UC San Diego
UC San Diego Electronic Theses and Dissertations

Title
Physical Design Methodologies for the More-than-Moore Era

Permalink
https://escholarship.org/uc/item/56h7w8qh

Author
Chan, Wei-Ting

Publication Date
2018

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Physical Design Methodologies for the More-than-Moore Era

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Wei-Ting Chan

Committee in charge:

Professor Andrew B. Kahng, Chair
Professor Chung-Kuan Cheng
Professor Rajesh K. Gupta
Professor Bill Lin
Professor Tajana Šimunić Rosing
Professor Dean M. Tullsen

2018

Copyright
Wei-Ting Chan, 2018
All rights reserved.

The dissertation of Wei-Ting Chan is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

____

____

____

____

____

____

Chair

University of California, San Diego

2018

DEDICATION

To my family.

TABLE OF CONTENTS

Signature Page . . . iii

Dedication . . . iv

Table of Contents . . . v

List of Figures . . . viii

List of Tables . . . xii

Acknowledgements . . . xiii

Vita . . . xvi

Abstract of the Dissertation . . . xix

Chapter 1 Introduction . . . 1
- 1.1 Emerging More-Moore Challenges . . . 2
- 1.2 Emerging More-Than-Moore Challenges . . . 4
- 1.3 New Opportunities of Machine Learning to Meet Scaling Challenges . . . 5
- 1.4 Organization of Chapters . . . 6

Chapter 2 Yield and Reliability Optimization . . . 9
- 2.1 Yield Challenge in Early Stage of Technology Adoption . . . 9
  - 2.1.1 Related Works . . . 12
  - 2.1.2 Problem Statement and Methodology . . . 13
  - 2.1.3 Yield Improvement by Exploiting Redundancy . . . 17
  - 2.1.4 Redundant Logic Insertion . . . 23
  - 2.1.5 Experimental Setup and Results . . . 29
  - 2.1.6 Conclusions . . . 34
- 2.2 Reliability Challenge in Advanced Nodes . . . 36
  - 2.2.1 Related Works . . . 38
  - 2.2.2 Methodology for Reliability Analysis . . . 39
  - 2.2.3 Experimental Setup and Results . . . 46
  - 2.2.4 Conclusions . . . 55
- 2.3 Acknowledgments . . . 55

Chapter 3 Machine Learning-based Routability Optimization . . . 57
- 3.1 Routability Challenge in Advanced Nodes . . . 58
- 3.2 Related Works . . . 60
- 3.3 Methodology for Routability Optimization . . . 61
  - 3.3.1 Machine Learning-based Optimization for Routability . . . 62
  - 3.3.2 Predictor Design . . . 63
  - 3.3.3 Predictor-guided Routability Optimization . . . 70
- 3.4 Experimental Setup and Results . . . 72

3.5 Conclusions . . . 74
3.6 Acknowledgments . . . 75

Chapter 4 Design Space Exploration for 3DIC . . . 76
4.1 Background of 3DIC . . . 76
4.2 Related Works . . . 79
4.3 Methodology for 3D Benefit Estimation . . . 81
4.3.1 Pseudo-1D Implementation . . . 81
4.3.2 Optimal 2D Implementation . . . 82
4.3.3 Power Benefit Estimation for 3DICs . . . 83
4.3.4 Implementation in Infinite Dimension . . . 85
4.4 Experimental Setup and Results . . . 86
4.4.1 Evaluation of 3D Benefits . . . 86
4.4.2 A More Realistic Evaluation . . . 91
4.4.3 Comparison Between Improved 2D versus 3D . . . 92
4.4.4 Assessment of 3D Benefit Across Technologies . . . 93
4.4.5 Netlist Study . . . 94
4.4.6 SoC-Level 3D Benefits . . . 99
4.5 Conclusions . . . 100
4.6 Acknowledgments . . . 102

Chapter 5 Modeling for Approximate Computing . . . 103
5.1 Background of Approximate Computing and Related Works . . . 103
5.2 Related Works . . . 107
5.2.1 Error Metrics . . . 107
5.2.2 Approximate Arithmetic Modules . . . 108
5.2.3 Analysis and Composition of Errors . . . 108
5.3 Methodology for Error Estimation . . . 110
5.3.1 Analysis of Existing Interval-based Approach . . . 110
5.3.2 Analysis for Computation of Error Metrics . . . 112
5.3.3 Proposed Approach to Estimate EMTs . . . 113
5.4 Experimental Setup and Results . . . 119
5.5 Conclusions . . . 125
5.6 Acknowledgments . . . 126

Chapter 6 Modeling and Optimization for Stochastic Computing . . . 127
6.1 Background of Stochastic Computing and Related Works . . . 127
6.1.1 Error Analysis . . . 135
6.1.2 Stochastic Number Generation . . . 140
6.2 Problem Description and Proposed Solutions . . . 141
6.2.1 Finding the Minimum Energy Point . . . 142
6.2.2 Solution of the MEOP Problem . . . 145
6.2.3 Error Estimation using a Markov Chain . . . 145
6.3 Methodology for Circuit-Level Optimization . . . 151
6.3.1 Arrival Time Misalignment Matters . . . 151
6.3.2 Optimization Methodologies . . . 153

6.4 Experimental Setup and Results 162
6.4.1 Circuit Optimization Results 162
6.4.2 Validation of MC Model 166
6.4.3 Comparison with Conventional Circuits 167
6.5 Conclusions 167
6.6 Acknowledgments 169

Chapter 7 Conclusion and Future Directions 170

Bibliography 174

LIST OF FIGURES

Figure 1.1: The scope of this thesis and the topic(s) covered by each chapter. 2
Figure 1.2: The projection of increasing current density on metal interconnect, which leads to deterioration of interconnect reliability due to electromigration [242]. 3
Figure 1.3: The projections of design rule numbers in advanced nodes. The rapid growth is due to complicated lithography and shrunk physical dimensions. The figure is reproduced from [234]. 3
Figure 1.4: Potential technology solutions for More-than-Moore challenges projected by [243]. . 4

Figure 2.1: The interactions among yield, amount of redundancy, and other design parameters. . 11
Figure 2.2: Layouts of original cells, redundant cells and overlay. 11
Figure 2.3: Illustration of choosing redundant logic clusters. 14
Figure 2.4: The flowchart of our proposed methodology for yield improvement by inserting redundancy. 15
Figure 2.5: Projected yield improvement by our model. 19
Figure 2.6: Fitting result for our yield model. 22
Figure 2.7: Yield simulation of a *JPEG* block. 23
Figure 2.8: The cluster area to MUX area ratio versus cluster size of *AES*, *JPEG*, and *VGA*. . . . 25
Figure 2.9: Algorithm runtime versus cell numbers in designs. 26
Figure 2.10: Timing path and inserted MUX cells. 27
Figure 2.11: The histogram of cluster size ratios of *AES*, *JPEG*, and *VGA*. 30
Figure 2.12: Number of terminals versus sizes of cluster of logic gates, reproduced from [84]. . . 31
Figure 2.13: Illustration of the chicken-and-egg loop for circuit signoff with EM and AVS to compensate for performance degradation due to BTI. 37
Figure 2.14: Interconnect resistance increase in percentage due to different current densities in 28nm BEOL technology. 41
Figure 2.15: Impact of ΔR on path delay with different capacitive loadings. $V_{dd} = 1.1V$. 43
Figure 2.16: Impact of ΔR on path delay due to EM stress with different gate sizes. $V_{dd} = 1.1V$. . 43
Figure 2.17: Illustration of P/G mesh, V_{dd}, and $V_{regulator}$. 45
Figure 2.18: Power comparison between different signoff corners with EM stress $\Delta R_{PG} > 0$. . . . 46
Figure 2.19: Lifetime and V_{final} of the eight implementations of the *AES* design. 49
Figure 2.20: Power and area due to EM fixes on AVS systems using both Black's Equation and [164] models for the *AES* design. 50
Figure 2.21: Power and area due to EM fixes on AVS systems using both Black's Equation and [164] models for the *DMA* design. 51
Figure 2.22: Impact on EM lifetime due to five different voltage schedules in AVS systems for *DMA* implementations (a) #3 and (b) #4. 53
Figure 2.23: Impact on EM lifetime due to five different voltage schedules in AVS systems for *AES* implementations (a) #3 and (b) #4. 54

Figure 3.1: Comparison between the GR-based congestion map and the actual DRVs (gcells with DRC violations) on a sub-14nm design. 59
Figure 3.2: An example of the analysis of a false negative prediction. 65
Figure 3.3: Accuracy comparisons between different parameter sets and mathematical models. (a) Linear regression; (b) logistic regression; and (c) SVM classifier. 67

Figure 3.4: Comparison between our learning-based DRV hotspot map and the actual DRVs (gcells with DRVs) on a sub-14*nm* design. 69
Figure 3.5: The predictor-guided routability optimization algorithm. 71
Figure 3.6: The experimental methodology used to evaluate the proposed predictor-guided routability optimization algorithm. 73

Figure 4.1: Design power with (a) varying synthesis clock period, and (b) placement utilization. Design: *JPEG*. Technology: 28*nm* FDSOI. 82
Figure 4.2: Design power and total cell area evaluated across various implementation dimensions. 89
Figure 4.3: Maximum potential (performance, power, area/cost) improvements from 3D integration. 90
Figure 4.4: The impact of larger clock skew (due to complexity of 3D CTS as well as inter-tier variation) on 3D power benefits. 91
Figure 4.5: Comparison between estimated 3D power and area benefits with two tiers versus power and area reductions from the latest version of a commercial P&R tool (i.e., 2D+). 92
Figure 4.6: 3D benefits assessed in infinite dimension showing consistency across various technologies. 93
Figure 4.7: Power and power benefits versus Rent parameters for the 2D and the 3D implementations with different tiers. 96
Figure 4.8: Power benefits correlate with Rent parameters. 97
Figure 4.9: Correlation between incidence of cells with >3 inputs vs. Rent parameter. 98
Figure 4.10: Power vs. Rent parameter with Rent Modulation. 98
Figure 4.11: Potential 3D benefit from optimized bock-level planning. 100

Figure 5.1: Illustration of approximation module replacement, in which accurate hardware modules are replaced by approximate ones. 105
Figure 5.2: Probability mass function (PMF) used in the interval-based approach. 111
Figure 5.3: (a) Five-node testcase. (b) Runtime from interval-based approach for each sample size. (c) ER estimation results from interval-based approach. 113
Figure 5.4: The structure of an ETAIIM approximate adder. CLAs are carry-lookahead subadders. RCAs are ripple-carry sub-adders. 114
Figure 5.5: The simulated EMT results for input distributions. A 24-bit signed ETAIIM adder is simulated in the analysis. 115
Figure 5.6: EMT estimation at a given node (approximate module) considering intrinsic and propagated EMTs. 116
Figure 5.7: Estimated STD_A or STD_B and EMT_{in} values obtained from lookup tables. 117
Figure 5.8: (a) Our proposed approach for error estimation, and (b) the lookup tables in the precharacterized libraries for EMT_{in} and STD_Z. 121
Figure 5.9: Configuration of (a) finite impulse response (FIR) filter and (b) multiply-accumulator (MAC) circuits used in the experiments. 122
Figure 5.10: Comparison of ER metrics between our approach and the interval-based approach. . 123
Figure 5.11: Comparison of ES metrics between our approach and the interval-based approach. The inaccuracy on the right side is (2×10^9). 124
Figure 5.12: Comparison of (a) runtime for error composition and (b) inaccuracy of EMT estimation for the MAC circuits with different testcase sizes. 124

Figure 5.13: Comparison of inaccuracy with respect to the number of nodes in randomly generated circuits. 125

Figure 6.1: Stochastic computing circuit implementing the function $Z = \frac{1}{4} + \frac{1}{2}X_1X_2$. The stochastic number represented by each bit-stream is the probability of seeing a 1 in a randomly chosen position. 128
Figure 6.2: Example showing how a representative SC circuit behaves under voltage/frequency scale. 130
Figure 6.3: (a) SC circuit implemented via conventional P&R tools, where the main objectives are area and power reduction. (b) SC circuit implemented via the proposed optimization flow, where path delays are balanced to improve accuracy. Inserted buffers are shown in red. 132
Figure 6.4: Basic SC components: (a) multiplier, (b) scaled adder, (c) stochastic number generator, and (d) stochastic-to-binary converter. 134
Figure 6.5: Effect of delay errors on an SN. 137
Figure 6.6: Three SNs with different activity factors representing $Z = 1/2$. 141
Figure 6.7: The effect of SN generation on output errors; they are lower when a low-activity SNG is used. 142
Figure 6.8: Misalignment of arrival times at z with respect to the clock capture phase can lead to a computation error. 144
Figure 6.9: Markov chain (MC) model for the proposed error estimation approach. The states are described in Table 6.1. 147
Figure 6.10: Flow to construct MC model and estimate computation errors across various operating points. 147
Figure 6.11: Distribution of the estimation errors across different operating points. 149
Figure 6.12: Plots showing correlation between the errors estimated by our proposed Markov chain (MC) model and the errors obtained from post-layout simulations. 150
Figure 6.13: Simulation results of the *PolySmall* testcase synthesized in $28nm$ FDSOI technology and clocked at $6.7GHz$: (a) path delay distributions of two implementations, and (b) computation error for different input delays. 152
Figure 6.14: Path delays (top) and average computation errors (bottom) at supply voltages ranging from $0.72V$ to $0.98V$ for testcase *PolySmall* at $28nm$ FDSOI. 154
Figure 6.15: Energy of designs optimized with different β values. The target average error rate is 0.02. Operating points are selected based on exhaustive simulation so as to minimize energy. 157
Figure 6.16: Energy comparison of three *GammaCorrection* circuits optimized with different β values. 157
Figure 6.17: Evaluation of the impact of process and temperature variations on the average error. The testcase *GammaCorrection* is optimized at 125°C and simulated at different corners and temperatures. 159
Figure 6.18: Path delay for the four different corners in Figure 6.17. 159
Figure 6.19: Applied buffering styles: a single-stage non-inverting buffer, and an inverter pair with routing detour. 160
Figure 6.20: (a) Buffering solution space, i.e., delay range and delay-voltage tradeoff range, with multiple stages of buffers/inverter pairs. The circle colors denote different numbers of stages of buffers/inverter pairs. (b) Pruned buffering candidates. 161

Figure 6.21: Layout of routing detour. 162
Figure 6.22: Optimization results for the *GammaCorrection* testcase. 163
Figure 6.23: Energy comparison for different accuracy requirements (err_{goal}) between unoptimized implementations (blue solid line) and optimized circuits (green dashed line). . 165
Figure 6.24: Energy comparison between operating points from MC-based search (green dashed line) and exhaustive search (blue solid line) for different accuracy goals (err_{goal}). . . 166
Figure 6.25: Voltage scaling results of *GammaCorrection* (top, design obtained from [191]) and *EdgeDetection* (bottom, design obtained from [12]) executed by conventional and stochastic circuits. 168

LIST OF TABLES

Table 2.1: Description of notations used in our formulation for redundancy insertion. 16
Table 2.2: Yield improvement and P&R qualities of (i) original routed layouts, (ii) post-ECO layouts with redundant logic, and (iii) post-optimization layouts. 33
Table 2.3: EM and BEOL parameters. 42
Table 2.4: Signoff corners for BTI. V_{min} = 0.90V, and V_{max} = 1.10V. 44

Table 3.1: Learning-based prediction: confusion matrix of our learning-based predictor. True-positive rate = 74% and false-positive rate = 0.2%. 68
Table 3.2: GR-based prediction: confusion matrix of prediction by GR shown in Figure 3.1. True-positive rate = 24% and false-positive rate = 0.5%. 69
Table 3.3: Comparisons between the default and the learning-optimized layouts. 74

Table 4.1: Summary of 3D benefits studied in previous works. 80
Table 4.2: Summary of benchmarks used to evaluate 3D benefits. 86
Table 4.3: Summary of Rentian testcases with different Rent parameters. 95
Table 4.4: Area scaling ratios of logic cells during synthesis in Figure 4.10. 99

Table 5.1: Categories of error analysis, propagation, and optimization works. 109
Table 5.2: (a) Regression coefficients derived with different hardware configurations, (b) estimation inaccuracy with and without regression, and (c) absolute estimation inaccuracy with and without regression. 120
Table 5.3: Estimation inaccuracy of a four-tap FIR filter shown in Figure 5.9(a). 121

Table 6.1: Description of each state in the MC model. 146
Table 6.2: Description of the notation used in the MILP. 155
Table 6.3: Summary of stochastic circuit testcases. 163

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Professor Kahng, for his insightful and sincere advice throughout my whole Ph.D. study, as well as opportunities to collaborate with top-notch researchers and have access to valuable research resources. I would also like to thank Mrs. Kahng (Son Young) for her years of support to the research group and my technical writing.

I appreciate the companionship of my wife Yadi and our daughter Iris. They have changed my life forever. I appreciate my father Chun-Ping Chan and my mother Mei-Huei Wang for everything. I also appreciate all the support from other family members and my friends.

I would also like to thank my labmates (Hyein Lee, Kwangsoo Han, Lutong Wang, Bangqi Xu, Ahmed Youssef, Tushar Shah, Sriram Venkatesh), and former members in the group (Professor Seokhyeong Kang, Dr. Tuck-Boon Chan, Dr. Siddhartha Nath, Dr. Jiajia Li, Dr. Ilgweon Kang, Mulong Luo, Yaping Sun), and especially to Professor Seokhyeong Kang, Dr. Tuck-Boon Chan, Dr. Siddhartha Nath, and Dr. Jiajia Li, who were extremely supportive throughout the learning process.

My sincere thanks also go to my thesis committee members Professor Chung-Kuan Cheng, Professor Rajesh K. Gupta, Professor Bill Lin, Professor Tajana Šimunić Rosing, and Professor Dean M. Tullsen for their time, encouragement and insightful comments.

I would like to thank Dr. Prashant Saxena and Dr. Pei-Hsin Ho for providing me with an excellent interternship opportunity to work with them. I would like to thank Dr. Juan-Antonio Carballo, Dr. Paolo Gargini, Mr. Gary Smith, and Mr. Ichiro Yamamoto for their inputs and shared knowledge when we worked on the System Integration chapter for ITRS 2.0.

I would like to express my special appreciation to Ms. Candace Gietzen and Mr. Richard Fridshal, who not only tutored me in language but also shared valuable life experiences with me. Their support and life wisdom were also tremendously helpful when Yadi and I were new parents.

The material in this thesis is based on the following publications.

Chapter 2 contains reprints of W.-T. J. Chan, A. B. Kahng and S. Nath, "Methodology for Electromigration Signoff in the Presence of Adaptive Voltage Scaling", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2014; and T.-B. Chan, W.-T. J. Chan and A. B. Kahng, "ILP-Based Identification of Opportunistic Redundant Logic Insertions for Opportunistic Yield Improvement During Early Process Learning", *Proc. IEEE International Conference on Computer Design*, 2017.

Chapter 3 contains a reprint of W.-T. J. Chan, P.-H. Ho, A. B. Kahng and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning", *Proc. ACM International Symposium on Physical Design*, 2017.

Chapter 4 contains reprints of W.-T. J. Chan, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2016; and W.-T. J. Chan, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Integration, the VLSI Journal*, 2017.

Chapter 5 contains a reprint of W.-T. J. Chan, A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Statistical Analysis and Modeling for Error Composition in Approximate Computation Circuits", *Proc. IEEE International Conference on Computer Design*, 2013.

Chapter 6 contains reprints of A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Optimizing Stochastic Circuits for Accuracy-Energy Tradeoffs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015; and A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Trading Accuracy for Energy in Stochastic Circuit Design", *ACM Journal on Emerging Technologies in Computing Systems*, 2017.

My coauthors (Dr. Armin Alaghi, Dr. Tuck-Boon Chan, Professor John P. Hayes, Dr. Pei-Hsin Ho, Professor Andrew B. Kahng, Professor Seokhyeong Kang, Professor Rakesh Kumar, Dr. Jiajia Li,

Dr. Siddhartha Nath, Professor John Sartori and Dr. Prashant Saxena, listed in alphabetical order) have all kindly approved the inclusion of the aforementioned publications in my thesis.

VITA

1981	Born, Tainan, Taiwan
2003	B.Sc., Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan
2005	M.Sc., Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan
2015	C.Phil., Electrical Engineering (Computer Engineering), University of California, San Diego
2018	Ph.D., Electrical Engineering (Computer Engineering), University of California, San Diego

All papers co-authored with my advisor Professor Andrew B. Kahng have authors listed in alphabetical order.

- T.-B. Chan, **W.-T. J. Chan**, A. B. Kahng, "ILP-Based Identification of Opportunistic Redundant Logic Insertions for Opportunistic Yield Improvement During Early Process Learning", *Proc. IEEE International Conference on Computer Design*, 2017, pp. 269-272.

- **W.-T. J. Chan**, P.-H. Ho, A. B. Kahng and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning", *Proc. ACM International Symposium on Physical Design*, 2017, pp. 15-21.

- **W.-T. J. Chan**, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Integration, the VLSI Journal* 53 (2017), pp. 226-235.

- A. Alaghi, **W.-T. J. Chan**, J. P. Hayes, A. B. Kahng and J. Li, "Trading Accuracy for Energy in Stochastic Circuit Design", *ACM Journal on Emerging Technologies in Computing Systems* 13(3) (2017), pp. 47:1-47:30.

- **W.-T. J. Chan**, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques", *Proc. IEEE International Conference on Computer Design*, 2016, pp. 41-48.

- **W.-T. J. Chan**, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2016, pp. 6:1-6:8.

- **W.-T. J. Chan**, K. Y. Chung, A. B. Kahng, N. D. MacDonald and S. Nath, "Learning-Based Prediction of Embedded Memory Timing Failures During Initial Floorplan Design", *Proc. Asia and South Pacific Design Automation Conference*, 2016, pp. 178-185.

- A. Alaghi, **W.-T. J. Chan**, J. P. Hayes, A. B. Kahng and J. Li, "Optimizing Stochastic Circuits for Accuracy-Energy Tradeoffs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 178-185.

- **W.-T. J. Chan**, A. B. Kahng, S. Nath and K. Samadi, "3D-IC Benefit Estimation and Implementation Guidance from 2D-IC Implementation", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 30:1-30:6.

- T.-B. Chan, **W.-T. J. Chan** and A. B. Kahng, "On Aging-Aware Signoff for Circuits with Adaptive Voltage Scaling", *IEEE Transactions on Circuits and Systems I* 61(10) (2014), pp. 2920-2930.

- **W.-T. J. Chan**, A. B. Kahng, S. Nath and I. Yamamoto, "The ITRS MPU and SOC System Drivers: Calibration and Implications for Design-Based Equivalent Scaling in the Roadmap", *Proc. IEEE International Conference on Computer Design*, 2014, pp. 153-160.

- J.-A. Carballo, **W.-T. J. Chan**, P. A. Gargini, A. B. Kahng and S. Nath, "ITRS 2.0: Toward a Re-Framing of the Semiconductor Technology Roadmap", *Proc. IEEE International Conference on*

Computer Design, 2014, pp. 139-146.

- **W.-T. J. Chan**, A. B. Kahng and S. Nath, “Methodology for Electromigration Signoff in the Presence of Adaptive Voltage Scaling”, *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2014, pp. 1-7.

- **W.-T. J. Chan**, A. B. Kahng, S. Kang, R. Kumar and J. Sartori, “Statistical Analysis and Modeling for Error Composition in Approximate Computation Circuits”, *Proc. IEEE International Conference on Computer Design*, 2013, pp. 47-53.

- T.-B. Chan, **W.-T. J. Chan** and A. B. Kahng, “Impact of Adaptive Voltage Scaling on Aging-Aware Signoff”, *Proc. Design, Automation and Test in Europe*, 2013, pp. 1683-1688.

ABSTRACT OF THE DISSERTATION

Physical Design Methodologies for the More-than-Moore Era

by

Wei-Ting Chan

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2018

Professor Andrew B. Kahng, Chair

In the past decades, device scaling along the Moore's Law trajectory has been the major focus of technology innovation in the semiconductor industry. However, this scaling has in recent years slowed down due to power limits, lithography complexity, and other physics limitations. The semiconductor industry has identified several looming technology challenges and expected new design paradigms that demand new "design-based equivalent scaling" approaches to enable continuation of Moore's Law. This thesis addresses several aspects of these challenges, for both the "More-Moore" and "More-than-Moore" domains.

