mu+3\sigma)$ critical delay. Such differences occur across benchmarks because of the differences in topology of the benchmarks.

Fig. 4.6 shows the PDF of the delay distributions of the baseline and the variability aware design for the benchmark *apex4*. It can be seen that the mean and the standard deviation of the critical delay reduces in the variability-aware design implementation leading to reduction in $(\mu+3\sigma)$ critical delay by 5.4%. As another example, in case of the benchmark *ex1010*, the mean value of the critical delay remains same in both the baseline and the variability optimized designs, however in the case of the variability optimized design the variance of the critical delay reduces by 18.8%, resulting in an overall improvement in the $(\mu+3\sigma)$ delay of 4.95%.

For a given design, process variations lead to variations in the timing of the fabricated designs. There can be two approaches to look at the timing variability in fabricated chips. The first approach is evaluating the $(\mu+n\sigma)$ critical delay point of the design, where, in

this work n is 3. This would ensure that 99.9% of the fabricated designs would meet the $(\mu+3\sigma)$ critical delay. For example, in the case of *apex2* benchmark, from Table 4.4, with variability-aware architecture and CAD, 99.9% of the apex2 designs on the FPGA will meet the critical delay target of 9.04ns, whereas in the case of the baseline architecture the critical delay target needs to be relaxed to 10.22ns, such that 99.9% of the *apex2* designs on the FPGA can meet the critical delay requirement. This implies that with the variability-aware architecture and CAD, the *apex2* design can be operated at a higher frequency.

The second approach works in the case when the design has to meet a specified target delay. The timing variability implies that not all chips will be able to meet the target delay. For example, in the case of the *apex2* design, if the target delay is 8.53ns, only 87.4% of the designs on the baseline FPGA will meet the target delay of 8.53ns. However, in the case of the FPGA with variability-aware architecture and CAD, 98.9% of the designs will meet the target delay of 8.53ns. The yield of a design depends on the minimum frequency requirement and the maximum allowed leakage [61]. The upper cut-off limit for the critical delay depends on the minimum frequency requirement, whereas the lower delay cut-off depends on the maximum allowed leakage. The timing yield of a given design for a target critical delay is then found by calculating the CDF of the critical delay distribution as follows:

$$Yield_{TargetDelay} = \int_{LowerCutoffDelay}^{TargetDelay} f(D)dD, \quad (4.24)$$

where $f(D)$ is the PDF of the design's critical delay, $Lower_Cutoff_Delay$ is the lower cutoff delay, which is due to the constraint on leakage, and $Target_Delay$ is the critical delay which must be met by the design. The target delay is intended for the minimum frequency requirement, whereas the lower cut-off delay is intended for the maximum allowed leakage. The lower cut-off selected in this work is $(\mu-2\sigma)$ delay, which discards about 2% of the chips in which variability can cause excessive leakage, thus rendering them unusable. Though this work targets timing yield, such a lower cut-off limit is required and the $(\mu-2\sigma)$ delay is selected for illustrative purposes, and any other value can be chosen depending on the maximum allowed leakage for the design.

The last column in Table 4.4 shows the best case yield improvement over the baseline implementation due to either from the architecture improvement or architecture improvement with variability-aware place and route. For calculating the yield improvement, the target and cutoff delays for the variability-aware design is selected as $(\mu+2\sigma)$ and $(\mu-2\sigma)$ respectively, corresponding to the 95% confidence level, i.e., 95% of the chips for a design will have their critical delay between these values. To compute the corresponding yield for the baseline design the $Target_Delay$ is selected as the $(\mu+2\sigma)_{Variability_Aware}$ of the variability-aware design, whereas the $Lower_Cutoff_Delay$ is selected as the $(\mu-2\sigma)_{baseline}$ of the baseline design. The above selection of $Target_Delay$ and $Lower_Cutoff_Delay$

are intended for computing the yield and any such delay values can be chosen based on the design constraints to estimate the yield of a design.

Fig. 4.7 shows the CDF of the critical delay for the benchmark *apex4* from which the yield for a given delay can be computed. For the benchmark *apex4* with the variability-aware implementation, a 95.5% yield is obtained if the target delay is 7.4 ns and the lower cut-off delay is 5.13 ns. For the same target delay, and the lower cutoff delay of 4.93 ns the baseline design implementation has a 91.5% yield. In cases of the benchmarks *bigkey*, *clma*, and *spla*, it can be seen that there is an yield loss. This occurs again because, although the standard deviation of the critical delay decreases in all the cases, but the mean value of the delay increases, such that there is an yield loss at the target delay. For instance in case of *spla*, the standard deviation of the critical delay reduces by 20%, however, the mean value of the critical delay increases by 5.8%. Again, this behavior is due to differences in topology of the benchmarks. This shows that just one architecture might not be suitable for all the applications. A possible solution to this can be providing a few flavors of different FPGA architectures, similar to what is provided by a commercial vendor. Though this work provides results for only one architecture, an FPGA designer might choose to provide more than one FPGAs such that all the validation benchmarks are satisfied. An an example, in the case of the benchmark *spla*, which has a small $(\mu+3\sigma)$ delay improvement of 1.48% with an yield loss of 0.22% for the architecture $arch4$ with varibility-aware CAD, the architecutre $arch3$ with the variability-aware CAD leads to an improvement of 6% in $(\mu + 3\sigma)$ delay, with an yield improvement of 3.8%.

Designing a routing architecture with shorter segment lengths requires more transistors. The area trade-off for the proposed architecture is such that the architecture requires 10% more transistors than the baseline implementation. The average dynamic power consumption (computed at mean frequency), for *alu4*, for the variability-aware implementation increases by 1% compared to the baseline implementation.

The optimization approach proposed in this work is a two step process, in which the first step is to determine the routing architecture and the second step is to optimize using variability-aware placement. The routing architecture improvement leads to most of the improvements in the $(\mu + 3\sigma)$ critical delay, with the variability-aware place and route optimization leading to further improvements. The run-times of the variability-aware CAD optimization and deterministic CAD optimization differ because of the statistical operations performed during the optimization steps. The runtimes of the benchmarks with variability-aware CAD vary between 20 minutes and 492 minutes for different benchmarks. The runtimes for deterministic place and route vary between 0.5 minutes and 7 minutes. For example, in the case of *alu4*, the deterministic place and route takes 1 minute of runtime, whereas the variability-aware place and route takes 31 minutes of run time. The runtime of the variability-aware CAD tool, depends on the number of levels chosen in the grid-model for SSTA, the number of grids on each level and the number of random

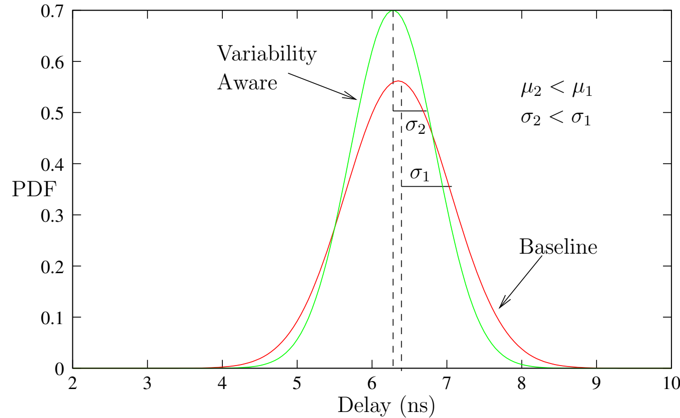


Figure 4.6: The PDFs for the baseline and variability aware implementations for the benchmark apex4

variables modeled. The higher these number, greater is the accuracy at the expense of runtime. In the case of the benchmark *alu4* if only channel length variations, which is the dominant variation, is considered, the runtime for the variability-aware CAD reduces from 31 minutes to 15 minutes, however, the standard deviation of the delay is underestimated by 5.9% . For the same benchmark, if variability-aware routing is turned off, the runtime of the CAD tool with just the variability-aware placement is 12 minutes. However, this comes at the expense of lesser improvement of 4.7% in $(\mu + 3\sigma)$ delay. If the number of levels in the grid model for SSTA is reduced by one and only channel length variation is modeled, the runtime of the variability-aware CAD reduces from 31 minutes to 9 minutes, with the standard deviation of the delay being underestimated by 9%. It should however, be noted that with architecture enhancements, even a deterministic place and route would lead to improvement in the timing variability as listed in column 5 of Table 4.4, with the runtime same as that of the deterministic place and route.

The technique proposed here for architecture and CAD enhancements is applicable to most of the industrial FPGAs, such as Virtex series from Xilinx or the Stratix from Altera. The technique is applicable because it follows the principle of incorporating shorter segments in the routing fabric, with more buffers, which can be easily applied to many FPGA architectures. The CAD enhancements are flexible and can be incorporated in any CAD tool by using statistical delay models.

4.5 Conclusions

This chapter presents a variability aware design technique to reduce the impact of process variations on the timing yield of FPGAs. The technique is twofold involving the co-design

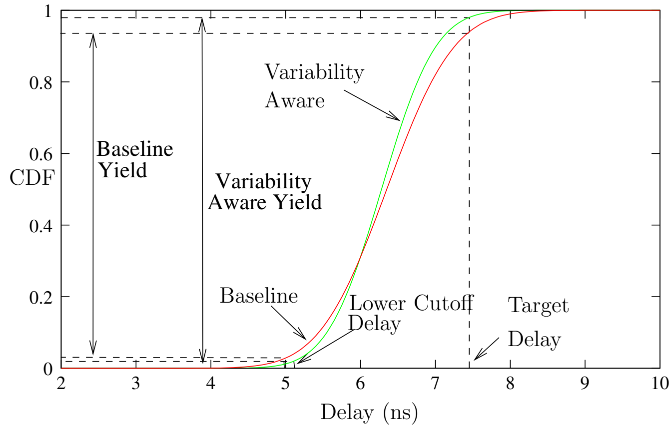


Figure 4.7: The CDFs for the baseline and variability aware implementations for the benchmark apex4

of the routing architecture and the variability-aware CAD optimizations. The routing architecture evaluations indicate that an architecture which has a certain proportion of shorter routing segments can provide a better trade-off for the timing variability. The CAD tool for placement and routing is enhanced to incorporate the timing variability to improve the timing yield of FPGAs. The results of the joint architecture and CAD optimizations indicate that the $(\mu + 3\sigma)$ delay improvement of up to 28% can be achieved depending upon the benchmark.

Chapter 5

Design for Power Yield

5.1 Introduction

This chapter discusses the design techniques for improving power yield in nanometer FPGAs. The techniques proposed in this chapter are CAD techniques. Every application has a power budget because of constraints due to battery life, thermal issues, etc., which should not be exceeded. All chips exceeding the power budget are discarded which leads to yield loss. The chips which have total power consumption less than the power budget contribute to the power yield of the design. For example, in mobile and low power biomedical applications, an appropriate power budget is pivotal to a long battery life. For such applications process variations can diminish the power yield of FPGAs resulting in significant financial loss. Several design techniques have been proposed for managing leakage power in FPGAs [5], [7], [62], [63]. However, none of these techniques have included the impact of process variations. Traditionally, process corners have been used for analyzing designs to meet the targeted performance, power and other design considerations at the best, nominal and worst case process corners. However, this may lead to pessimistic or optimistic designs. Moreover, it is very difficult to determine whether a particular process corner is indeed a best, nominal or worst case corner, because of the significant increase in the number of varying process parameters and operating conditions with technology scaling. Therefore, to design VLSI circuits under process variations, statistical techniques need to be adopted. Yield improvements directly translate to profit and even small yield improvements are desirable because of economic considerations [30]. These challenges in designing low power FPGAs in nanometer technologies, and simultaneously accounting for process variations, motivate the following, which are contributions of this work [64]. The contributions of this work are as follows:

- **Variability aware placement methodology for reducing the leakage varia-**

tion to increase the power yield: The leakage variation in FPGAs is significantly affected by spatial correlations of the process parameters variations. In this work, since the leakage is modeled as a random variable, it is shown that the leakage variation can be reduced if spatial correlations between leakage from the different spatial regions are reduced. A placement technique is proposed to reduce these spatial correlations and thus reduce the intra-die leakage variation. Such a technique is uniquely applicable to FPGAs since there is flexibility of placing the logic blocks at different CLBs by programming the FPGAs, and the un-utilized blocks can be power gated. Since the placement is a spatial operation on a two dimensional space, and the spatial correlation of process parameters is also on the same two dimensional space, managing placement can effectively reduce the impact of spatial correlations. Inter-die leakage variations for FPGAs and microprocessors are usually handled by binning [43], and therefore this work is targeted towards reducing intra-die leakage variation, resulting in improved power yield.

- **Variability aware dual-Vdd assignment:** A programmable dual-Vdd FPGA architecture is used for implementing the proposed CAD methodology. A new dual-Vdd assignment scheme is proposed for the programmable dual-Vdd FPGA which reduces the spatial correlations of the leakage to reduce its variation. This dual-Vdd assignment scheme is used after the placement and routing, to evaluate the total improvement in power yield.

5.2 Targeted FPGA Architecture

The basic structure of the FPGA under consideration in this work is the same as that described in Chapter 2. For high performance FPGAs, that need power efficient techniques, an implementation that reduces power consumption is required. Designs which target power yield should first implement a power reduction mechanism, and then a variability-aware design approach should be adopted to reduce the variability in the power. This is because enhancing power yield attempts the reduction of power variability such that the target power is met for as many chips as possible, so that the yield loss is small. In case the target power dissipation is achieved by applying a low power technique (considering power variability), for a certain confidence level, power yield loss is not a concern in such a design. However, if, after applying a low power technique, power variability causes a large number of chips to exceed the power budget, leading to yield loss beyond the acceptable confidence limit, variability-aware techniques need to be employed for enhancing power yield of the design. Therefore, to consider a low power FPGA design, the targeted architecture in this work is a dual-Vdd FPGA architecture, which is the first step. Such an architecture leads to significant power reduction [7], [65], [66]. A dual-Vdd FPGA architecture reduces power by

applying low-Vdd to FPGA elements on non-critical paths, thereby reducing the dynamic power, and also by turning off the un-utilized parts of the FPGA, resulting in savings in static power too. The granularity of power gating was investigated in [63]. The analysis indicated the trade-offs associated with the area and power savings, for different power gating granularity, with the finer power gating achieving higher power savings but also consuming more area. The results show that even for fine grained power gating granularity, at the level of each slice within a CLB, provides a good area trade-off. It argued that since the similar logic blocks which are idle during the same period tend to lie closer, a coarse power gating granularity can be employed for the logic elements in the FPGA. However, it concludes that a best architecture for area and power trade-off should have a combination of coarse and fine grained power gating flexibility, though no definite best architecture was proposed. In this work the architecture proposed in [66] is chosen, which is in line with industrial FPGAs. It has two types of power supply rails, a low voltage supply rail and a high voltage supply rail. Each of the FPGA logic and routing resource can be connected to either a low voltage supply rail or a high voltage supply rail by programming the transistors connecting the logic/routing resource to either the high voltage or the low voltage supply rail. Such an implementation for a logic block is shown in Fig. 5.1. The use of two supply voltages requires a level converter when a signal crosses from a low voltage net/logic to a high voltage net/logic and vice versa. The architecture which has the level converters at the inputs of the CLBs is chosen [66]. All the nets that are driven by a CLB operate at the same supply voltage as the CLB. The unused routing switches and the CLBs are power gated by switching off both power supply transistors. Furthermore, the SRAM cells have high-Vth transistors to reduce the leakage. This reduces leakage without any delay penalty because the SRAMs need to be programmed only once and do not contribute to the run time performance of the FPGA. The power gating granularity in this architecture can be classified as coarse for the logic elements while it is fine for the routing resources, which is along the lines of the argument presented in [63]. A baseline FPGA implementation, against which the methodology proposed in this work is compared, consists of the dual-Vdd FPGA with the placement and the routing of a netlist on the FPGA followed by a dual-Vdd assignment.

Another architecture which is an extension of the above, is the use of a dynamic Vdd architecture in which the supply voltage can be configured during runtime based on the desired frequency of operation. The dynamic Vdd can be implemented using a dedicated controller which can control the supply voltage to different parts of the chip. However, the methodology proposed in this work would remain similar even with dynamic Vdd implementation. Statically configured voltage islands, as used in this work, are ideally suited for the scenario where the chip is required to run at the constant frequency throughout it duration of operation. Statically configured voltage islands have lesser complexity.

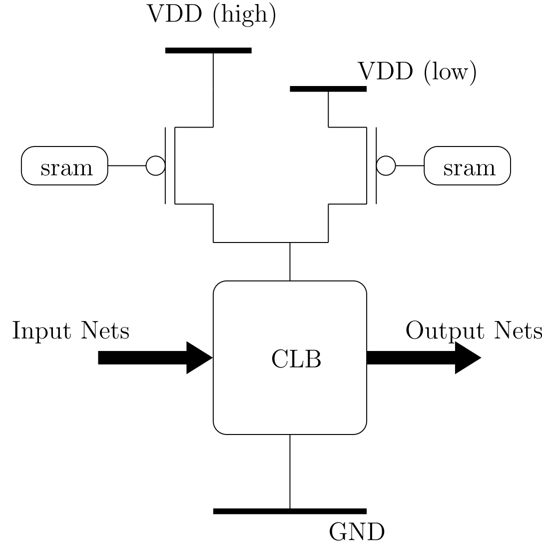


Figure 5.1: Dual-Vdd logic block implementation for power reduction

5.2.1 Statistical Power Model

The total power of any VLSI circuit can be divided into dynamic power and leakage power. Dynamic power is dissipated due to switching of the circuit nodes, whereas leakage power in a transistor is consumed when it is not switching. During the active mode of a system, some parts of the circuit consume dynamic power and the rest leakage power.

Dynamic power is not very sensitive to process parameter variations, and therefore is modeled deterministically [67]. The dynamic power is not very sensitive to process parameters because it is linearly dependent on the process parameter such as the gate length, whereas the leakage power is exponentially dependent on process parameters, such as gate length and threshold voltage. The power model proposed in [68] is adopted to compute dynamic power. The power model computes the dynamic power of logic and routing resources by taking into account the associated capacitances, switching activities and voltage swings at various nodes of the circuit.

The leakage power has two main components, subthreshold leakage and gate leakage. The subthreshold leakage current through a MOSFET is modeled as [57]

$$I_{sub} = I_0\left[1 - exp\left(-\frac{V_{ds}}{V_T}\right)\right].exp\left(\frac{V_{gs} - Vth - V_{off}}{n.V_T}\right), \quad (5.1)$$

where I_0 is a constant dependent on the device parameters for a given technology, V_T is the thermal voltage, V_{off} is the offset voltage which determines the channel current at $V_{gs} = 0$, Vth is the threshold voltage, and n is the subthreshold swing parameter. It

can be seen that the subthreshold leakage is exponentially dependent on the threshold voltage of the MOSFET. This makes the subthreshold leakage very sensitive to variations in the threshold voltage. Also, the subthreshold leakage depends on the channel length of the MOSFET. It should be noted that the threshold voltage is also dependent on the channel length of the transistors due to short channel effects. In particular, a roll-off in the threshold voltage is observed as the channel length is reduced. Since the subthreshold leakage is very sensitive to the threshold voltage, it is important to accurately model the threshold voltage of a MOSFET. In this work, the threshold voltage model from BSIM4 [69] is used. The threshold voltage of a transistor is modeled as

$$\begin{aligned} Vth &= Vth_0 + Vth_{body} - Vth_{SCE} - Vth_{DIBL} \\ &+ Vth_{halo} - Vth_{DITS}, \end{aligned} \tag{5.2}$$

where Vth_{body}, is the body effect, Vth_{SCE} is the short channel effect, Vth_{DIBL} is the drain induced barrier lowering effect, Vth_{halo} is the threshold voltage shift due to the halo pocket implants at the drain and source junctions, and Vth_{DITS} is drain induced threshold shift due to the source and the drain pocket implants. The various components of the threshold voltage are functions of *Leff*, gate oxide thickness and channel doping. The BSIM4 analytical models for threshold voltage dependence on these process parameters have been used.