Interconnect reliability increases the design uncertainty in advanced node technologies. Electromigration is a growing concern in sub-22*nm* technology. To close a costly "chicken-egg" loop that spans library characterization and signoff in the presence of design adaptivity, we study the interlock among front-end (device) aging, voltage scaling, and electromigration; we furthermore quantify timing and power costs of meeting lifetime requirements. Based on this, we provide new signoff guidelines and demonstrate that suboptimal choice of voltage step size and scheduling strategy can result in decreased product lifetime.

As semiconductor technology advances, leading-edge product companies must rapidly improve yield for designs that seek to reach mass production while still early in the adoption of a new technology node. We study the possible mitigation of yield loss by opportunistic, last-stage redundant logic insertion in early advanced-node production. We describe a yield estimation methodology, and propose an integer linear programming-based optimization of redundant logic insertion for opportunistic yield optimization.

In sub-14nm processes, routability challenges arise from multiple patterning and pin access constraints that drastically weaken the correlation between global-route congestion and detailed-routing design rule violations. We present a method that applies machine learning techniques to effectively predict detailed-routing design rule violations after global routing, as well as detailed placement techniques to effectively reduce detailed-routing design rule violations.

Beyond conventional design paradigms, three-dimensional integrated circuits (3DICs) with multiple tiers are expected to achieve large benefits (e.g., in terms of power and area) as compared to conventional two-dimensional designs. However, upper bounds on the potential power and area benefits from 3DIC integration with multiple tiers are not well-explored. We use the concept of implementation with infinite dimension to estimate upper bounds on power and area benefits achievable by 3DICs versus 2DICs. We observe that design power sensitivity to implementation with different dimensions correlates

well with placement-based Rent parameter of the netlist. We show that the quality of netlist synthesis and optimization benefits from awareness of the target implementation dimension (e.g., 2D versus 3D).

Last, aggressive requirements for low power and high performance in VLSI designs have led to increased interest in non-conventional computation paradigms. Approximate and stochastic hardware can achieve improved energy efficiency compared to accurate, traditional hardware modules. To exploit any benefits of approximate and stochastic hardware modules, design tools should be able to quickly and accurately estimate the output quality of composed approximate designs. We propose new accuracy estimation methodologies for approximate hardware and stochastic hardware, respectively. For stochastic circuits, we further investigate opportunities to optimize circuits under aggressive voltage scaling. We find that logical and physical design techniques can be combined to significantly expand the already powerful accuracy-energy tradeoff possibilities of stochastic circuits.

Chapter 1

Introduction

This thesis proposes potential solutions to meet several major technology challenges inherent in continued scaling as projected by Moore's Law. The challenges are divided into two categories as shown in Figure 1.1: (i) the *"More-Moore"* challenges of continued device scaling along the Moore's-Law trajectory, and (ii) the *"More-than-Moore"* challenges of enabling integration and design paradigms which require new *"design-based equivalent scaling"* approaches to complement future device scaling. We discuss the More-Moore challenges in Chapter 2 and Chapter 3, covering reliability, yield and routability issues. Then we discuss the More-than-Moore challenges in Chapters 4 through 6, including enablements of 3DICs and of approximate and stochastic computing circuits. For each challenge, we start with new analysis and modeling methodologies for design challenges, and then propose our optimization methodology.

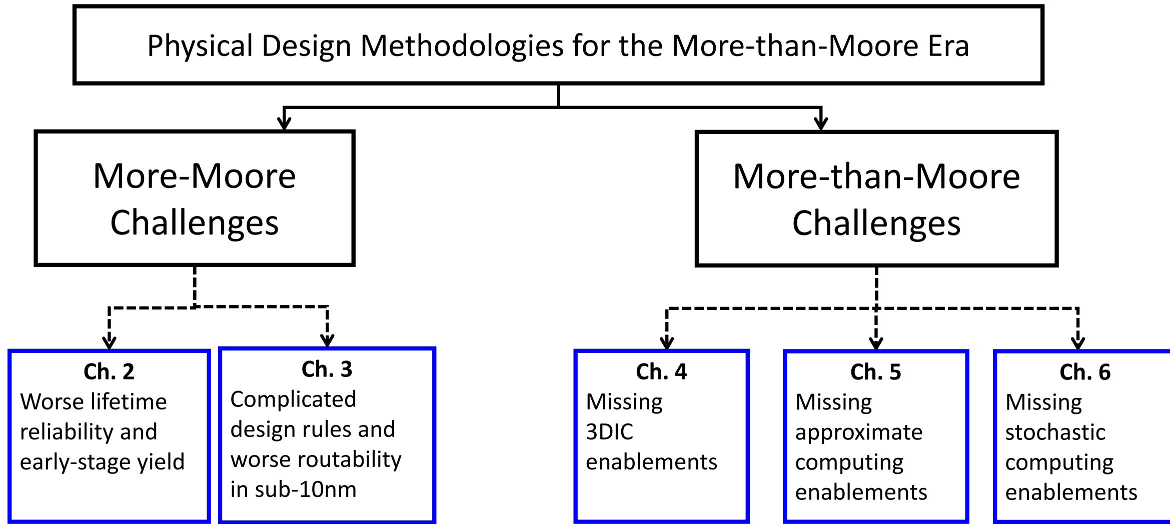


Figure 1.1: The scope of this thesis and the topic(s) covered by each chapter.

1.1 Emerging More-Moore Challenges

In the past decades, device scaling as projected by Moore's Law has been driving technology innovations in the semiconductor industry. Designers have put forward numerous application innovations with increased device integration in single chips, substantially reshaping end-users' daily lives. However, continued scaling has been challenged by power limits, lithography complexity, interconnect density, margins for reliability, design rule explosion, early-stage yield loss, etc. More-Moore challenges refer to various challenges of continued scaling. To maintain scaling, Kahng [107] proposes the concept of "design-based equivalent scaling" to leverage design technologies. Electronics design automation (EDA) technologies and tools have accordingly played an important role in response to the two challenges. Recent research has explored new technologies that potentially help the continuum of the scaling, the results of which are new scaling boosters (*buried interconnects*, *backside power delivery*, *supervias*), next device architectures (*vertical gate all around* (VGAA) FETs), or improved design-technology co-optimizations [159] [108] [109].

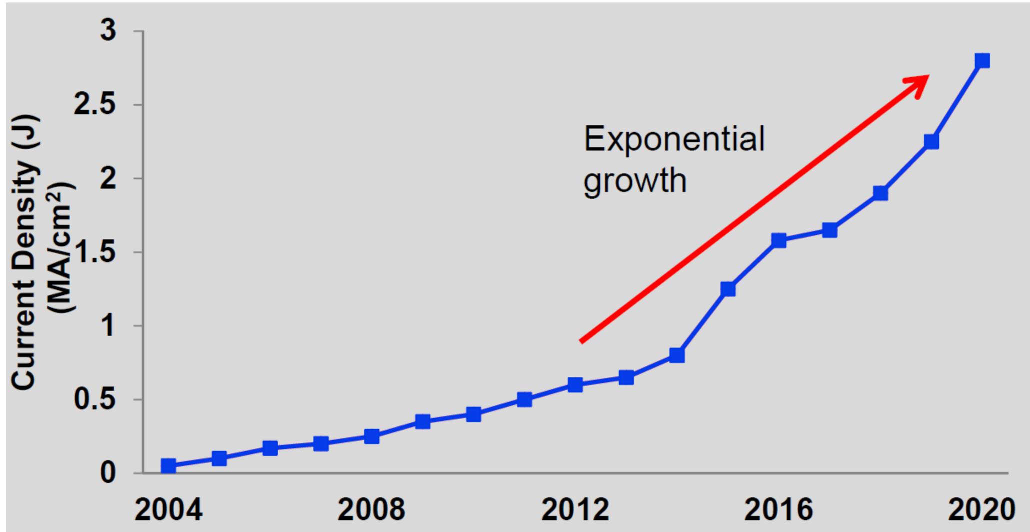


Figure 1.2: The projection of increasing current density on metal interconnect, which leads to deterioration of interconnect reliability due to electromigration [242].

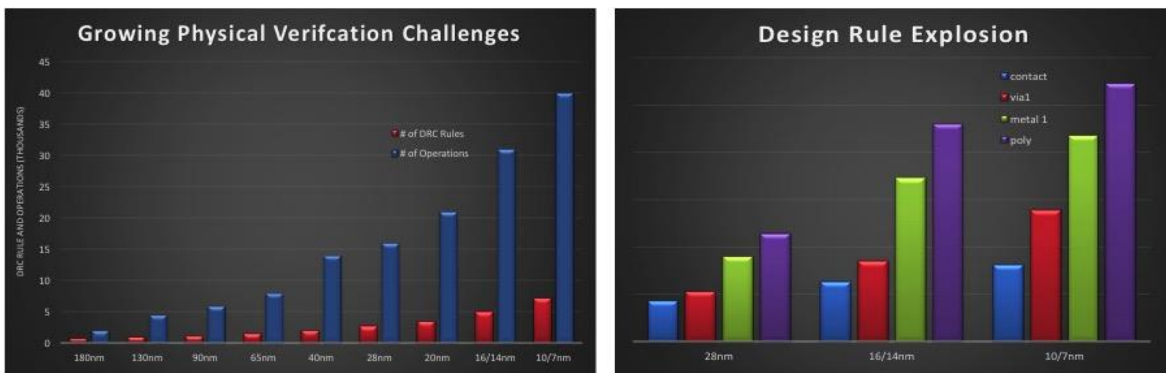


Figure 1.3: The projections of design rule numbers in advanced nodes. The rapid growth is due to complicated lithography and shrunk physical dimensions. The figure is reproduced from [234].

Among the More-Moore challenges, this thesis focuses on three main threats to scaling: (i) yield loss, (ii) excessive margins for reliability, and (iii) design rule explosion. The first part of Chapter 2 discusses the yield loss problem due to the area scaling and complication of process tuning, and we propose a strategy of redundancy logic insertion as a solution to the problem. The second part of Chapter 2 discusses the complicated relationship between interconnect reliability and signoff in the presence of *adaptive voltage scaling* (AVS). The reliability challenge has been looming in recent technology nodes. Figure 1.2 from the *International Technology Roadmap for Semiconductors* (ITRS) [242] shows that

current density on metal interconnects has been growing rapidly due to shrunk wire dimensions and relatively slow voltage scaling. The increasing current density leads to worse wire reliability, which may worsen in the presence of AVS. The need for signoff corners that consider cell aging and AVS in design flow also complicates the interactions among voltage, wire reliability, and cell aging.

Figure 1.3 points out that the explosion of layout rules is a severe challenge in advanced technology nodes, along with *system-on-chip* (SOC) and physical design complexity. Chapter 3 discusses the challenge of predicting design-rule check (DRC) violation hotspots and proposes a machine learning-based design methodology to tackle this problem.

1.2 Emerging More-Than-Moore Challenges

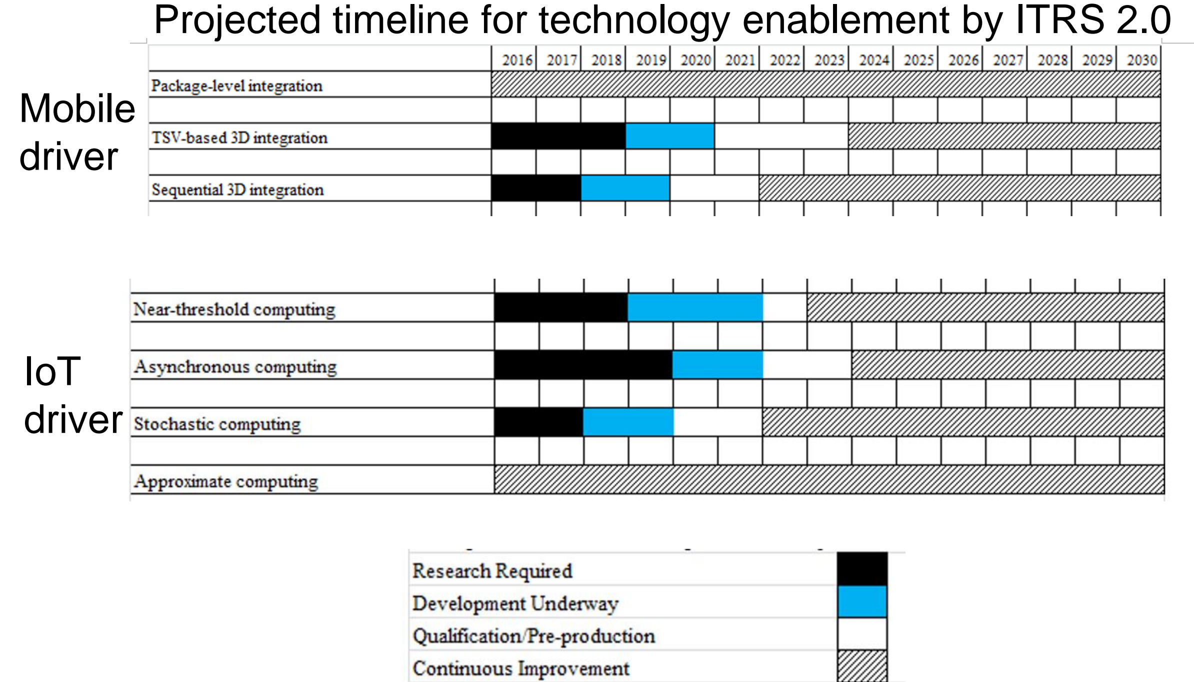


Figure 1.4: Potential technology solutions for the More-than-Moore challenges, projected by [243]. Examples of the *Mobile* driver include smartphones and tablets. Examples of the *Internet of things* (IoT) driver include low-power sensor nodes, smart home appliances, and smart power grids. The solutions are proposed to fulfill identified application requirements, such as low-cost system integration and low lifetime energy.

Recent research has also advanced new design paradigms to enable additional scaling. For example, designers now use the vertical dimension (heterogeneous multi-die integration, monolithic 3DIC) and various *"rebooting computing"* paradigms such as *quantum*, *approximate*, *stochastic*, *adiabatic*, and *neuromorphic* computing. *ITRS 2.0* [243] has outlined new technology enablements to continue the scaling. Figure 1.4 reproduced from the *System Integration chapter* of ITRS 2.0 has roadmapped timelines for these "More-than-Moore" technology enablements. The *sequential 3D integration* (i.e., monolithic 3DIC), which fabricates devices on stacked layers sequentially, is in demand by *Mobile* driver to achieve higher transistor densities in limited system form factors. The inexact (approximate and stochastic) computing technologies are also in demand to accommodate the *Internet of Things* (IoT) driver's ultra low-power requirement. However, the EDA research community is still developing new enabling technologies to maximize the benefits of leveraging these rebooting computing paradigms. Chapters 4 through 6 discuss the enablements for these rebooting computing paradigms. Chapter 4 deals with finding the upper bound of benefits from the adoption of sequential 3DIC integration and proposes a heuristic to improve 3D benefits. Chapters 5 and 6 investigate two issues to exploit inexact computing for ultra low-power design: (i) error estimation for approximate and stochastic computing, and (ii) optimization of stochastic circuits for power and accuracy in the presence of delay skews.

1.3 New Opportunities of Machine Learning to Meet Scaling Challenges

A powerful tool to take up these challenges in both the More-Moore and More-than-Moore contexts is the use of *machine learning* (ML) techniques. ML applications in EDA research span a wide range, covering function verification and physical design. The examples in the verification regime include specification mining, debugging and root cause identification, and logic structure identification [177] [67] [25]. The examples in physical design are detection of lithography hotspots [227], identification of datapath

cells in placement [220], timing convergence [87] [114], pathfinding [40] [38], and routability estimation and improvement [39] [41]. This thesis resorts to machine learning-based approaches to tackle new design challenges in physical design. The example in the More-Moore context is the use of machine learning-based classifiers to identify hotspots of DRC violations in Chapter 3. The example in the More-than-Moore context is the use of a Markov chain model, trained in advance by simulation data, to estimate errors of stochastic circuits in Chapter 6.

1.4 Organization of Chapters

The chapters of this thesis are organized as follows. Chapter 2 and Chapter 3 focus on design methodologies for the "More-Moore" context, which concern yield, reliability, and routability challenges to continue the Moore's-Law scaling. The theme from Chapter 4 to Chapter 6 is design methodologies for non-conventional design paradigms (3D and inexact computing) to accommodate technology demands in the "More-than-Moore" context.

- Chapter 2 presents mitigations of two emerging design challenges rising from device and interconnect scaling. The first challenge is yield loss during early technology adoption. While the device dimensions keep shrinking, process tuning takes a longer time before design yield can ramp up to mass production. The low yield during the adoption stage increases the risk of advancing to new technology nodes, and moreover, slows down the Moore's-Law scaling. The chapter proposes a design methodology for opportunistically adding redundant logic into design to improve yield. We first investigate a yield model that considers redundancy, and then we propose to use an iterative min-cut partition algorithm to generate candidate logic clusters for duplication. *Integer linear programming* (ILP) is employed to select optimum clusters and prevent timing violation due to

excessive insertion of multiplexers. Finally, we demonstrate yield improvement and insignificant timing impact after redundancy insertion.

The second challenge is the deterioration of reliability due to device and interconnect aging. Unlike the yield challenge during the adoption stage of a new technology, reliability issues lead to performance degradation or malfunction after manufacturing and testing. To guarantee performance throughout the whole lifetime, guardbanding for reliability is necessary in advanced nodes, and this reduces the benefit of scaling. In this thesis, we focus on two major reliability issues, which respectively impact device performance (*bias temperature instability*, BTI) and interconnect integrity (*electromigration*, EM). Conventionally, the two reliability issues are guardbanded by flat constraints of timing margin and maximum current density. However, the flat guardband approaches do not comprehend time-varying supply voltage in the presence of *adaptive voltage scaling* (AVS). Although an elevated supply voltage compensates for degraded performance, the higher voltage also hastens the aging of devices and interconnect. To find better design signoff corners that comprehend varying voltage, aging device, and degraded interconnect, we first investigate the latest BTI and EM models, apply the models to quantify the suboptimality without awareness of varying voltages, and then propose a methodology for selecting signoff corners.

- Chapter 3 presents a routability optimization work guided by a machine learning model. Routability challenges are more severe in advanced technology nodes. Standard-cell design is constrained by *multiple patterning* and the tendency toward fewer tracks in cell libraries for better area utilization. However, compact cells potentially lead to worse pin accessibility. At the design level, physical design engineers may need to increase cell spacing to resolve possible DRC violations. But the added spacing obstructs the benefit of scaling. The chapter proposes a cell spreader guided by a machine learning model. In the first step, we demonstrate that the machine learning model can efficiently

close the gap between the prediction by congestion map and the outcome of detailed routing. Then, we demonstrate that our model-guided cell spreader can improve the design routability with little design cost.

- Chapter 4 presents a design space exploration methodology for monolithic 3DIC design. 3DIC is a promising design technology to extend the Moore's-Law scaling. However, due to the lack of golden 3DIC design flows, designers are unable to efficiently evaluate the achievable benefits of implementing existing designs in 3D. We propose a concept of "infinite dimension" to evaluate 3D benefits of a given design. It is observed that the 3D benefit correlates well with the designs' Rent exponents, which indicate the complexity of interconnect. Based on this observation, we heuristically modulate designs' Rent exponents during synthesis to improve the power benefit obtained by 3D integration.

- Chapter 5 presents design methodologies for approximate computing paradigms. Approximate computing allows errors in computation in order to relax the design cost requirement. However, a lack of error estimation blocks the enablement of approximate computing. We propose a methodology for evaluating error metrics in approximate computing.

- Chapter 6 presents design methodologies for stochastic computing paradigms. To enable more aggressive energy saved by stochastic computing, we exploit the timing-error resilience of stochastic computing to achieve aggressive voltage scaling. In the literature, Najafi et al. [169] point out that the study presented in this thesis is among the first to introduce and employ skew tolerance to save energy in stochastic circuit design. We demonstrate significant energy savings without severe output quality degradation compared to conventional binary computation paradigms.

- Chapter 7 concludes the thesis with a brief look toward the future.

Chapter 2

Yield and Reliability Optimization

This chapter presents mitigations of two emerging design challenges due to device and interconnect scaling. The first challenge is the yield loss during the technology adoption. While the device dimensions keep shrinking, process tuning takes longer before design yield can ramp up to enable mass production. The low yield during the adoption stage increases the risk of advancing to new technology nodes, and essentially slows down the Moore's-Law scaling. The second challenge is the worsening of reliability issues due to device and interconnect aging. Unlike the yield challenge during the adoption stage of a new technology, reliability issues lead to performance degradation or malfunction after manufacturing and testing. To guarantee performance throughout the whole lifetime, guardbanding for reliability is necessary in advanced nodes, and this reduces the benefit of scaling.

2.1 Yield Challenge in Early Stage of Technology Adoption

Defect-limited yield is always a challenge in new technology nodes because manufacturing recipes are still being refined and processing steps are not tightly controlled. Low yield during the adoption stage of a new technology node significantly impacts product schedules. Defect-limited yield can be improved

through redundancy (by duplicating logic gates), but this approach has not been seriously considered in mature nodes, and/or when designers can achieve high area utilizations, because it incurs obvious gate area overheads. However, in advanced technology nodes, achievable placement utilization decreases due to increasing placement and routing congestions, constrained pin accessibility, and complex design rules. For example, more than 20% of a chip is whitespace in 7*nm* technology [34]. With the available whitespace, designers can potentially add redundancy to a design without increasing chip area, given a methodology for judiciously avoiding harmful perturbations of timing and congestion hotspots. Figure 2.1 qualitatively summarizes interactions among yield, amount of redundancy, and other design parameters.

The above-mentioned evolution of design consideration motivates us in this section to rethink the potential benefits of adding redundancy for yield. Redundancy insertion in very early stages of technology learning is preferable because of the greater uncertainties in tuning advanced-node processes. Any major process tuning may lead to slower progress of defect reduction. Although redundancy insertion requires (i) available whitespace for added cells, (ii) sufficient routing resource for control wires, (iii) sufficient timing margin for potential timing overhead, and (iv) additional control logics, the chip designer and product organization may still benefit from a new lever that gives some controllability of yield from the design perspective. Figure 2.2 demonstrates the layouts after adding redundancy to visualize the outcome of redundancy insertion. Our work focuses on preventing yield loss due to random defects. (Note that parametric yield is beyond the scope of our study; we assume that parametric yield loss has been addressed by proper signoff criteria.) We propose an algorithm aimed at reducing yield loss due to random defects, by exploiting multiple facets of design slack – area, timing, routing congestion – that may exist in late stages of physical implementation. In our approach, we add redundant (i.e., duplicate) logic cells to a design and then add multiplexers (MUX cells) to switch between original logic cells and redundant logic cells. Furthermore, we consider timing and congestion as we add the redundant logic and MUX cells.

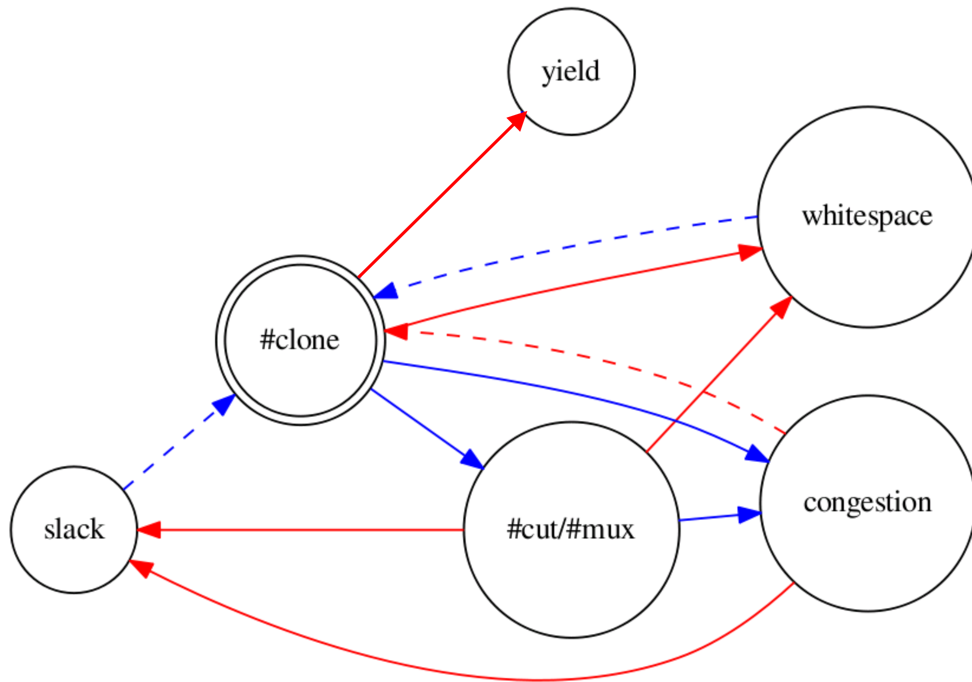


Figure 2.1: The interactions among yield, amount of redundancy, and other design parameters. Red arrows indicate positive correlation between two design parameters, and blue arrows indicate negative correlation. Red dashed arrows indicate tighter constraints to prevent addition of more redundant cells. Blue dashed arrows indicate looser constraints due to more resources.