The variation in threshold voltage due to random dopant fluctuations, $\sigma_{Vth_{rdf}}$, is modeled as [58]:

$$\sigma_{Vth_{rdf}} = \frac{Q.T_{ox}}{\epsilon_{ox}} \sqrt{\frac{N_{ch}.W_{dm}}{3.L.W}}, \tag{5.3}$$

where W_{dm} is the channel depletion width, N_{ch} is the channel doping concentration, and L, W are the channel length and width, respectively. For a function $y = g(x)$, where x is a random variable with variance σ_x^2, the variance of y can be approximated by [59]:

$$\sigma_y^2 = \left(\frac{dg(x)}{dx}\right)^2 \cdot \sigma_x^2. \tag{5.4}$$

The threshold voltage and hence leakage expressed as a function, $f(Leff)$, and the variance in the leakage, $\sigma^2_{sub_{Leff}}$, is computed by (5.4). Similarly, the variance in leakage due to gate oxide thickness variation, $\sigma^2_{sub_{Tox}}$, is calculated. Since the channel length variations, random dopant fluctuations and gate oxide thickness variations are independent the total variance of the threshold voltage is calculated by

$$\sigma^2_{I_{sub}} = \sigma^2_{sub_{rdf}} + \sigma^2_{sub_{Leff}} + \sigma^2_{sub_{Tox}}. \tag{5.5}$$

To compute the variance and mean of the subthreshold leakage, it is modeled as a function of *Leff*, Tox and Vth. Using the statistical model, the mean and variance of subthreshold leakage I_{sub} for a logic element at location (j, k) for i^{th} level, are computed as follows

$$\begin{aligned} E\{I_{sub}\} &= I_{sub}(L_{nom}, Vth_{nom}, Tox_{nom}) \\ &+ \frac{1}{2}\sum_{i=0}^{n}\Bigg(\frac{\partial^2 I_{sub}}{\partial Leff^2}.\sigma_{Leff_{i,(j,k)}}{}^2 \\ &+ \frac{\partial^2 I_{sub}}{\partial Vth^2}.\sigma^2_{Vth(rdf)_{i,(j,k)}} \\ &+ \frac{1}{2}\sum_{i=0}^{n}\Bigg(\frac{\partial^2 I_{sub}}{\partial Tox^2}.\sigma^2_{Tox_{i,(j,k)}}\Bigg), \end{aligned} \tag{5.6}$$

$$\begin{aligned} \sigma^2_{I_{sub}} &= \sum_{i=0}^{n}\Bigg(\Big(\frac{\partial I_{sub}}{\partial Leff}\Big)^2.\sigma^2_{Leff_{i,(j,k)}} \\ &+ \Big(\frac{\partial I_{sub}}{\partial Vth}\Big)^2.\sigma^2_{Vth(rdf)_{i,(j,k)}} \\ &+ \Big(\frac{\partial I_{sub}}{\partial Tox}\Big)^2.\sigma^2_{Tox_{i,(j,k)}}\Bigg), \end{aligned} \tag{5.7}$$

where all the partial derivatives, which represent the sensitivities of the leakage to the process parameters, are computed at the nominal values of these process parameters, at the ith level and location (j, k). Since the random variables, $Leff_{i,(j,k)}$, for the different values of i, j, k are independent, the leakage variances are added to obtain the total leakage variance. By extending this to the complete chip each of the sensitivities (i.e., for grid location i, j and k) represents the total sensitivity due to all the logic elements lying in the location (j, k) at level i. The total leakage current for a chip is the sum of the leakage currents for all the logic elements. Consequently, this amounts to computing the sensitivities of the leakage of each location at each level and using equations 5.6 and 5.7 to compute the variance and mean at each location for all the levels. Since all these locations have independent random variables across the various locations within the level and across the levels, the total leakage variance can be obtained by adding these variances. If $(x_1, x_2, x_3, x_4...)$ are independent random variables, and $y = \sum_i x_i$, then

$$\sigma^2_y = \sum_i \sigma^2_{x_i} \tag{5.8}$$

$$E\{y\} = \sum_i E\{x_i\}. \tag{5.9}$$

An analytical state dependent leakage power model for FPGAs is proposed in [70] and is adopted for this work. The leakage power model takes into account the *probability* of different states for a logic element for the leakage computation, because the leakage current through a logic element depends on its inputs [71]. The work in [70] models the total leakage for a logic element as

$$I_{leak} = \sum_{i} P_i.Leak_i, \tag{5.10}$$

where P_i represents the probability for state i, and $Leak_i$ represents the leakage of the logic element in state i. By extending this, the sensitivity to the process parameters of the leakage is also dependent on the state of the logic element. Therefore, for computing the *total sensitivity* of the leakage for a logic element at i^{th} level and (j,k) location, following equations are used. (The subscripts i, j, k are dropped to retain simplicity of the expressions and convey the essential idea):

$$\frac{\partial I_{sub}}{\partial Leff} = \sum_{n} P_n.\left(\frac{\partial I_{sub}}{\partial Leff}\right)_n \tag{5.11}$$

$$\frac{\partial I_{sub}}{\partial Vth} = \sum_{n} P_n.\left(\frac{\partial I_{sub}}{\partial Vth}\right)_n, \tag{5.12}$$

$$\frac{\partial I_{sub}}{\partial Tox} = \sum_{n} P_n.\left(\frac{\partial I_{sub}}{\partial Tox}\right)_n \tag{5.13}$$

where P_n represents the probability for state n, and $\left(\frac{\partial I_{sub}}{\partial Leff}\right)_n$, $\left(\frac{\partial I_{sub}}{\partial Vth}\right)_n$, and $\left(\frac{\partial I_{sub}}{\partial Tox}\right)_n$ represent the sensitivities for state n of the logic element.

The gate leakage models are described in [70]. Although the gate leakage is orders of magnitude smaller than subthreshold leakage, its variation is modeled in a similar way to the methodology described for the subthreshold leakage. In the remainder of the chapter the term, *leakage*, refers to total leakage including both the gate and subthreshold leakage.

5.3 Proposed Methodology

5.3.1 Preliminaries

In this section, the placement methodology for improving the leakage yield of FPGAs is described. The methodology is used to minimize the impact of the systematic process variations in FPGAs. Since the FPGAs have regular structure and therefore a significant

amount of spatial correlation can be anticipated. Same structures with same layout and orientations but separated by distance lead to similar variability, and hence high spatial correlations. Traditionally, inter-die variations are handled by dividing the chips into different bins for the FPGAs and microprocessors [43]. However, the binning technique cannot manage intra-die variations. In this work the reduction of the impact of intra-die process variations on the leakage is targeted.

Since in a dual-Vdd based FPGA architecture, all the CLBs are identical, the mean leakage of FPGA for an application simply depends on the number of CLBs and routing resources used by the application. The unused CLBs and routing resources do not contribute to variation in leakage because these are turned OFF. The placement location of the logic blocks on the CLBs and the location of the used routing resources do not impact the mean leakage of the application. However, the variance of the leakage is impacted by the placement of the logic blocks because of the spatial correlations in the variations of process parameters. To illustrate this, consider the example in Fig. 5.2. It shows two logic blocks placed on two CLBs, where the other CLBs are not used, and hence, are power gated. The total leakage, its mean, and variance are expressed as

$$I_{leak} = I_{leak1} + I_{leak2}, \quad (5.14)$$

$$E\{I_{leak}\} = E\{I_{leak1}\} + E\{I_{leak2}\}, \quad (5.15)$$

$$\sigma^2_{I_{leak}} = \sigma^2_{I_{leak1}} + 2.r_{i,j}.\sigma_{I_{leak1}}.\sigma_{I_{leak2}} + \sigma^2_{I_{leak2}}, \quad (5.16)$$

where $r_{i,j}$ represents the leakage correlation coefficient when logic blocks 1 and 2 are placed on CLB_i and CLB_j, respectively. In Fig. 5.2 (a), the logic blocks are placed on CLB_1 and CLB_2, with the coefficient of correlation $r_{1,2}$, whereas in Fig. 5.2 (b), the logic blocks are placed on CLB_1 and CLB_9 with leakage correlation coefficient of $r_{1,9}$. Since CLB_1 and CLB_2 are closer, compared to CLB_1 and CLB_9, the leakage correlation coefficient $r_{1,2} > r_{1,9} > 0$, because of a stronger spatial correlation in the former case. This means that $\sigma^2_{I_{sub_a}} > \sigma^2_{I_{sub_b}}$. Therefore, to reduce the variance in the leakage, the logic blocks should be placed far apart to reduce the effect of the positive spatial correlation. It should also be pointed out that placing the logic blocks far apart might also lead to an increase in the critical path delay. However, in FPGAs, there are many logic blocks which do not lie on the critical path and have a large amount of slack available with them. The placement of these logic blocks can be adjusted to reduce the total leakage variance without incurring a large delay penalty. It will be shown in the subsequent section how the trade-off between the leakage variance and timing can be achieved.

5.3.2 Placement Methodology

The statistical leakage power model described in the previous section is implemented in the framework of VPR tool [1]. The VPR implements a timing driven placement for a

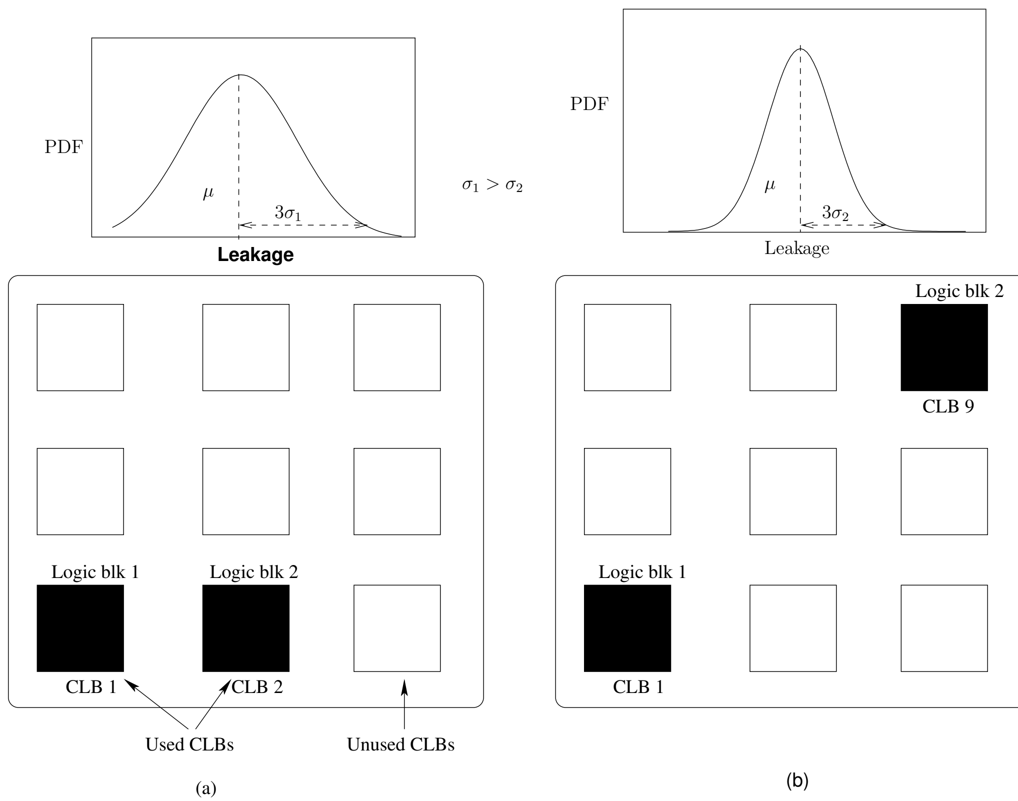


Figure 5.2: Example illustrating the impact of placement on leakage pdf. Spatial correlation causes the variance of leakage to increase.

netlist to map an application to FPGAs[16]. VPR uses the simulated annealing algorithm for the placement of logic blocks on the CLBs. Simulated annealing is an algorithm which mimics the annealing procedure to cool molten metal slowly for producing high quality metal structures. The algorithm begins with a random placement, and repeatedly moves the logic blocks to newer locations and evaluates whether the move can be accepted or not. The acceptance or rejection of a move depends on the placement cost, computed by the VPR. If the move results in a reduction of the placement cost, the move is accepted. If the placement cost increases as a result of the move, there is still some probability that the move is accepted. Accepting some bad moves allows the placement tool to avoid being stuck at some local minimum.

The placement cost used by the VPR is the sum of the timing cost and the wiring cost. The timing cost is computed on a source sink basis. The timing cost for a source sink pair (i, j) is given as [16]

$$Timing_Cost(i,j) = Delay(i,j).Crit(i,j)^{crit_exp}, \tag{5.17}$$

$$Crit(i,j) = 1 - \frac{Slack(i,j)}{D_{max}}, \tag{5.18}$$

where $Delay(i, j)$ is the delay between the source (i) and sink (j), $Crit(i, j)$ is the criticality of the connection between them, $Slack(i, j)$ is the slack available with the source and sink pair, D_{max} is the critical path delay, and $crit_exp$ is for assigning large weights to critical timing connections. The total timing cost is then computed as

$$Timing_Cost = \sum_{(i,j)} Timing_Cost(i,j). \tag{5.19}$$

The wiring cost is estimated by computing the *bounding box* of the placed logic blocks [16]. Essentially, the bounding box is the smallest rectangle within which all the logic blocks lie for the current placement. The wiring cost is an estimate of the wire length used by the netlist. The authors in [16] propose an *auto-normalizing* cost function for the placement. The cost function, $\triangle C$, used for placement is defined as

$$\begin{aligned} \triangle C &= \lambda.\frac{\triangle Timing_Cost}{Previous_Timing_Cost} \\ &+ (1-\lambda).\frac{\triangle Wiring_Cost}{Previous_Wiring_Cost}, \end{aligned} \tag{5.20}$$

where λ is factor for giving different weights to the timing cost and the wiring cost, $\triangle Timing_Cost$ is the change in the $Timing_Cost$ because of the current move, and $\triangle Wiring_Cost$ is the change in the $Wiring_Cost$ because of the current move. The timing driven placement in the VPR optimizes both the wiring cost and the timing cost depending

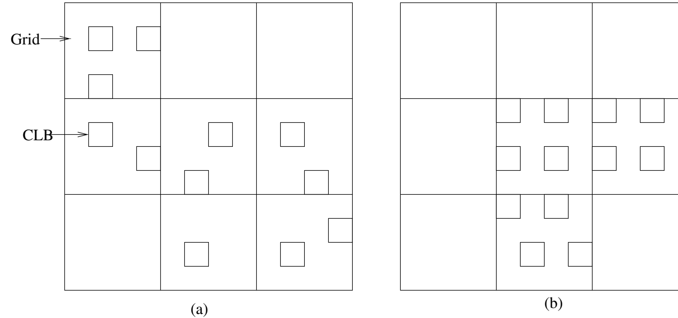


Figure 5.3: (a) Placement is fairly spread out throughout the FPGA, which leads to reduced leakage variance. (b) Placement more concentrated, higher leakage variance.

on the value of λ. In [16], it is proposed that $\lambda = 0.5$ and $crit_exp = 8$, are the best values for the timing and wiring cost trade-offs.

The mean value of leakage depends only on the utilized logic and routing resources, since the un-utilized resources are turned OFF. The variance of the leakage depends on the utilized resources and their location because of the spatial correlations in the process parameters. So the concern is only with the utilized CLBs and routing switches for the power yield. For improving the power yield either the leakage variance should be reduced, or the dynamic power and mean leakage power should be reduced, or all of these should be reduced. Reducing the leakage variance can be achieved by reducing the spatial correlations among the utilized CLBs. This is accomplished by placing the logic blocks further apart. A placement in which the utilized CLBs are evenly spread in the FPGA would have lesser leakage variance than that where the utilized CLBs are concentrated in some part of the FPGA. Typically, a placement tool minimizes the wiring and delay of the circuit by using the placement cost as in 5.20. In general this leads to a closely packed placement to reduce net delays. However, all the logic blocks do not need to be placed close together since many nets have timing slack available with them. In Fig. 5.3 two cases of the placement of the logic blocks on an FPGA is depicted. Fig. 5.3 (a) has the placement which is more evenly spread out than the one in Fig. 5.3 (b), such that the total leakage variance is more in the case of Fig. 5.3 (b) because of increased spatial correlation of the process parameters in Fig. 5.3 (b).

The proposed placement methodology is based on making the placement more uniform, while the delays of the nets are taken into account. The proposed placement algorithm is outlined in Algorithm 1. The parts which are highlighted in italics in the algorithm represent the description of the proposed variability-aware placement technique. First the algorithm is applied to divide the FPGA chip into smaller square grids as shown in Fig. 5.3, and then the occupancy of each of the grids is computed. The occupancy cost is the *grid density*, $Grid_Occ_Factor$, calculated as $\frac{Grid\ Occupancy}{Grid\ Area}$. The occupancy cost, due

Algorithm 1: Variability aware placement algorithm

Divide FPGA into smaller grids;
$Curr_Occ_Cost = 0$;
for *each grid* **do**
  Grid Occupancy = no. of logic blocks in the grid;
  $Grid_Occ_Factor = \frac{Grid\ Occupancy}{Grid\ Area}$;
  $Curr_Occ_Cost = Curr_Occ_Cost + (Grid_Occ_Factor)^{\alpha}$
end
$\triangle\ Occ_Cost = Prev_Occ_Cost$ - $Curr_Occ_Cost$;
$\triangle Placement_Cost = \triangle Timing_Cost + \triangle Wiring_Cost + \triangle Occ_Cost$;
Proceed with placement based on Simulated Annealing;

to a grid, is then computed as $(Grid_Occ_Factor)^{\alpha}$, where α is a factor to control the aggressiveness of the cost function. The occupancy cost is then summed for all the grids to obtain the total occupancy cost. The value of α should be greater than 1, else, if it is 1, the total occupancy cost always remains same for all the placement iterations, and does not have any impact on the placement. If the value is less than 1, the cost function would lead to more concentrated placement, because the $Grid_Occ_Factor$ is always less than 1, leading to higher occupancy costs, $Curr_Occ_Cost$, for lesser grid densities. The total placement cost is then computed as the sum of the timing cost, the wiring cost and the grid occupancy cost.

The new normalized placement cost function which takes into account leakage variance is

$$\begin{aligned} \triangle C &= \lambda . \frac{\triangle Timing_Cost}{Previous_Timing_Cost} \\ &+ (1 - \lambda - \beta) . \frac{\triangle Wiring_Cost}{Previous_Wiring_Cost} \\ &+ (\beta) . \frac{\triangle Occ_Cost}{Previous_Occ_Cost}, \end{aligned} \tag{5.21}$$

where β is a factor to provide a weight to the leakage variation term of the cost function. After the acceptance of a move, the occupancy cost, $Curr_Occ_Cost$, in the current configuration is re-calculated. The $Timing_Cost$ and $Wiring_Cost$, will tend to bring the logic blocks closer together in order to reduce the delay and the wiring length, whereas the Occ_Cost tends to evenly spread out the placement in an FPGA. However, the timing critical logic blocks have a higher $Timing_Cost$ as compared to non-critical logic blocks, and thus will tend to be closer together.

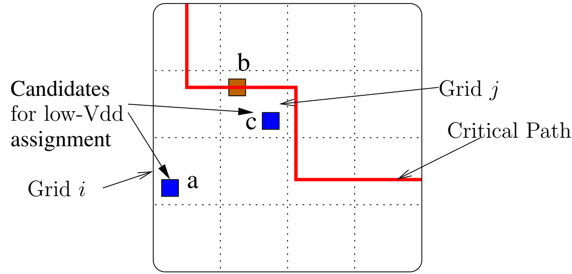


Figure 5.4: Variability-Aware Dual-Vdd assignment technique

5.3.3 Dual-Vdd Assignment

For a programmable dual-Vdd FPGA, once a circuit is placed and routed, a dual-Vdd assignment algorithm assigns high/low Vdd to different CLBs and routing switches according to the slacks available to them. The dual-Vdd assignment algorithms are widely used and one such dual-Vdd assignment algorithm is used here for baseline implementation [72]. The algorithm first creates a graph of the circuit and then arranges all the nodes in the graph according to their level in the graph. Then it begins assigning dual-Vdd with the nodes at the highest level (i.e. primary outputs) and moves backwards towards the lowest level (i.e. primary input). At each level it assigns the low-Vdd to a selected node and check for the timing violation. If the timing the of the circuit gets violated, the node is re-assigned with high-Vdd. This baseline dual-Vdd implementation is then compared with the implementation based on the proposed variability aware methodology to evaluate the improvement in the power yield.

To further improve the power yield, a new variability aware dual-Vdd assignment is proposed in this work as outlined in Algorithm 2. This dual-Vdd assignment algorithm is applied after the variability aware placement and routing step is completed. The algorithm divides the FPGA with all high-Vdd implementation into square grids (Fig. 5.3). Then all the grids are arranged in a Priority Queue, PQ, such that the head of the priority queue contains the grid with the maximum number of logic blocks. Then the dual-Vdd assignment algorithm is applied with the assignment in the grid with maximum number of logic blocks being chosen first. A grid with a high density of logic blocks leads to greater leakage variability due to increased spatial correlation effect.

For example, consider the case with two grids i, j, such that each grid can contain a maximum of four logic blocks, and the spatial correlation is restricted to only one grid (ignore the grid boundary case for simplicity), i.e., there is no parameter spatial correlation across the grids. Also, say, the occupancy of grid i is 1 CLB, a, and that grid j has two CLBs, b and c. The critical path is such that either a or c can be assigned a low-Vdd as

shown in Fig. 5.4. When all the utilized CLBs are high-Vdd, the total leakage variance is

$$\begin{aligned} \sigma_{leak}^2 &= \sigma_a^2(HVdd) + \sigma_b^2(HVdd) \\ &+ 2r_{bc}.\sigma_b(HVdd).\sigma_c(HVdd) + \sigma_c^2(HVdd), \end{aligned} \tag{5.22}$$

when CLB a is assigned low-Vdd, the leakage variance is expressed by

$$\begin{aligned} \sigma_{leak}^2 &= \sigma_a^2(LVdd) + \sigma_b^2(HVdd) \\ &+ 2r_{bc}.\sigma_b(HVdd).\sigma_c(HVdd) + \sigma_c^2(HVdd), \end{aligned} \tag{5.23}$$

when CLB c is assigned low-Vdd, the leakage variance is given by

$$\begin{aligned} \sigma_{leak}^2 &= \sigma_a^2(HVdd) + \sigma_b^2(HVdd) \\ &+ 2r_{bc}.\sigma_b(HVdd).\sigma_c(LVdd) + \sigma_c^2(LVdd), \end{aligned} \tag{5.24}$$

where r_{bc} is the correlation coefficient between blocks b and c. Under the assumption that all the low-Vdd CLBs have the same leakage and its variance, and all the high-Vdd logic blocks have the same leakage and its variance, it can be seen from 5.22 - 5.24, that assigning a low-Vdd to CLB c, leads to the least leakage variance. This rationale is followed in this work by using Algorithm 2, because for larger circuits such a behavior is typically observed. The parts in the algorithm highlighted in italics represent the description of the proposed technique. Therefore, assigning low-Vdd to regions with a higher utilized CLBs density results in a smaller leakage variance.

Algorithm 2: Variability aware dual-Vdd assignment algorithm

```
Assign high-Vdd to all blocks and routing resources;
 Divide FPGA into smaller grids;
Perform placement as in Algorithm 1;
Perform routing;
 Sort the grids in a priority queue, PQ, such that highest occupancy grid is at the
head;
while PQ not empty do
    igrid = PQ(head);
    for each logic block, iblk, in igrid do
        Assign low-Vdd to iblk and associated routing resources;
        if slack < 0 then
            Reassign high-Vdd to iblk and associated routing resources;
        end
    end
end
```

5.4 Evaluation, Results and Discussions

5.4.1 Experimental Details

In this work 45nm Berkeley Predictive Models is chosen as the technology node for the simulations [69, 73]. It is demonstrated in [24] that three levels in the quad-tree model for the process variations lead to sufficiently accurate results for the timing analysis. Instead of using 4^i grids at level i, a scheme in which the grid size at zeroth level is the size of the FPGA, first level has a grid size of 12x12 FPGA tiles, second level has a grid size of 8x8 FPGA tiles, and the third level has the size of 4x4 FPGA tiles. The last level represents the random independent variations. In the absence of actual measurement results, five levels for in the quad-tree model is chosen for modeling process parameter variability. The channel length variations have been modeled at levels 1, 2, and 3 which represent intra-die variations and models the spatial correlation in process parameters between different parts of a chip. A 3σ variation of 20% in L_{eff}, and a 3σ variation of 15% in Tox, is assumed, and this is distributed equally over these three levels, in absence of actual fabrication data for spatial correlation, similar to the work in [23]. The variation in the threshold voltage due to random dopant fluctuations is modeled at the last level representing an independent random variable. The delay and the power of level converters at the input of the CLBs is ignored. This is because the power of a level converter is negligible compared to that of the logic block [6]. The delay of the interconnects dominate the delay of a path and the delay of the level converter is only a fraction of the delay of a logic block [6]. The A set of MCNC benchmarks has been selected in this work for obtaining the results.

5.4.2 Estimating leakage distribution and yield

The leakage distribution of a circuit element is close to lognormal [74]. Consequently, the leakage distribution of the complete circuit is approximated by a lognormal distribution, but the distribution tends to become a normal distribution as explained below. Given the quad-tree model for modeling the variations in the process parameters, and thus, for computing the leakage mean and variance, independent random variables are added for computing the total leakage. This amounts to adding n random variables with lognormal distribution as,

$$I_{sub} = I_1 + I_2 + ... + I_n = e^{g_1} + e^{g_2} + ... + e^{g_n}, \tag{5.25}$$

where $I_1, I_2, ...I_n$ have a lognormal distribution, and $g_1, g_2, ...g_n$ have a normal distribution. The *Wilkinson* approximation [75] is used to approximate the sum of the lognormals as another lognormal. Wilkinson's approach approximates the mean and variance of the sum

of lognormals by matching the first two moments of the distribution as follows:

$$E\{I_{sub}\} = \mu_1 + \mu_2 + \mu_3 + ... + \mu_n, \quad (5.26)$$

$$\sigma^2_{I_{sub}} = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + ... + \sigma_n^2, \quad (5.27)$$

where μ_i and σ_i are the means and standard deviations associated with a grid (j,k) for a layer, in the layered grid model. The probability distribution function of a lognormal is given by

$$f(x) = \left(\frac{1}{x\sqrt{2\pi q}}\right).exp\left(\frac{-(ln(x) - p)^2}{2q^2}\right), \quad (5.28)$$

where p, and q are the parameters of the lognormal distribution. The mean and variance of a lognormal distribution are expressed as [74]

$$E(X) = exp\left(p + \frac{q^2}{2}\right), \quad (5.29)$$

$$\sigma^2_X = exp\left(2.(p + q^2)\right) - exp(2.p + q^2). \quad (5.30)$$

The lognormal random variable X is expressed as $X = exp(Y)$, where Y is a normal random variable with the mean and standard deviation of (p, q). Since 5.26 and 5.27 compute the mean and variance of the lognormal, the parameters p and q of the lognormal distribution in 5.28 needs to be computed to calculate the PDF of the lognormal. The parameters can be computed as follows to obtain the PDF of the distribution [74]

$$p = \left(\frac{1}{2}\right).log\left(\frac{E^4(X)}{E^2(X) + \sigma^2_X}\right) \quad (5.31)$$

$$q = log\left(\frac{\sigma^2_X + E^2(X)}{E^2(X)}\right). \quad (5.32)$$

For circuits with a large number of elements, the leakage current distribution approaches a normal distribution, as predicted by the Central Limit Theorem [59]. Therefore, the shape of the leakage distribution approaches a Gaussian distribution [74]. For the distribution of total power, the dynamic power is added to the mean of the leakage power to get the mean of the total power. The standard deviation of the total power is the standard deviation of the leakage power, because the dynamic power does not vary.

Power yield estimation is carried out as follows. The Cumulative Distribution Function (CDF) of the power distribution of the chip is obtained by calculating

$$CDF(P) = \int_{-\infty}^{\infty} f(P)dP \quad (5.33)$$

The CDF of the power distribution directly gives the power yield for the FPGA.

5.4.3 Results and Discussions

Table 5.1 lists the results of the leakage variability aware placement for FPGAs. Column 7 shows the reduction in leakage variability from the proposed methodology. The reduction in leakage variability increases the probability of a design to meet a target power budget for a design and thereby improve the power yield of the design. For comparing the improvement in the power yield for a benchmark, the total power of the baseline implementation for which the power yield is 90% is computed. Then the yield with variability aware placement for the same value of total power is calculated as follows

$$CDF_{baseline}(P_1) = 0.90, \quad (5.34)$$

$$Yield_Improvement = CDF_{variability_aware}(P_1) - 0.90. \quad (5.35)$$

The last column in Table 5.1 shows that the power yield improvement due to the variability-aware placement is between 3% and 9%. The improvements in yield translate directly into savings in the number of chips being discarded and hence economic benefits. The results of 3%-9% yield improvements are important and previous works on statistical optimization (timing and/or power) have reported similar improvements for ASICs [26].

To analyze the impact of the proposed technique on the speed of the circuit, a deterministic timing analysis is performed for both the baseline, and the variability aware implementations. FPGAs have longer critical paths as compared to those of custom VLSI designs and ASICs. For longer critical paths the variations in the delay is smaller because of the averaging effect. Furthermore, the delay is not as sensitive to the channel length variations, gate oxide thickness variations and the random dopant fluctuations as the leakage is, because leakage is exponentially dependent on threshold voltage which is affected by these process parameters. As a result, deterministic delay computation gives a reasonably accurate insight into the impact of variability aware placement methodology on the performance of a circuit. Some delay penalty is associated with the variability aware placement. This is the result of the reduced weight to the wire length cost in the placement cost function due to some weight being attributed to occupancy cost term of the placement cost function. An average delay penalty of 5% is observed for the proposed leakage variability-aware CAD technique.

Also, the reduction in leakage variability depends on the logic utilization factor. The second column in Table 5.1 indicates the logic utilization of the FPGA for the different benchmarks. When the logic utilization factor is low, the variability aware placement tool has more flexibility for optimization. The logic utilization for some benchmarks, such as *des*, and *bigkey* are low because these benchmarks are I/O intensive. For example, *des* has 200 CLBs with 256 inputs and 245 outputs requiring a larger FPGA to fit the I/O pads on the FPGA. Similarly *bigkey* has 214 CLBs with 229 inputs, 197 outputs.

Table 5.1: Results of variability aware placement

Benchmark	Utilization	Baseline		Variability Aware		Leakage Variability Reduction	Power Yield Improvement
		Mean Total Power μW	$3.\sigma_{leak}$ μW	Mean Total Power μW	$3.\sigma_{leak}$ μW		
alu4	40%	5711	944.3	5404	805.5	14.7%	9.1%
apex2	50%	6702	1068	6654	962	9.9%	3.9%
apex4	34%	3717	881	3650	746.5	15.3%	5.8%
bigkey	24%	9236	718.5	9063	618.8	13.9%	8.5%
des	16%	8487	621	8427	610.5	1.7%	4.2%
dsip	19%	8400	585	8275	477.7	18.3%	8.6%
elliptic	50%	9781	1239	9049	1190	4%	9.4%
ex1010	67%	7175	1293	6416	1066	17.6%	9.6%
frisc	50%	8834	1948	6345	1151	41%	9.7%
misex3	37%	5752	1000	5472	890	11%	8.6%
s298	50%	5157	821.7	5054	750	8.7%	6.1%
seq	47%	6965	1141	6810	1040	8.9%	6.3%
spla	52%	8013	1478	7402	1323	10.5%	9.3%
tseng	27%	4385	892.1	3939	475.6	47%	9.7%

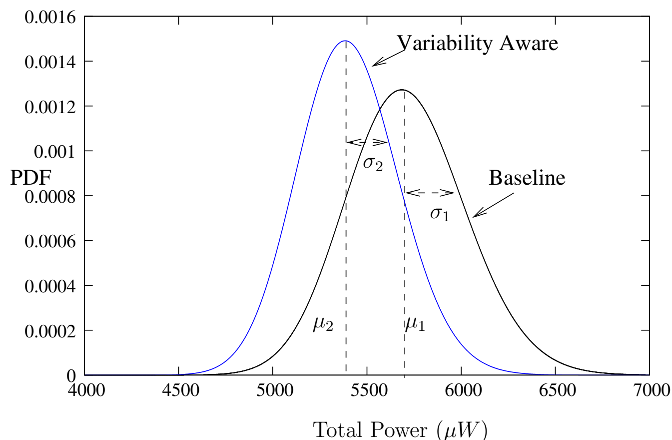


Figure 5.5: Power distribution without and with variability aware placement for alu4

Fig. 5.5 shows the power distributions for the benchmark *alu*4, for baseline and variability aware implementations, where mean power and its variance decrease for the variability aware implementation. Similarly, Fig. 5.6 shows the power distributions for the benchmark *seq*, where the mean power and its variance decrease for the variability aware implementation.

The variability aware implementation spreads out the placement of CLBs leading to increased usage of routing resources with a small delay penalty. However, the dynamic power does not increase significantly, even though more routing resources are used in variability-aware implementation, because more logic and routing resources are assigned low-Vdd due to extra slacks available in different paths of the circuit. This, coupled with reduced leakage variability lead to a total power yield (dynamic and leakage) improvement between 3% and 9%. However, it is important to note that for low power applications in which devices tend to remain in standby mode for most of the time followed by short bursts of activity, the leakage power would be the dominant factor contributing to the total energy consumption and hence the reducing leakage variability becomes even more important for such low power applications.

The contribution of the random dopant fluctuations to the total leakage variance is insignificant compared to the contribution of channel length variations and the gate oxide thickness variations. For example, the benchmark alu4 has 3σ total leakage variation of $944\mu W$, where only $27\mu W$ is due to random dopant fluctuations, because the channel length variations and gate oxide thickness variations exhibit spatial correlation, whereas the random dopant fluctuations are independent for each transistor. Spatially correlated variations have a higher impact on the variance as compared to random variations because the random variations have an averaging effect which reduces the variance. This causes the variation in the leakage due to the random dopant fluctuations to be very small. However,

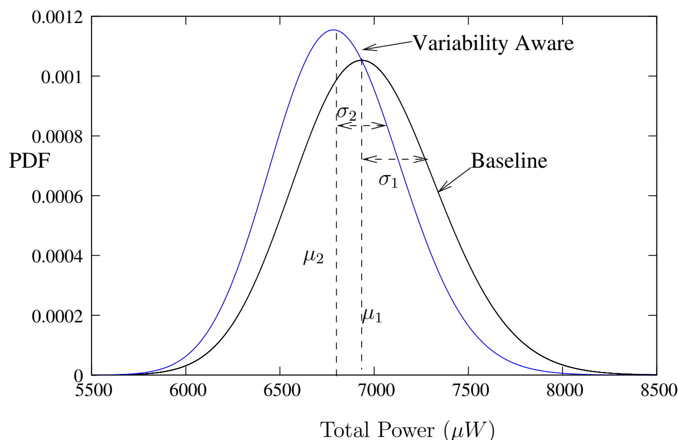


Figure 5.6: Power distribution without and with variability aware placement for seq

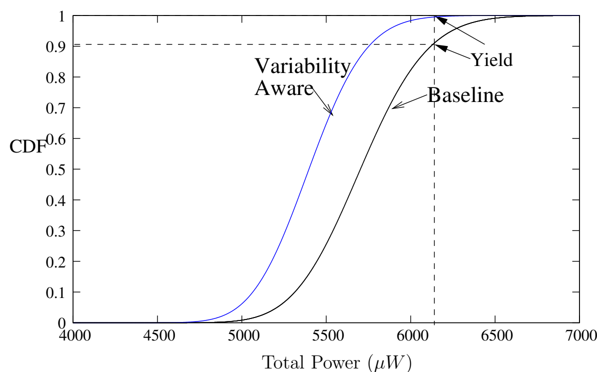


Figure 5.7: CDF of power distributions for alu4 for the baseline implementation and variability aware placement

these variations affect the mean value of the leakage and ignoring the random dopant fluctuations leads to error in the mean leakage value estimation.

Fig. 5.7 shows the CDF of the power distributions for *alu*4, computed analytically, for the baseline implementation and variability aware implementation. It can be seen from the two curves that the power yield improves for the variability aware implementation.

The technique proposed in this thesis is applicable to such class of FPGAs which supports turning off of the un-utilized parts of the FPGA, or turning off of those parts of the FPGA which are idle. Although, the analysis in this thesis does not include the active and idle time of different parts of the FPGA, incorporating the idle durations (if known) for different parts of the FPGA would yield even better results, because that would not only increase the flexibility in placement but would also provide additional parts of the FPGA which can be turned off during idle periods, resulting in lesser leakage and its variability.

5.5 Conclusions

Leakage power has become a significant challenge in the design of FPGAs in nanometer technologies, with process variations aggravating the problem. This chapter introduces a leakage variability aware CAD for dual-Vdd FPGAs, to improve the power yield of FPGAs. The chapter proposes a power variability aware placement for FPGAs, which is intended for reducing the leakage variance due to the spatial correlation in the process parameters. A new placement cost function is proposed for the variability aware placement. The results indicate that there is as much as 9% power yield improvement by using the proposed CAD algorithms. Also, the power distributions are computed for the entire chip and it is indicated that the variability aware implementation has less power variation compared to an implementation with deterministic algorithms. The novel CAD methodology is flexible enough to incorporate any number of process parameters in the model. In addition, environmental parameters such as the power supply and the temperature across the chip vary. In future work the intention is to incorporate these environmental parameters in the model for analyzing and improving the architecture and CAD tools for FPGAs.

Chapter 6

Interconnect Design under Process Variations

6.1 Introduction

Interconnects in FPGAs occupy most of the chip area and contribute a major portion to the total chip delay and leakage power consumption. This is because the routing flexibility needs to be high enough to allow complex circuits to be implemented on FPGAs, resulting in longer interconnects with many switches and buffers. Motivated by the above challenges in designing low power FPGAs in nanometer technologies this work develops and proposes a variability-aware leakage power optimization technique under delay constraints for FPGA interconnects. The routing architecture in an FPGA consists of evenly spaced buffers connected by routing wires. Such an architecture is shown in Fig. 6.1. Here all the buffered switches are identical and the distance between any two buffers is same throughout the interconnect.

Each of the switches is composed of two inverters and a pass transistor as shown in Fig. 6.2. The pass transistor is controlled by an SRAM cell. If the SRAM cell is programmed to output 1, the pass transistor is turned on, otherwise it is switched off. The aim of this work is to estimate the optimum sizes and threshold voltages of the different transistors in the

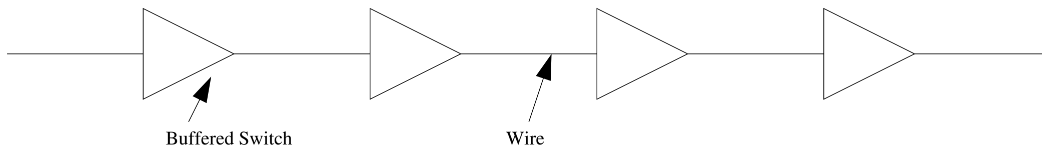


Figure 6.1: Interconnect in FPGAs having buffered switches evenly spaced.

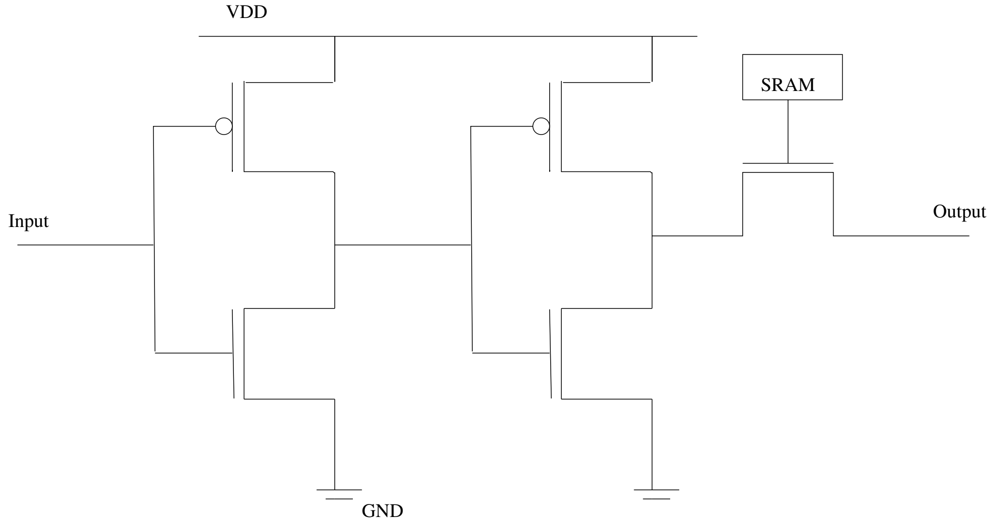


Figure 6.2: Schematic of a buffered switch. The SRAM cell controls the pass transistor.

switch, such that the target delay for the interconnect is achieved while the leakage power and its variability is reduced under process variations [76]. SRAM cell is not considered for optimization because an extensive published research exists on optimizing SRAMs for leakage power, and it is assumed that the FPGA fabric uses such an optimized SRAM cell. Further, the SRAM cells in FPGAs do not change state during run time and therefore can be easily optimized for leakage.

6.2 Impact of Process Variations on Leakage and Delay

6.2.1 Process Parameters and Variations

The process variation under consideration in this work are channel length, and random dopant fluctuations. This results in three model parameters having variability, L_{eff} (effective channel length of transistor), Vth_{nmos} (threshold voltage of NMOS), and Vth_{pmos} (threshold voltage of PMOS). This is because the gate oxide thickness is a well controlled process. Therefore gate oxide thickness is not considered as a parameter having variations in this work [77]. Due to process variations these parameters need to be modeled as random variables. In this work the random variables for the process parameters are modeled as Gaussian random variables. Each of the buffered switches consists of two inverters and a pass transistor connected as shown in Fig. 6.2. In each inverter, NMOS and PMOS have a fixed ratio of their widths, sized to provide equal rise and fall times.

6.2.2 Leakage Modeling

The subthreshold leakage current through a MOSFET is modeled as (BSIM4):

$$I_{sub} = I_0 \frac{W}{L_{eff}} \left[1 - exp\left(-\frac{V_{ds}}{V_T}\right)\right].exp\left(\frac{V_{gs} - Vth - V_{off}}{n.V_T}\right) \tag{6.1}$$

where I_0 is a constant dependent upon device parameters for a given technology, W is the width of the transistor, L_{eff} is the effective channel length of the transistor, V_T is the thermal voltage, V_{off} is the offset voltage which determines the channel current at $V_{gs} = 0$, Vth is the threshold voltage and n is the subthreshold swing parameter. It can be seen that the subthreshold leakage is exponentially dependent on the threshold voltage of the MOSFET. This makes the subthreshold leakage sensitive to variations in threshold voltage. It should be noted that the threshold voltage is also dependent on the channel length of the transistors because of short channel effects. In particular, a roll off in threshold voltage is observed as the channel length is reduced. This makes subthreshold leakage also sensitive to the channel length of the MOSFET. In this work analytical models from BSIM4 are used. Gate leakage is orders of magnitude smaller than subthreshold leakage because of use of high-K gate dielectric materials and hence it is not considered in this work.

6.2.3 Delay Modeling

A simplified expression for the delay of the complete interconnect can be written as:

$$\begin{aligned} Delay &= n.\big(Tdel_{sw} + Rsw_{on}.(l_{wire}.C_{wire} \\ &+ 4.Cin_{sw})\big) \end{aligned} \tag{6.2}$$

where n is the number of switches in the interconnect, $Tdel_{sw}$ is the intrinsic delay of the switch, Rsw_{on} is the on-resistance of the switch, l_{wire} is the length of the wire between two switches, C_{wire} is the total capacitance of the wire between two switches, and Cin_{sw} is the input capacitance of the switch. It is assumed that on an average each switch sees a load equivalent to up to four switches in the routing fabric, apart from the wire load. This means that each switch sees a total load capacitance of $(l_{wire}.C_{wire}+4.Cin_{sw}))$. Although a more accurate modeling considering the resistances of the interconnects can be formulated, the above expression provides a fairly good fidelity and since the main purpose of the work is to improve the leakage power yield, the above expression is sufficiently accurate. The on

resistance of the switch Rsw_{on}, $Tdel_{sw}$, and Cin_{sw} are

$$Cox = \frac{e_{ox}}{t_{ox}} \tag{6.3}$$

$$k = \mu.Cox \tag{6.4}$$

$$Vd_{sat} = L_{eff}.\frac{Vsat}{\mu} \tag{6.5}$$

$$Id_{sat} = k.\frac{W}{L_{eff}}.\big((Vdd - Vth).Vd_{sat} - \frac{Vd_{sat}^2}{2}\big) \tag{6.6}$$

$$R_{eq} = (0.69)\frac{3}{4}.\frac{Vdd}{Id_{sat}}.\big(1 - \frac{7}{9}\lambda.Vdd\big) \tag{6.7}$$

$$Ron_{inv1} = \frac{Req_n + Req_p}{2.Size_{inv1}} \tag{6.8}$$

$$Ron_{inv2} = \frac{Req_n + Req_p}{2.Size_{inv2}} \tag{6.9}$$

$$R_{pass} = \frac{Req_n}{Size_{pass}} \tag{6.10}$$

$$Rsw_{on} = R_{pass} + R_{inv2} \tag{6.11}$$

$$Tdel_{sw} = R_{inv1}(Cd_{nmos_inv1} + Cd_{pmos_inv1} + Cox_{inv2}) \tag{6.12}$$

$$Cin_{sw} = Cox_{inv1} \tag{6.13}$$

where e_{ox} is the permittivity of the gate oxide, t_{ox} is the thickness of the gate oxide layer, μ is the mobility of charge carriers, $Vsat$ is the saturation velocity of the charge carriers, λ is an empirical parameter, $Size_{inv1}$ is the multiples of minimum width of transistors for first inverter, Cd_{nmos} is the diffusion capacitance of NMOS, Cox_{inv1} is the gate oxide capacitance of the first inverter.