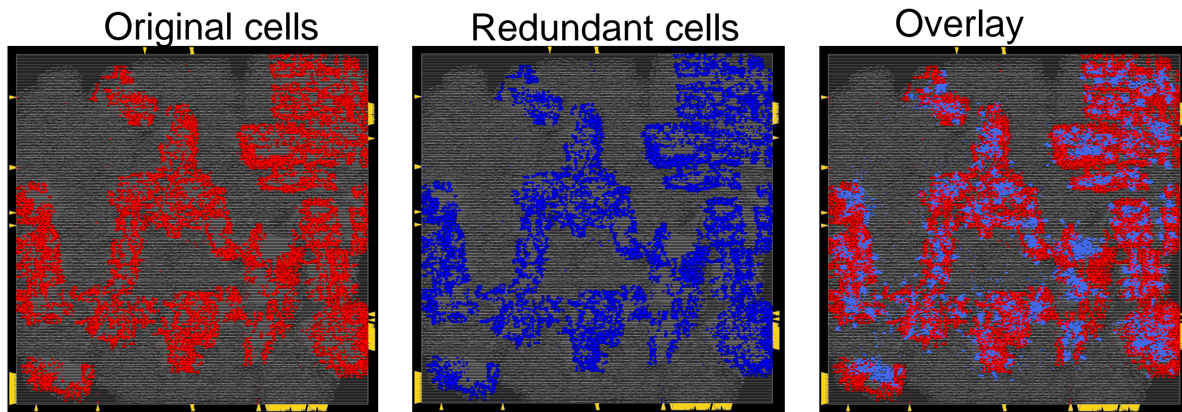


Figure 2.2: Layouts of original cells, redundant cells and overlay.

We assume that the control logic to select clusters can be realized using programmable memories. With extra testing effort, designers can reprogram the control logic to select non-defective logic. We

leave the testing strategy and controller design for future work, and focus solely on the strategy to insert redundancy.

2.1.1 Related Works

Since manufacturing yield has always been a major determinant of die cost, many previous works have investigated yield improvement from different perspectives. Nardi and Sangiovanni-Vincentelli [170] propose a synthesis framework to improve yield by reducing the number of instances of standard cells with higher *critical areas*. Wo et al. [222] consider block-level critical areas during architecture optimization. Iizuka et al. [100], and Bourai and Shi [23] focus on reducing critical areas by relaxing layouts of standard cells at the cost of higher areas.

In the arena of placement and routing (P&R), yield improvement has been achieved by a wide variety of approaches: (i) whitespace modulation during detailed placement [16]; (ii) track and layer assignment with critical-area awareness during detailed routing [135] [61] [144]; and (iii) post-route optimization to reduce critical area by widening wires [57], inserting redundant vias and wires [52] [141] [140] [27], and non-tree routing [113]. Although introduction of redundant logic is considered in the literature, previous works focus on using majority-voting to reduce *single-event upset* (SEU) of important function blocks. By contrast, our goal here is to provide *opportunistic* redundancy insertion when the utilization and timing criticality allow this.[1] To the best of our knowledge, our work is the first to present yield analysis and improvement by considering instance-level redundancy during physical implementation.

[1] *Razor* [71] is a self-repair mechanism for parametric timing error. Although it can improve parametric yield, it addresses a different yield loss mechanism and use case.

2.1.2 Problem Statement and Methodology

The amount of yield improvement depends on the amount of redundancy that can be inserted into a design. The redundancy insertion as shown in Figure 2.3 should consider identification of redundant logic at a cluster level to reduce the hardware overhead, as shown in Figure 2.3(b). The MUX cells used to select between original logic clusters and redundant clusters are critical since the MUX cells have no redundancy. On the other hand, MUX cells are larger than logic cells,[2] and their insertion therefore detracts from the potential yield benefit of logic cell redundancy. Moreover, a higher number of MUX cells will lead to more complicated connections after performing *engineering change order* (ECO) placement and routing; this may lead to routing congestion issues or loss of back-end-of-line (BEOL) yield. As shown in Figure 2.3(a), inserted MUX cells also cause timing degradation. (We do understand that additional control logics are required at the full-chip integration to select proper redundant logic after testing. Our present treatment does not include the discussion of testing strategy and controller design; we leave this for future work, and focus solely on the strategy to insert redundancy.)

[2] The largest (resp. smallest) 2-input MUX cell in the 28*nm* FDSOI foundry library that we use is roughly six (resp. three) times larger than the minimum-size NAND2 logic cell, and occupies 18 (resp. 9) placement sites.

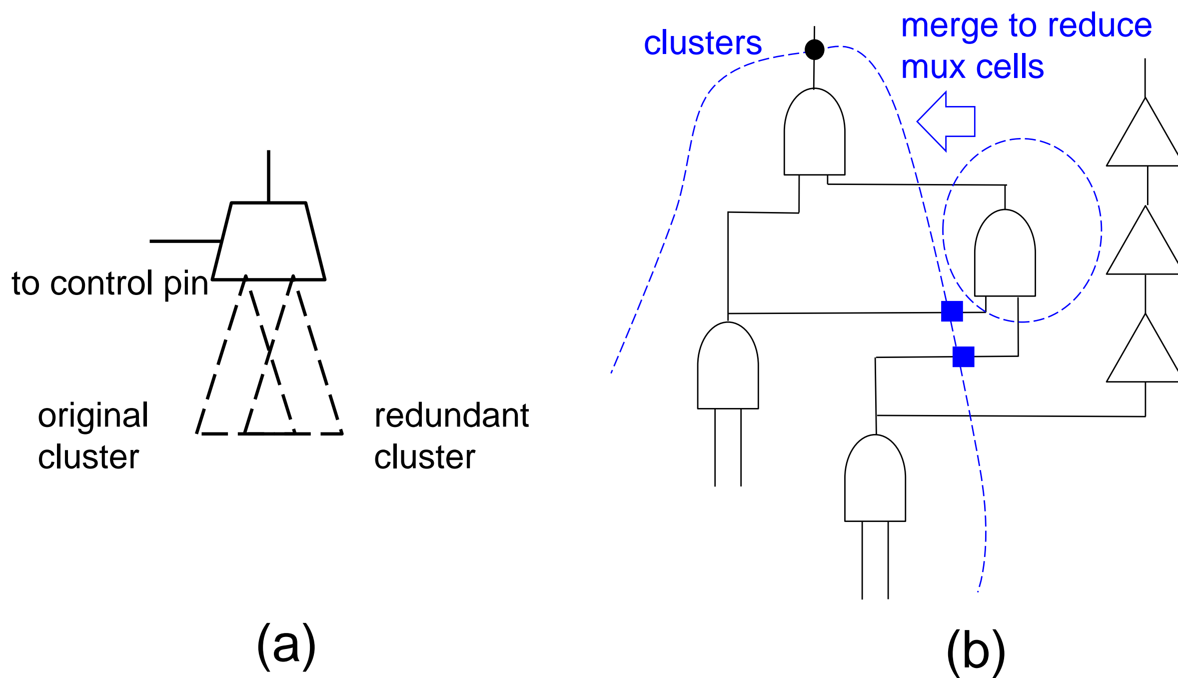


Figure 2.3: Illustration of choosing redundant logic clusters. (a) Illustration of logic clusters and a MUX cell to select between original cluster and redundant cluster. (b) Moving proper cells into a cluster can reduce the overhead of MUX cells. The blue squares indicate fanout connections that will no longer require MUX cells after merging the circled AND cell into the cluster.

We state our problem as follows. Our objective is to minimize random-defect yield loss. A problem instance consists of a routed design, with timing constraints, and a post-redundancy insertion target utilization. We seek to determine a *post-engineering change order* (post-ECO) design that maximizes redundant logic area while meeting timing and utilization constraints.

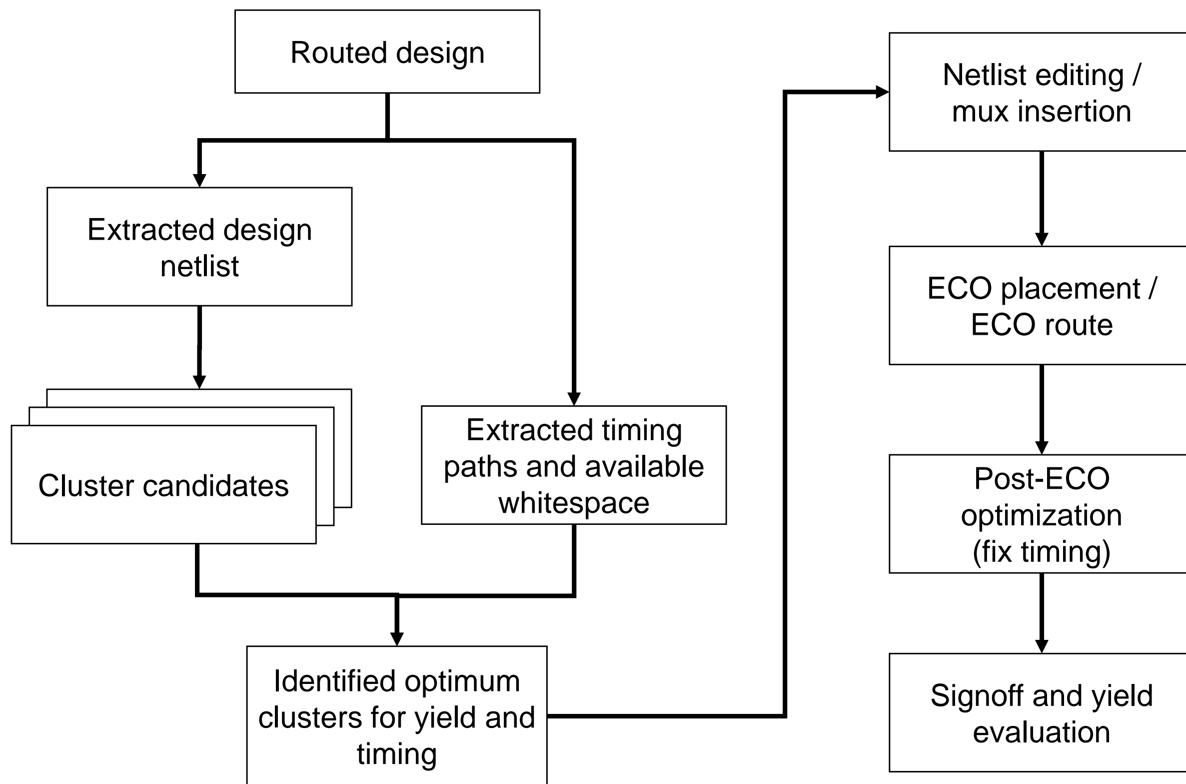


Figure 2.4: The flowchart of our proposed methodology for yield improvement by inserting redundancy.

We describe our proposed methodology in Figure 2.4. We start from the existing P&R database and extract netlist, critical paths, and available whitespace. Based on the extracted information, we generate cluster candidates and identify optimum combination of clusters. We then edit the netlist and insert the redundant logic clusters. We add MUX cells to the output nets of clusters. The P&R tool uses the edited netlist to execute ECO placement and route. We apply timing optimization and yield evaluation after the ECO flow.

Table 2.1: Description of notations used in our formulation for redundancy insertion.

Term	Meaning
T	netlist
i	grid index in yield analysis
k	cluster index
S_{min}	the minimum cell number in a cluster
F_{nand}	defect rate of a NAND2 cell
U_i	metal utilization of grid i
λ	Poisson exponent used in the yield model
$\{a,\ b\}$	fitting parameters in the yield model
Y, Y_{BEOL}, Y_{FEOL}, Y_{MUX}, Y_{redun}, $Y_{non-redun}$	design yield, BEOL yield, FEOL yield, MUX yield, yield of redundant area, and yield of non-redundant area
A_k, A, A_{redun}, A_{MUX} A_{chip}, A_{init}	area of cluster k, area of design, area of redundant logic cells, area of MUX cells, area of chip, and area of input layout
$n,\ m$	index of timing paths and timing points
N	number of extracted critical paths
M_n	number of timing points of nth path
K	number of candidate clusters
c_k	logic cluster
C_k	binary variable of selection of a cluster
D_{MUX}	MUX cell delay
SL_n	path slack of nth path
SL_{min}	minimum slack after redundancy insertion
$P_{m,n}$	binary indicator of whether m^{th} timing point of nth path has a MUX
UT	target utilization
$\|G\|$	number of cells in a design

In the remainder of Section 2.1, we first describe our yield model to evaluate the benefit of inserting redundant logic in Section 2.1.3. Section 2.1.4 then presents our heuristics to identify logic clusters to duplicate. We present experimental results in Section 2.1.5, and conclusions in Section 2.1.6. Table 2.1. summarizes the notations that we use in the remainder of this section. Below, we present more details of yield modeling in Section 2.1.3 and describe our approach to identify low-overhead clusters (for potential replication) in Section 2.1.4.

2.1.3 Yield Improvement by Exploiting Redundancy

We describe our yield model and our methodology to improve yield by adding redundant logic cells in this section. We propose a yield model that comprehends both (i) yield loss due to random defects of cells or wires, and (ii) yield recovery due to redundant cells and wires. Then we propose our yield-improving methodology by adding redundant logic cells.

Yield Impact Evaluation During Early Process Learning

For random defects, the probabilities of failure in redundant and non-redundant logic area are independent. Therefore, we can calculate design yield (of a design with redundancy) as follows.

$$\begin{aligned} Y &= Y_{non-redun} \cdot ([Y_{MUX} \cdot Y_{redun}^2] + 2 \cdot [Y_{MUX} \cdot Y_{redun}(1 - Y_{redun})]) \\ &= Y_{non-redun} \cdot Y_{MUX} \cdot Y_{redun} \cdot (2 - Y_{redun}) \qquad (2.1) \end{aligned}$$

where Y is design yield, $Y_{non-redun}$ is yield of non-redundant area, Y_{redun} is yield of redundant area, and Y_{MUX} is yield of MUX area (for logic redundancy). The first curly bracket denotes the yield while original and redundant areas have no defects, and the second curly bracket denotes the yield while exactly one of

the original and redundant clusters is defective. For a design with area A, random defect yields can be modeled by the Poisson Yield model [246]

$$Y = e^{-\lambda \cdot A} \tag{2.2}$$

where λ is a process-dependent Poisson exponent. Given a design with area A, if we duplicate A_{redun} area of the design, with MUX area A_{MUX}, the yield of the design with redundancy is given by

$$\begin{aligned}
Y &= Y_{non-redun} \cdot (Y_{MUX} \cdot Y_{redun}^2 + 2 \cdot Y_{MUX} \cdot Y_{redun}(1 - Y_{redun})) \\
&= Y_{non-redun} \cdot Y_{MUX} \cdot Y_{redun}(2 - Y_{redun}) \\
&= e^{-\lambda \cdot (A - A_{redun})} \cdot e^{-\lambda \cdot A_{MUX}} \cdot e^{-\lambda \cdot A_{redun}} \cdot (2 - e^{-\lambda \cdot A_{redun}}) \\
&= e^{-\lambda \cdot A} \cdot e^{-\lambda \cdot A_{MUX}} \cdot (2 - e^{-\lambda \cdot A_{redun}})
\end{aligned}$$

$$\text{yield gain} = e^{-\lambda \cdot A_{MUX}} \cdot (2 - e^{-\lambda \cdot A_{redun}}) \tag{2.3}$$

Equation 2.3 shows that yield improvement from redundancy is a function of redundant logic and MUX areas (and independent of A). Figure 2.5 shows yield improvements for various A_{redun} and A_{MUX} with $\lambda = 10^{-6}$.[3] Based on the figure, we can see that yield improvement increases with A_{redun}/A_{MUX} ratio and saturates for a fixed A_{redun}. This implies that achievable yield improvement is limited by A_{redun}. We

[3]The redundant area in the simulation starts from 20000 to match the realized redundant area we observed in our testcase *VGA*. See Table 2.2.

also notice that the A_{redun}/A_{MUX} ratio must be sufficiently large (e.g., larger than 2) to achieve noticeable yield improvement.

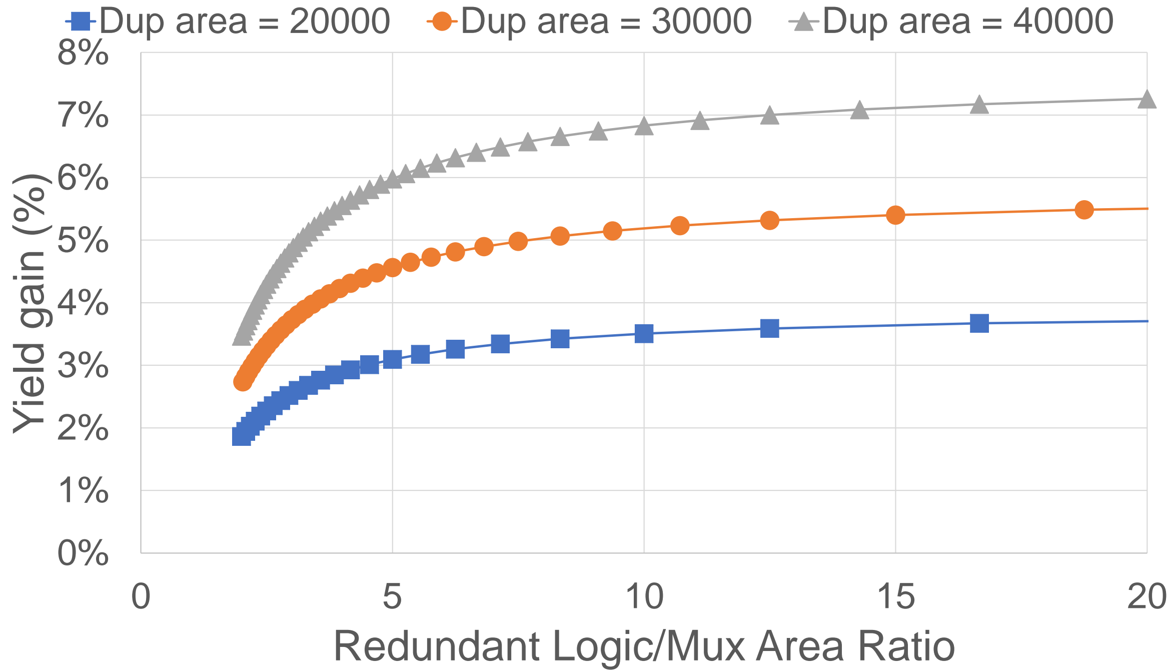


Figure 2.5: Projected yield improvement by our model. Three different redundant areas (cell area in redundant clusters) are projected in this simulation.

Yield Model

To connect the cluster-level yield calculation in Equation (2.1) with component-level (cells and nets) yield, we split the yield of a logic cluster into *back-end-of-line* (BEOL) and *front-end-of-line* (FEOL) yields. Therefore, the yield of a redundant cluster can be determined by Equation (2.5).

$$Y_{cluster} = Y_{BEOL}(nets) \cdot Y_{FEOL}(cells) \tag{2.4}$$

$$Y_{redun} = Y_{BEOL}(redun_nets) \cdot Y_{FEOL}(redun_cells) \tag{2.5}$$

We then estimate Y_{BEOL} and Y_{FEOL} in different ways. For Y_{FEOL}, we make an assumption regarding the defect rate of the minimum NAND2 cell and then extrapolate defect rates of other cells based on the area ratio, as shown in Equation (2.6).[4] The overall FEOL yield is given by the products of all the cells in the cluster, as shown in Equation (2.7).

$$\ln(Y_{cell}) = \ln((1 - F_{nand}) \cdot A_{nand2}/A_{cell}) \tag{2.6}$$

$$Y_{FEOL} = \prod_{\text{all cells}} Y_{cell} \tag{2.7}$$

For the BEOL, we interpolate the yield based on the calibration of *Mentor Calibre*. We first characterize the yield of a routed layout in a 28*nm* foundry FDSOI technology, and then use the yield data for curve fitting. We first run through the P&R flow to generate a GDSII file, then use *Mentor Calibre Yield Analyzer* [246] to estimate the BEOL yield for each 10*μm* by 10*μm* grid. The probability distribution of defect sizes is obtained from the *Calibre* reference flow. We use six metal layers in this yield characterization and other experiments. Since the pitches are the same for each layer, we use the same BEOL yield model for all the six layers.

We observe the relationship between BEOL yield and the metal utilization on each layer. We use a Poisson distribution assumption for the yield. The yield of a grid and the BEOL is given by Equation (2.8).

[4] If not stated, the NAND2 defect rate is assumed to be 1*ppm* in the rest of this section.

$$\begin{aligned} Y_{BEOL} = & \prod\nolimits_{i \in \text{all grids}} Y_i, \\ where\ Y_i = & e^{-\lambda(U_i)} \end{aligned} \tag{2.8}$$

The Poisson exponent is assumed to be a function of track utilization U_i, and the data points of exponent $\lambda(U_i)$ are given by the *Calibre* results. Based on the curve fitting in Figure 2.6, we use a = 3.91×10^{-7} and $b = 2.9 \times 10^{-9}$ in our yield evaluation.

$$\lambda(U_i) = a \cdot U_i + b, \tag{2.9}$$

where i is the grid index

We obtain the Y_{BEOL} in Equation (2.8) by decomposing the design into grids, calculating U_i of each grid, and using the fitting equation in Equation (2.9) to derive Y_i. We need per-net yield for design yield calculation in Equation (2.5) after adding redundancy. The per-net yield Y_{net} is then derived from the yield Y_{BEOL} by the following equations.

$$Y_{BEOL} = \prod_{allnets} Y_{net} \tag{2.10}$$

$$Y_{net} = e^{\frac{ln(Y_{BEOL})}{TOTAL_WL} \cdot NET_WL} \tag{2.11}$$

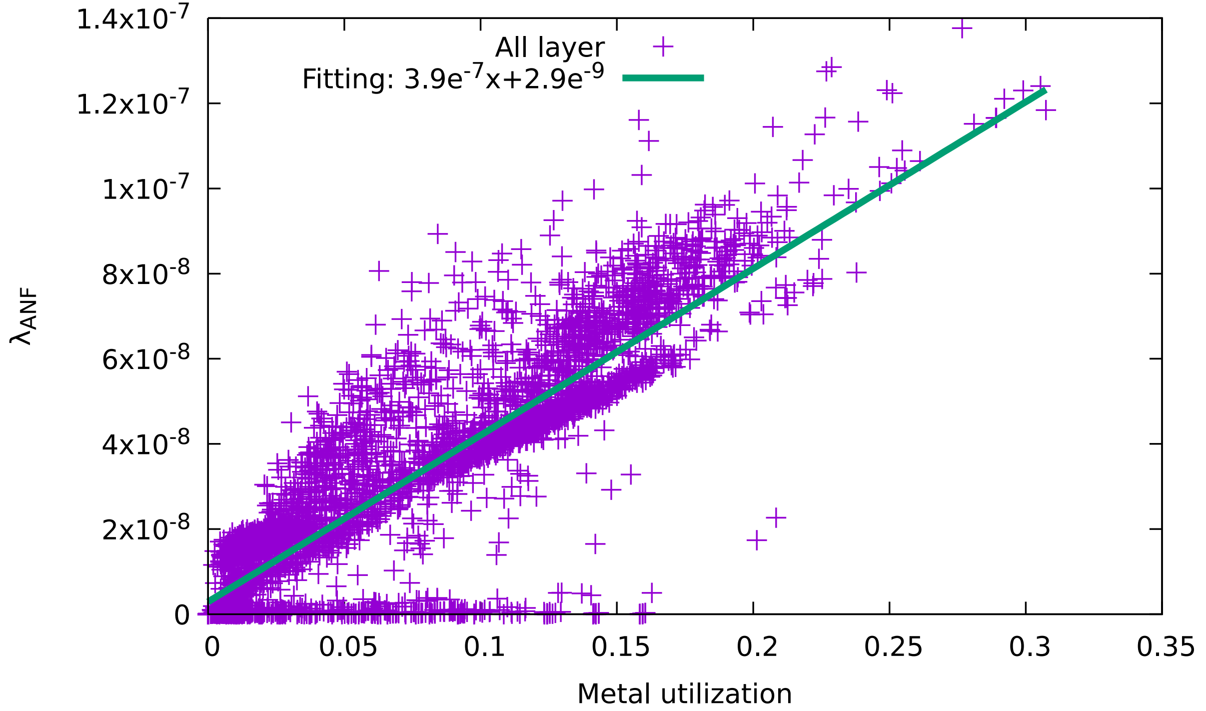


Figure 2.6: Fitting result for our yield model. The data points are from the analysis of a *JPEG* design in a 28*nm* FDSOI technology by *Calibre Yield Analyzer* [246]. We use linear fitting as an approximation to obtain BEOL yield. The details are described in Equation (2.8) to Equation (2.11).