6.2.4 Variation Modeling

The leakage and delay of the switches in the interconnect are functions of threshold voltage and channel length of the transistors. The standard deviation of threshold voltage is modeled using (5.3). The variance of a function of a random variable can be calculated using (5.4) The threshold voltage can be expressed as a function, $f(L_{eff})$, and the variance in threshold voltage $\sigma^2_{Vth_{Leff}}$ can be computed using (5.4). Since the channel length variance and discrete dopant variations are independent the total variance of threshold voltage can now be calculated as:

$$\sigma^2_{Vth} = \sigma^2_{Vth_{rdf}} + \sigma^2_{Vth_{Leff}} \tag{6.14}$$

The process parameters have intra-die and inter-die variations. The inter-die variations are such that the process parameters vary across dies, whereas in the case of inter-die variations, there is within-die spatial variation of process parameters. The intra-die process variations are handled by binning, and therefore is not considered in this work, and this work is focused on inter-die spatial variation of process parameters. The random dopant fluctuations and channel length variations are independent variations because their sources are different. This means that for a single switch we need to calculate variation in delay and leakage due to three independent random variations, δL_{eff}, δVth_{nmos_rdf}, and δVth_{pmos_rdf}. Spatial correlation of process parameters decrease quadratically with distance. Therefore, in absence of empirical data, the correlation coefficient for L_{eff} is modeled as follows:

$$r(i,j) = \frac{1}{1+(d_i - d_j)^2} \tag{6.15}$$

where, $r(i,j)$ is the correlation coefficient for i^{th} and j^{th} switches, and d_i, d_j are the location coordinates of ith and j^{th} switches. Once the correlation coefficient is known, the corresponding covariance matrix can be calculated.

To compute the variance and mean of subthreshold leakage for a MOSFET, subthreshold leakage is modeled as a function of L_{eff} and Vth using the First Order Second Moment (FOSM) technique as follows, similar to (5.6) and (5.7), and are shown here as a function of L_{nom} and Vth_{nom} to make this section more readable and for quick reference:

$$\begin{aligned} E\{I_{sub}\} &= n.I_{sub}(L_{nom}, Vth_{nom}) \\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Bigg(\frac{\partial^2 I_{sub}}{\partial L_{eff}^2}.\sigma_{Leff_i}.\sigma_{Leff_j} \\ &. \quad Cov(L_{eff_i}, L_{eff_j})\Bigg) \\ &+ \frac{1}{2}.n.\frac{\partial^2 I_{sub}}{\partial Vth_{nmos}^2}.\sigma^2_{Vth_{nmos}} \\ &+ \frac{1}{2}.n.\frac{\partial^2 I_{sub}}{\partial Vth_{pmos}^2}.\sigma^2_{Vth_{pmos}}, \end{aligned} \tag{6.16}$$

$$
\begin{aligned}
\sigma^2_{I_{sub}} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\Bigg((\frac{\partial I_{sub}}{\partial L_{\mathit{eff}_i}})(\frac{\partial I_{sub}}{\partial L_{\mathit{eff}_j}}) \\
&. \quad Cov(L_{\mathit{eff}_i}, L_{\mathit{eff}_j})\Bigg) \\
&+ \quad n.(\frac{\partial I_{sub}}{\partial Vth_{nmos}})^2.\sigma^2_{Vth_{nmos}} \\
&+ \quad n.(\frac{\partial I_{sub}}{\partial Vth_{nmos}})^2.\sigma^2_{Vth_{nmos}},
\end{aligned} \tag{6.17}
$$

where all the partial derivatives, which represent the sensitivities of leakage to the process parameters, are computed at nominal values of these process parameters.

The above expressions were developed for leakage power, which is a function of threshold voltage and L_{eff}. Proceeding in a similar way, the expected value, $E\{Delay\}$, and variance, σ^2_{Delay}, of the total interconnect delay can be calculated.

6.3 Proposed Methodology

This section discusses the optimization methodology used in this work for minimizing leakage and its variability under constraints. The objective is to minimize the leakage and its standard deviation, under the constraints of the device sizes, threshold voltages and interconnect delay. The design variables in this problem are as follows:

- s_1 is the size of the first inverter in terms of its minimum size. s_2 is the size of the second inverter in terms of its minimum size. This means that the widths of transistors of first and second inverters in the switch are multiplied by s_1 and s_2.
- s_3 is the size of the pass transistor in terms of minimum width of the transistor.
- s_4, s_5 are the sizes of the channel lengths in terms of minimum channel length for the first and the second inverters respectively. Gate length biasing is a technique which is used to reduce leakage [78].
- s_6 is the size of the channel length of the pass transistor in terms of minimum channel length.
- s_7, s_8 are the scaling factors for the threshold voltages of the NMOS and PMOS. This allows adjusting the threshold voltage of the devices.

6.3.1 Deterministic Optimization

The deterministic optimization technique is implemented without accounting for variability in process parameters. The deterministic optimization is implemented to show the improvement that a variability-aware optimization technique would have over a deterministic optimization technique. Based on the results from deterministic optimization we compute the improvement in leakage power yield that the variability aware optimization would provide. Essentially, the deterministic optimization has the objective function as $f = Total_Leakage$, where $Total_Leakage$ is the leakage value computed without considering process variability. The constraints for the deterministic optimization are same as for variability aware optimization and are given by (6.19)-(6.28), except for the constraint on delay, where again the delay is computed without accounting for process variability.

6.3.2 FOSM Based Model: Accounting for Variability

In this section, the mathematical programming technique using the FOSM model for leakage and delay is described. The model is shown below.

Objective Function :

$$\sigma_{I_{sub}} \tag{6.18}$$

Subject to :

$$1 \leq s_1 \leq 4 \tag{6.19}$$
$$1 \leq s_2 \leq 4 \tag{6.20}$$
$$1 \leq s_3 \leq 4 \tag{6.21}$$
$$1 \leq s_4 \leq 1.2 \tag{6.22}$$
$$1 \leq s_5 \leq 1.2 \tag{6.23}$$
$$1 \leq s_6 \leq 1.2 \tag{6.24}$$
$$0.7 \leq s_7 \leq 1.4 \tag{6.25}$$
$$0.7 \leq s_8 \leq 1.4 \tag{6.26}$$
$$0 \leq E\{Delay\} + 3.\sigma_{Delay} \leq Target_Delay \tag{6.27}$$
$$0 \leq E\{Leakage\} \leq Target_Leakage \tag{6.28}$$

The objective function is the standard deviation of leakage, which is a measure of leakage variability. Therefore the leakage variability is directly optimized in the proposed variability-aware optimization. The constraints on $s_1, s_2, s_3, s_4, s_5, s_6$ are based on the biggest size transistors that can be used without increasing significantly the total area of a

chip, and not significantly altering the layout of the chip. Therefore the maximum channel length allowed is only 20% larger than the minimum channel length for this technology. The constraints on s_7 and s_8 keep the threshold voltages of NMOS and PMOS within bounds. Finally, the $Target_Delay$ value used in the variability-aware optimization is chosen as the $(\mu + 3\sigma)$ delay value from the deterministic optimization. Therefore this implies that the circuit delay remains the same for both the deterministic and the variability aware optimization techniques, which provides a fair basis for comparison. The $Target_Leakage$ value is the expected value of leakage as obtained from deterministic optimization. This is done to ensure that the mean leakage for the variability-aware leakage optimization is bounded in the same way as that for the deterministic optimization.

6.4 Evaluation, Results and Discussions

The 65nm predictive model is chosen as the technology node for simulations [69], [73]. A 3σ variation of 30% in L_{eff} is assumed. For experimental purposes it is assumed that the number of switches is 16 and length of wire between switches is $200\mu m$. The wires are assumed to be on the intermediate metal layer. The minimum sized inverter consists of NMOS with minimum channel length and minimum width, and PMOS with minimum channel length and 1.8 times the minimum width. This is to ensure that the rise and fall times at the output remain same. The $(\mu + 3\sigma)$ delay for both the deterministic and variability-aware optimizations are kept the same. A standard non-linear constrained optimization package from MATLAB is used for solving the optimization problems. The non-linear optimization technique uses the Sequential Quadratic Programming (SQP) method to solve the optimization problem. This method is based on the solution of the Kuhn-Tucker (KT) equations [79]. Constrained quasi-Newton methods are employed to guarantee superlinear convergence. The leakage distribution of a circuit element is close to lognormal [74]. Therefore, the leakage distribution of the complete circuit is approximated as a lognormal [74].

Table 6.1 shows the results of variability aware optimization compared with the deterministic optimization. It can be seen that s_2 and s_3, which represent the sizes of the second inverter and the pass transistor are obtained as the upper limit for these variables for both variability-aware and deterministic optimizations. This is because the second inverter and the pass transistor drive the wire having large capacitance. The channel lengths of the transistors in the first inverter, second inverter and pass transistor are 20%, 15% and 15.0% larger than the minimum L_{eff}, respectively, for the variability-aware optimization. This is because leakage and its variability are reduced exponentially with increase in channel length. In case of deterministic optimization, the channel length of only the first inverter is non-minimum. Finally, it can be seen that the deterministic optimization

Table 6.1: Results of variability aware and deterministic optimizations

Parameter	Deterministic Optimization	Variability-Aware Optimization
s_1	1.39	1.31
s_2	4.0	4.0
s_3	4.0	4.0
s_4	1.18	1.2
s_5	1.0	1.15
s_6	1.0	1.15
s_7	1.05	0.97
s_8	1.14	1.03
Delay $(\mu + 3\sigma)$	2.32ns	2.33ns
Mean Leakage, $\sigma_{Leakage}$	28.1 nA, 10.6 nA	28.4 nA, 7.8 nA

actually increases the threshold voltage of the NMOS and PMOS transistors by 5% and 14% respectively, whereas in the case of the variability-aware optimization the threshold voltages of the NMOS transistors decrease by 3% and that of PMOS transistors increase by 3%. Since leakage is exponentially dependent on the threshold voltage, even small changes in the threshold voltage impacts the leakage current.

The leakage and delay results shown in the table are obtained from Monte Carlo simulations for accurate results. Monte-Carlo simulations were used by generating samples of process parameters for normal distribution and then simulating the circuit for each sample to generate the data for power and delay. Once the power and delay values for each sample were determined, the mean and variance of delay and power can be computed. The results for leakage indicate that although the mean leakage remains almost the same for both the optimizations, the standard deviation of leakage reduces by 26.4% for the variability-aware optimization. This leads to improvement in the leakage yield of the design. Even small improvements of the order of few percents in yield result in significant savings in cost. To estimate the improvement in leakage yield, the yield point is selected as 40 nA. Any other target value can be chosen, and this value is selected just for illustration purposes. The deterministic optimization leads to an yield of 87.5%, whereas the variability-aware optimization leads to an yield of 92.04%. This results in an improvement of 4.54% in leakage power yield. It should be noted that the $(\mu + 3\sigma)$ delay of the circuit remains almost constant as can be seen from Table 6.1. This means that the 4.54% improvement in leakage yield and 26.4% reduction in leakage variability is obtained without any delay penalty, using the proposed variability-aware optimization technique.

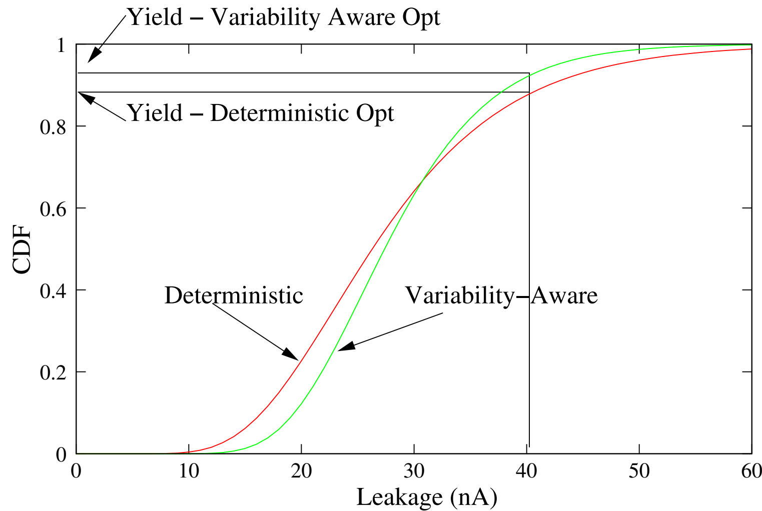


Figure 6.3: CDFs for the deterministic and the variability-aware optimizations.

Fig. 6.3 shows the Cumulative Distribution Function (CDF) plots for the deterministic and variability-aware optimizations. It can be seen from the plots that at the given target leakage value of 40 nA, the leakage yield is more in the case of variability-aware optimization compared to the deterministic optimization.

6.5 Conclusion

This chapter presented a CAD technique for modeling and optimizing leakage under variability with delay constraints for an interconnect in FPGAs. The proposed CAD technique is based on a mathematical programming methodology. The results indicate that leakage variability is reduced by 26% and the leakage yield improves by 4.54% for the chosen target leakage value. Since, the dominant power is consumed in the routing fabric of an FPGA, the proposed technique can lead to significant savings in leakage power.

Chapter 7

IR-Drop Aware Place and Route

7.1 Introduction

The state of the art CAD techniques for FPGAs do not manage IR drops or voltage profile, and no such published work is known, apart from the work proposed in this thesis. Improving the minimum V_{dd} is important from reliability perspective and power grid design typically involves a constraint to be met for minimum V_{dd} [41, 36]. It is also important to reduce the variation in the V_{dd} across the chip for clock skew management [80, 81, 82]. Further, reducing the V_{dd} variation is also important because coupled with process variations, this can lead failures in implementing the functionality of the chip at desired frequency of operation. Motivated by the above this chapter proposes a novel CAD technique for FPGAs to reduce the IR drop, such that the minimum supply voltage V_{dd} improves and the spatial variation of V_{dd} reduces in the FPGA chip, while making the current distribution more uniform. The main contributions of this work are as follows:

1. IR-drop aware placement technique
2. IR-drop aware routing technique
3. Complete CAD framework for the analysis

The proposed IR-drop aware placement and routing techniques do not require solving the power grid network at every iteration and is therefore an efficient methodology. To the best of our knowledge, this is the first work which propose place and route techniques for improving the voltage distribution profile in the power grid of FPGAs [83, 84].

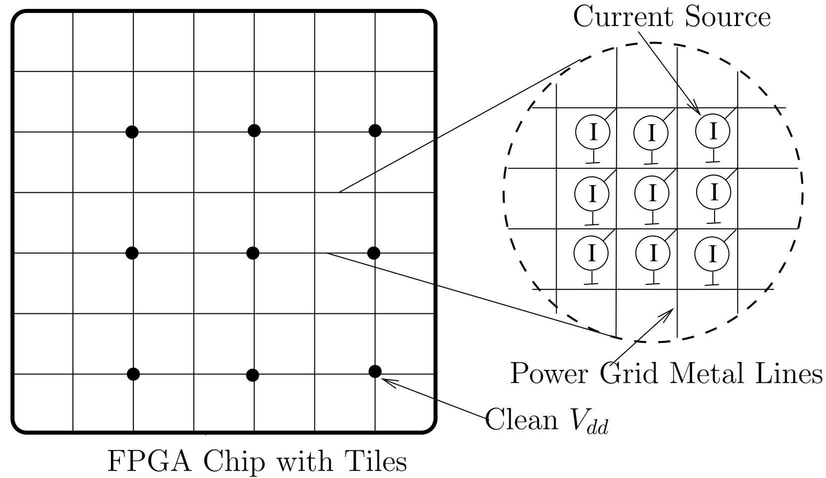


Figure 7.1: Mesh style power grid model

7.2 Power Grid Model

The power grid model selected in this work is a mesh power grid with the horizontal and vertical metal lines forming a mesh structure [41]. The structure is shown in Fig. 7.1. The currents drawn by the devices are modeled by independent current sources in the power grid. Such a power grid model has been widely used [85, 41, 86]. The current sources at each node represent the average current drawn from the node. Some of the nodes, at a regular spacing, are connected to clean V_{dd} supply as shown in Fig. 7.1. The power grid can be modeled as a network of resistive elements with current and voltage sources. The ground network has a similar mesh style grid.

The current sources are modeled by computing the current drawn by each tile and then distributing the total current equally over the mesh nodes supplying current to the elements in the tile. Thus, for a single FPGA tile, several current sources are modeled such that all the current sources in the tile will have the same value, however, the value of the current drawn by these current sources will be different across different tiles. Although this is an approximation, but the model is good for this work because the clustering optimization is carried out at the granularity of a logic cluster. Further, the area of a FPGA tile is small compared to the FPGA, hence representing it as composed of several equal valued current sources distributed uniformly over its area do not introduce inaccuracies in the model. For computing the current in the independent current sources, the total power consumed by the elements in the tile, i.e., logic and routing resources, is calculated. Then the steady state current for the current sources of a tile is calculated as $I_{steady_state} = \frac{P_{total}}{n.V_{dd}}$, where n is the number of current sources modeled in a tile, P_{total} is the total power consumed by the tile. The total power is composed of two parts, the dynamic power and the leakage power

as in (7.1). The dynamic power model proposed in [68], and the leakage power model proposed in [70], have been adopted in this work. The detailed expressions for leakage power are discussed in [70]. The total power is given by:

$$P_{total} = P_{dynamic} + P_{leakage}. \tag{7.1}$$

The dynamic power at a node is given by

$$P_{dynamic} = \sum_{all\ nodes} 0.5 C_i V_{dd}^2 D(i) f_{clk}, \tag{7.2}$$

where C_i is the node capacitance, V_{dd} is the supply voltage, $D(i)$ is the transition density at node i, and f_{clk} is the clock frequency [68]. The transition density at a node gives the expected number of toggles per clock cycle at the node. The dynamic power computation model adopted in [68], and the software tool for FPGA power computation developed by the authors of [68] has been selected for this work, which is based on transition density model [87, 88]. This model takes signal probabilities and transition densities at its inputs and propagates these values to the internal nodes in the circuit. Given, an output y, the inputs x_i, corresponding signal probabilities as $P(x_i)$, the transition density at the output node y is calculated as:

$$\frac{\partial y}{\partial x_i} \triangleq y|_{x_i=1} \oplus y|_{x_i=0} \tag{7.3}$$

$$D(y) = \sum_{i=1}^{i=n} P\big(\frac{\partial y}{\partial x_i}\big) D(x_i) \tag{7.4}$$

The term $\frac{\partial y}{\partial x_i}$ is called Boolean Difference and is used to compute the transition density. The signal probabilities need to be propagated through the boolean network to compute the transition densities at the nodes. However, this model assumes spatial independence while propagating the signal probabilities, but takes temporal correlations into account. The assumption of spatial independence makes the algorithm fast. The power model adopted in this work assumes the signal probabilities for the inputs as 0.5 and transition densities as 0.5 [68], which are propagated in the boolean network to obtain the transition densities at all the internal nodes. The authors of the paper [68] verify the accuracy of the transition density power model through HSPICE simulations by simulating different sizes of LUTs and multiplexers with the inputs delayed by varying values.

However, it should be noted that the techniques proposed in this chapter for IR-drop reduction does not rely on the power model. Rather, it relies on an indirect parameter which can predict power consumption in different parts of the chip. Furthermore, the proposed IR-drop aware clustering technique does not need an accurate model for power, rather it requires some estimates which can provide the IR-drop aware clustering technique with a relative measure of power in different parts of the chip. Hence, any other measure for power would be equally applicable, and would not change the adopted methodology.

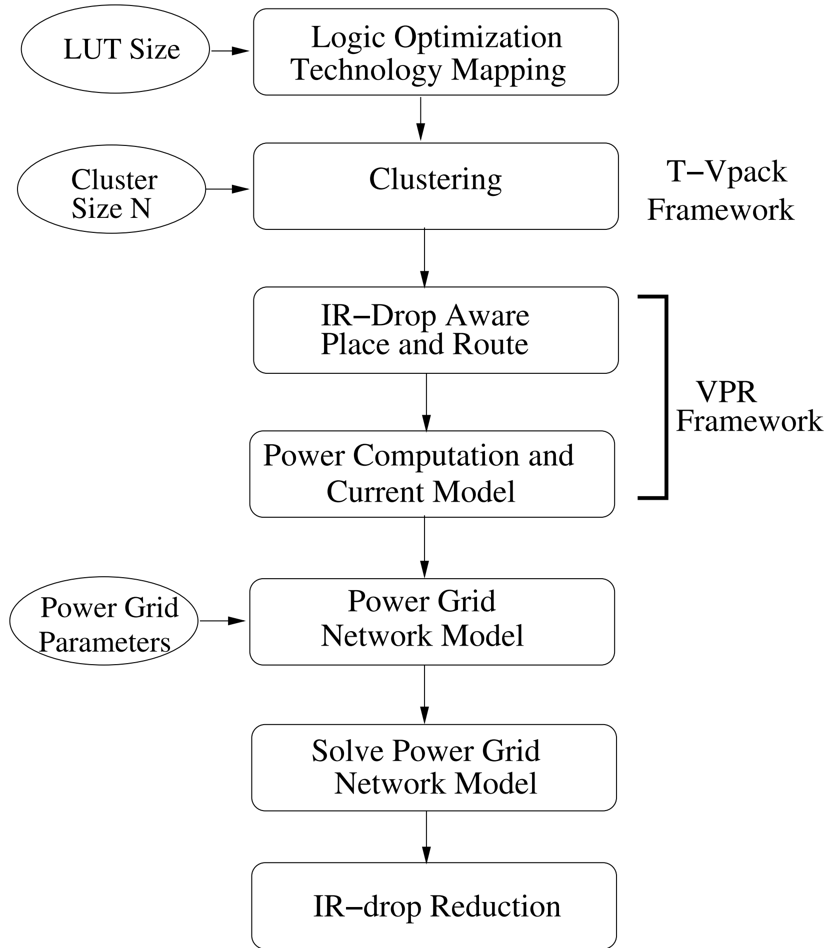


Figure 7.2: Proposed IR-drop aware Place and Route CAD flow

7.3 Proposed Methodology

Any region in the chip which has a high local transition density will experience higher IR-drops. Therefore, the techniques proposed in this work reduce the transition densities in a region, which has high transition densities, to achieve a lower IR drop and also a lower variation in the supply voltage across the chip. This work implements the proposed methodologies in the framework of the academic tool called Versatile Place and Route (VPR) [1].