Based on our yield model, we demonstrate how to emulate yield loss during early process learning with different Y_{nand2}. In order to demonstrate the capability to extrapolate severe yield losses of large design blocks during early process learning stage, we start the yield simulation with a *JPEG* design and expand the design size by duplicating the layout by $\{10\times,\ 15\times,\ 100\times,\ 500\times\}$. Figure 2.7 shows the results of the yield simulation.

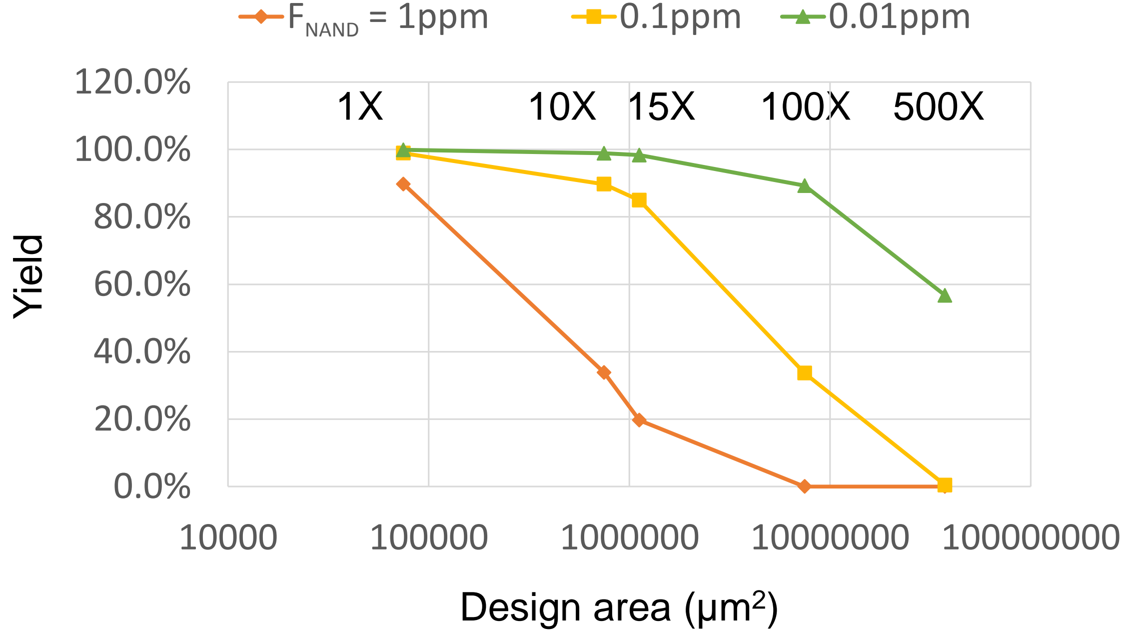


Figure 2.7: Yield simulation of a *JPEG* block. Different values of F_{nand} are used to project yield loss at the early process learning stage. We also duplicate the block multiple times to emulate the yield loss of larger designs.

2.1.4 Redundant Logic Insertion

Analysis in Section 2.1.3 shows that to maximize the yield gain, we need to maximize redundancy area with minimal MUX area overhead. Our redundant logic insertion methodology addresses this problem through two optimization steps. First, we identify high-quality candidate clusters with large redundancy to MUX area ratio. Second, we select clusters through an *integer linear programming* (ILP) solver to maximize the redundancy area. Details of these optimization steps are described in the rest of this section.

2-Way FM-Based Cluster Generation

Quality of a logic cluster is closely related to its *intrinsic* Rent parameter [84], i.e., the ratio of number terminals to cluster size. This is because intrinsic Rent parameter defines the minimum possible

Rent parameter, which is defined by the ratio of logarithm of terminals to logarithm of cluster size. (Note that number of terminals is the upper bound for MUX cell insertion, while cluster size is proportional to redundant logic area.) This means that cluster candidates with a small Rent parameter is also a good cluster for redundant logic insertion. Based on this observation, we propose to use the min-cut partitioning algorithm to generate candidate clusters. In this section, we use *MLPart* library [30], a known-good implementation of 2-way Fiduccia-Mattheyses (FM) [75] partition algorithm, to generate candidate clusters. The flow to generate the clusters are described in Algorithm 1.

Algorithm 1 Cluster generation by recursive 2-way FM-based partitioning.

Input: Netlist T, minimum cluster size S_{min}
Output: The solution of cluster candidate
$Sol = \{c_1, c_2, ..., c_K\}$

1: $Sol \leftarrow T$; $Sol' \leftarrow \emptyset$;
2: $stop \leftarrow FALSE$
3: **while** $(!stop)$ **do**
4:   **for** c_k in Sol **do**
5:     $\{c'_k, c''_k\} \leftarrow$ minCutPart(c_k)
      // tolerance of area balance = 10%
6:     **if** $(|c'_k| < S_{min}$ or $|c''_k| < S_{min})$ **then**
7:       $stop \leftarrow TRUE$
8:       break
9:     **end if**
10:     $Sol' \leftarrow Sol' \cup \{c'_k, c''_k\}$
11:   **end for**
12:   $Sol \leftarrow Sol'$
13: **end while**
14: **return** Sol

The iterative 2-way partition is implemented based on the infrastructure of *RentCon* [250]. We first read in the design DEF and construct a hypergraph in *C++*, then we iteratively call *MLPart* to generate bipartitioned clusters until the constraint of lower bound of cluster size (S_{min}) is reached (Algorithm 1). We avoid using extreme (large) S_{min} values because a large cluster is more difficult to be duplicated due to placement and routing congestions. Also, a larger cluster is more likely to degrade circuit timing during

logic redundancy insertion. On the other hand, a small S_{min} is likely to produce small clusters which are sensitive to area overheads.

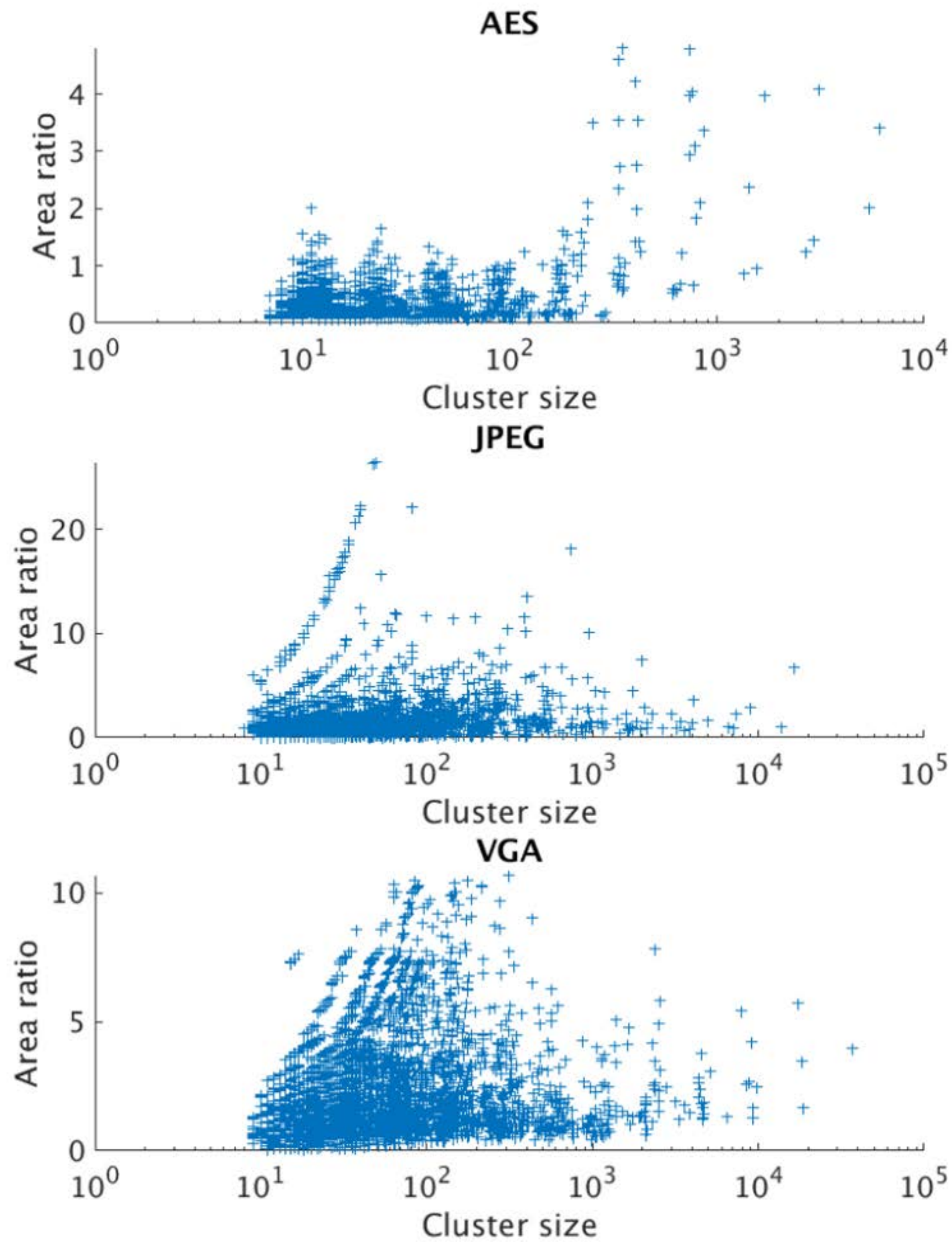


Figure 2.8: The cluster area to MUX area ratio versus cluster size of *AES*, *JPEG*, and *VGA*.

To find a suitable S_{min}, we plot all the clusters generated from the initial bipartitioned logic clusters to the clusters with more than five instances. The ratios between the number of MUX cells and logic cells of different cluster sizes are reported in Figure 2.8. Figure 2.8 shows that the clusters with high area ratios are fewer when cluster sizes are less than 200 (for *AES*). On the other hand, larger cluster sizes may limit the solution space in ILP due to utilization constraint. Therefore, we fix S_{min} to 200 in this section.

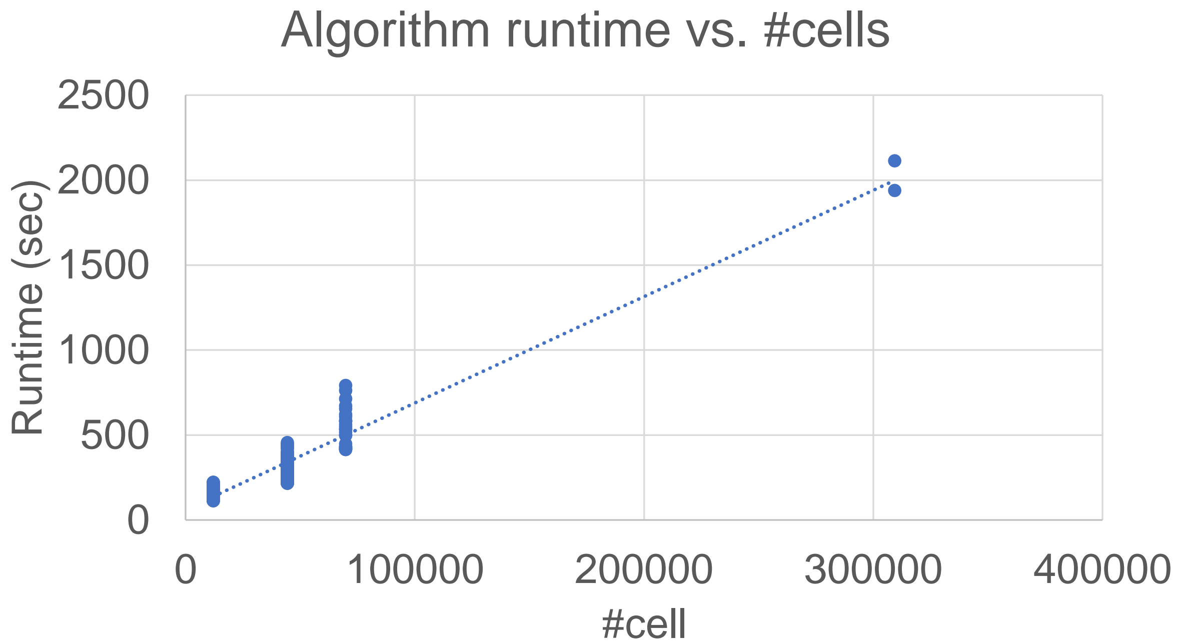


Figure 2.9: Algorithm runtime versus cell numbers in designs.

We analyze runtime complexity as follows. Let there be $|G|$ cells in the design. The FM algorithm is well-understood to have time complexity that is linear in $|G|$.[5] Ignoring the (dominated) time to write out / read in the input netlist for each given call to FM, assuming an $O(1)$ bipartitioning cost for small end-case netlists, and applying the Master Theorem for solution of divide-and-conquer recurrences, we obtain the recurrence $T(|G|) = 2 \cdot T(\frac{|G|}{2}) + \Theta(1)$ for the runtime of the recursive bipartitioning process.

[5]The time complexity of FM is linear in the number of pins in the netlist, and this is a *per-pass* time complexity. However, in practice, FM uses only a small (single-digit) number of passes, and the number of pins in the netlist is bounded by a (technology- and library-dependent) constant times the number of cells.

The complexity of the cluster generation is then $\Theta(|G|)$. (Including a $\Theta(|G|)$ term for the writing/reading of each netlist would add a logarithmic factor to the solution of the recurrence.) Figure 2.9 shows the runtime versus different design size.

ILP-based Cluster Selection

The quality of identified logic clusters during redundancy insertion has a significant impact on P&R quality after ECO placement and route. As shown in Figure 2.10, we use MUX cells to provide programmability to switch between original logic cluster and the redundant logic cluster. The inserted MUX cells not only occupy available whitespace but also increase the risk of worse post-ECO timing. To reduce the overhead, we propose our methodology to first identify low-overhead logic clusters and then select compatible clusters which do not hurt the existing critical path. The details will be described in the rest of this section.

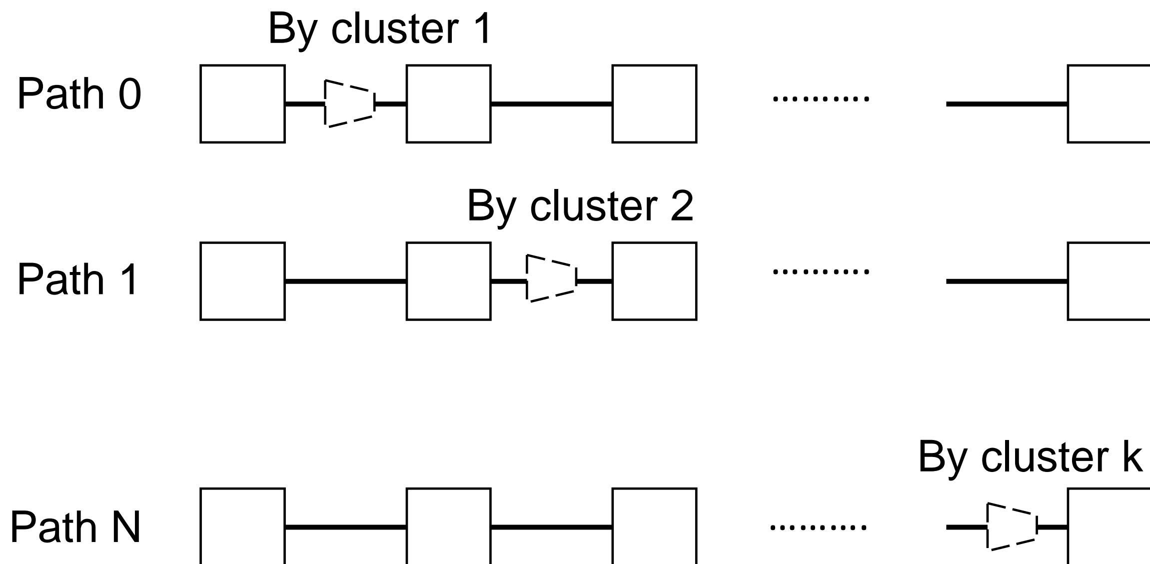


Figure 2.10: Timing path and inserted MUX cells. For each near-critical timing path, we use a corresponding constraint in ILP to constrain the delay. When any added MUXes appear in the near-critical paths, we formulate the appearance as binary variables. The details are described in the ILP formulation from Equation (2.12) to Equation (2.15).

We duplicate logic clusters and then insert MUX cells to the output nets to switch between the original logic and the insert redundant logic. In order to avoid timing impact on the design, we apply an ILP-based cluster selection flow to duplicate the maximum amount of logic under given timing constraints. The formulation uses binary variables to denote if a selected cluster will impact any timing critical path due to the MUX cells associated with the clusters in Figure 2.10. The details of our ILP formulation are described from Equation (2.12) to Equation (2.15).

Maximize

$$\sum_{k=1}^{K} A_k \cdot C_k \tag{2.12}$$

Subject to:

For $1 \leq n \leq N$,

$$SL_n - \sum_{m=1}^{M_n} P_{m,n} \cdot D_{MUX} \geq SL_{min} \tag{2.13}$$

For $1 \leq n \leq N,\ 1 \leq m \leq M_n,\ 1 \leq k \leq K$

$$P_{m,n} \geq C_k, \tag{2.14}$$

where C_k and $P_{m,n}$ are binary variables,

$$(\sum_{k=1}^{K} A_k \cdot C_k + A_{init})/A_{chip} \leq UT \tag{2.15}$$

Our objective in Equation (2.12) guides the ILP to maximize the area of redundant logic clusters. To avoid the inserted MUX cells from hurting critical timing paths as illustrated in Figure 2.10, we extract timing paths with less than 200*ps* slack from P&R tool and use Equations (2.13) and (2.14) to account

for the timing slack after inserting redundancy. To prevent the inserted redundancy from excessively using available whitespace, we use Equation (2.15) to constrain the target die utilization after redundancy insertion. We will demonstrate our results based on our proposed approaches in Section 2.1.5.

2.1.5 Experimental Setup and Results

We implement our flow with *C++* code and *Cadence Innovus Implementation System* [235] and *Synopsys Design Compiler* [251]. We solve the ILP formulation by calling *IBM ILOG CPLEX*[6] and feed our solutions to commercial tools. We use a 28*nm* foundry FDSOI technology in our experiments. The testcases *AES*, *JPEG*, *VGA*, *LEON3MP*, and *NETCARD* are obtained from *OpenCores* [247] and *ISPD Contests* [175].

[6]The runtimes of the 2-way partitioning and ILP are less than 20 minutes for *VGA* on a Xeon E5-2690 machine.

Evaluation of Generated Clusters

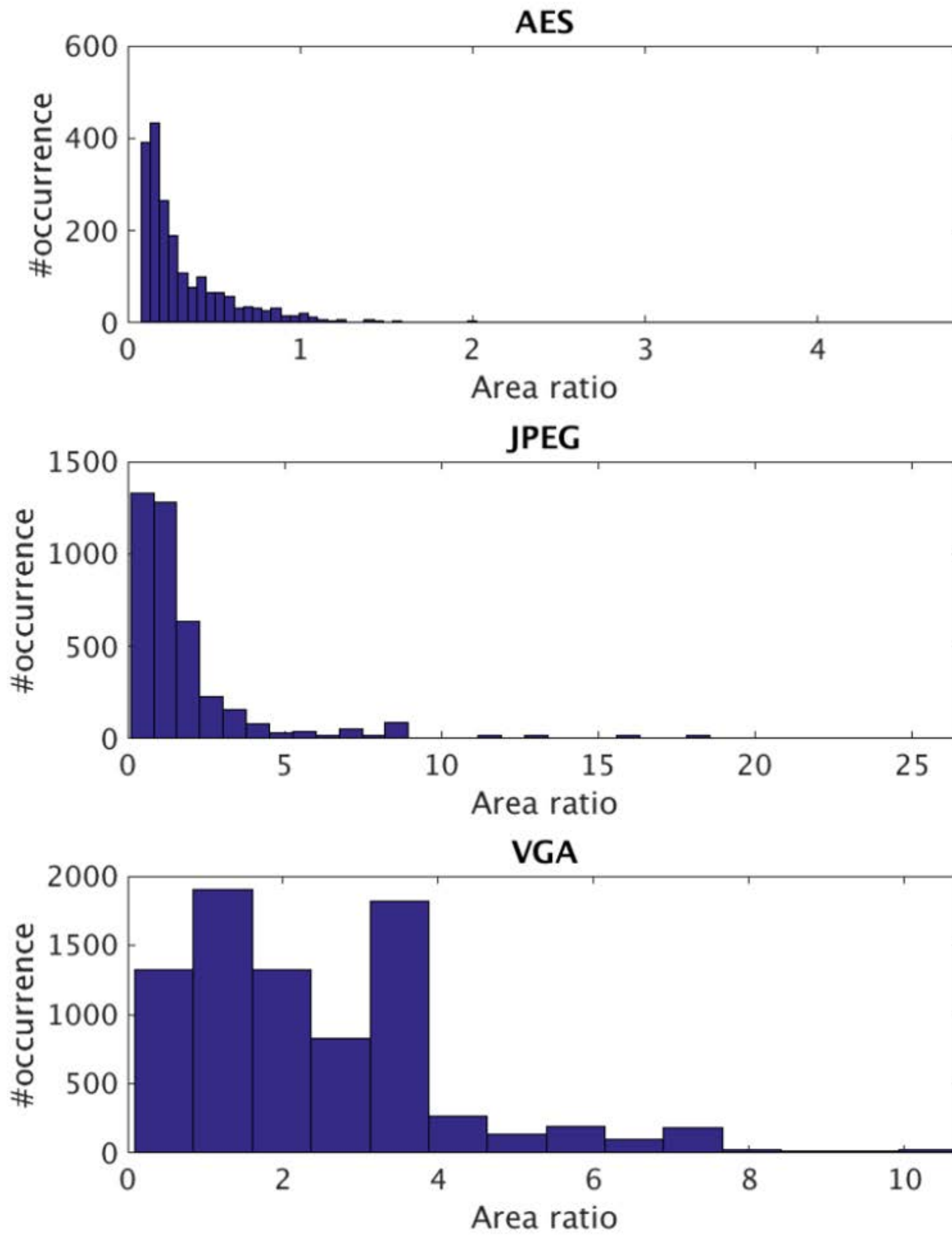


Figure 2.11: The histogram of cluster size ratios of *AES*, *JPEG*, and *VGA*.

MLPart is two-way multilevel FM. *RentCon* is top-down application of *MLPart* to identify a given netlist's *intrinsic* (partitioning-based) Rent parameter (e.g., the lowest-slope trace possible in the plot of $\log(\overline{T}_i)$ (y-axis) versus $\log(\overline{C}_i)$ (x-axis), where $\overline{C}_i$ is the size of a cluster of logic gates and $\overline{T}_i$ is the associated number of terminals (cut nets at the boundary of the cluster)). (See Figure 4 of [84], reproduced in Figure 2.12, for example.) Figure 2.11 shows the cluster to MUX area ratio across different designs.

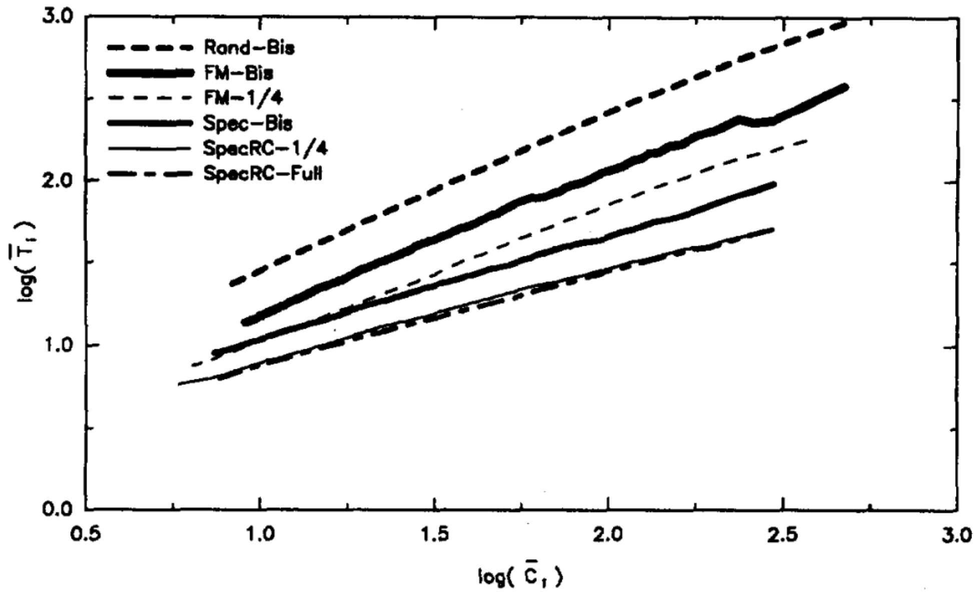


Figure 2.12: Number of terminals versus sizes of cluster of logic gates, reproduced from [84].

Yield Evaluation and P&R Metrics

The main results are presented in Table 2.2. Since the initial layouts constrained at lower frequencies and initial utilizations show better yield improvement, we report details of these data points to show the design tradeoffs under different settings. For redundant logic insertion, we target 10%, 20%, and 30% of the total core area (up to 40% for the three smaller testcases: *AES*, *JPEG* and *VGA*). Thus, for each design, we show metrics targeting 60%, 70%, and 80% utilizations (up to 90% for the smaller testcases).

The limited yield gain of *AES* is caused by the following reasons. First, achievable redundant area is a strong function of original design area. *AES* has an original logic area of $84400um^2$, which is much smaller than *LEON3MP* and *NETCARD*. Based on analysis in Figure 2.5, we do not expect any significant yield gain for such a small design. To speculatively project the yield improvement in large designs, we scale up the design footprints (except the two larger ones) so that the post-routed yield is around 50%.[7] Based on the same footprint sizes, we report extrapolated yield (Ex. Yield) in Table 2.2. We observe significant yield improvement in this speculative context.

For *LEON3MP* (*70*× size of *AES*), the yield gain is noticeably larger. For the same testcase, we observe that the yield improvement increases when more redundant logic cells are inserted. For example, the yield improvements of *LEON3MP* are 1.20×, 1.41× and 1.62× when target utilizations after redundancy insertion are 60%, 70% and 80%, respectively. We observe timing degradation right after ECO place and route, and the timing slacks are restored after optimization, which is indicated by near-zero *worst negative slack* (WNS) and *total negative slack* (TNS) in *LEON3MP*. *NETCARD* also shows high yield improvement because of larger design area, but we notice that DRC violations (DRVs) occur in *NETCARD* since its physical implementation is wire-dominated.

For all testcases, the power, leakage, and routed wirelength increase with higher amount of redundancy. We observe high power overhead in the post-ECO results, due to the logic redundancy including flip-flops. In addition, we propagate switching activity (we use an activity factor of 0.02) from output pins of flip-flops in vectorless power estimation; this makes the flip-flop power overhead more obvious. Note that the timing of *AES* is tight and hence the ILP is infeasible at high clock frequency. We use "–" in those rows.

[7] The footprint area increase by the ratio X between log(post-routed yield) and $log(50\%)$. Then we scale the extrapolated yield by equation $Y_{ex} = Y^X$.

Table 2.2: Yield improvement and P&R qualities of (i) original routed layouts (routed), (ii) post-ECO layouts with redundant logic (ECO), and (iii) post-optimization layouts (opt). WNS denotes *worst negative timing slack* and TNS denotes *total negative timing slack*. (For the two larger testcases *LEON3MP* and *NETCARD*, we do not present the extrapolated yield because their yields at routed stage are lower than 50%.)

Design	Clock (ns)	Init util	Target util		#Inst	Area (μm^2)	Util	Yield	WNS (ps)	TNS (ns)	Power (mW)	Leak (mW)	WL (μm)	#DRV	#MUX	#dup	MUX area	dup area	Ex. yield
JPEG	2.0	50%	60%	routed	30110	42129	50.3%	91.8%	24	0.001	14.9	1.01	406459	0	0	0	0	0	49.9%
				ECO	1.16×	1.17×	1.17×	1.01×	-116	-2.251	1.23×	1.50×	1.24×	0	188	4526	552	6608	1.84×
				opt	1.16×	1.17×	1.17×	1.01×	1	0.001	1.22×	1.50×	1.24×	0	188	4526	552	6608	1.84×
JPEG	2.0	50%	70%	routed	30110	42129	50.3%	91.8%	24	0.001	14.9	1.01	406459	0	0	0	0	0	49.9%
				ECO	1.28×	1.32×	1.31×	1.02×	-300	-15.751	1.47×	2.15×	1.43×	0	412	8118	1210	12089	1.85×
				opt	1.28×	1.32×	1.32×	1.02×	3	0.001	1.47×	2.16×	1.43×	0	412	8118	1210	12089	1.85×
JPEG	2.0	50%	80%	routed	30110	42129	50.3%	91.8%	24	0.001	14.9	1.01	406459	0	0	0	0	0	49.9%
				ECO	1.39×	1.44×	1.44×	1.03×	-395	-59.621	1.72×	2.87×	1.66×	0	671	11086	1971	16439	1.84×
				opt	1.40×	1.44×	1.44×	1.03×	-1	0.001	1.73×	2.92×	1.66×	0	671	11086	1971	16439	1.84×
JPEG	2.0	50%	90%	routed	30110	42129	50.3%	91.8%	24	0.001	14.9	1.01	406459	0	0	0	0	0	49.9%
				ECO	1.43×	1.48×	1.48×	1.03×	-410	-63.941	1.83×	3.11×	1.73×	0	749	12091	2200	18098	1.83×
				opt	1.43×	1.49×	1.49×	1.03×	-2	0.001	1.83×	3.18×	1.74×	0	749	12091	2203	18098	1.83×
JPEG	0.8	50%	90%	routed	43478	52276	54.3%	89.9%	-2	-0.011	46.3	6.96	466599	0	0	0	0	0	42.3%
				ECO	1.06×	1.08×	1.08×	1.01×	-307	-7.131	1.13×	1.08×	1.12×	0	108	2711	317	3654	2.13×
				opt	1.07×	1.08×	1.08×	1.01×	-19	-0.211	1.13×	1.10×	1.12×	0	108	2711	317	3705	2.13×
JPEG	0.8	80%	90%	routed	43544	51522	85.6%	90.0%	-3	-0.011	44.4	6.13	374089	4	0	0	0	0	42.8%
				ECO	1.03×	1.03×	1.03×	1.00×	-335	-25.401	1.06×	1.04×	1.10×	4	41	1320	120	1555	2.11×
				opt	1.03×	1.03×	1.03×	1.00×	-99	-1.011	1.08×	1.07×	1.11×	4	41	1320	120	1556	2.11×
AES	2.0	50%	60%	routed	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0	50.0%
				ECO	1.23×	1.19×	1.19×	1.00×	-504	-12.941	1.12×	1.83×	1.44×	103	55	2559	161	1422	1.94×
				opt	1.23×	1.19×	1.19×	1.00×	0	0.001	1.12×	1.94×	1.44×	93	55	2559	161	1430	1.94×
AES	2.0	50%	70%	routed	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0	50.0%
				ECO	1.42×	1.36×	1.36×	1.00×	-549	-17.111	1.24×	2.83×	1.74×	102	125	4743	367	2685	1.91×
				opt	1.42×	1.36×	1.37×	1.00×	-8	-0.011	1.29×	2.89×	1.74×	67	125	4743	367	2692	1.91×
AES	2.0	50%	80%	routed	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0	50.0%
				ECO	1.57×	1.50×	1.50×	1.01×	-990	-66.251	1.47×	4.00×	2.01×	171	221	6280	649	3608	1.87×
				opt	1.57×	1.51×	1.51×	1.01×	-21	-0.051	1.47×	4.22×	2.01×	183	221	6280	649	3616	1.87×
AES	2.0	50%	90%	routed	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0	50.0%
				ECO	1.67×	1.66×	1.67×	1.01×	-1538	-134.501	1.82×	5.94×	2.30×	1000	380	7360	1116	4494	1.87×
				opt	1.68×	1.67×	1.67×	1.01×	3	0.001	1.88×	6.56×	2.30×	1000	380	7360	1117	4504	1.87×
AES	0.8	50%	90%	routed	11942	10242	60.8%	97.9%	-5	-0.011	6.1	1.74	146107	65	0	0	0	0	50.0%
				ECO	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
				opt	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AES	0.8	80%	90%	routed	11719	9538	90.5%	98.1%	-17	-0.161	5.8	1.5	120016	77	0	0	0	0	43.2%
				ECO	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
				opt	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 2.2 continued: Results of *VGA*, *LEON3MP* and *NETCARD*.

Design	Clock (ns)	Init util	Target util		#Inst	Area (μm^2)	Util	Yield	WNS (ps)	TNS (ns)	Power (mW)	Leak (mW)	WL (μm)	#DRV	#MUX	#dup	MUX area	dup area	Ex. yield
VGA	2.0	50%	60%	routed	68924	96732	50.2%	82.1%	7	0.001	44.7	2.66	900535	29	0	0	0	0	45.7%
				ECO	1.16×	1.17×	1.17×	1.03×	-619	-517.711	1.17×	1.36×	1.22×	29	360	10457	1057	14906	0.00×
				opt	1.16×	1.17×	1.17×	1.03×	-4	-0.011	1.17×	1.36×	1.22×	29	360	10457	1057	14907	0.00×
VGA	2.0	50%	70%	routed	68924	96732	50.2%	82.1%	7	0.001	44.7	2.66	900535	29	0	0	0	0	50.2%
				ECO	1.29×	1.31×	1.31×	1.05×	-744	-1568.201	1.35×	1.81×	1.43×	33	801	19174	2353	27368	1.67×
				opt	1.29×	1.31×	1.31×	1.05×	-12	-0.021	1.35×	1.85×	1.43×	35	801	19174	2358	27369	1.67×
VGA	2.0	50%	80%	routed	68924	96732	50.2%	82.1%	7	0.001	44.7	2.66	900535	29	0	0	0	0	50.2%
				ECO	1.41×	1.44×	1.43×	1.07×	-796	-4050.701	1.53×	2.26×	1.64×	31	1208	27031	3549	38606	1.70×
				opt	1.42×	1.44×	1.44×	1.07×	-28	-0.111	1.54×	2.32×	1.65×	31	1208	27031	3559	38607	1.70×
VGA	2.0	50%	90%	routed	68924	96732	50.2%	82.1%	7	0.001	44.7	2.66	900535	29	0	0	0	0	50.2%
				ECO	1.52×	1.55×	1.55×	1.09×	-833	-4917.701	1.70×	2.70×	1.88×	33	1666	34103	4894	48685	1.72×
				opt	1.53×	1.56×	1.56×	1.09×	-44	-0.201	1.70×	2.79×	1.89×	33	1666	34103	4904	48685	1.72×
VGA	0.8	50%	90%	routed	69750	101968	52.8%	81.2%	0	0.001	121.1	6.48	914560	15	0	0	0	0	50.2%
				ECO	1.12×	1.13×	1.13×	1.02×	-200	-216.491	1.16×	1.24×	1.20×	15	433	7815	1272	11551	1.74×
				opt	1.13×	1.13×	1.13×	1.02×	-68	-1.041	1.18×	1.31×	1.20×	15	433	7815	1281	11558	1.74×
VGA	0.8	80%	90%	routed	69476	99955	82.8%	81.5%	-8	-0.011	118.2	5.06	754471	59	0	0	0	0	49.1%
				ECO	1.06×	1.07×	1.07×	1.01×	-479	-249.731	1.09×	1.14×	1.24×	57	184	4254	540	6161	1.67×
				opt	1.09×	1.09×	1.09×	1.01×	-175	-22.721	1.13×	1.32×	1.29×	63	184	4254	542	6163	1.67×
LEON3MP	4	50%	60%	routed	442613	621758	50.4%	28.1%	-13	-0.042	147.9	17.63	9562023	2	0	0	0	0	–
				ECO	1.17×	1.18×	1.18×	1.21×	-1213	-10854.600	1.20×	1.46×	1.21×	0	3456	71694	10153	102006	–
				opt	1.17×	1.18×	1.18×	1.20×	-18	-0.070	1.20×	1.47×	1.21×	1	3456	71694	10153	102029	–
LEON3MP	4	50%	70%	routed	442613	621758	50.4%	28.1%	-13	-0.042	147.9	17.63	9562023	2	0	0	0	0	–
				ECO	1.33×	1.35×	1.35×	1.41×	-1585	-30674.900	1.42×	2.09×	1.42×	7	8350	136489	24532	194484	–
				opt	1.33×	1.36×	1.36×	1.41×	-26	-0.082	1.43×	2.14×	1.42×	7	8350	136489	24533	194523	–
LEON3MP	4	50%	80%	routed	442613	621758	50.4%	28.1%	-13	-0.042	147.9	17.63	9562023	2	0	0	0	0	–
				ECO	1.47×	1.51×	1.51×	1.63×	-1694	-48297.300	1.67×	2.77×	1.66×	0	13324	196155	39145	279319	–
				opt	1.48×	1.52×	1.52×	1.62×	-31	-0.091	1.68×	2.87×	1.66×	1	13324	196155	39145	279361	–
NETCARD	4	50%	60%	routed	303000	399021	50.2%	44.1%	39	0.000	98.1	12.84	12381761	90	0	0	0	0	–
				ECO	1.15×	1.17×	1.17×	1.09×	-2276	-755.229	1.24×	1.67×	1.19×	1000	3870	41472	11370	54595	–
				opt	1.15×	1.17×	1.17×	1.09×	-22	-0.038	1.24×	1.71×	1.19×	1000	3870	41472	11377	54725	–
NETCARD	4	50%	70%	routed	303000	399021	50.2%	44.1%	39	0.000	98.1	12.84	12381761	90	0	0	0	0	–
				ECO	1.29×	1.33×	1.33×	1.18×	-3468	-5453.600	1.49×	2.46×	1.38×	1000	8546	79095	25108	104952	–
				opt	1.29×	1.33×	1.33×	1.17×	-234	-1.311	1.51×	2.61×	1.38×	1000	8546	79095	25151	105227	–
NETCARD	4	50%	80%	routed	303000	399021	50.2%	39.1%	39	0.000	98.1	12.84	12381761	90	0	0	0	0	–
				ECO	1.42×	1.48×	1.48×	1.43×	-3506	-18876.000	1.76×	3.33×	1.56×	1000	13709	114146	40277	152329	–
				opt	1.43×	1.50×	1.50×	1.41×	-1590	-30.580	1.81×	3.68×	1.57×	1000	13709	114146	40436	152620	–

2.1.6 Conclusions

Yield is now a dominant challenge in a new technology node, and yield loss during early learning stages of a new process can make leading-edge product chips economically unviable. To mitigate yield loss due to random defects, we propose a redundant logic insertion methodology that copies clusters of logic cells and connects their outputs (i.e., fanouts) to original nets through MUX cells. Based on a

Poisson yield model, we derive *yield gain* as a function of redundancy cells and MUX areas. We show that maximum achievable yield gain is determined by redundant cell area.

Our methodology optimizes redundant logic insertion through two optimization steps. First, we extract candidate clusters with minimum cuts through a recursive bipartitioning algorithm. Second, we maximize the area of redundant logic by selecting the best clusters via solution of an integer-linear program. Experimental results on our benchmark circuits show that for large design areas, logic redundancy can improve defect-limited yield by up to 1.62× from an initial value of 28.1%. Such a yield improvement could be highly significant especially for products in a new technology node, where profit margins are large due to the lack of competition.

Although our study focuses on defect-limited yield, the concept of opportunistic redundant logic insertion, as well as our methodology, can be applied toward other purposes such as (i) improvement of product lifetime against aging through redundancy and (ii) mitigation of impact of random soft defects on chip performance. Our ongoing and future works include early stage routability consideration, timing recovery, and exploration of algorithms to improve the quality of clusters for purposes of redundancy insertion for yield gain. For example, the use of top-down multilevel FM bipartitioning might be adapted to incorporate elements of classic "replication cut" approaches [99] [130]. Another potential direction is to select optimum clusters from a richer set that is derived using multiple runs of partitioning. Intelligent heuristics to choose the minimum cluster size for different designs may also improve the current approach.

2.2 Reliability Challenge in Advanced Nodes

Reliability signoff is critical in modern sub-22*nm* technology nodes to guarantee that an IC product operates at a minimum acceptable performance throughout its lifetime [115]. Recently, the most important reliability mechanisms for IC design teams are *bias temperature instability* (BTI) and *electromigration* (EM) [115] [78] [19]. BTI degrades chip performance by slowing down device switching speeds in critical paths. To meet performance requirements, design teams overcome performance degradation due to BTI by *applying adaptive voltage* scaling (AVS) [37]. EM can increase wire resistance, which can cause voltage drop resulting in device slowdown; it can also cause permanent failures in circuits due to shorts or opens. To meet lifetime requirements, design teams overcome *mean time to failure* (MTTF) with respect to EM-induced interconnect voids and shorts by applying design guardbands [148] [116] [174] [164]. Sometimes, design teams can try to make interconnects 'immortal' by limiting segment lengths to be less than or equal to the *Blech length* [21]. However, recent analysis shows that immortality is not guaranteed under all operating conditions [164]. The authors of [164] demonstrate that under long-time electrical stress, all power and ground interconnects suffer from EM degradation which results in larger wire resistances and can cause supply voltage drop (IR drop). IR drop can result in delay degradation and timing failures.

Even as modern designs use AVS to compensate for the delay degradation, the higher supply voltages from AVS can accelerate EM failures [115]. Hence, design teams are forced to use larger margins to guardband against delay degradation, that is, sign off at either a lesser lifetime or lesser performance. This dilemma induces a "chicken-and-egg" loop in the signoff flow as shown in Figure 2.13. To our best understanding, the interaction between BTI-induced AVS and EM in the context of this chicken-and-egg loop has not been studied in previous works. This motivates us to study EM-aware signoff for systems that use AVS.

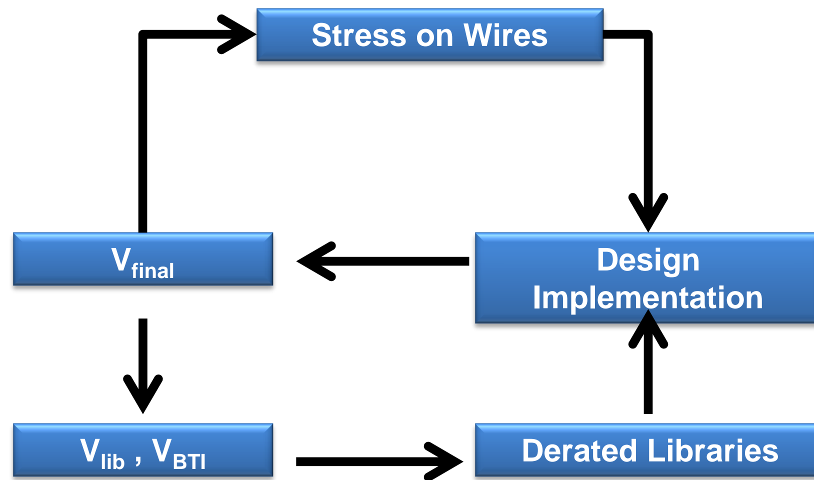


Figure 2.13: Illustration of the chicken-and-egg loop for circuit signoff with EM and AVS to compensate for performance degradation due to BTI.

In Section 2.2, we study the effects of EM on both signal and power/ground interconnects using recent models [164]. We quantify impact on circuit performance, power and lifetime in a system that uses AVS. We empirically demonstrate that EM-awareness can change power and area tradeoffs when designers choose corners for AVS signoff. Furthermore, we demonstrate that different AVS strategies have different impacts on EM lifetimes. We conduct our experiments with a 28*nm* foundry *fully depleted silicon on insulator* (FDSOI) technology and commercial tool flows.

Our contributions are summarized as follows.

1. We analyze interactions between EM degradation, and AVS signoff and scheduling, in a 28*nm* foundry FDSOI technology using commercial EDA tool flows. We address the three-way interactions between EM, BTI and AVS, and demonstrate that these interactions are significant at 28*nm* and below.

2. We study the incremental design cost to account for EM during signoff. We perform our signoff analyses in two ways: (i) modeling AVS and considering EM worsened by escalated voltages; and (ii) ignoring escalated voltages assigned by AVS.

3. We demonstrate that the inaccuracy (penalty) of EM lifetime due to improper guardband against BTI at signoff can be as high as 30% in a 28*nm* technology. Furthermore, we demonstrate that choice of AVS voltage step size and scheduling strategies can result in up to 1.5 years of decreased EM lifetime.

4. We provide signoff guidelines which suggest that the EM lifetime penalty can be compensated by paying an area penalty of up to 1.6% and a power penalty of up to 6%.

The remainder of Section 2.2 is organized as follows. In Section 2.2.1, we review related works on AVS and EM signoff. In Section 2.2.2, we study implications of a recent statistical EM model in AVS systems, and in Section 2.2.3, we describe our design of experiments and present results. In Section 2.2.4, we describe future works and conclude the section.

2.2.1 Related Works

Several previous works address different aspects of EM. We taxonomize these works into three categories.

(a) Models for lifetime and wire degradation due to EM. Black [20] proposes an empirical mean time to failure (MTTF) model of wires, which is the well-known Black's Equation. Arnaud et al. [14] and Federspiel et al. [73] study the failure processes in copper wires and extend Black's Equation. While previous works focused on developing physical models, Mishra et al. [164] propose a statistical model of the process of EM degradation. They demonstrate that their modeling approach can reduce the pessimism in Black's Equation-based EM signoff criteria.

(b) Approaches for EM evaluation and signoff. EM evaluations and optimizations in EDA tools rely on CPU-intensive computations and layout optimizations to constrain current densities that satisfy EM lifetime requirements with design tools, such as *Apache Redhawk* [233], *Cadence Virtuoso* [238], or

Synopsys PowerRail [254]. As technology advances, more physical design factors such as temperature [56], process variation [174], and growing dominance of BTI and AVS must be considered in order to achieve accurate EM evaluation and signoff.

(c) Physical design approaches for EM-durable circuits. In this category, we include wire-sizing and wire segmentation methods for the design of EM-durable circuits. Adler et al. [1] [2], Jiang et al. [102], Lienig et al. [148] and Yan et al. [225] develop wire-sizing algorithms for current-aware routers that avoid high current densities in circuits. They guarantee lifetime by ensuring that their algorithms always deliver current densities that are within the limits specified in technology files. Li et al. [145] develop wire segmentation and via insertion algorithms to create immortal wires by constraining interconnect segment lengths to be less than the Blech length [145].

Previous works on BTI in AVS systems [17] [55] [133] [134] [139] [162] [200] focus on the interactions between behaviors of AVS and BTI while ignoring EM degradation. Recently, Chan et al. [37] study signoff strategies that consider interactions between AVS and BTI, but do not consider EM in their work.

Our work falls at the intersection of categories (b) and (c), building on that of [37]. We consider EM signoff strategies in the presence of BTI and AVS, and we investigate approaches to the design of EM-durable circuits. To the best of our knowledge, we are the first to address the three-way interactions between EM, BTI and AVS, and to demonstrate that these interactions are significant at 28*nm* and below.

2.2.2 Methodology for Reliability Analysis

We study the implications of a recent statistical EM model [164] on system performance. The model is a function of wire geometry, electric field due to the supply voltage V_{dd}, and ambient temperature, and estimates degradation due to increased resistance on wires. We first verify correctness of the model

in a 28*nm* foundry FDSOI technology and then quantify the delay impact due to EM on AVS systems. In our evaluations in Section 2.2.3, we use both the statistical as well as the traditional lifetime model using Black's Equation [20].

Characterization of EM Model

In previous works, such as [14] and [73], circuit measurements and empirical models demonstrate that the failure of wires due to EM has two stages. In the first stage (nucleation), height of void is less than the thickness of copper layer, so the wire resistance does not change significantly. Once the void occupies the whole cross-section of the wire, the degradation enters the second stage (growth) and the resistance of the wire suddenly jumps since the liner layer (tantalum, which has relatively higher resistance than copper), below the copper layer becomes the only conducting path in the void region. The jump in resistance occurs when the void height reaches the thickness of the copper layer. After this, the resistance increases linearly and finally causes defects in the wire. We model the time to reach the growth stage and the corresponding void lengths as random variables. We then apply the model in [164] to evaluate the impact of EM on circuit performance.