The complete CAD flow for computing the minimum node voltage and the standard deviation of the node voltages for both the baseline (using classical VPR algorithms) and the proposed IR-drop aware design methodology is as follows.

1. **Place and Route:** The first step is to place and route the design. For the baseline implementation the classical place and route of VPR is employed. For the IR-drop aware implementation, the enhanced place and route, as proposed in this work are used.

2. **Power Computation:** The second step is to compute the total power consumed by each of the FPGA tiles. The computation of power, for a benchmark, is done at a fixed clock frequency, same for both the baseline and the IR-drop aware implementations for comparison purposes.

3. **Model Current Sources:** The third step is to compute the current sources from the power for each of the tiles.

4. **Build Power Grid Network Model:** The fourth step is to build the circuit model for the power grid network with the current sources. The power grid model includes both the supply voltage network and the ground network.

5. **Solve the Circuit Model:** The fifth step is to solve the circuit model to determine the node voltages in the power grid. The voltage value of concern in this work is the voltage difference across the current sources, i.e., the voltage across the points of a current source connected to V_{dd} and V_{ss}, $(V_{dd} - V_{ss})$, because the V_{ss} is not at perfect zero voltage.

6. **Minimum and Std. Dev. of V_{dd}:** The final step is to find the minimum V_{dd} and compute the standard deviation of V_{dd} across all the nodes in the power grid. The minimum V_{dd} reported in this work is the minimum of the voltages across the current sources in the power network model, i.e., it is minimum of $V_{dd} - V_{ss}$ across all the current sources. For the sake of brevity the remainder of the chapter will refer to this value as minimum V_{dd} denoting the voltage at which the devices connected to the node are operating. The same notation holds for standard deviation of $V_{dd} - V_{ss}$.

7.3.1 IR-Drop Aware Placement

The classical placement routine implemented in VPR is based on a simulated annealing algorithm [16]. The authors in [16] develop an *auto-normalizing* cost function for the placement consisting of timing cost, for optimizing circuit delay and wiring cost, for reducing wire length. The cost function, $\triangle C$, used for the placement routine, is defined as

$$\begin{aligned} \triangle C &= \lambda . \frac{\triangle Timing_Cost}{Previous_Timing_Cost} \\ &+ (1-\lambda) . \frac{\triangle Wiring_Cost}{Previous_Wiring_Cost}, \end{aligned} \tag{7.5}$$

where λ is a factor for giving different weights to the timing cost and the wiring cost, $\triangle Timing_Cost$ is the change in the $Timing_Cost$ because of the current move, and $\triangle Wiring_Cost$ is the change in the $Wiring_Cost$ because of the current move.

There are four main steps involved in the IR-drop aware placement proposed in this work. 1) Divide the chip using *grid based model*, 2) compute the transition density cost for each grid, 3) formulate the placement cost function and, 4) perform the placement with the augmented cost function. The flow of the proposed IR-drop aware placement routine is shown in Algorithm 3. The grid model for the FPGA chip is developed by

Algorithm 3: IR-Drop Aware Placement Algorithm

```
Partition FPGA into small regions ;
Begin Placement;
repeat
    for each CLB do
        Determine grid location (j,k);
        Compute transition density cost (eqn (7.7), (7.7));
        Update the total transition density cost (eqn (7.8)-(7.10));
    end
    Compute Normalized Placement Cost;
until Placement Exit Criterion ;
```

dividing the chip into several regions of square grids. Each of the square grids represents the corresponding area on the chip, with a coordinate (j, k) associated with the grid. The transition density cost for a grid at location (j, k) is given by,

$$D_{(j,k)} = \sum_{n,i} d_{n,i}, n \in S'_{Vdd} \tag{7.6}$$

$$d_{n,i} = 0, n \in S_{Vdd} \tag{7.7}$$

where $D_{(j,k)}$ represents the total transition density for the grid, $d_{n,i}$ is the transition density of the ith net connected to n^{th} logic element located in the grid (j, k). S_{Vdd} represents the set of tiles which have a clean V_{dd} supply node, whereas S'_{Vdd} represents the set of remaining tiles, i.e., those tiles which do not have a clean V_{dd} supply. This is because those tiles which have a clean V_{dd} supply node will have a much less IR drop due to proximity to the clean V_{dd} node. Therefore such tiles need not contribute to the transition density cost. This helps the placement tool to better optimize the critical delay of the circuit. The reduction of total transition density in a region is performed by moving some of the blocks with high transition density nets from the region of high total transition density to a region where the total transition density is low. This is achieved by augmenting the placement cost function in (7.5) with the transition density cost, D_Cost in (7.10), which is computed as

follows,

$$
\begin{aligned}
Avg_D &= \frac{Total_D}{Num_Grids}, && (7.8) \\
D_Cost_{j,k} &= |(D_{(j,k)} - Avg_D)|^{\beta}, && (7.9) \\
D_Cost &= \sum_{j,k} D_Cost_{j,k},
\end{aligned}
$$

where Avg_D is the average transition density cost for each grid, obtained by computing the total transition density for the chip and dividing it by the total number of grids in the model. $D_Cost_{(j,k)}$ is the transition density cost at the grid location (j,k), and D_Cost is the total cost obtained by adding all the individual transition density costs for each grid. The cost function formulated in (7.9) penalizes those grids which have a transition density deviating from the Avg_D value. A strategy is adopted to give a higher value to the cost for a higher deviation from the average value, i.e., instead of adding a linearly increasing penalty, an exponentially increasing penalty is employed for deviations from the average transition density per grid. The factor β in (7.9) is employed for this purpose and its value is experimentally chosen as 1.3. Another technique adopted in this work to reduce the IR drops is a strategy to reduce the wire length of the nets with high transition densities. The $Wiring_Cost$ in (7.5) is composed of the net_cost for each net as explained in [16]. The net cost for each net is modified to incorporate the transition density factor as follows, $new_net_cost(i) = net_cost(i) * transition_density(i)$, where $new_net_cost(i)$ is the modified net cost for net i, $net_cost(i)$ is the classical net cost as proposed in [16] for net i, and $transition_density(i)$ is the transition density of net i. Therefore, the final IR-drop aware placement cost function proposed and implemented in this work is given by

$$
\begin{aligned}
\triangle C &= \lambda.\frac{\triangle Timing_Cost}{Previous_Timing_Cost} && (7.10) \\
&+ (1-\lambda-\gamma).\frac{\triangle Wiring_Cost_{tran_density}}{Previous_Wiring_Cost_{tran_density}} \\
&+ (\gamma).\frac{\triangle D_Cost}{Previous_D_Cost},
\end{aligned}
$$

where $Wiring_Cost_{tran_density}$ is computed using the new net cost, $\triangle D_Cost$ is the change in transition density cost due to current move, and $Previous_D_Cost$ is the previous transition density cost. The factor, γ, is the IR-drop trade-off for the cost function and the experimentally obtained value of the trade-off factor in this work is 0.2.

It can be seen from the placement algorithm described above that it relies on redistributing the placement of the high transition density CLBs such that the local transition density of a region reduces. Similar techniques have been employed in thermal-aware placement to reduce the hot spots in the circuit, though the formulations and placement techniques have been different [89, 90].

7.3.2 IR-Drop Aware Routing

The routing in the FPGA is based upon the Pathfinder algorithm [1]. The cost for using node n during routing expansion for connecting the sink j of the net i is expressed as

$$\begin{aligned} cost(n) &= crit(i,j).delay(n, topology) \\ &+ [1 - crit(i,j)].b(n).h(n).p(n), \end{aligned} \tag{7.11}$$

where $crit(i,j)$ is the criticality of the connection, $delay(n, topology)$ is the delay of the connection after including node n in the path, and $b(n), h(n), p(n)$ are the base cost, the historical congestion, and the present congestion [1]. The IR-drop aware routing methodology proposed in this work relies on avoiding routing high transition density nets through the same region or spatially close regions. This is because the routing buffers draw current from the power supply grid and if there are many nets with high transition densities in close proximity, then that part of the chip will tend to draw more current leading to larger IR drop in the region. The main steps involved in IR-drop aware routing flow are: 1)Build the *grid based model* and start route of a net, 2)determine the location of the current node, during wave expansion, in the grids, 3)compute the cost due to transition density of this node and, 4)augment the cost function with the transition density cost function and repeat until the net is routed. The algorithm flow for the proposed IR-drop aware routing is shown in Algorithm 4. For developing the new cost function, at each wave expansion in the routing algorithm, in which a node is given a cost, the location of the routing switches to be used is determined, and thus the corresponding grid in the grid based model is located. Each grid in the IR-drop aware routing keeps a historical record of the total transition density, for the previously routed nets as well as the current net. In the wave expansion during the routing this historical record is augmented with the transition density cost of the current net to determine the cost of using this node in the routing of the net.

The switch boxes are distributed evenly throughout the routing fabric and the number of switch boxes used is directly dependent on the length of the routed net. The proposed IR-drop aware routing algorithm takes into account the transition densities of the nets. The transition densities at the switch boxes are the same as the transition densities of the nets that are being routed through them. Therefore the wire length cost and the transition density cost of the nets, together account for the power consumption in the switch boxes. Also, while calculating power, all the nodes are accounted for, including that of the switch boxes and hence power calculation and consequently IR-drop calculations are realistic.

The cost function of a node n, for the IR-drop aware routing is computed as follows,

$$D_{(j,k)} = D_{(j,k)}(prev) + D_{net} \tag{7.12}$$

$$\begin{aligned} cost_{IR-drop}(n) &= crit(i,j).delay(n, topology) \\ &+[1- crit(i,j)].\left(b(n).h(n).p(n) + b(n).D_{(j,k)}\right) \end{aligned} \tag{7.13}$$

Algorithm 4: IR-Drop Aware Routing Algorithm

```
Partition FPGA into small regions ;
Begin Routing Nets;
foreach net i do
    while net i not routed do
        Expand to node n ;
        Find grid location (j,k) for node n;
        Determine IR-Drop Cost (eqn (7.13));
        Update the historical IR-Drop Aware Cost;
        Augment classical VPR route cost;
    end
end
```

where $D_{(j,k)}$ is the total transition density for the grid (j, k), D_{net} is the transition density for the current net being routed.

The cost in (7.12) is incorporated in that part of the routing cost function which governs the congestion cost for the routing of a net. The congestion cost for a net signifies the congestion in the routing channels due to many nets being routed through the same region. Now, the transition density cost similarly represents a form of *congestion* which can be termed as the *transition density congestion*, which occurs due to many high transition density nets being routed through the same region. Therefore, the form of the cost function in (7.13) not only reduces the physical congestion, but also the *congestion* due to high transition density nets being routed through a local region.

7.4 Experimental Details, Results and Discussions

7.4.1 Experimental Details

The 45nm Predictive Model is chosen as the technology node for simulations [69, 73]. The intermediate metal layers are used for the power grid mesh. The pitch of the power grid network is taken as $20\mu m$ [91, 92]. The clean Vdd nodes are available at a pitch of $300\mu m$. The length of the FPGA tile is assumed to be $100\mu m$ which was obtained by the scaling of an FPGA tile as proposed in [1]. The V_{dd} for this technology node is 1V. The software package used to compute the voltages at the circuit nodes is GNU Circuit Analysis Package [93]. The power for the baseline and the IR-drop aware implementations for a benchmark are computed at the same clock frequency which is chosen such that the critical circuit delays of both the implementations can meet the clock frequency. The routing has been done with the same channel widths for both the implementations. The

Table 7.1: Results of IR-Drop Aware Design

Benchmarks	Baseline		IR-Drop Aware		Improvement		Maximum Current Reduction
	Min V_{dd}	Std. Dev. V_{dd}	Min V_{dd}	Std. Dev. V_{dd}	IR-Drop Reduction	Std. Dev. V_{dd}	
alu4	0.86412	0.031810	0.93668	0.014052	53.40%	55.83%	43.90%
apex2	0.88656	0.028923	0.92431	0.009666	33.27%	66.58%	27.04%
apex4	0.92163	0.020254	0.94525	0.011911	30.14%	41.19%	25.40%
bigkey	0.88334	0.028303	0.91958	0.020652	31.07%	27.03%	23.95%
des	0.88610	0.021701	0.93377	0.009311	41.85%	57.09%	16.82%
diffeq	0.93277	0.018474	0.95927	0.006765	39.42%	63.38%	18.64%
dsip	0.89061	0.026784	0.93130	0.017201	37.20%	35.78%	22.39%
elliptic	0.90965	0.023156	0.92859	0.011971	20.96%	48.30%	-4.21%
ex1010	0.93001	0.015642	0.95133	0.006571	30.46%	57.99%	32.31%
ex5p	0.90682	0.024753	0.92918	0.019379	24%	21.71%	31.35%
frisc	0.94018	0.016708	0.95626	0.007870	26.88%	52.90%	3.597%
misex3	0.89436	0.028901	0.89770	0.017819	3.16%	38.35%	7.951%
pdc	0.91467	0.021078	0.93726	0.009569	26.47%	54.60%	37.11%
s298	0.92973	0.020038	0.94601	0.009006	23.17%	55.06%	27.02%
s38417	0.89757	0.018936	0.90697	0.009888	9.18%	47.78%	-25.7%
seq	0.86230	0.034070	0.90686	0.014250	32.36%	58.17%	23.18%
spla	0.91176	0.020004	0.94308	0.009303	35.49%	53.49%	16.48%
tseng	0.91943	0.022891	0.93667	0.015020	21.40%	34.39%	30.93%

grid based model has each grid of size 2x2 tiles which is selected experimentally. Changing the size may affect the result and speed of the algorithm. A very large grid size may not lead to any improvement in the IR-drops whereas too small a size would also not lead to any improvement as the placement granularity is an FPGA tile. However, an FPGA designer is free to experimentally select a grid size which would give best results for the specific FPGA architecture.

7.4.2 Results and Discussions

The results for the baseline and the IR-drop aware placement and routing for the minimum V_{dd} and standard deviation of V_{dd} are shown in Table 7.1. Columns 2 and 3 list the minimum

V_{dd} and the standard deviation of V_{dd} for all the nodes in the power grid for the baseline implementation. Columns 4 and 5 show the minimum V_{dd} and standard deviations of V_{dd}, respectively, for different benchmarks with IR-drop aware implementation. Columns 6 and 7 show the reduction in maximum IR-drop and standard deviation of V_{dd}. It can be seen from the table that a reduction of up to 53.4% in the IR-drop can be obtained. The standard deviation of V_{dd} reduces by up to 66.85%. The last column shows the reduction in maximum average current at any FPGA tile. It can be seen that up to 43.9% reduction in maximum average current can be obtained from the proposed technique. Reducing maximum average current is important from the perspective of electromigration effects. Although, this work does not attempt to quantify the electromigration effects, the reduction in maximum current reduces the impact of electromigration.

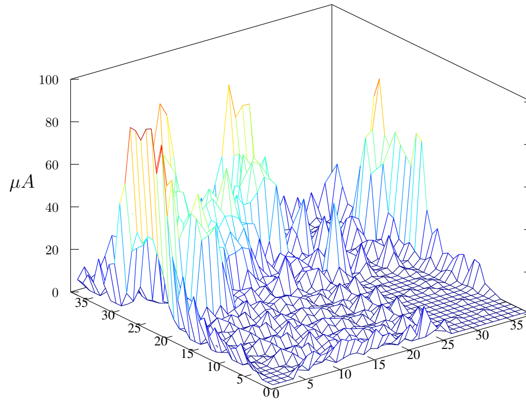


Figure 7.3: Current distribution for baseline implementation: *des*

Figures 7.3 and 7.4 show the current and voltage distributions for the benchmark *des*, which has been placed and routed using classical VPR placement and routing (baseline

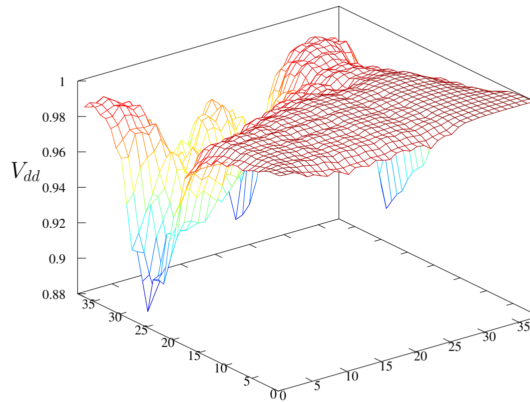


Figure 7.4: Voltage distribution for baseline implementation: *des*

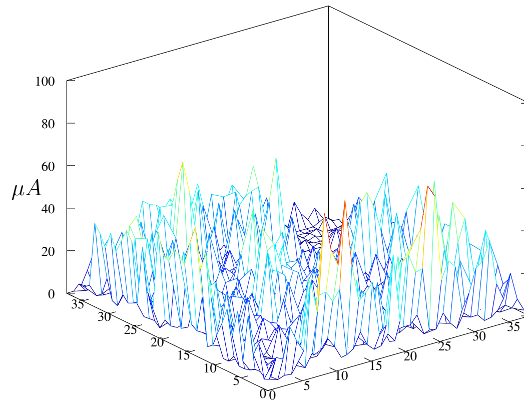


Figure 7.5: Current distribution for IR-drop aware implementation: *des*

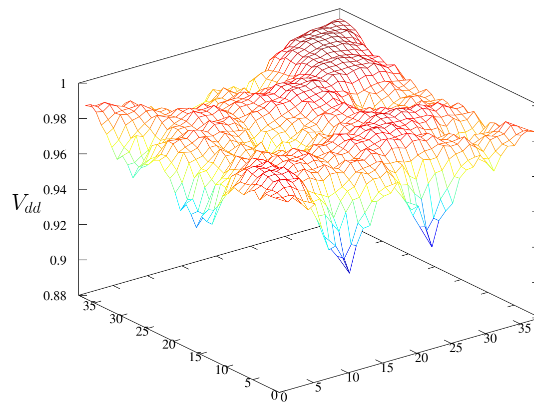


Figure 7.6: Voltage distribution for IR-drop aware implementation: *des*

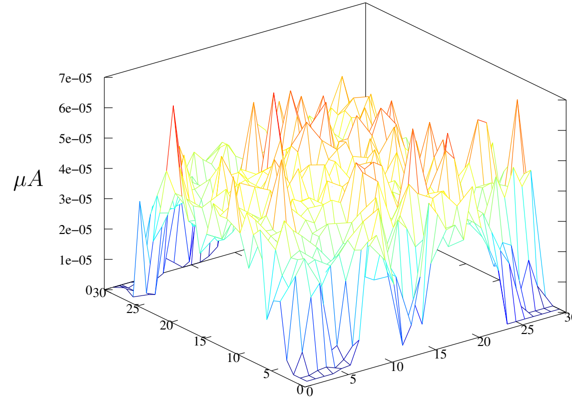


Figure 7.7: Current distribution for baseline implementation: *s38417*

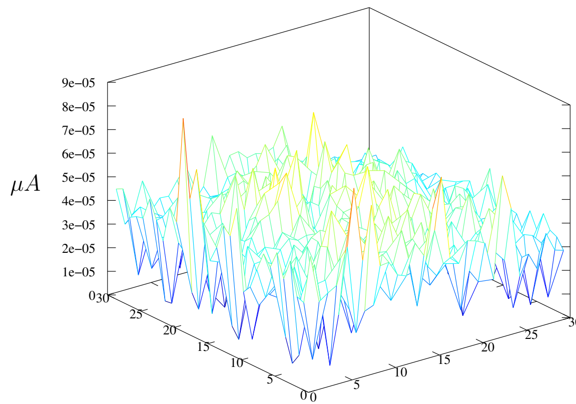


Figure 7.8: Current distribution for IR-drop aware implementation: *s38417*

implementation). It can be seen from Fig. 7.3, that there are regions in the FPGA chip which draw larger current than other parts, resulting in large current peaks in parts of the FPGA chip. Consequently, those parts have a larger IR drop leading to lower V_{dd} values in the regions as shown in Fig. 7.4. However, there are some regions in the FPGA chip which draw small currents and therefore those parts show small IR-drops. Further, the current distribution shows a large variation across the chip. Figures 7.5 and 7.6 show the current and voltage distributions for the benchmark *des*, with the IR-drop aware design implementation as proposed in this chapter. It can be seen from Fig. 7.5 that the current is now more uniformly distributed across the chip compared to that in Fig. 7.3. Further, there are no large current peaks in the IR-drop aware implementation as there are in the baseline implementation. Consequently, the V_{dd} distribution profile is smoother in this case, as shown in Fig. 7.6, compared to the baseline implementation. This results in improved minimum V_{dd} by 5.4%, and a consequent reduction in IR-drop by 41.85% along with improved standard deviation of the V_{dd} distribution by 57.09%.

In some cases, such as *s38417* and *misex3*, it can be seen that the improvement in the minimum V_{dd} is smaller compared to other benchmarks, or there is no improvement. This is attributed to the fact that the baseline implementation already has a uniform current distribution throughout the FPGA. This can be seen in figures 7.7 and 7.8 which depict the current distributions for the baseline and IR-drop aware implementations for the benchmark *s38417*. It can be seen that the baseline implementation in Fig. 7.7 has a good uniform current distribution throughout the chip. Also, the difference between the current peaks across the chip is small which implies that there is a small scope of improving the minimum V_{dd} in the FPGA chip by reducing the current peaks. This is because the IR-drop aware design re-distributes the current profile in such a way that, while the larger current peaks are reduced, the smaller current peaks provide the *space* for accommodating the current profile re-distribution. The IR-drop aware implementation smooths the current distribution profile, shown in Fig. 7.8 but has similar values of current for the different parts of the FPGA compared to the baseline implementation in Fig. 7.7. This results in small reduction in IR-drop for the benchmark *s38417* by 9.18%. The trade-off of the IR-drop aware design methodology, proposed in this work, with the circuit delay is shown in Fig. 7.9. The figure shows the ratio of the circuit delay between the proposed IR-drop aware implementation and the baseline implementation. It can be seen from the figure that IR-drop aware implementations are slower compared to baseline implementations, on an average by about 3%.