We use similar EM parameters as in [164] and apply the model to a 28*nm* technology BEOL process with $W = 100nm$ ($2\times$ minimum width to avoid pessimism) and $L = 50\mu m$. Table 2.3 summarizes all the parameters that we use, as confirmed with [163]. The simulation results of ΔR due to EM are shown in Figure 2.14. When the current density is low ($0.01MA/cm^2$), then the wire is immortal because the process of nucleation is too slow to form voids in copper wires. When the current density is high ($0.1MA/cm^2$), the process of nucleation is fast, and enters the growth stage where the resistivity increases rapidly and results in significant delay degradation. When the current density is in between the low and high values ($0.05MA/cm^2$), nucleation starts later than, but degrades at a similar rate as, when the current density is high ($0.1MA/cm^2$).

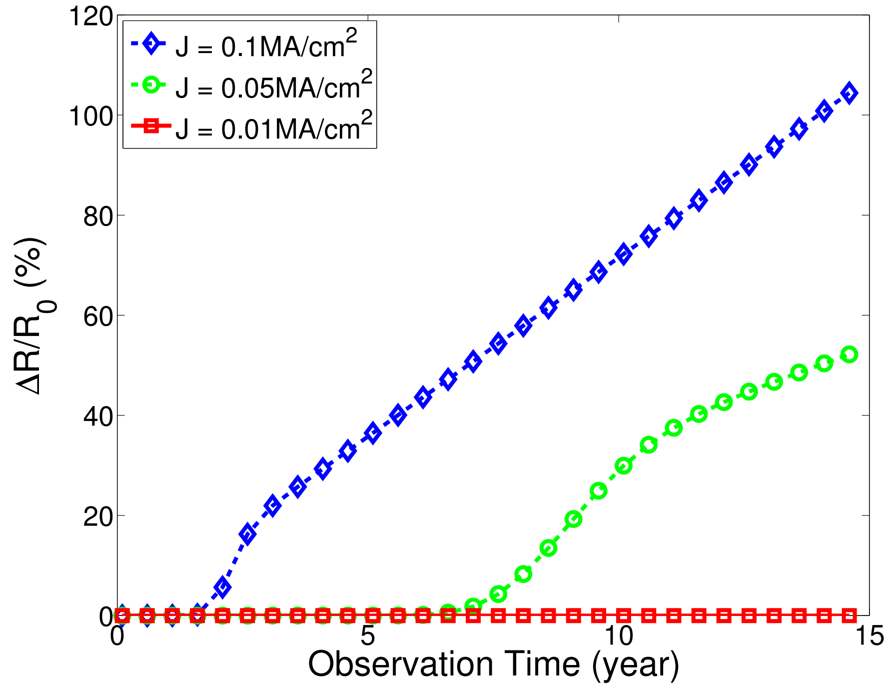


Figure 2.14: Interconnect resistance increase in percentage due to different current densities in 28*nm* BEOL technology.

Impact of EM-Induced Performance Degradation in AVS Systems

We now assess the impact of EM degradation on wires for logic and clock signals, and for power and ground distribution.

Impact of EM stress on signal wires. To evaluate the impact of EM-induced resistance increase (ΔR) from its initial resistance R_0 on circuits, we use *Synopsys HSPICE* to simulate a 30-stage chain of buffers in a 28*nm* foundry FDSOI technology library. In Figure 2.15, we simulate delay of the buffer chain[8] due to increase in resistance because of EM. We vary resistance from the nominal resistance of the wire[9] to 556% of the nominal resistance. We use multiple fanout-of-four (FO4) capacitive loadings, that is, we multiply

[8] To consider the worst-case loading, we set the nominal buffer size to 8× the minimum-sized buffer from the 28*nm* library FDSOI .

[9] For this buffering experiment, we assume the nominal wire resistance to be 600Ω, which is extracted from 28*nm* BEOL technology with $L = 150\mu m$ and $W = 50nm$.

Table 2.3: EM and BEOL parameters.

	Definition	**Value**	**Units**
ρ_{Cu}	Resistivity of copper	2.50×10^{-8}	Ω/m
ρ_{Ta}	Resistivity of Tantalum liner	3.10×10^{-7}	Ω/m
T	Temperature	125	°C
σ_c	Effective critical stress for void nucleation	4.10×10^{7}	Pa
σ_{th}	Threshold for thermal stress	0	Pa
Z^*_{eff}	Effective charge number	5	
D_0	Diffusivity constant in Arrhenius' relation	9.80×10^{-9}	m^2/sec
k_B	Boltzmann constant	1.38×10^{-23}	J/K
e	Charge of single electron	1.60×10^{-19}	C
Ω_{Cu}	Atomic volume of copper	1.18×10^{-29}	m^3
B	Effective bulk modulus	10^9	Pa
μ_{Ea}	Mean of activation energy	7.52×10^{-20}	J
σ_{Ea}	Standard deviation of activation energy	8.00×10^{-22}	J
W	Copper wire width	100	nm
L	Copper wire length	150	μm
T_{wire}	Thickness of (Cu + Ta)	100	nm
T_{Ta}	Thickness of Ta	5	nm

the FO4 capacitive load by factors $\{1.0\times,\ 1.6\times,\ 2.1\times\}$. EM stress slows down circuit performance by increasing stage delay and by decreasing drive current due to increased output transition times.

We also study the impact of different gate sizes other than the $8\times$ of the minimum-sized buffer to evaluate the impact of ΔR after gate sizing. For the delay shown in Figure 2.16, we observe that larger gates do not necessarily help to reduce the impact of ΔR. The possible reason is that the drive current of larger gates cannot compensate for the increase in gate capacitance from the next stage. As we size up all the cells at the same time, the capacitive load due to large input capacitances of large-sized gates also increases. The experiments above indicate that for typical signal wires, the model from [164] estimates that delay increases by $\sim$8% through cell chains with buffers smaller than $4\times$ the minimum-sized buffer,

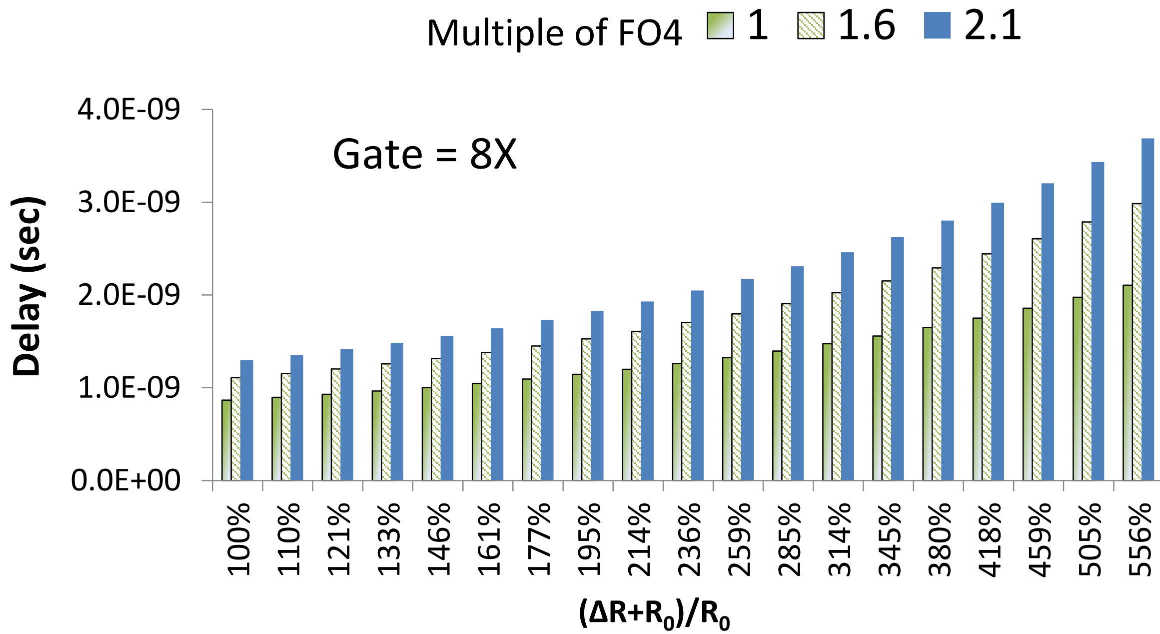


Figure 2.15: Impact of ΔR on path delay with different capacitive loadings. $V_{dd} = 1.1V$.

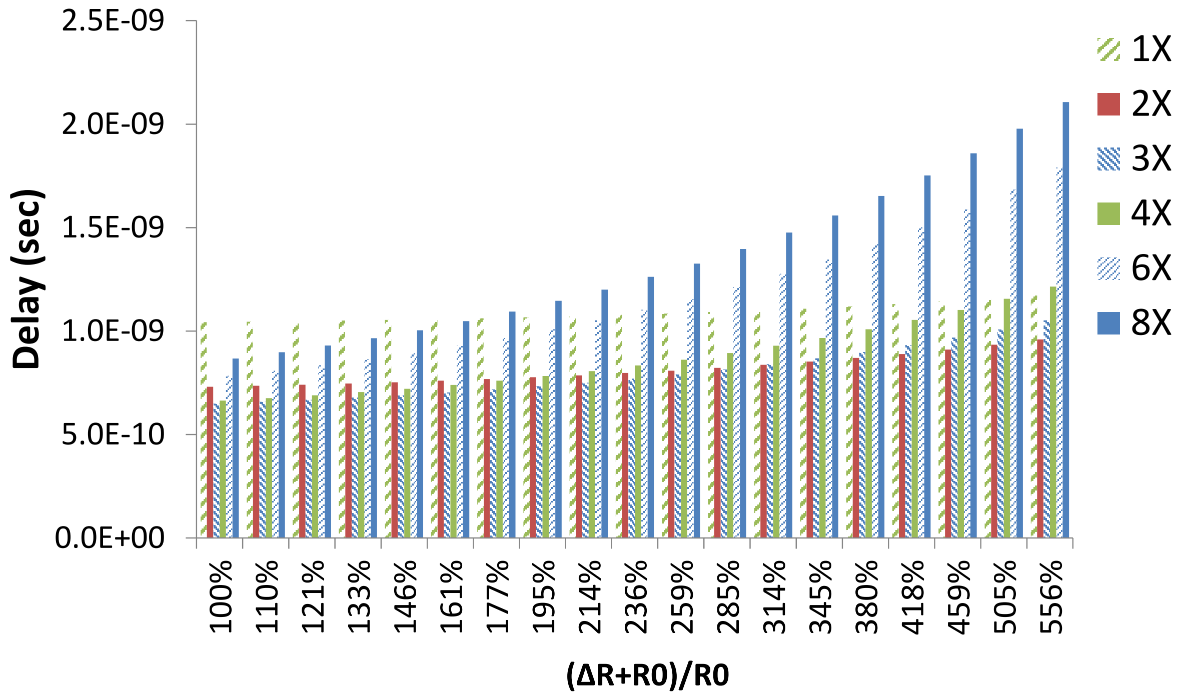


Figure 2.16: Impact of ΔR on path delay due to EM stress with different gate sizes. $V_{dd} = 1.1V$.

when the wire resistance increases to high values (∼146%). According to our evaluation in Figure 2.14, this is equivalent to 10 years' degradation when the current density is $0.1 MA/cm^2$.

Impact of EM stress on IR drop on power and ground mesh. We use the circuit shown in Figure 2.17 to study the IR drop impact of EM stress on power and ground mesh. We define the supply voltage used by the core logic as V_{dd} and the voltage applied on the power and ground mesh (P/G mesh) and power ring by the regulator as $V_{regulator}$. When the resistance of the P/G mesh (R_{PG}) increases due to EM, the power consumed increases when the drive current remains the same so as to achieve the same performance. However, the degradation of P/G mesh depends on the scheduling of the supply voltage to compensate for IR drop and BTI effects. Since different guardbands for BTI at signoff can change the behavior of voltage scheduling, the impact of EM on P/G mesh also changes with different guardbands.

Table 2.4: Signoff corners for BTI. V_{min} = $0.90V$, and V_{max} = $1.10V$.

$\mathbf{V_{lib}}$	V_{max}	$0.98V$	$0.97V$	$0.96V$	$0.95V$	V_{min}	V_{min}	V_{min}
$\mathbf{V_{BTI}}$	V_{max}	$0.98V$	$0.97V$	$0.96V$	$0.95V$	V_{min}	N/A	V_{max}
Corner#	**3**	**5**	**6**	**7**	**8**	**1**	**4**	**2**

We choose eight different signoff corners defined by different V_{BTI} and V_{lib} [37] as shown in Table 2.4 to evaluate the impact on different-aged *AES* (from *OpenCores* [247]) implementations.[10] From [37], we estimate the final AVS voltage V_{final}, as obtained from cell chain simulations, to be $0.98V$. Therefore, we sweep V_{dd} across $\{0.98V, 0.97V, 0.96V, 0.95V\}$ to cover different margins for our AVS simulations. With the libraries characterized at the eight signoff corners, we perform synthesis, place and route (SP&R), and sign off the implementations with *Synopsys PrimeTime vH-2013.06-SP2* [253]. We report core power (defined as the power consumed by the design other than P/G mesh) as well as the P/G

[10]Implementation #4 has no BTI degradation in the signoff libraries, so V_{BTI} is not applicable.

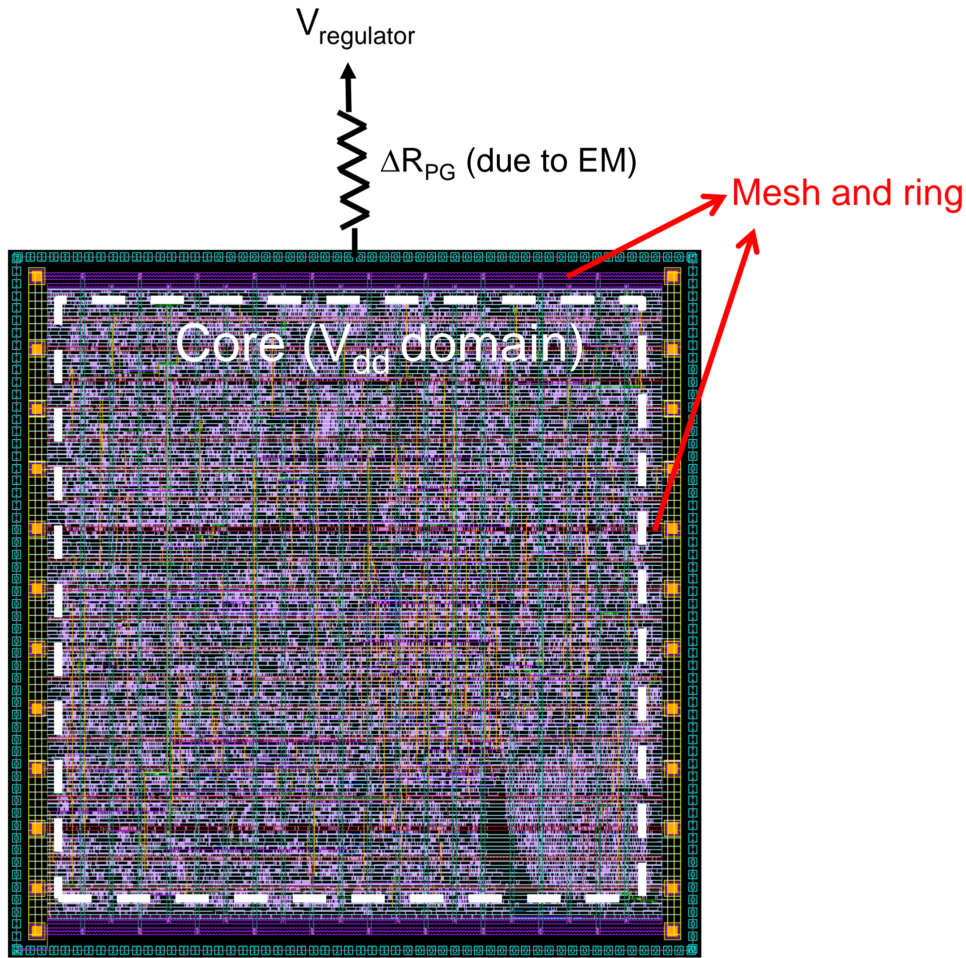


Figure 2.17: Illustration of P/G mesh, V_{dd}, and $V_{regulator}$.

mesh power. We account for EM stress and perform AVS simulations in *MATLAB* [245] and calculate the core power by interpolating power simulation results obtained from *PrimeTime*.

We calculate the P/G mesh power by obtaining the initial R_{PG} = 2Ω from 28*nm* FDSOI BEOL foundry technology library, and the increase of wire resistance is modeled using the statistical EM model in [164]. We obtain the average current on the P/G mesh to be 10mA from our SP&R implementation and power simulations of the design *AES*. The calculated P/G power with the above assumptions are shown in Figure 2.18. When there is EM stress, the statistical model predicts a ∼1% power increase due to worst-case BTI degradation (Implementation #2). In summary, our experimental results with the

statistical model in [164] indicate that it is optimistic with respect to the amount of EM degradation, leading to smaller estimated delay and power penalties. Therefore, we use both this model as well as Black's Equation in our further studies.

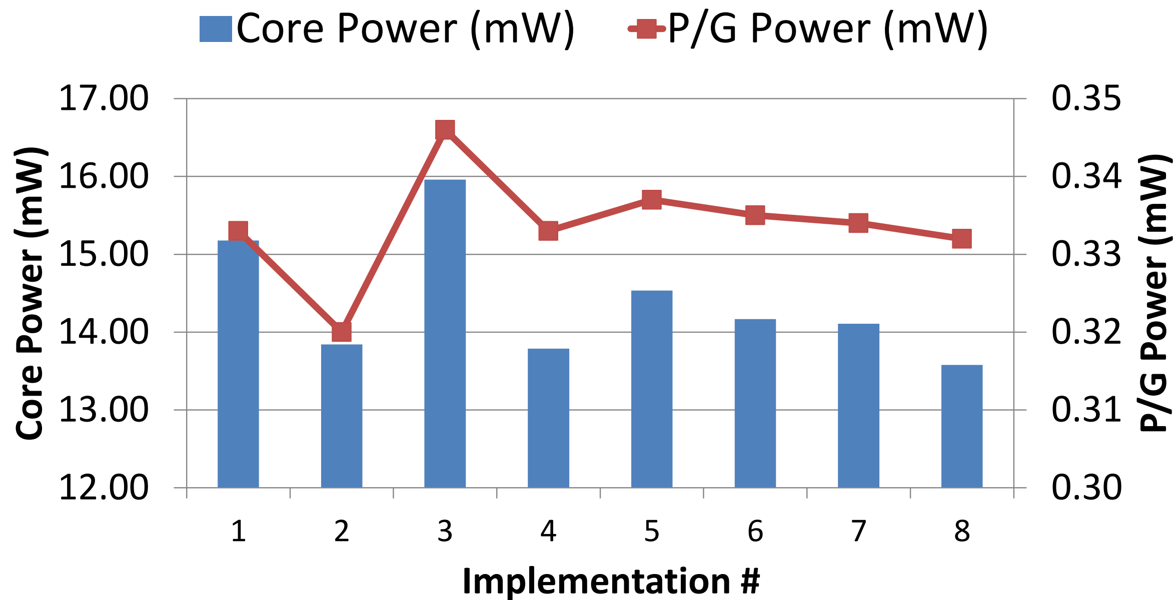


Figure 2.18: Power comparison between different signoff corners with EM stress $\Delta R_{PG} > 0$.

2.2.3 Experimental Setup and Results

Our studies in Section 2.2.2 indicate that cell chains with buffers or inverters of different sizes have different behaviors due to EM stress. This relationship between gate sizes and EM vulnerability can lead circuit designers to change the *wire and gate widths* to build more reliable circuits when both EM and BTI can degrade the circuits.

In this section, we report experiments aimed at quantifying design costs in terms of power, lifetime and area of the implemented circuits. Total power changes with voltage scheduling due to AVS as well as due to effects of both EM and BTI. We use the following steps to implement our test circuits and evaluate design costs.

1. We use *Synopsys SiliconSmart v2013.06-SP1* [255] to re-characterize the 28nm FDSOI library at different supply voltage V_{dd} and ΔV_{th} to obtain the impact of BTI on delay, dynamic power, and leakage power. We characterize a total of 48 libraries by setting V_{dd} to $\{0.9V, 1.0V, 1.1V\}$, threshold voltage of the PMOS V_{thp} to $\{0mV, 40mV, 80mV, 110mV\}$, and threshold voltage of the NMOS V_{thn} to $\{0mV, 20mV, 40mV, 55mV\}$.

2. To compare the impact with different aging due to BTI, the circuits are synthesized, placed, and routed at eight different corners shown in Table 2.4. We use *Synopsys Design Compiler vH-2013.03-SP3* [251] and *Synopsys IC Compiler vI-2013.12-SP1* [252] in the flow.

3. The timing, dynamic power, and leakage power of the implemented circuits are measured using the 48 libraries, characterized above, with *Synopsys PrimeTime vH-2013.06-SP2* [253].

4. The behaviors of AVS, BTI degradation, and EM degradation are simulated in *MATLAB* [245]. The BTI analytical model is from [212] and calibrated to the data in [228]. For each time step, the circuit power, delay, and the wire degradation due to EM are updated using the two models. When the V_{dd} and ΔV_{th} are not exactly the same as the 48 data points in the previous step, we use interpolation to calculate the values.

We conduct the following three experiments.

- **Experiment 1.** The goal is to demonstrate the impact of final AVS voltage on EM lifetime.

- **Experiment 2.** The goal is to quantify the design costs to fix EM in systems with AVS.

- **Experiment 3.** The goal is to demonstrate the impact of voltage step size and scheduling due to AVS on EM lifetime.

Experiment 1 Results

We study the impact of the final AVS voltage on EM lifetime by using the following steps.

- Step 1. Assume all the circuits have an initial lifetime of 10 years at a nominal voltage of $0.90V$.
- Step 2. For each AVS timestep i, the remaining lifetime is calculated as follows [20] [116].

$$MTTF(i) = MTTF(i-1) \times \left(\frac{V_{dd}(i-1)}{V_{dd}(i)} \right)^2 \tag{2.16}$$

- Step 3. Update the lifetime after each AVS timestep until all the lifetime is used up.

Figure 2.19 shows how the final AVS voltage V_{final} affects EM lifetime for eight implementations of the *AES* design from *OpenCores* [247]. EM lifetime decreases for implementation #3 more than others because implementation #3 has large negative slack at signoff, hence higher V_{dd} is required to compensate for the negative slack. The degradation for implementation #3 is 30% of its lifetime, that is, decrease from 10 years to seven years. We notice that implementation #3 has significantly lower lifetime ($>$ one year difference) compared to #1 even though implementation #1 and #3 have similar V_{final} (difference $<$ $50mV$). The reason is that implementation #3 increases the V_{dd} very early due to less margin for BTI, so it degrades EM lifetime sooner than implementation #1. This figure demonstrates how AVS impacts EM lifetime and thereby EM signoff. We further study the impact of voltage scheduling on EM lifetime in Experiment 3.

Experiment 2 Results

To quantify design costs due to fixing EM, we implement two designs *AES* and *DMA*, and perform gate-sizing for timing closure in *Synopsys IC Compiler* before signoff. We then apply EM constraints in an ALF file to the test circuits and use the following steps to fix the violations.

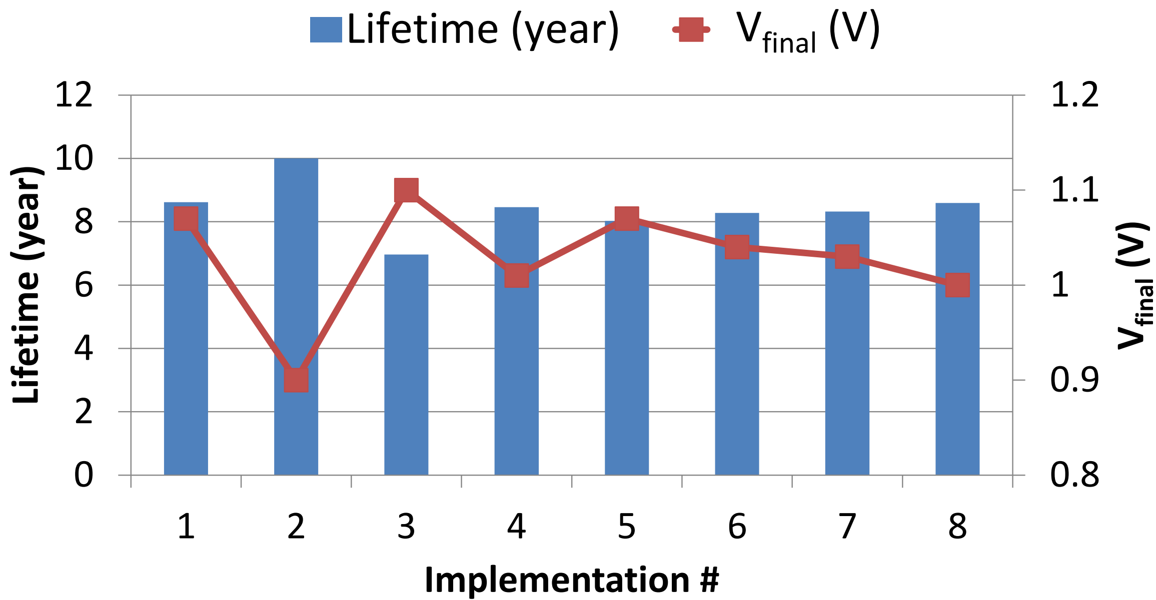


Figure 2.19: Lifetime and V_{final} of the eight implementations of the *AES* design.

1. Step 1. Generate initial EM violation report of wires with the EM constraints.
2. Step 2. For each wire segment that has EM violation, use *non-default rules* (NDRs) to fix EM violations.
3. Step 3. Repeat Steps 1 and 2 until EM violations are fixed.
4. Step 4. Re-route and apply gate-sizing to fix the remaining EM violations, if any, and then perform timing recovery to fix all setup and hold time violations.[11]

Figures 2.20 and 2.21 respectively show power and area for the *AES* and *DMA* designs across eight implementations in Table 2.4, and for three cases: (i) BTI signoff is not aware of EM, (ii) BTI signoff is EM-aware and uses the traditional Black's Equation model, and (iii) BTI signoff is EM-aware and uses the recent statistical EM model. As demonstrated in Section 2.2.2, the statistical EM model is optimistic and has almost same power as Case (i) for most implementations. EM-awareness decreases power in

[11]Note that the baseline signoff corner is set at $V_{dd} = 0.9V$. We do not guardband BTI at signoff in this experiment, and assume AVS will be used to compensate the aging. Since we downsize cells to fix EM violations, the circuit timing will be affected.

AES for implementations #1 and #2 due to downsizing of gates and constraining maximum fanout in the clock network. In *DMA*, EM-awareness increases area per the statistical model because of resistance increase and addition of smaller buffers to fix setup violations. The worst-case area penalty (∼1.6%) is for implementation #2 of *DMA*, and the worst-case power penalty (∼6%) is for implementation #6 of *AES*. Note that the EM fixing may cause buffer downsizing and power reduction due to less wire delay. However, this is at the cost of worse congestion and difficulty in detailed routing due to different wire-sizing rules.

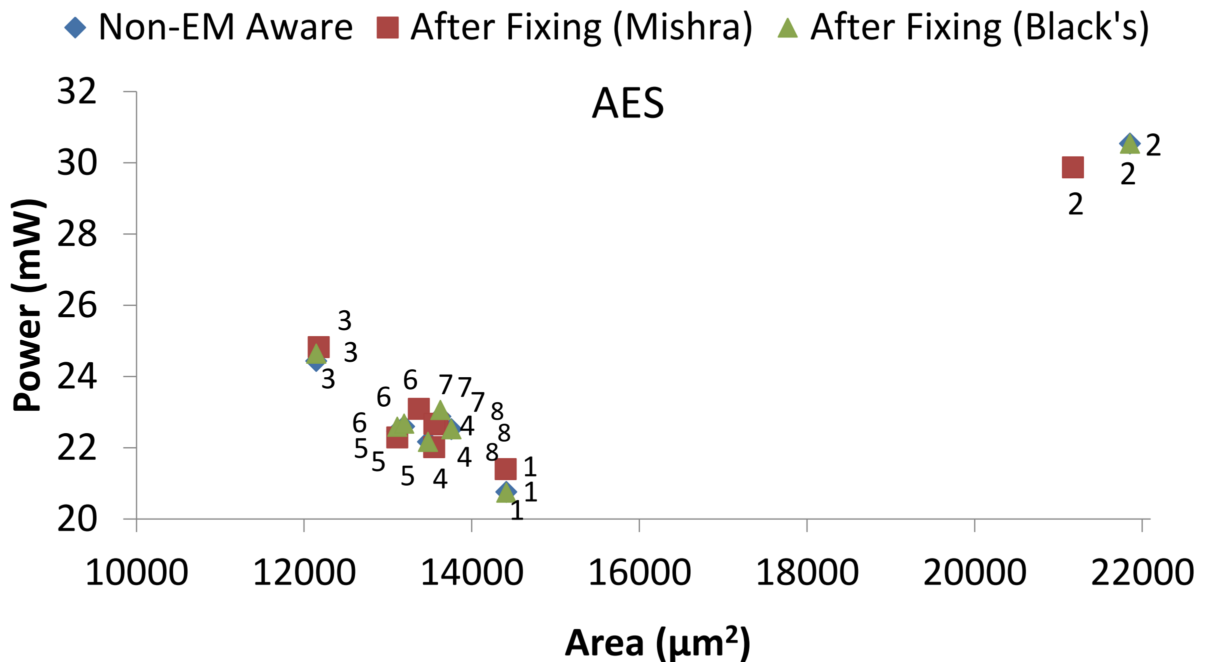


Figure 2.20: Power and area due to EM fixes on AVS systems using both Black's Equation and [164] models for the *AES* design.

Experiment 3 Results

EM lifetime is affected due to voltage scheduling in systems using AVS. We use five different voltage schedules S1–S5. The initial voltage for all these schedules is $0.90V$. The steps by which each schedule is increased is shown as the tuple (schedule, voltage step): (S1, $8mV$), (S2, $10mV$), (S3, $15mV$),

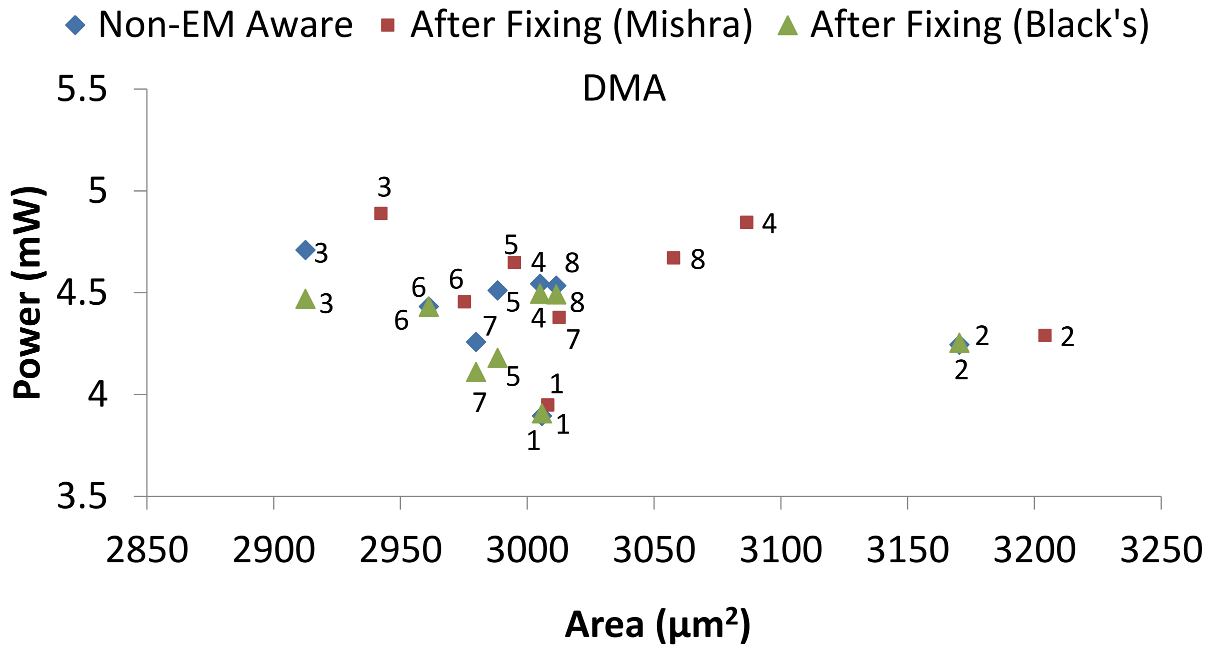


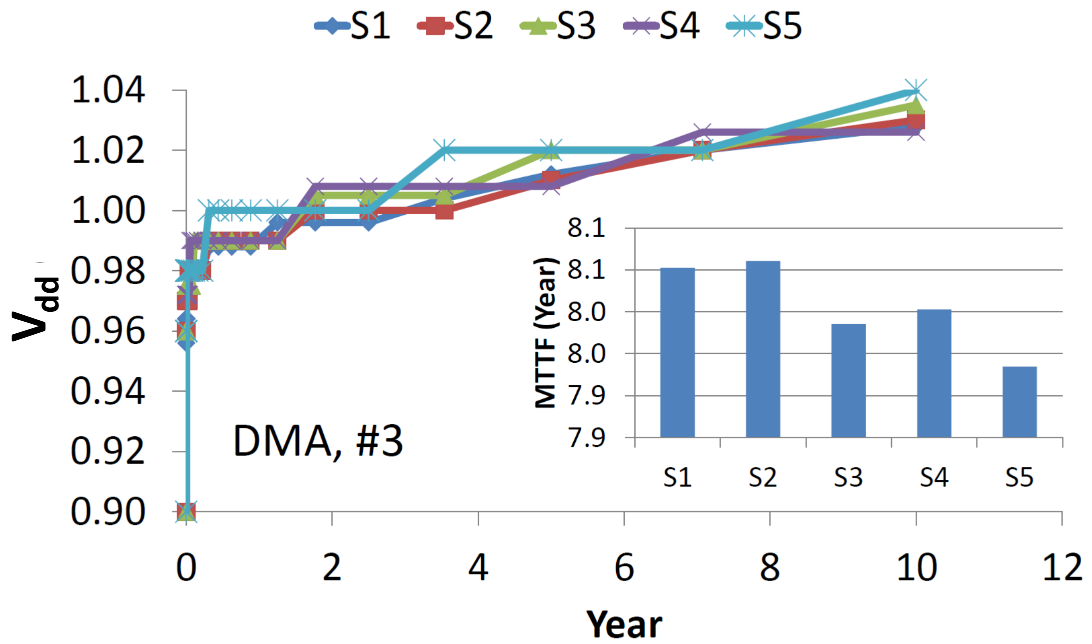
Figure 2.21: Power and area due to EM fixes on AVS systems using both Black's Equation and [164] models for the *DMA* design.

(S4, $18mV$) and (S5, $20mV$), The voltage step is made whenever delay is higher than target due to EM degradation. We conduct the experiment as follows.

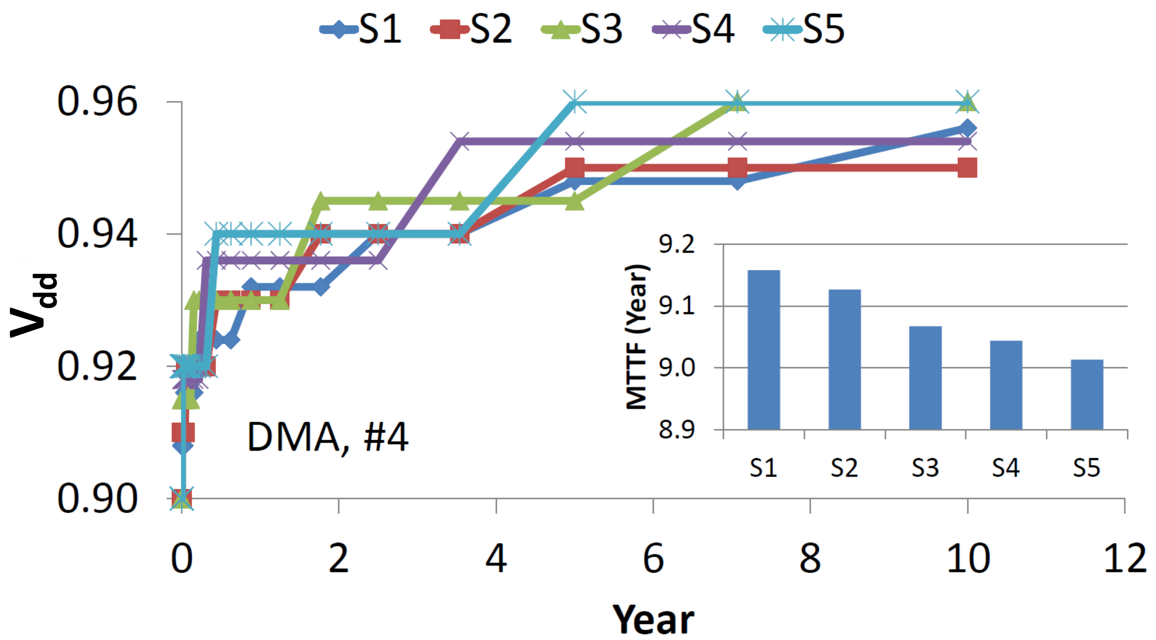
- Step 1. Signoff the *AES* and *DMA* designs at $0.90V$ with no guardband for EM and assume that AVS will compensate aging due to BTI.
- Step 2. Simulate the delay degradation with BTI [212] and EM [164] models. To compensate delay degradation, increase the supply voltage V_{dd} to a higher value in each timestep according to the schedule that is being used.
- Step 3. Update remaining lifetime using Equation (2.16) and repeat Steps 1 and 2 until the entire lifetime is consumed.

Figures 2.22(a) and (b) show the EM lifetime for the five schedules for the *DMA* design. Figures 2.23(a) and (b) show the EM lifetime for the five schedules for the *AES* design. Implementation #4 is the

baseline with no BTI degradation. We observe that a scheduling with voltage steps of 18mV or 20mV can result in up to 1.5 years of decreased EM lifetime, which indicates that small fluctuations in guardband can result in significant lifetime changes due to different voltage scheduling requirements.

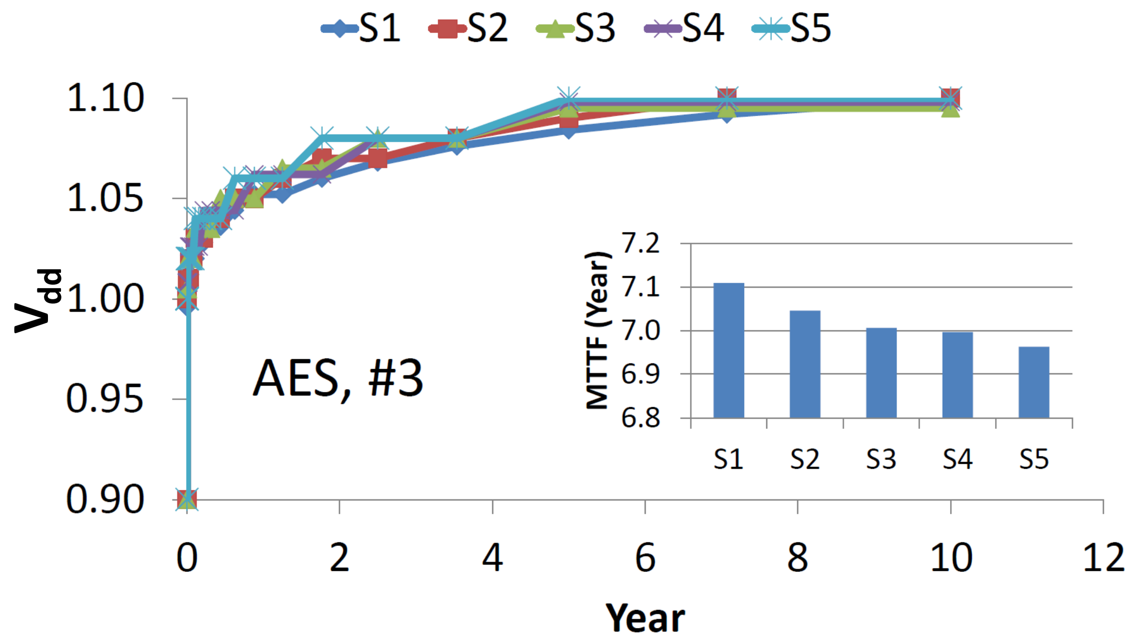


(a)

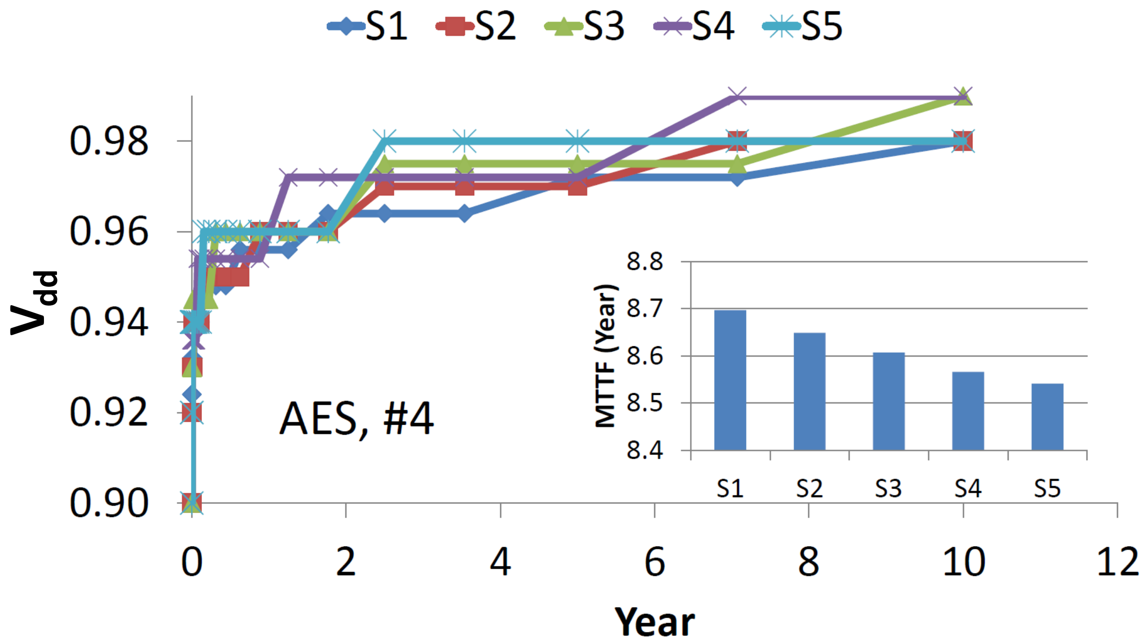


(b)

Figure 2.22: Impact on EM lifetime due to five different voltage schedules in AVS systems for *DMA* implementations (a) #3 and (b) #4.



(a)



(b)

Figure 2.23: Impact on EM lifetime due to five different voltage schedules in AVS systems for *AES* implementations (a) #3 and (b) #4.

2.2.4 Conclusions

Reliability signoff affects circuit performance, power and lifetime. In Section 2.2, we study the joint impact of BTI, AVS and EM on signoff. We use a statistical EM model recently proposed by [164] and apply it to our AVS simulations to study the signoff of both EM and BTI. We use two EM models (i) resistance increase as predicted by the model in [164] in AVS systems and (ii) the traditional Black's Equation-based EM model, implement fixes in commercial tool flows, and use EM constraints from a 28*nm* foundry FDSOI library. The model from [164] is easier to apply in earlier implementation stage by derating the resistance in commercial tools, but it tends to be optimistic in our characterization. Our experimental results indicate that for signal wires, large drivers should be avoided as they suffer from more delay degradation due to EM-induced resistance increase. For P/G mesh, we quantify the impact of resistance increase and conclude that the area-power curve formed by different BTI signoff corners is shifted due to EM. We empirically analyze the cost of fixing EM at signoff with different guardbands against BTI. In our studies, this cost is up to 1.6% increase in area and 6% increase in power. Our ongoing work seeks to (i) explore signoff methodologies for other reliability mechanisms at advanced foundry nodes, (ii) improve accuracy of signoff by considering thermal gradient effects, and (iii) develop a learning-based modeling approach to quantify design costs of reliability.

2.3 Acknowledgments

Chapter 2 contains reprints of W.-T. J. Chan, A. B. Kahng and S. Nath, "Methodology for Electromigration Signoff in the Presence of Adaptive Voltage Scaling", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2014; and T.-B. Chan, W.-T. J. Chan and A. B. Kahng, "ILP-Based Identification of Opportunistic Redundant Logic Insertions for Opportunistic Yield Improvement

During Early Process Learning”, *Proc. IEEE International Conference on Computer Design*, 2017.

I would like to thank my coauthors Tuck-Boon Chan, Andrew B. Kahng and Siddhartha Nath. For the SLIP publication, I also like to thank Vivek Mishra and Sachin Sapatnekar for their very helpful clarifications regarding the EM model described in their work [164].

Chapter 3

Machine Learning-based Routability Optimization

This chapter presents a routability optimization work guided by a machine-learning model. Routability issue is more severe in advanced technology node. Standard cell design is constrained by multiple patterning and the design tendency of less tracks for better area utilization. However, compact cells potentially lead to worse pin accessibility. At the design level, physical design engineers may need to increase cell spacing to resolve possible *design-rule check violations* (DRVs). We first demonstrate the machine-learning model can efficiently close the gap between the prediction by congestion map and the outcome of detailed routing. Then, we demonstrate our model-guided cell spreader can improve the design routability at insignificant design cost.

3.1 Routability Challenge in Advanced Nodes

As semiconductor technology advances, the EDA and design communities have seen increasing unpredictability in the IC implementation flow. In particular, design tapeouts are increasingly placed at risk by the inability of the router to complete the routing successfully. Historically, any risk of unroutability could be identified prior to the runtime-intensive detailed routing (DR) stage, based on congestion maps generated using global routing (GR). However, the miscorrelation between these congestion maps and the actual routing design rule check (DRC) violation maps has increased significantly at current process nodes due to the many new, complicated design rules defined at these nodes. This unpredictability causes added iterations (and consequent schedule slippage) during design implementation, sometimes endangering the design tapeout itself.

At advanced process nodes, GR-based congestion maps do not correlate well with DRC violation (DRV) maps obtained at the end of detailed routing. This is a consequence of the numerous complicated design rules imposed upon design layouts to ensure viable fabrication; these DRVs, most of which are not visible in the GR routing model, constrain the detailed router significantly. (The study of Han et al. [85] quantifies wirelength overheads due to increasing numbers of design rules.) As a result, GR-based congestion maps are no longer good predictors either for evaluating overall design routability or for identifying potential DRV hotspots prior to detailed routing. Therefore, they can easily mislead any routability optimization engines that rely on these maps, resulting in poor effectiveness at resolving routability problems, even as they sacrifice timing and area metrics to ameliorate spurious congestion problems. Figure 3.1 shows an example of such a miscorrelation on a sub-14*nm* design. The figure compares a map of actual DRVs with a map of congestion hotspots obtained by running a state-of-the-art industrial global

router on the same layout; an overlay of these two maps is also shown. The GR-based congestion map is thresholded so that both maps display the same number of violating grid cells.[12]

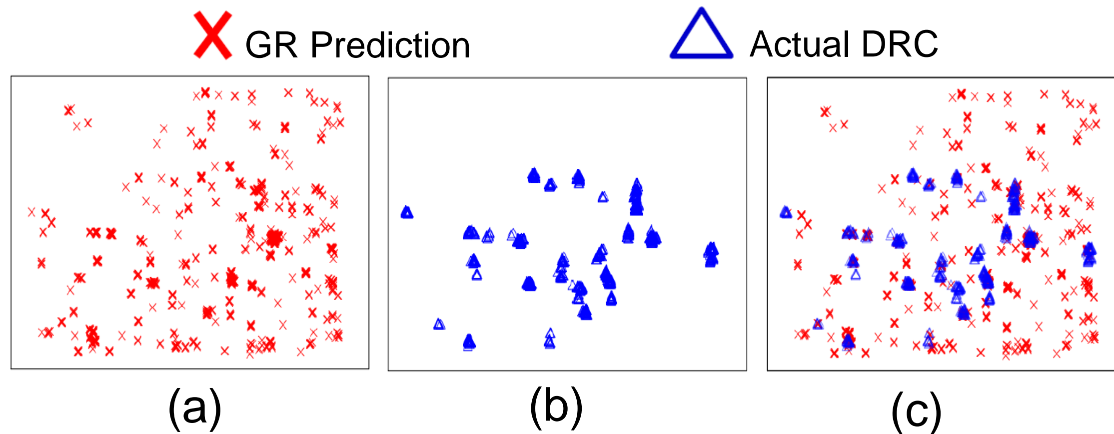


Figure 3.1: Comparison between the GR-based congestion map and the actual DRVs (gcells with DRC violations) on a sub-14*nm* design. (a) Overflows extracted from GR-based congestion map (per gcell). (b) DRVs after detailed routing. (c) An overlay of the GR-predicted overflows and actual DRVs, highlighting the numerous false positive and false negative predictions. The placer and global router are from a state-of-the-art industrial physical design platform.