It can be seen from the results table that the maximum average current for the benchmarks *elliptic* and *s38417* increase for the IR-drop aware implementation. This is because the proposed technique does not directly attempt to reduce the peak current at a node, but rather *redistribute* the current profile to make it more uniform across the chip for reducing the voltage variations. Therefore, it is possible, that this redistribution would lead to a scenario, in which although the overall current profile for the chip becomes more uniform, the peak current might show an increase. Further, the V_{dd} at a node does not depend only on the current at a node but also on currents being drawn by the surrounding nodes. Hence, in this case it is observed that although the peak current increases, still the IR-drop reduces.

The reduction in maximum IR-drop can directly translate into metal area savings. So, for instance, in the case of *frisc* in which the minimum V_{dd} is improved by 1.7% using the proposed CAD techniques, if instead only power grid metal widening was employed to improve the minimum V_{dd}, then the power grid metal line width needs to be increased by 26.88% to achieve 1.7% improvement in minimum V_{dd}, i.e., to achieve a 26.88% reduction in maximum IR-drop.

The proposed IR-drop aware place and route generates a new placement and routing solution in which the net lengths might increase, leading to increased number of switching routing resources. This results in some increase in total FPGA power because of increase

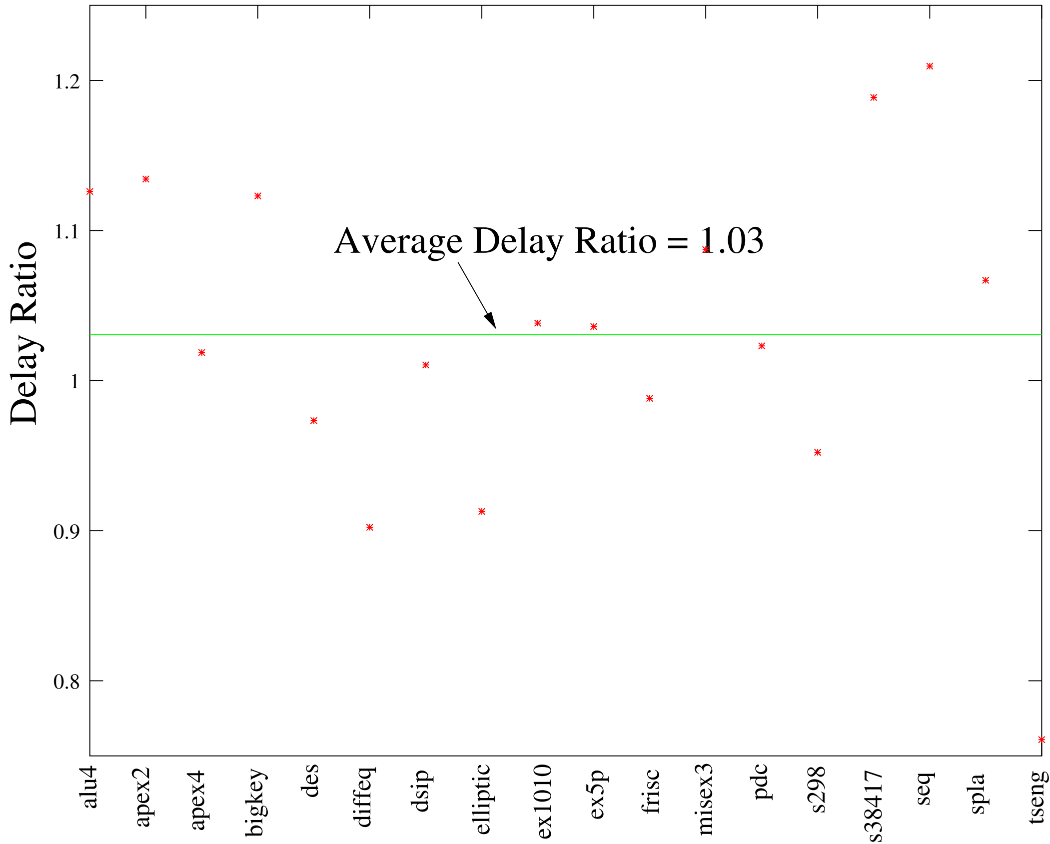


Figure 7.9: Ratio of the circuit delay for the IR-drop aware and baseline implementation

in the dynamic power. On an average there is a 6.7% increase in the total FPGA power. The IR-drop in the power grid computed for the IR-drop aware implementation takes this increase in power into account.

The IR-drop aware place and route requires 3.95X runtime, on an average, as compared to the VPR place and route. This is justifiable because power grid optimization techniques which generally employ iterative techniques, need to solve the power grid at each iteration which would be computationally much more expensive. The proposed CAD techniques can be used in conjunction with other power grid design techniques, such as wire sizing, in case the CAD techniques do not alone suffice to improve the reliability of the power grid. In such a case, the proposed technique can provide savings in the metal area for the power grid by reducing the amount by which the widths of the power metal lines need to be increased.

7.5 Conclusions

The chapter proposes IR-drop aware placement and routing techniques which re-distributes the current drawn by different parts of the FPGA chip such that there is a more uniform current distribution for the entire chip. The IR-drop aware placement technique redistributes the placement of the logic blocks in such a way so that the blocks having high switching activities are not placed close together. The IR-drop aware routing technique avoids routing high switching activity nets close to each other. The results from the benchmarks indicate up to 53% reduction in maximum IR-drop and up to 66% reduction in standard deviation of the V_{dd} distribution. As an additional remark, it should be noted that all the work in this chapter and previous chapters, except Chapter 6, used VPR as a place and route tool and modified the placement and routing in VPR to achieve the desired results. However, any other place and route tool can be used and modifications can be made in the tool to achieve the desired results. The main ideas would remain the same, however, the implementation would change. For example, VPR uses simulated annealing with a global cost function to optimize the placement. In this work this global cost function was augmented to optimize the performance. If a force-directed placement tool is employed the appropriate cost functions would need to be enhanced in that case.

Chapter 8

IR-Drop Aware Clustering

8.1 Introduction

Chapter 7 investigated and proposed novel place and route techniques to mitigate IR-drops in the power grid of FPGAs. It explained the proposed approach to show how the supply voltage profile can be improved by reducing the maximum IR-drops and the spatial variation of supply voltage in the power grid. This chapter proposes techniques at the clustering stage of the CAD flow to reduce IR-drops in the power grid network. The proposed IR-drop aware clustering technique is a faster technique compared to place and route, however, the IR-drop aware place and route produces better results. To the best of our knowledge, this is the first work which proposes a clustering technique for improving the voltage profile in the power grid of an FPGA [94].

8.2 Proposed CAD Flow

The IR-drops in the power grid network stems from the current drawn by the transistors consuming dynamic and leakage power and the resistance of the metal lines of the power grid. Therefore, that part of the chip which draws larger currents will experience larger IR drops. It can be seen from (7.2) that the dynamic power consumed by a chip is directly proportional to the transition densities, $D(i)$, of nodes in the circuit. Therefore, the part of the chip which has nodes with higher transition densities experiences larger IR-drops. The CAD flow proposed in this work, shown in Fig. 8.1, and outlined below, redistributes these nodes in such a way that the local transition density of a region of the chip reduces resulting in improved voltage profile.

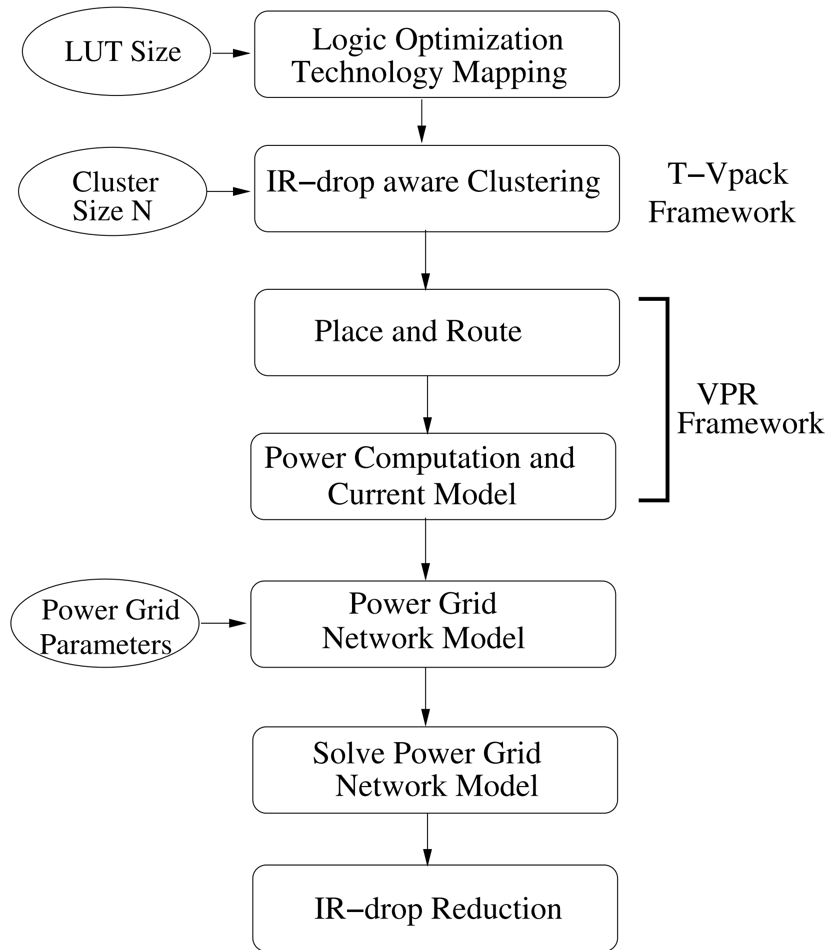


Figure 8.1: Proposed IR-drop aware CAD flow

1. **Clustering:** The second step, which is after technology mapping, is to cluster the Look-up Tables (LUTs) and Flip Flops (FFs) into logic clusters, where a logic cluster consists of N LUTs and FFs. Here, for baseline implementation the classical T-VPack is employed, whereas for the IR-drop aware implementation the clustering technique as proposed in this work is used.

2. **Place and Route:** The third step is to place and route the netlist consisting of logic clusters which is the output of the previous step. Once the netlist is placed and routed the critical delay of the circuit is computed.

3. **Power Computation:** The fourth step is to compute the total power consumed by each of the FPGA tiles. The computation of power, for a benchmark, is done at a fixed clock frequency, same for both the baseline and the IR-drop aware implementations for comparison purposes.

4. **Determine Current Sources:** The fifth step is to compute the current sources from the power for each of the tiles.

5. **Build Network Model:**The sixth step is to build the circuit model for the power grid network with the current sources. The power grid model includes both the supply voltage network and the ground network.

6. **Solve the Circuit Model:** The seventh step is to solve the circuit model to determine the node voltages in the power grid. The voltage value of concern in this work is the voltage difference across the current sources, i.e., the voltage across the points of a current source connected to V_{dd} and V_{ss}, $(V_{dd} - V_{ss})$, because the V_{ss} is not at perfect zero voltage.

7. **Minimum and Std. Dev. of V_{dd}:** The final step is to find the minimum V_{dd} and compute the standard deviation of V_{dd} across all the nodes in the power grid. The minimum V_{dd} reported in this chapter is the minimum of the voltages across the current sources in the power network model, i.e., it is minimum of $V_{dd} - V_{ss}$ across all the current sources. For the sake of brevity the remainder of the chapter will refer to this value as minimum V_{dd} denoting the voltage at which the devices connected to the node are operating. The same notation holds for standard deviation of $V_{dd} - V_{ss}$.

8.3 Proposed Clustering Technique

Fig. 8.2 shows a logic cluster with input and output nets and few routing switches for the nets depicting an example of the connectivity of a netlist. The logic cluster has some input nets and some output nets. It can be seen that a net can fan out in such a way that it can

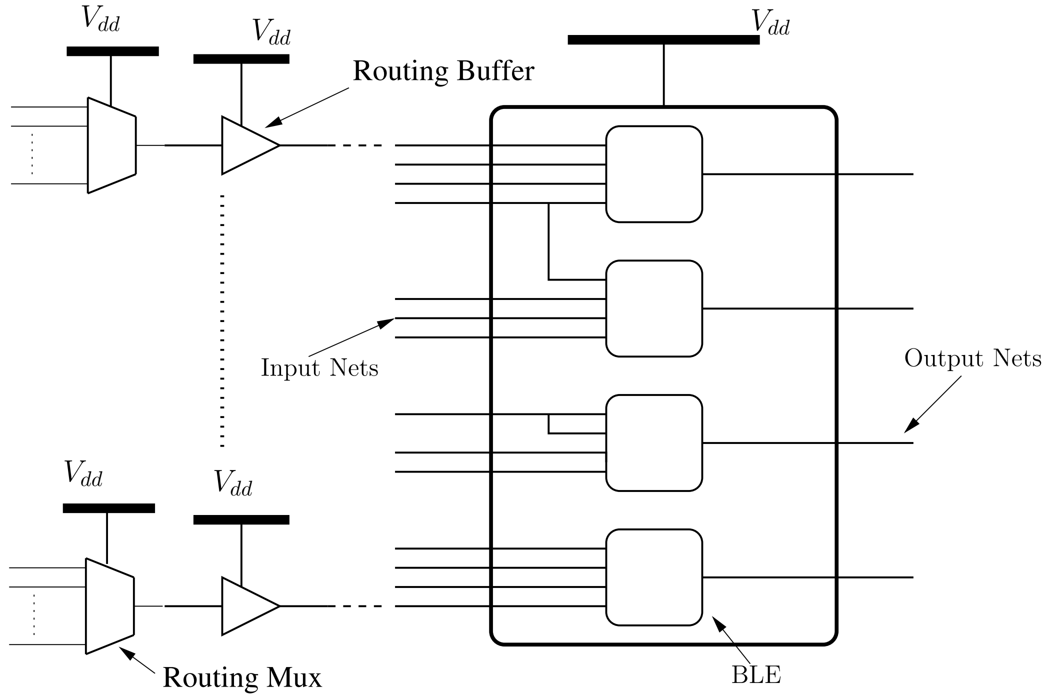


Figure 8.2: Logic cluster with input and output nets.

drive multiple BLE inputs. Now, if many of the nets connected to this cluster have high transition densities, then the current drawn by the cluster and the neighboring switches, which route the nets to the logic cluster, will be high, resulting in a large IR-drop in the local region. This means that if crowding of high transition density nets are avoided in a cluster then the IR-drops can be reduced.

The framework used in this work for clustering of BLEs is based on the academic tool T-VPack [3]. Cluster size of an FPGA refers to the maximum number of BLEs that can be accommodated in a cluster. This is determined by the physical structure of the CLBs in the FPGA. The packing algorithm outputs a netlist of clusters in which each cluster can contain up to a maximum of N BLEs, where N is the cluster size. Some of the clusters might have less than N BLEs, which is also a feasible clustering. The T-Vpack algorithm attempts to create clusters of BLEs based on the constraints of keeping the number of BLEs less than or equal to N, the cluster size, and also keeping the total number of unique external inputs to the cluster less than or equal to I, which is the maximum number of inputs allowed due to the physical structure of the FPGA as shown in Fig. 2.5. The T-Vpack algorithm tries to pack a cluster to its maximum capacity possible under the above constraints. The T-Vpack is a timing driven clustering algorithm such that it attempts to minimize the critical path delay of the netlist. The inter-cluster delay, which is the delay

of a net connecting two different clusters is larger than the intra-cluster delay, which is the delay within a cluster. The T-VPack algorithm essentially tries to reduce the number of inter-cluster connections on the critical path of the netlist. This is because the inter-cluster connections have larger delay than the intra-cluster connections. The T-VPack algorithm proceeds by computing the connection criticalities of the input pins of the BLEs and selecting the most critical BLE as the seed cluster. The connection criticality of the connection driving input i of a BLE is computed as follows,

$$Connection_Criticality(i) = 1 - \frac{slack(i)}{MaxSlack}, \tag{8.1}$$

where $MaxSlack$ is the maximum slack available, $slack(i)$ is the slack of the ith connection, and the delays are computed by assuming the inter cluster delay of 1 unit and intra cluster delay of 0.1 units. This is because the actual delays are not available until the circuit has been placed and routed. Also, the inter cluster delays are significantly larger than the intra cluster delays, therefore such values of these delays are selected. The seed BLE, which is the first BLE for a cluster is the one which has most critical connection among the unclustered BLEs. Consider a cluster which has some BLEs and to which more BLEs are being added until it reaches its capacity or it becomes infeasible to add any more BLE to the cluster. The candidate BLEs for the current cluster are those BLEs which share a connection with the BLEs in the current cluster. The $Base_BLE_Criticality(B)$, of a candidate BLE B, is defined as the maximum of the $Connection_Criticality$ values of all connections joining B to BLEs in the current cluster [3]. However, there are chances that some candidate BLEs might have same $Base_BLE_Criticality$ value, so a tie-breaker mechanism is adopted. The tie-breaker mechanism essentially computes the number of paths that are affected if a BLE is selected for addition to the current cluster [3]. Consider the example shown in Fig. 8.3, which shows nets and BLEs on critical paths in bold such that the $Base_BLE_Criticality$ of BLEs (A, ... I) are the same, and they are candidate BLEs for the current cluster. It can be seen that if BLEs G or H or I is added to the current cluster then it reduces the delays of more critical paths than if either of (A,B, C, D, E or F) is added to the current cluster. Therefore to handle this situation two parameters are defined, $input_paths_affected$, and $output_paths_affected$ for each candidate BLE. The parameter $input_paths_affected$ define the number of critical paths between the sources and the current BLE, and $output_paths_affected$ define the number of critical paths between the sinks and the current BLE. The parameter $total_paths_affected$ is then computed as the sum of the parameters $input_paths_affected$ and $output_paths_affected$. So in the example shown in Fig. 8.3, the parameter $total_paths_affected$ for the BLEs, G, H, and I is same and is equal to 3 (2 for $input_paths_affected$ and 1 for $output_paths_affected$), whereas for the other BLEs the parameter $total_paths_affected$ is 2 (1 for $input_paths_affected$

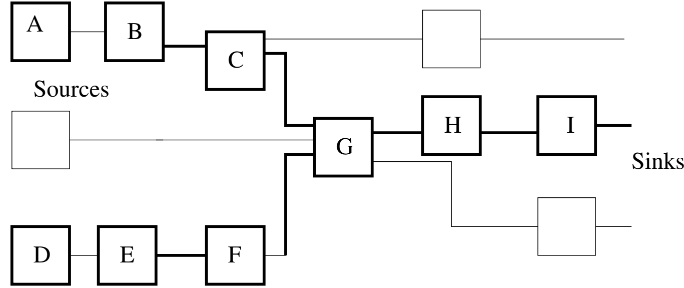


Figure 8.3: Criticality tie breaking technique [3]

and 1 for $output_paths_affected$). The final criticality of a BLE B, is then calculated as:

$$Criticality(B) = Base_BLE_Criticality(B) + \epsilon.total_paths_affected \quad (8.2)$$

The attraction function which finally determines the BLE B to be added to the current cluster C is defined as follows,

$$Attraction(B) = \alpha.Criticality(B) + (1-\alpha).\frac{Nets(B) \cap Nets(C)}{G}, \quad (8.3)$$

where $Nets(B) \cap Nets(C)$ determines the shared nets between BLE B and the cluster C, and G is the maximum number of unique nets to which a BLE can connect, and is given by $G = \#BLE_Inputs + \#BLE_Outputs + \#BLE_Clocks$. The default value of α is 0.75.

This work develops the following clustering technique to account for high transition densities in local regions of FPGA, which lead to larger IR-drops. The methodology efficiently reduces transition densities in local regions which cause the current distribution in the power grid to be more uniform, resulting in improved minimum voltage and reduced spatial variation in V_{dd} in the supply network. Consider the cluster being built up by adding BLEs to the partially full cluster as shown in Fig. 8.4. It can be seen that at the current stage there are two BLEs in the cluster and three candidate BLEs which are under consideration for addition to the cluster. There are a total of nine nets under consideration, $n_1, n2, \ldots, n_9$, with the transition densities of the nets as $td_1, td_2, \ldots, td_9$. The BLEs BLE_1, BLE_2, BLE_5 are the potential candidates for the current cluster, whereas BLE_3, BLE_4 are already present in the cluster. A transition density cost associated with each of the potential BLE candidates for the current cluster is computed. The following procedure is adopted for computing the new attraction function of a candidate BLE to the current cluster.

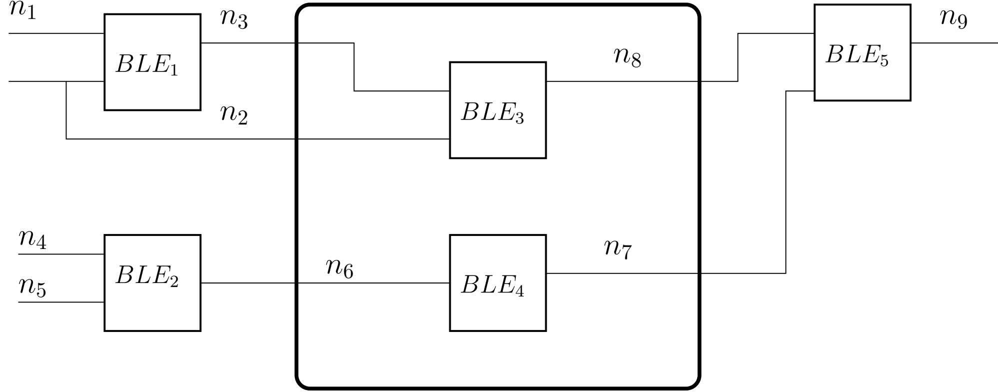


Figure 8.4: Computing the transition density cost during clustering.

1. Identify the set of potential candidate BLEs, SP_{BLE}, for addition to the current cluster, and let S_{BLE} denote the set of BLEs in the current cluster. So, for the case shown in Fig. 8.4, $SP_{BLE} = \{BLE_1, BLE_2, BLE_5\}$, $S_{BLE} = \{BLE_3, BLE_4\}$.

2. Let NC denote the set of unique nets connected to the set of BLEs, S_{BLE}, in the current cluster, and $NBLE_i$ denote the set of unique nets connected to the ith BLE which is a potential candidate for addition to the current cluster. For the case shown in Fig. 8.4, $NC = \{n_2, n_3, n_6, n_7, n_8\}$, and for example, $NBLE_1 = \{n_1, n_2, n_3\}$. Compute the set NTD_i as:

$$NTD_i = NBLE_i - (NBLE_i \cap NC), i \in SP_{BLE} \tag{8.4}$$

So for the case shown in Fig. 8.4, $NTD_1 = \{n_1, n_2, n_3\} - \{n_2, n_3\} = \{n1\}$. Similarly, $NTD_2 = \{n_4, n_5\}$, and $NTD_5 = \{n_9\}$.

3. Define the transition density of the current cluster as

$$TDC = \sum_{net_k \in NC} TD_{net_k}, \tag{8.5}$$

which is the sum of transition densities of all the nets (net_k) in the set NC. The IR-drop aware clustering aims at reducing the local power consumption in the areas which have a high power dissipation. It should be noted that a local region consists of not only one cluster but several other clusters surrounding that cluster. During the clustering phase, there is no information about the buffers and routing switches that will be encountered by the net once the design is placed and routed. After final placement and routing the buffers and the routing switches that lie on a net will be determined by the wire segments that compose the routing of the net. Also, a single

net uses several routing buffers and switches and therefore the current drawn due to switching of the net is distributed throughout the net rather than only the first gate that drives the net. Therefore if a BLE is being added to a cluster it causes all the nets connected to its input to get routed to the cluster it is being added to. This means that once the netlist is placed and routed then both the input and outputs nets of the BLE would have to get routed through the neighboring regions of the cluster which would involve usage of switches and buffers in the local region and hence power consumption in the local region, which would lead to IR-drops. Therefore the proposed technique considers both the input and the output nets while computing the transition density cost.

Define the target cluster transition density as $TargetTDC$. This is the upper limit value that the clustering technique should target for TDC. Selecting an appropriate value for $TargetTDC$ is important for the performance of the proposed technique and is explained later in this section.

4. For each ith candidate BLE for addition to the current cluster, compute

$$TD_i = \sum_{net_k \in NTD_i} TD_{net_k}, \tag{8.6}$$

which represents the additional transition density that would be added to the current cluster if BLE_i is added to the current cluster. It should be noted here that transition densities of only those nets, connected to BLE_i, are added to compute TD_i which are not present in the current cluster. The set of such nets is denoted by NTD_i, computed as in (8.4). This is because the nets which are already present in the current cluster, and which are also connected to BLE_i under consideration for addition to the cluster, have already been accounted for in the term TDC. Further, such nets would in any case be routed to the current cluster whether BLE_i is finally added or not, and would draw current from the power grid in the local region of the cluster.

5. Calculate the value of the *available transition density*, $ATDC$, per BLE, for the current cluster, such that the $TargetTDC$ is not exceeded, as follows,

$$ATDC = \frac{TargetTDC - TDC}{ClusterSize - NumBLEs}, \tag{8.7}$$

where $ClusterSize$ is the maximum number of BLEs that can be present in any logic cluster, and $NumBLEs$ is the number of BLEs present in the current cluster. This represents an average transition density that can be added per BLE, until the limit $TargetTDC$ is reached, for the remaining space in the logic cluster.

6. The gain function for the BLE_i, due to transition density is calculated as,

$$TDGain_i = 1 - \frac{TD_i}{ATDC} \tag{8.8}$$

It can be seen that if TD_i is greater than $ATDC$, then the gain would be negative which would discourage BLE_i from being added to current cluster. On the other hand if TD_i is much less than $ATDC$ the gain function would provide a strong attraction for BLE_i to be added to current cluster. TD_i can be small if the *new* nets connected to BLE_i have small transition density values.

7. Determine the new attraction function for BLE_i by modifying the cost function in (8.3) as follows,

$$Attraction(i) = (\alpha - \beta).Criticality(i) + (1-\alpha).\frac{Nets(B) \cap Nets(C)}{G} + \beta.TDGain_i, \tag{8.9}$$

where β is the transition density trade-off factor. The value of β is empirically determined to be 0.6 for best trade-offs.

8. Select the BLE based on the best *Attraction*(i).

9. Repeat the above steps until all the blocks have been clustered.

10. **Calculating TargetTDC:** $TargetTDC$ is calculated by first performing the clustering with the classical T-VPack clustering cost function in (8.3), and computing TDC_n, $n \in C$, where C is the set of all logic clusters.

$$\mu_{TDC} = \sum_{n \in C} \frac{TDC_n}{|C|} \tag{8.10}$$

$$\sigma_{TDC} = \sum_{n \in C} \sqrt{\frac{(TDC_n - \mu_{TDC})^2}{|C|}} \tag{8.11}$$

$$TargetTDC = \mu_{TDC} + \sigma_{TDC} \tag{8.12}$$

It can be seen from (8.12) that the $TargetTDC$ is based on the T-VPack clustering and provides an estimate of the target value of transition density for each cluster. Such a value is selected because it provides only a small deviation in the maximum targeted value of TDC for any cluster from the average TDC value. Further, the value of $TargetTDC$ computed in (8.12), gives a reasonable target value for the clustering of the netlist such that it tends to discourage building of clusters which have TDC much larger than the average value. When the clustering based on T-Vpack algorithm is carried out, which tries to reduce the circuit delay, the transition densities, TDC_i, are distributed in such a way so that the

some of the clusters have nets with many high transition density nets connected to it, such that the total transition density cost is high, whereas some of the clusters have low total transition density because the nets connected to it have smaller transition densities. From the perspective of transition density distribution, in the ideal case the transition densities would be equal for all the clusters and would have the value μ_{TDC}. Based on the value of $TargetTDC$, the IR-drop aware clustering algorithm computes the penalty associated with increase in transition density of a cluster beyond the value of $TargetTDC$. This essentially means that as the TDC of a cluster increases, the algorithm prefers adding low transition density BLEs to the cluster. If $TargetTDC$ is selected as μ_{TDC}, then it would be too restrictive and would have an adverse impact on the circuit delay. Since the TDC_i is distributed across the clusters, to take into account the variation of TDC distribution, $TargetTDC$ is computed as $TargetTDC = \mu_{TDC} + \sigma_{TDC}$. This ensures that a small deviation from the μ_{TDC} is allowed for $TargetTDC$ for good performance of the algorithm, and the value of the small deviation is computed based on the distribution of TDC. The form of (8.7) and (8.8), makes it harder for BLEs with new high transition density nets to get added to the current cluster if the $ATDC$ is small. This means if few BLEs with high transition density nets have already been added to the current cluster, then the proposed clustering routine would tend to select such BLEs which have lesser number of new high transition density nets. This avoids crowding of high transition density nets at a cluster and its neighboring switches, effectively reducing the IR-drops, while accounting for the critical delay. Also, the value of $TargetTDC$ in (8.12) is an estimate and a starting value, and it can be increased if it is found that choosing such a value leads to larger delays after the circuit is placed and routed, or if it leads to significant increase in the number of BLEs over the classical T-VPack clustering.

Apart from the above steps a final transition density cutoff value called $LimitTDC$ is employed which prevents more BLEs from being added to the current cluster. So if $TDC > LimitTDC$, addition of more BLEs to the current cluster is stopped, and the clustering procedure starts building a new cluster. The value of $LimitTDC$ is selected in the same way as the value for $TargetTDC$ is selected. It should be noted that if $TargetTDC$ and $LimitTDC$ is selected as a low value, then the total number of clusters would increase which can also lead to usage of more routing resources for high transition density nets, thereby causing the routing switches to draw more current from the power supply network. This leads to no reduction in IR-drops. However, $LimitTDC$ need not be employed always, as is the case in this work, where it is not used for some benchmarks for which it does not lead to any reduction in IR-drops. A brief outline of the proposed algorithm is given in Algorithm 5.

Algorithm 5: IR-Drop Aware Clustering Algorithm

repeat
  while *Current_Cluster_Size* $<$ *Cluster_Capacity* & $TDC < LimitTDC$ **do**
    Identify S_{BLE} and SP_{BLE};
    for $i \in SP_{BLE}$ **do**
      Calculate TD_i, (8.6);
      Calculate $ATDC$, (8.7);
      Find the gain value $TDGain_i$, (8.8);
      Compute the new attraction function $Attraction(i)$, (8.9);
    end
    Select the best BLE based on *Attraction*;
  end
until *Until all BLEs clustered* ;

8.4 Results and Discussions

The 45nm Predictive Model is chosen as the technology node for the simulations [69, 73]. The intermediate metal layers are used for the power grid mesh for delivering the current to the logic and routing cells. The pitch of the power grid network is taken as $20\mu m$. The clean V_{dd} nodes are available at a pitch of $300\mu m$. The V_{dd} for this technology node is 1V. The software package used to compute the voltages at the circuit nodes is GNU Circuit Analysis Package [93]. The netlist is clustered using the proposed technique, and it is then placed and routed to determine the actual circuit delay and IR-drops. The clock frequency at which a circuit can operate depends upon the critical delay of the circuit. The maximum clock frequency that a circuit can operate at is given by $f_{max} = \frac{1}{T_{crit}}$, where T_{crit} is the critical delay of the circuit. However, during actual operation the clock frequency can be kept smaller than f_{max}. The power for the baseline and the IR-drop aware implementations for a benchmark are computed at the same clock frequency which is chosen such that the critical circuit delays of both the implementations can meet the target clock frequency. For most of the benchmarks the clock frequency is selected as 100 MHz, except for those benchmarks which have critical delays larger than 10ns. For such benchmarks a slower clock speed is selected. Choosing such values of clock frequency provides the correct basis for comparison as both the baseline and IR-drop aware implementations run at the same clock frequency. Further, such clock speeds are normal for FPGAs [95, 96]. The cluster size in this work is selected as 8 because such a cluster size provides good speed and area trade-offs [1].

The results for the baseline implementation, using classical T-VPack clustering, and the IR-drop aware implementation employing the IR-drop aware clustering technique proposed in this work are shown in Table 8.1. The table lists the minimum V_{dd} at any node in the FPGA, and standard deviation of V_{dd} for both the baseline and IR-drop aware im-

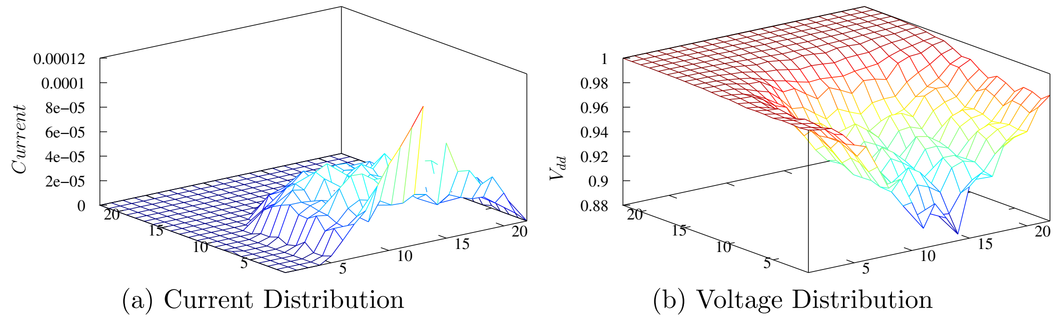
Table 8.1: Results of IR-Drop Aware Clustering

Benchmarks	Baseline		IR-Drop Aware		Improvement	
	Min V_{dd}	Std. Dev. V_{dd}	Min V_{dd}	Std. Dev. V_{dd}	IR-Drop (Reduction)	Std. Dev. V_{dd}
alu4	0.88533	0.027115	0.92258	0.024365	32.49%	10.14%
apex2	0.89635	0.026444	0.92042	0.021812	23.22%	17.52%
apex4	0.93317	0.017511	0.94534	0.015924	18.21%	9.06%
des	0.89611	0.019823	0.92375	0.017908	26.60%	9.66%
elliptic	0.91744	0.021169	0.93548	0.019267	21.85%	8.98%
ex1010	0.93380	0.014728	0.93981	0.013444	9.1%	8.72%
ex5p	0.92056	0.021204	0.93627	0.018312	19.77%	13.64%
frisc	0.94571	0.015190	0.95239	0.013242	12.30%	12.82%
misex3	0.91054	0.024679	0.92451	0.022697	15.62%	8.03%
pdc	0.94134	0.011542	0.94961	0.008921	14.10%	22.71%
s298	0.93035	0.019720	0.94402	0.015933	19.63%	19.2%
seq	0.88365	0.029062	0.92581	0.021004	36.23%	27.73%
spla	0.91792	0.018579	0.93862	0.016594	25.22%	10.68%
tseng	0.93824	0.017621	0.94424	0.015759	9.72%	10.57%

plementations. It can be seen from the table that a reduction of up to 36% in IR drop can be achieved using the proposed clustering technique. Further, a reduction in variance of up to 27% is also observed.

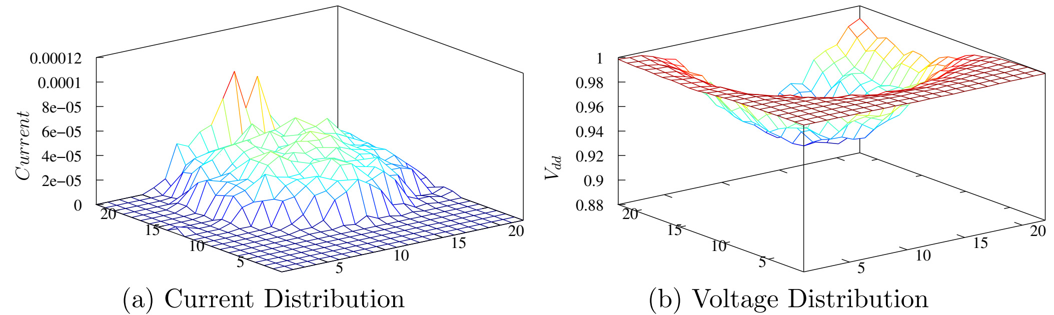
Fig. 8.5 shows the current and voltage distributions for the benchmark $alu4$, implemented using the T-VPack clustering technique. It can be seen that the maximum voltage drop is observed near the region having a large peak current. Fig. 8.6 shows the current and voltage distributions for the IR-drop aware implementation. In Fig. 8.6, the current peak is reduced resulting in improved minimum voltage. Additionally, since the proposed clustering technique reduces crowding of high transition density nets in a local region, the current distribution in the power grid in Fig. 8.6a is more uniform than the current distribution in Fig. 8.5a. This results in reduced IR-drop variance for the IR-drop aware clustering based implementation.

For some benchmarks such as $ex1010$, it can be seen that the reduction in IR-drop is small. This is because the baseline implementation has a good current distribution profile, which is relatively uniform across the different regions of the chip. This provides a limited scope for improvement. Fig. 8.7 shows the current distributions for the benchmark $ex1010$ for the baseline and the IR-drop aware implementations. It can be seen that the current distribution is relatively uniform in the baseline implementation also. However, a reduction



(a) Current Distribution (b) Voltage Distribution

Figure 8.5: Current and voltage distribution for the baseline implementation: *alu4*



(a) Current Distribution (b) Voltage Distribution

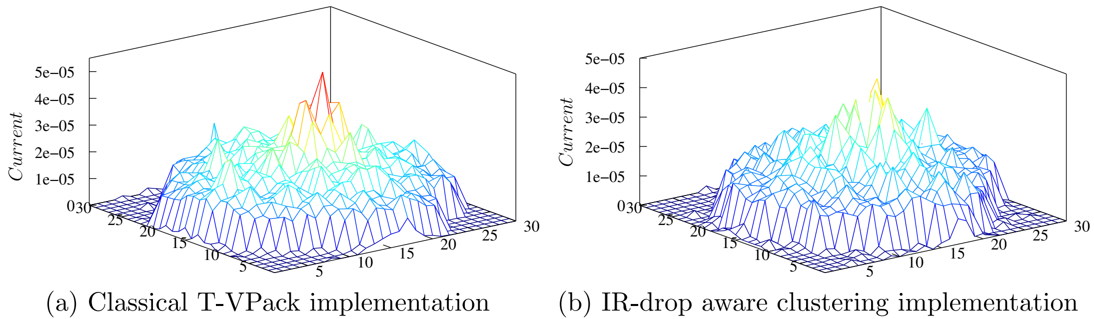
Figure 8.6: Current and voltage distribution for the IR-drop aware implementation: *alu4*

of 9.1% in IR-drop is still observed. Typically, a highly skewed current distribution profile would mean a larger IR-drop and somewhat larger variance in IR-drops across the chip. Therefore, the scope of improvement in minimum V_{dd} is more for such benchmarks which exhibit larger IR-drops with large current peaks. For example in the case of the benchmark $alu4$, the maximum average current at an FPGA tile is $121.7\mu A$, with an average value of $29.5\mu A$ for the complete chip, whereas in the case of the benchmark $ex1010$, the maximum average current at an FPGA tile is $54.6\mu A$, with an average value of $15.5\mu A$ for the complete chip. This means that the benchmark $alu4$ has tiles with current values having a large deviation from the average value and hence reducing those peak current values would result in improved voltage profile.

8.4.1 Trade-offs and Advantages

Delay and Number of Clusters

The IR-drop aware clustering technique proposed in this work alters the clustering with a trade-off associated with the critical delay of the final placed and routed circuit. The



(a) Classical T-VPack implementation (b) IR-drop aware clustering implementation

Figure 8.7: Current distributions for the baseline and IR-drop aware implementations: *ex1010*

delay of the IR-drop aware implementation is different from that of the classical T-VPack clustering because the composition of the clusters are different in the two cases. In Fig. 8.8, it can be seen that the average delay ratio of IR-drop aware clustering compared with the classical T-VPack based clustering is close to 1.12, if one of the benchmarks, *tseng* is excluded. The benchmark *tseng* is excluded because its critical delay is 40% larger for IR-drop aware clustering technique, and is not representative of the typical behavior. This can be attributed to the fact that the benchmark *tseng* has 131 logic blocks, with 52 inputs and 122 outputs, i.e., it has a high ratio of inputs/outputs to logic blocks. Since the IR-drop aware clustering technique uses a different cost function to build clusters, the number of logic clusters built in IR-drop aware implementation is different from that of the classical T-VPack implementation. The maximum size of a cluster is determined by the physical structure of the FPGA on which the netlist is to be mapped. So, for example, in this work the FPGA has been assumed to have a structure which can accommodate up to 8 BLEs per CLB. In the packing of the BLEs if some clusters have less than the maximum permissible BLEs, then also it is a feasible packing. T-Vpack packs the BLEs in such a way so that the circuit delay is minimized. This is primarily done by packing BLEs together which share connections. However, the IR-drop aware clustering attempts to pack clusters in such a way so that the transition density of a local region is reduced. Therefore it is somewhat likely that the addition of BLEs to clusters might not be as dense as the T-Vpack clustering. Even, in that case there is only a slight increase in the total number of clusters finally packed by the IR-drop aware clustering. On an average IR-drop aware implementation has 1.5% more clusters than the classical T-VPack implementation, with the maximum of 5.8% extra clusters for the benchmark *ex*5*p*.

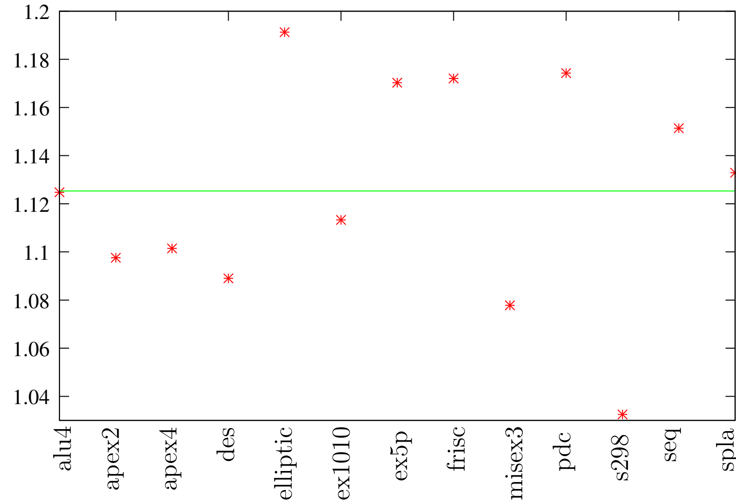


Figure 8.8: Ratio of the circuit delay for the IR-drop aware and baseline implementations.

Power Consumption

The impact of the proposed clustering technique on power is also evaluated, the results of which are shown in Table 8.2. The power for both the baseline and the IR-drop aware implementations were estimated at the same clock frequency. It is interesting to note that the IR-drop aware clustering technique reduces the power for all the benchmarks, except *tseng* and *des* for which the power remains almost the same for both the baseline and the IR-drop aware implementations. On an average IR-drop aware implementation reduces the power consumption by 12%. The power savings is obtained from the savings in the dynamic power, while the leakage power remains almost same for both the baseline and IR-drop aware implementations. The dynamic power reduces because the proposed clustering technique favors adding those blocks to the current cluster which share *high transition density* nets with the cluster being built up as explained in detail in section 8.3. This reduces the length of the high transition density nets resulting in reduced capacitance of those nets when they are finally routed. Therefore, this leads to reduction in dynamic power which is an additional advantage of the proposed clustering technique.

Algorithm Performance and Savings in Metal Area

The proposed CAD technique is fast because it does not rely on solving the power grid model during clustering stage, and therefore, do not lead to large penalties in run-time. Rather, an indirect methodology based on transition densities is employed. The time complexity of T-Vpack clustering is $O(n^2)$ [3], where n is the number of BLEs, and the proposed IR-drop aware clustering is also $O(n^2)$ since the proposed technique only com-

Table 8.2: Power savings and runtime for IR-drop aware clustering

Benchmarks	Power Results			Clustering Runtime (ms)	
	T-Vpack	IR-drop Aware	Reduction	T-Vpack	IR-drop Aware
alu4	7.1	6.74	5.1%	136	416
apex2	8.8	7.5	14.8%	159	271
apex4	4.2	3.6	14.3%	120	205
des	13.1	13.1	0%	147	296
elliptic	15.3	13.7	10.5%	410	1734
ex1010	13.8	11.6	14.4%	501	1097
ex5p	4.1	3.6	12.2%	107	207
frisc	10.4	8.5	18.3%	358	1067
misex3	6.4	5.7	10.9%	117	230
pdc	14.8	12.9	12.8%	505	1053
s298	8.8	6.5	26.1%	184	695
seq	8.22	6.6	19.7%	145	281
spla	11.8	10.8	8.5%	386	753
tseng	4.0	4.0	0%	101	265

putes and modifies the cost function. However, because of additional computations the IR-drop aware technique requires more runtime as compared to T-Vpack clustering. The runtime for the T-Vpack and the IR-drop aware techniques are shown in Table 8.2. The clustering algorithms were run on a linux machine with 3.06GHz Intel Xenon processor. On an average the IR-drop aware clustering is 2.5X slower than the T-Vpack clustering. The number of BLEs, i.e., n, varies from 1047 for $tseng$ to 4598 for $ex1010$. However, it should be noted that the most time consuming part in an FPGA synthesis flow is the placement and routing which requires orders of magnitude time compared to the packing. Therefore the overall impact of the IR-drop aware clustering algorithm is very small on the FPGA synthesis flow.

Another advantage of the proposed CAD techniques, is that the they do not impose any restriction on other power grid design and optimization techniques that can be applied on the FPGAs. The proposed CAD techniques can be used in conjunction with other power grid design techniques, such as wire sizing, in case the CAD techniques do not alone suffice to improve the reliability of the power grid. In such a case, the proposed technique can provide savings in the metal area for the power grid by reducing the amount by which the widths of the power metal lines need to be increased. For example, consider the case of the benchmark $apex2$, where the minimum voltage at any node in the FPGA using classical T-VPack clustering is 0.89635, and with IR-drop aware clustering technique the minimum

voltage at any node improves to 0.92042. To provide the same amount of improvement by widening the power grid metal lines instead of using the proposed IR-drop aware clustering technique, the metal lines need to be widened by approximately 33%. Since in an FPGA the end application is unknown the power grid metal lines need to be widened throughout the chip, unlike the case of ASICs, where selective widening can be carried out [41]. This will require a large increase in the metal area. Therefore, the proposed CAD techniques can provide a significant savings in metal area.

8.5 Conclusions

This chapter discussed a novel IR-drop aware clustering technique to improve the power grid reliability in FPGAs. The clustering technique reduces maximum IR-drops and reduces its variance. The technique relies on avoiding the crowding of high transition density nets near a cluster. It is observed that a reduction of up to 36% in IR-drop and 27% in standard deviation of V_{dd} can be achieved. The proposed technique has an additional advantage that an average reduction of 13% in total power is obtained due to dynamic power reduction.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis proposed novel CAD, architecture and circuit techniques for design of FPGAs under process variations for improving the timing yield and power yield of the FPGA. Subsequently, the power grid reliability improvement techniques were also proposed to reduce the IR-drops and improve the supply voltage profile in the power grid of the FPGAs. One of the key ideas in this work is to keep the proposed modifications in the architecture and circuit to minimum and propose most of the enhancements at the CAD level.

The design techniques for timing yield improvement are proposed at both architecture and CAD levels for reducing the impact of process variations on timing variability in FPGA designs. Results indicate that up to 28% improvement in (μ+3σ) of the critical delay can be obtained from the proposed methodology. CAD techniques for power yield improvement are proposed for dual-V_{dd} FPGA architecture. A variability aware placement technique is proposed which reduces the correlation between leaking blocks to reduce the leakage variability. Additionally, a variability aware dual-V_{dd} assignment technique is proposed to reduce the leakage variability. Results indicate that an average reduction of 15% in leakage variability can be obtained from the proposed methodology, with an average of 7.8% power yield improvement. A variability-aware transistor sizing and parameter optimization technique based on mathematical programming is proposed for FPGA interconnects to reduce leakage variability under timing constraints. Results show a reduction of 26% in leakage variability without any delay penalty.

Two CAD techniques have been proposed to improve the power grid reliability and the supply voltage profile of the FPGAs by reducing IR-drops and its spatial variability in the power grid of FPGAs. The first technique is an IR-drop aware place and route technique which reduces currents drawn in the local regions of the FPGA to reduce IR-drop and also

Table 9.1: Summary of Proposed Techniques

Proposed Technique	**Enhancements**	**Comments**
Timing yield improvement	Architecture, Placement and Routing	Routing architecture with different segment lengths explored
Power yield improvement	Placement, Dual-Vdd assignment	A dual-Vdd FPGA architecture selected for this work
Interconnect buffer optimization	Circuit	Optimization of different transistor parameters through mathematical programming
IR-drop aware place and route	Placement, Routing	Placement and routing accounts for transition densities in the nets to improve supply voltage profile
IR-drop aware clustering	Clustering	Clustering of LUTs accounts for transition densities in the nets to improve supply voltage profile

its spatial variation. The IR-drop aware place and route result in a maximum IR-drop reduction of up to 53% and a reduction in standard deviation of spatial supply voltage distribution by up to 66%. Another technique based on clustering is proposed which packs the LUTs in such a way so as to reduce the density of high activity nets in a logic block. This reduces the current drawn in a local region and hence reduces IR-drops in the power grid. Table 9.1 summarizes the list of techniques proposed in this thesis.

9.2 Future Work

The future work which can be explored for FPGA design under variability are as follows:

- **Timing-aware IR-drop improvement:** IR-drops in the power grid degrade the performance of the circuit which can lead to timing failures. Under such circumstances, the CAD tools should be designed to incorporate timing optimization under IR-drops to improve the performance of the circuit.

- **Architecture exploration for better optimization of the FPGA design under variability:** Architecture optimization can have a major impact on the robustness of FPGAs against variations. Understanding the impact of architectural parameters on the sensitivity of the FPGA to variations is the first step. The next step is finding if the architectural parameters can be altered to improve its robustness. It is also necessary to investigate if existing architectures need to modified in a fundamental way for the scaled nanometer technologies. Here, both the interconnect and the logic architectures need to be explored.

- **Incorporating temperature variability:** Temperature variations affect leakage power and can lead to high leakage variability. Temperature varies across the chip and would therefore lead to different leakage currents in different parts of the chip. However, this work considered a constant temperature across the chip. Assuming a constant worst case temperature across the chip can lead to pessimistic design resulting in increased cost. Incorporating temperature variability will lead to better designs with less cost.

References

[1] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for deep-submicron FPGAs.* Kluwer Academic Publishers, MA 1999. xii, 4, 6, 7, 8, 9, 10, 13, 27, 41, 44, 46, 60, 87, 91, 92, 110

[2] S. Borkar *et al.*, "Parameter variations and impact on circuits and microarchitecture," in *DAC*, 2003, pp. 338–342. xii, 1, 17

[3] A. Marquardt, V. Betz, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," in *FPGA*, 1999, pp. 37–46. xiv, 103, 104, 105, 114

[4] T. Tuan and B. Lai, "Leakage power analysis of a 90nm FPGA," in *CICC*, 2003, pp. 57–60. 2

[5] J. Anderson, F. Najm, and T. Tuan, "Active leakage power optimization for FPGAs," in *FPGA*, 2004, pp. 33–41. 2, 53

[6] A. Gayasen, Y. Tsai, N. Vijayakrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing leakage energy in FPGAs using region-constrained placement," in *FPGA*, 2004, pp. 51–58. 2, 67

[7] F. Li, Y. Lin, L. He, and J. Cong, "Low power FPGA using predefined dual-vdd/dual-vt fabrics," in *FPGA*, 2004, pp. 42–50. 2, 53, 54

[8] A. Kumar and M. Anis, "Dual threshold CAD framework for subthreshold leakage power aware FPGAs," *IEEE Trans. on CAD*, vol. 26, no. 1, pp. 53–66, Jan 2007. 2

[9] A. Sangiovanni-Vincentelli, A. E. Gamal, and J. Rose, "Synthesis methods for field-programmable gate arrays," *Proc. of IEEE*, vol. 81, no. 7, pp. 1057–1083, July 1993. 9

[10] R. Brayton, G. Hachtel, and A. Sangiovanni-Vincentelli, "Multilevel logic synthesis," *Proc. of IEEE*, vol. 78, no. 2, pp. 264–300, Feb 1990. 9

[11] J. Cong and Y. Ding, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Trans. on CAD*, vol. 13, no. 1, pp. 1–12, 1994. 9, 15

[12] J. Cong, J. Peck, and Y. Ding, "RASP: A general logic synthesis system for SRAM-based FPGAs," in *FPGA*, 1996, pp. 137–143. 9

[13] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," in *Science*, 1983, pp. 671–689. 11

[14] M. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing," in *ICCAD*, 1986, pp. 381–384. 11

[15] W. Swartz and C. Sechen, "New algorithms for placement and routing of macro cells," in *ICCAD*, 1990, pp. 336–339. 11

[16] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *FPGA*, 2000, pp. 203–213. 12, 43, 44, 62, 63, 88, 90

[17] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, "Placement and routing tools for Triptych FPGA," *IEEE Trans. on VLSI*, vol. 3, no. 4, pp. 473–482, Dec 1995. 12

[18] V. Betz, *VPR and T-Vpack User's Manual (Version 4.30).* [Online]. Available: http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html 13

[19] E. M. Sentovich *et al.*, "SIS: A system for sequential circuit analysis," University of California, Berkeley, Tech. Rep., 1992. 15

[20] O. Unsal *et al.*, "Impact of parameter variations on circuits and microarchitecture," *IEEE Micro*, vol. 26, no. 6, pp. 30–39, Nov 2006. 17, 19

[21] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1440, Oct 1989. 20

[22] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. on CAD*, vol. 24, no. 9, pp. 1467–1482, September 2005. 20, 21, 33

[23] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *ICCAD*, Nov. 2003, pp. 900 – 907. 20, 33, 34, 45, 67

[24] B. Cline, K. Chopra, D. Blaauw, and Y. Cao, "Analysis and modeling of CD variation for statistical static timing," in *ICCAD*, 2006, pp. 60–66. 23, 45, 67

[25] S. Kulkarni, D. Sylvester, and D. Blaauw, "A statistical framework for post-silicon tuning through body bias clustering," in *ICCAD*, 2006, pp. 39–46. 25

[26] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester, "Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation," in *ICCAD*, 2005, pp. 1023–1028. 25, 69

[27] M. Mani, A. Singh, and M. Orshansky, "Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization," in *ICCAD*, 2006, pp. 19–26. 25

[28] A. Agarwal, K. Chopra, D. Blaauw, and V. Zoltov, "Circuit optimization using statistical static timing analysis," in *DAC*, 2005, pp. 321–324. 25

[29] A. Srivastava, S. Shah, K. Agarwal, D. Sylvester, D. Blaauw, and S. Director, "Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance," in *DAC*, 2005, pp. 535–540. 25, 26

[30] A. Datta, S. Bhunia, J. Choi, S. Mukhopadhyay, and K. Roy, "Speed binning aware design methodology to improve profit under parameter variations," in *ASPDAC*, 2006, pp. 712–717. 25, 26, 53

[31] X. Bai, C. Visweswariah, P. Strenski, and D. Hathaway, "Uncertainty-aware circuit optimization," in *DAC*, 2002, pp. 58–63. 25

[32] A. Srivastava, D. Sylvester, and D. Blaauw, "Statistical optimization of leakage power considering process variations using dual-Vth and sizing," in *DAC*, 2004, pp. 773–778. 25

[33] S. Sapatnekar and H. Su, "Analysis and optimization of power grids," in *IEEE Design and Test*, 2002, pp. 7–15. 26

[34] R. Saleh, S. Z. Hussain, S. Rochel, and D. Overhauser, "Clock skew verification in the presence of IR-drop in the power distribution network," *IEEE Trans. on CAD*, vol. 19, no. 6, pp. 635–644, June 2000. 26

[35] X. Tan, C. Shi, and J. Lee, "Reliability-constrained area optimization of vlsi power/ground networks via sequence of linear programmings," *IEEE Trans. on CAD*, vol. 22, no. 12, pp. 1678–1684, Dec 2003. 26

[36] K. Wang and M. Sadowska, "On-chip power supply network optimization using multigrid-based technique," *IEEE Trans. on CAD*, vol. 24, no. 3, pp. 407–417, Mar 2005. 26, 84

[37] M. Zhao, Y. Fu, V. Zoltov, S. Sundareswaran, and R. Panda, "Optimal placement of power supply pads and pins," in *DAC*, 2004, pp. 165–170. 26

[38] J. Singh and S. Sapatnekar, "Congestion-aware topology optimization of structured power/ground networks," *IEEE Trans. on CAD*, vol. 24, no. 5, pp. 683–695, May 2005. 26

[39] H. Su, K. Gala, and S. Sapatnekar, "Fast analysis and optimization of power/ground networks," in *ICCAD*, 2000, pp. 477–480. 26

[40] H. Chen and D. Ling, "Power supply noise analysis methodology for deep submicron vlsi chip design," in *DAC*, 1997, pp. 638–643. 26

[41] J. Singh and S. Sapatnekar, "Partition-based algorithm for power grid design using locality," *IEEE Trans. on CAD*, vol. 25, no. 4, pp. 664–677, Apr 2006. 26, 84, 85, 116

[42] Y. Lin and L. He, "Stochastic physical synthesis for FPGAs with pre-routing interconnect uncertainty and process variation," in *FPGA*, 2007, pp. 80–88. 27, 29

[43] Y. Lin, M. Hutton, and L. He, "Placement and timing for FPGAs considering variations," in *FPL*, 2006, pp. 1–7. 27, 54, 60

[44] G. Nabaa, N. Azizi, and F. N. Najm, "An adaptive FPGA architecture with process variation compensation and reduced leakage," in *DAC*, 2006, pp. 624–629. 27, 28

[45] H. Wong, L. Cheng, Y. Lin, and L. He, "FPGA device and architecture evaluation considering process variations," in *ICCAD*, 2005, pp. 19–24. 27, 28

[46] P. Sedcole and P. Y. K. Cheung, "Parameteric yield in FPGAs due to within-die delay variations: A quantitative analysis," in *FPGA*, 2007, pp. 178–187. 27

[47] S. Sivaswamy and K. Bazargan, "Variation aware routing for FPGAs," in *FPGA*, 2007, pp. 71–79. 27, 28

[48] S. Srinivasan and V. Narayanan, "Variation aware placement for FPGAs," in *ISVLSI*, 2006, pp. 422–423. 27

[49] L. Cheng, J. Xiong, L. He, and M. Hutton, "FPGA performance optimization via chipwise placement considering variations," in *FPL*, 2006, pp. 1–6. 27

[50] A. Kumar and M. Anis, "FPGA design for timing yield under process variations," *IEEE Trans. on VLSI*, vol. 18, no. 3, pp. 423–435, Mar 2010. 32

[51] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *DAC*, 2004, pp. 331–336. 33

[52] J. Singh and S. Sapatnekar, "A scalable statistical static timing analyzer incorporating correlated non-gaussian and gaussian parameter variations," *IEEE Trans. on CAD*, vol. 27, no. 1, pp. 160–173, January 2008. 33

[53] S. Garg and D. Marculescu, "3D-GCP: An analytical model for the impact of process variations on the critical path delay distribution of 3D ICs," in *ISQED*, 2009, pp. 147–155. 33

[54] K. Bowman, S. Duvall, and J. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 183 –190, Feb 2002. 33

[55] A. Agarwal, D. Blaauw, V. Zoltov, and S. Vrudhula, "Computation and refinement of statistical bounds on circuit delay," in *DAC*, 2003, pp. 348–353. 33

[56] K. Chopra, B. Zhai, D. Blaauw, and D. Sylvester, "A new statistical max operation for propagating skewness in statistical timing analysis," in *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design.* New York, NY, USA: ACM, 2006, pp. 237–243. 35

[57] BSIM4 MOS Models. [Online]. Available: http://www-device.eecs.berkeley.edu/bsim3/bsim4.html 35, 36, 56

[58] Y. Taur *et al.*, "CMOS scaling into the nanometer regime," *Proc. of IEEE*, vol. 85, no. 4, pp. 486–504, April 1997. 35, 57

[59] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes.* McGraw Hill, 2002, 4th Edition. 35, 57, 68

[60] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits.* Prentice Hall, 2003, Second Edition. 37

[61] J. W. T. et. al., "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1396–1402, Nov 2002. 49

[62] J. Anderson and F. Najm, "A novel low power FPGA routing switch," in *CICC*, 2004, pp. 719–722. 53

[63] A. Rahman, "Determination of power gating granularity for FPGA fabric," in *CICC*, 2006, pp. 9–12. 53, 55

[64] A. Kumar and M. Anis, "Power-yield enhancement for FPGAs under process variations," *ASP Journal of Low Power Electronics*, vol. 6, pp. 280–290, Aug 2010. 53

[65] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-vdd," in *DAC*, 2004, pp. 735–740. 54

[66] A. Gayasen *et al.*, "A dual-Vdd low power FPGA architecture," in *FPL*, 2004. 54, 55

[67] A. Devgan and S. Nassif, "Power variability and its impact on design," in *VLSI Design*, 2005, pp. 679–682. 56

[68] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *FPL*, 2002, pp. 312–321. 56, 86

[69] Predictive MOS Models. [Online]. Available: http://ptm.asu.edu/ 57, 67, 81, 92, 110

[70] A. Kumar and M. Anis, "An analytical state dependent leakage power model for FPGAs," in *DATE*, 2006, pp. 612–617. 59, 86

[71] S. Sirichotiyakul *et al.*, "Duet: an accurate leakage estimation and optimization tool for dual-Vt circuits," *IEEE Trans. on VLSI*, vol. 10, pp. 79–90, April 2002. 59

[72] K. Usami *et al.*, "Automated low power technique exploiting multiple supply voltages applied to a media processor," *IEEE JSSC*, vol. 33, no. 3, pp. 463–472, 1998. 65

[73] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov 2006. 67, 81, 92, 110

[74] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical estimation of leakage current considering inter and intra die process variation," in *ISLPED*, 2003, pp. 84–89. 67, 68, 81

[75] S. Schwartz and Y. Yeh, "On the distribution function of the moments of power sums with lognormal components," *Bell Systems Technical Journal*, vol. 61, pp. 1441–1462, 1982. 67

[76] A. Kumar and M. Anis, "Interconnect design for FPGAs under process variations for leakage power yield," in *IEEE International NEWCAS Conference*, 2010, pp. 249–352. 75

[77] W. Zhao, F. Liu, K. Agarwal, D. Acharyya, S. Nassif, K. Nowka, and Y. Cao, "Rigorous extraction of process variations for 65-nm cmos design," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 22, no. 1, pp. 196 –203, feb. 2009. 75

[78] P. Gupta, A. B. Kahng, P. Sharma, and D. Sylvester, "Selective gate-length biasing for cost-effective runtime leakage control," in *DAC*, 2004, pp. 327–330. 79

[79] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, J. Neyman, Ed. University of California Press, Berkeley, CA, USA, 1950, pp. 481–492. 81

[80] A. H. Ajami, K. Banerjee, A. Mehrotra, and M. Pedram, "Analysis of IR drop scaling with implications for deep submicron P/G network designs," in *ISQED*, 2003. 84

[81] M. Z. et. al., "Worst case clock skew under power supply variations," in *TAU*, 2002, pp. 22–28. 84

[82] M. Hashimoto, T. Yamamoto, and H. Onodera, "Statistical analysis of clock skew variation in h-tree structure," in *ISQED*, 2005, pp. 402–407. 84

[83] A. Kumar and M. Anis, "IR-drop management in FPGAs," *IEEE Trans. on CAD*, vol. 29, no. 6, pp. 988–993, Jun 2010. 84

[84] ——, "IR drop management CAD techniques in FPGAs for power grid reliability," in *ISQED*, 2009, pp. 746–752. 84

[85] H. Su, K. Gala, and S. Sapatnekar, "Analysis and optimization of structured power/ground networks," *IEEE Trans. on CAD*, vol. 22, no. 11, pp. 1533–1544, Nov 2003. 85

[86] H. Chen, C. Cheng, A. Kahng, M. Mori, and Q. Wang, "Optimal planning for mesh based power distribution," in *ASPDAC*, 2004. 85

[87] F. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Trans. on CAD*, vol. 12, no. 2, pp. 310–323, Feb 1993. 86

[88] ——, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. on CAD*, vol. 2, no. 4, pp. 446–455, Dec 1994. 86

[89] G. Chen and S. Sapatnekar, "Partition driven standard cell thermal placement," in *ISPD*, 2003, pp. 75–80. 90

[90] C. Tsai and S. Kang, "Cell level placement for improving substrate thermal distribution," *IEEE Trans. on CAD*, vol. 19, no. 2, pp. 253–266, Feb 2000. 90

[91] N. Srivastava, X. Qi, and K. Banerjee, "Impact of on-chip inductance on power distribution network design for nanometer scale integrated circuits," in *ISQED*, 2005. 92

[92] Q. K. Zhu, *Power Distribution Network Design for VLSI.* Wiley-Interscience, 2004. 92

[93] A. Davis, *The GNU Circuit Analysis Package Users Manual.* [Online]. Available: http://www.gnucap.org 92, 110

[94] A. Kumar and M. Anis, "IR-drop aware clustering for robust power grid in FPGAs," *IEEE Trans. on VLSI (In Press).* 100

[95] Altera Stratix 4. [Online]. Available: http://www.altera.com 110

[96] Xilinx Virtex 4. [Online]. Available: http://www.xilinx.com 110