The miscorrelation between the congestion map and the actual DRV hotspot map demonstrates why the routability improvement techniques traditionally employed during physical synthesis are not very effective at advanced process nodes (since they are driven by GR-based congestion maps). At the same time, it is critical to model and optimize routability during the physical synthesis stage, since the netlist and layout transforms that are permissible during routing-based optimization—when the DRVs actually manifest themselves—are very limited in scope. This motivates the study of better DRV hotspot prediction techniques and their use for routability improvement without hurting the timing convergence of the design. In this chapter, we describe a new algorithm that employs machine learning to accurately identify and optimize routability hotspots during physical synthesis without any timing or area overhead;

[12]We use the GR-based congestion map as the reference because this is still the most common way to estimate routability in industrial physical implementation tools and flows. We generate the congestion maps by summing up numbers of overflows on each metal layer.

furthermore, we demonstrate the effectiveness of our approach on industrial benchmarks in a sub-14*nm* process node from a leading foundry.

3.2 Related Works

Ensuring the routability of designs has always been a central challenge in IC implementation. Attempts to address this problem are usually most effective at the physical synthesis stage. In parallel, the problem of routability misprediction has also attracted interest in the EDA research community. We categorize and summarize previous works in these domains as follows.

Congestion predictors. Taghavi et al. [208] propose *MILOR* to identify local routing hotspots by examining pin-shape layouts and their densities or proximities within the placed design. Chan et al. [39] propose a learning-based methodology to predict the overall routability of a design by using placement information. Zhou et al. [231] use MARS to model DR congestion.

Routability-aware global routing. Qi et al. [188] improve the DR model from [231] to guide the global router. Wang et al. [219] and Zhong et al. [230] also propose the use of DR model-guided global routers. However, such works typically apply only to the routing stage, during which the allowed netlist and layout transforms are limited.

Congestion-aware placement. There has been significant work on congestion-aware placement in both industrial and academic coarse placers. For example, [103] [94] [90] [150] [151] [152] [176] [121] use global routers to predict congestion and feed the information back to the placer in order to improve routability. The works [29] [211] [91] [205] [117] [194] [221] use spreading regions, densities of nets, or routing patterns to estimate congestion and guide placement. However, such works are typically limited by their reliance on GR-based congestion maps.

Whitespace optimization. When the pin accessibility problem dominates the DRVs, it is effective to use cell inflation during placement to improve the routability. Sadakane et al. [195] first addressed pin accessibility in the context of metal gate arrays, using a simulated annealing approach. [93] and [24] incorporate empirical heuristics to guide cell inflation in two academic placers to improve the routability. Instead of considering added space as attachments to (bloated) cells, [226] [147] [31] [3] handle the whitespace as separate components in placement, which enables more proactive control of whitespace to improve routability. However, the introduction of whitespace typically involves a timing and area overhead, especially when it is conservatively driven by poor routability predictor metrics such as GR-based congestion maps.

Our work is different from the previous works in several significant ways. (i) Rather than merely predicting routability, we show how to use machine learning to automatically *improve* the routability of the design. (ii) Our engine focuses on improving route completion at the detailed routing stage, rather than optimizing GR congestion (and, does so without hurting the timing closure of the design). (iii) Rather than merely predicting whether the overall design is routable or not, we use learning to predict the actual locations of the DRV hotspots. (iv) Our predictor comprehends global routing, netlist structure, and cell-layout level information to capture routability risks due to both routing resource shortage and complicated design rules.

3.3 Methodology for Routability Optimization

In order to overcome the miscorrelation between the GR-based congestion map and the actual DRV map, we use machine learning to improve the accuracy of our identification of potential DRV hotspots. We model this prediction problem as a supervised classification problem. We label DRC-violating *global routing cells* (gcells) in our training set of IC layouts with true labels, and cleanly-routed

gcells with false labels. We then extract various parameters from the training netlists and layouts and use them to build an accurate predictor. Using this predictor, we propose an engine that minimally perturbs the converged physical synthesis netlist in a way that surgically redistributes the whitespace in the vicinity of the predicted DRV hotspots so as to ameliorate those hotspots without hurting timing, area or wirelength. We demonstrate the effectiveness of this approach on several layouts at a sub-14*nm* process node from a leading foundry. Our results show that we can reduce DRVs by up to 76.8% and by an average of 20.6%, without hurting design convergence.

The key contributions of our work are as follows.

1. We present the first application of the machine learning paradigm to actually *optimize* design routability (in contrast to *predicting* routability, as in [39]).

2. We quantify the miscorrelation between a GR-based congestion map and the actual DRV map in designs at a sub-14*nm* process node.

3. We use machine learning to predict actual DRV locations in a design layout and use the prediction to improve post-placement routability. This is a significantly more difficult problem than the binary prediction made in [39] on whether the overall design would be routable or not.

4. We develop an engine that employs our new learning-based predictor of DRV hotspots to ameliorate these hotspots without hurting timing, area or wirelength.

3.3.1 Machine Learning-based Optimization for Routability

We use machine learning to generate a model to close the gap between DRV hotspot prediction using congestion map and actual DRVs located after detailed routing. To enable accurate predictions, our model incorporates diverse parameters that we describe in Section 3.3.2.

With the help of this robust and accurate predictor, we are able to guide the optimization effectively to improve design routability. We design an algorithm that can leverage this predictor to ameliorate the DRV hotspots with minimal layout perturbation, thus avoiding timing or area penalties. This algorithm is described in more detail in Section 3.3.3.

3.3.2 Predictor Design

We use *hotspots to* refer to gcells with DRVs. The objective of our predictor is to separate hotspot gcells from non-hotspot gcells. In order to analyze the root causes of routability problems, we first partition training layouts into small grids on top of gcells (we use the term *local windows*[13] for these grids) to generate training data. We extract numerous netlist and layout parameters for each local window. These parameters include:

- **Density parameters** such as local pin density and local cell density;
- **GR parameters** obtained from a global routing invocation, such as local overflow, demand and capacity of each metal layer and via layer;
- **Pin proximity** measuring the average and minimum spacings between pins in each local window (proposed in [39]);
- **"Unfriendly" cells**, which are library cells that occur in the DRV hotspots (local windows with more than one DRV) at a rate significantly higher than their overall rate of incidence in the netlist;
- **Multi-height and sequential cells** and parameters relating to their fanins, fanouts, and occurrence frequencies in local windows;

[13]We use two sizes of local windows, including 1×1 (gcell itself) and 3×3 windows (including a central gcell and the surrounding gcells). The 3×3 local windows have overlapping regions. Overlapping regions capture the influence among gcells in the DRV prediction.

- **Connectivity parameters** such as the number of buried nets completely enclosed inside the local windows, the number of non-buried nets crossing these window boundaries [39], and the number of connected pins lying outside the windows; and
- **Structural parameters** such as the number and depth of fanin and fanout logic stages in paths crossing local windows.

Multi-height cells are cells with heights greater than that of a single cell row, which are typically sequential cells in this technology. Fanin or fanout numbers refer to the numbers of sequential cells transitively connected (incident or outgoing, respectively) to the cells within the current local window. The intuition of "structural parameters" is to evaluate the likelihood of cells to be placed around the sequential cells. The sequential cells are especially of interest because they have lower pin densities and different track heights.

We then go through an iterative process of training the predictor model using our parameter list. The iterative process contains both layout observation and evaluation of statistical significance.[14] In each iteration, we measure the statistical significance of the various parameters, and analyze the locations of both false-positive and false-negative predictions, in order to refine the parameter list and identify additional predictive features of the design. This process is repeated with several mathematical models of machine learning, *viz.*, linear regression, logistic regression, and *support vector machines* (SVM) [88] with various kernel choices.[15] As an illustration of the physical analysis involved in this iterative process, consider the following example.

[14]For the layout observation, we check the DRV locations and the cell placements. We also examine the false-positive and false-negative rates and the p-values (derived from the confusion matrices) after adding the parameters. We keep parameters that contribute to accuracy improvement in the process.

[15]We test among linear, polynomial, and RBF kernels. We apply different weights to the DRC-violating gcells during model training and evaluate the model accuracy with testing gcells. RBF shows the best true-positive rate with similar false-positive rate with the first few parameters (density and GR). We choose RBF for our main experiments reported here.

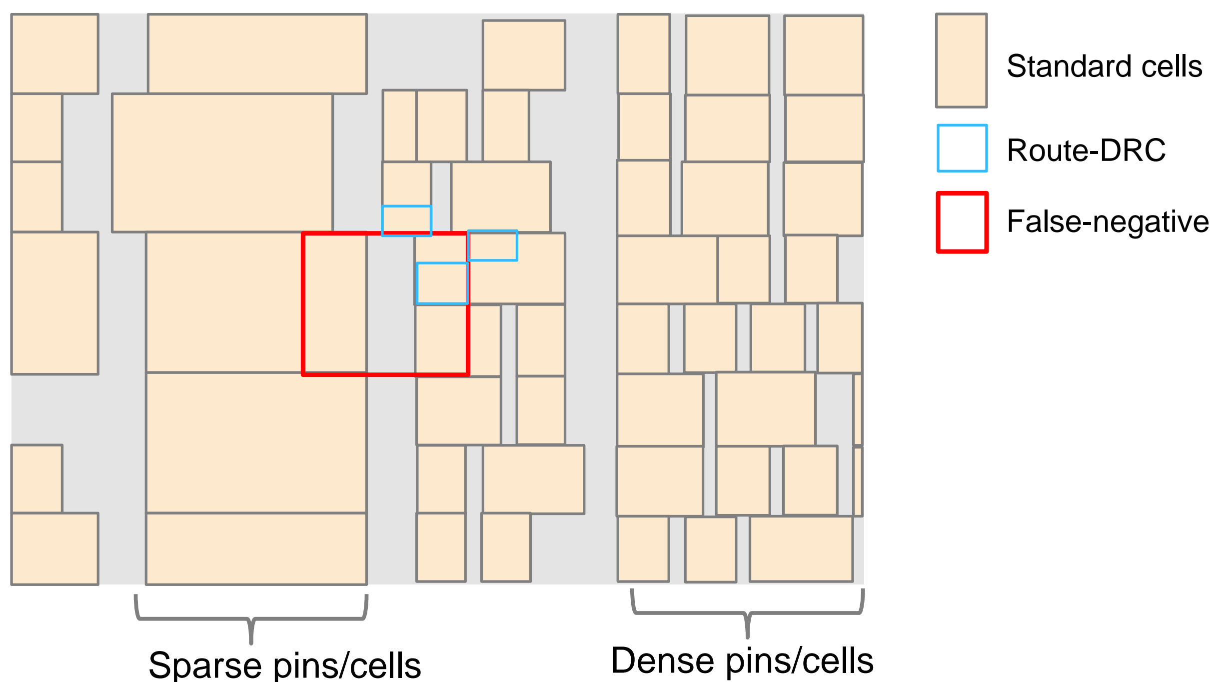


Figure 3.2: An example of the analysis of a false negative prediction.

Figure 3.2 shows an example of a DRV hotspot that was predicted to be DRV-clean in one of our earlier predictor models.[16] In our manual analysis of this hotspot, we find that the red false-negative region itself has low pin and cell density but is located close to a region with higher pin and cell density. We capture this anomaly (caused by the tendency of the router to introduce small detours around DRV hotspots) in our predictor model by using larger local windows for such parameters, and by distinguishing between the parameter values inside a gcell, and the parameter values in a larger window that is centered on the gcell.

Besides incorporating different parameters to improve the prediction accuracy, we apply the following approaches to improve the accuracy and robustness of our local hotspot predictor. (i) Since the majority of the gcells in a typical layout do not have DRVs, the training of the predictor is easily misguided by the biased distribution of the few DRC-violating gcells and the many DRV-clean gcells. We address

[16]This reference predictor model has only basic parameters (density, GR overflow, etc.) and uses a single size (1×1) of local windows. The mathematical model for prediction is SVM.

this problem by emphasizing the DRC-violating gcells, increasing their weights[17] during the training stage. (ii) Given that there are very few real-world sub-14*nm* designs and layouts available at this time, it becomes important to choose the training methodology carefully so as to avoid overfitting. We do this by randomly choosing 20% of the gcells from the layout to be the training data set and use the remaining 80% for testing. We repeat this randomized 20%-80% evaluation 12 times and use the average and distribution of the prediction accuracy from these 12 runs[18] to draw any conclusions about the tested parameter set and mathematical prediction model (*viz.*, linear regression, logistic regression, or SVM). (iii) In order to take the neighborhood effect into consideration, we use both small and large local windows to annotate the central gcells. In addition to the multiple local window sizes, we also annotate the central gcell of each window with the extremal (i.e., maximum and minimum) values of selected parameters within the *expanded observation windows*.[19]

We use the *R* [248] statistical analysis package to prototype our predictor. As an illustration, the average true-positive and false-negative rates (over the 12 evaluations) for a series of evolving parameter sets are reported in Figure 3.3. We incrementally update the parameter set and mathematical model from predictor P1 through to predictor P9 based on our physical and statistical analysis of each of these predictors.

We use true-positive rate and false-negative rate to evaluate the statistical significance of prediction results. We compare the true-positive rates among combinations of different weighting[20] and the 12 evaluations for P{i} with or without a new set of parameters (unfriendly cells, sequential cells, etc.) If

[17]The DRV-clean cells are always weighted with one. We swept the weight of DRC-violating gcells with {2, 3, 4, 5,.... 10, 20, 30, 40, 50}.

[18]We pick 12 evaluations (larger than two times of 100%/20%) to avoid overfitting on specific training sets since we can only access one design in this technology.

[19]Two types of windows are mentioned in our discussion. The first type is the local window with two sizes (1×1 and 3×3 gcells) to extract per-layer GR information and other parameters. The second type is the expanded observation window with four sizes (3×3, 5×5, 7×7, 9×9) to keep track of max/min values within a certain range.

[20]We then choose the weight according to the true-positive rate for those runs with false-positive rates lower than a certain threshold (typically 0.5%).

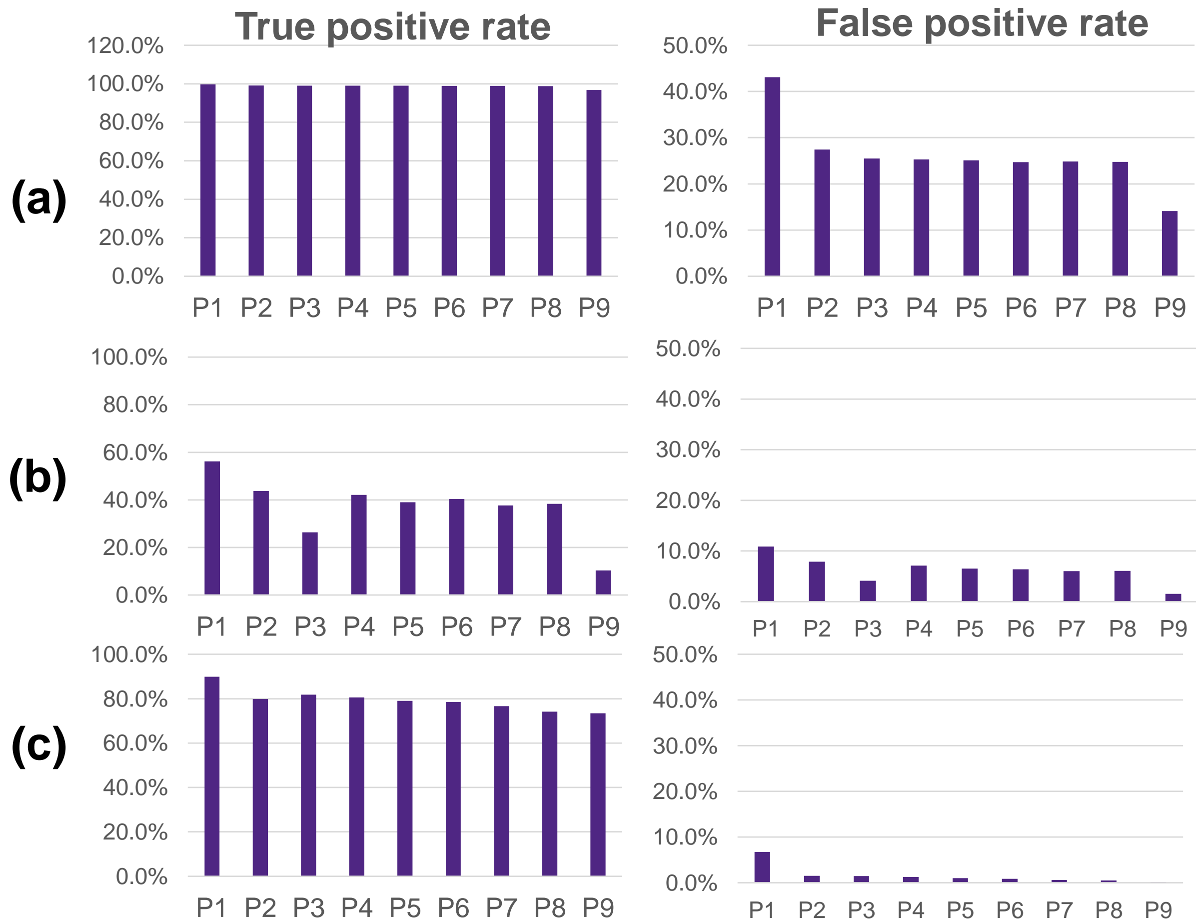


Figure 3.3: Accuracy comparisons between different parameter sets and mathematical models. (a) Linear regression; (b) logistic regression; and (c) SVM classifier.

improvement is observed in the true-positive rate with the same false-positive rate constraint (typically, 0.5%), we accept the P{i} with new parameters as the P{i+1} predictor.

In Figure 3.3, P1 is the baseline set of predictors with 1×1 local windows, including pin density, cell density, and per-layer (metals and vias) GR capacity, demand, and overflow. P2 to P3 are variations of P1 with 3×3 windows and combinations of 1×1 and 3×3 windows.[21] P4 to P8 are the baseline predictors, combined with pin proximity, number of multi-height cells, number of unfriendly cells, connectivity parameters, and structural parameters. P9 uses the *Leaps* [249] package in *R* to pick predictors with high correlation to DRVs.[22]

Table 3.1: Learning-based prediction: confusion matrix of our learning-based predictor. True-positive rate = 74% and false-positive rate = 0.2%.

		Actual	
		FALSE	**TRUE**
Prediction	**FALSE**	98571	117
	TRUE	170	344

We observe significant improvements in the prediction accuracy, especially in the false-positive rate, across this series. These plots also show that SVM (with a *radial basis function* (RBF) kernel) can provide better separation between DRV gcells and non-DRV gcells than linear or logistic regression models. We also plot the predicted DRV hotspots (red squares) and actual DRV hotspots (blue squares) in Figure 3.4. The contrast with the corresponding figure (Figure 3.1) for GR-based predictions is readily

[21] P1 to P3 are the same predictors with different local window sizes. We observe significant false-positive rate improvement when we compare P2 and P3 with P1 ($\sim 1\%$ vs. $> 6\%$), due to different local window sizes.

[22] Note that P1 essentially uses only cell and pin density parameters from global routing and thus has a high false-positive rate even if SVM is used ($\sim$6%). This again indicates that global routing is not sufficient to predict DRV hotspots.

Table 3.2: GR-based prediction: confusion matrix of prediction by GR shown in Figure 3.1. True-positive rate = 24% and false-positive rate = 0.5%.

		Actual	
		FALSE	TRUE
Prediction	FALSE	98260	350
	TRUE	481	111

apparent.[23] Our learning-based predictor provides a significantly more accurate prediction of the DRV hotspots, achieving 74% true-positive rate with a false-positive rate less than 0.2% (see Table 3.1). This is in contrast to a true-positive rate of 24% and a false-positive rate of 0.5% obtained from the GR-based predictor (Figure 3.1), as shown in Table 3.2.

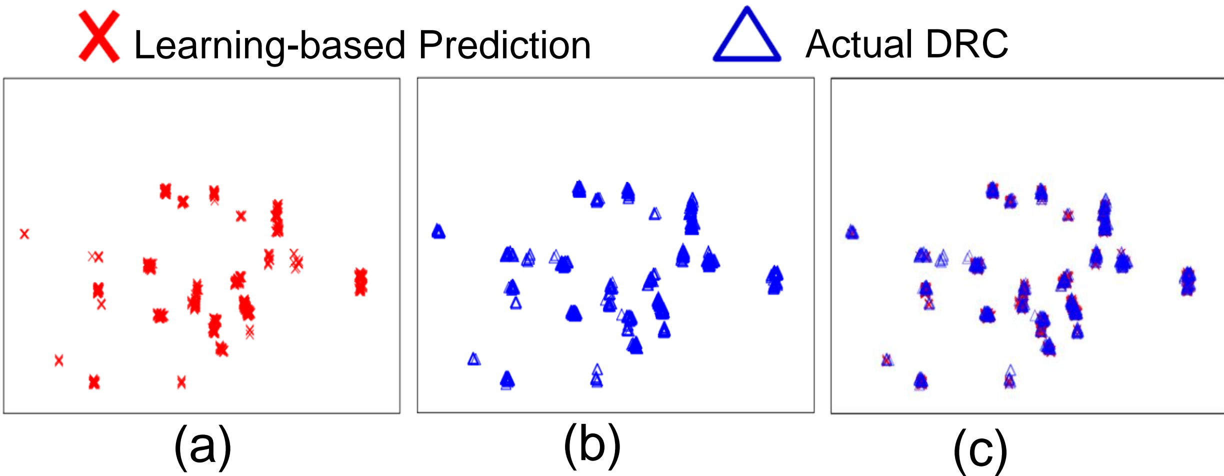


Figure 3.4: Comparison between our learning-based DRV hotspot map and the actual DRVs (gcells with DRVs) on a sub-14*nm* design. (a) DRV hotspots predicted by our learning-based model. (b) Actual DRVs after detailed routing (note that this is the same as Figure 3.1(b) presented earlier, and is reproduced here merely for comparison with the predicted map in (a)). (c) An overlay of the predicted and actual DRVs.

[23] Note that the learning-based model has an advantage over GR-based congestion map because it generates proper thresholds (i.e., support vectors) in the training stage.

3.3.3 Predictor-guided Routability Optimization

We present an optimization engine to improve the routability of a converged netlist with minimal perturbation of the layout. This algorithm is summarized in Figure 3.5. Given the placement of a timing-converged netlist, we first use our DRV prediction model to identify the potential DRV hotspots in that layout. We have developed a local whitespace optimization engine that redistributes the whitespace already present in the neighborhood of the predicted DRV hotspots in a way that improves the routability of these hotspots. Real-world physical implementation flows are almost invariably run under some form of local cell density constraints in order to improve the flow convergence. Such constraints ensure the existence of whitespace everywhere in the layout when measured at some level of spatial granularity. However, this granularity is typically much larger than that of individual cells; therefore, this default whitespace distribution is not always able to resolve detailed routing DRV problems. This shortcoming is addressed effectively in our work by introducing a new, detailed whitespace redistribution stage that relies on our hotspot prediction model to minimize the perturbation to the layout while maximizing the routability impact of the redistribution.

The spreading is constructed based on the legalizer. First, the available whitespace size is collected in a given window. Then, we distribute a certain fraction of the available whitespace by adding small temporary keepout regions adjacent to the cells in the window, and run the legalizer to increase the cell spacing. This step incrementally moves the cells from the initial locations obtained from the original converged layout. The legalizer tends to abut the cells in order to minimize the routed wirelength; the temporary keepouts help counter this behavior by introducing porosity in order to improve routability.

During the whitespace redistribution process, we first mark out local windows around each hotspot, and calculate the amount of whitespace available in these windows in order to compute a local whitespace budget (splitting overlapping windows appropriately during this process). We then incremen-

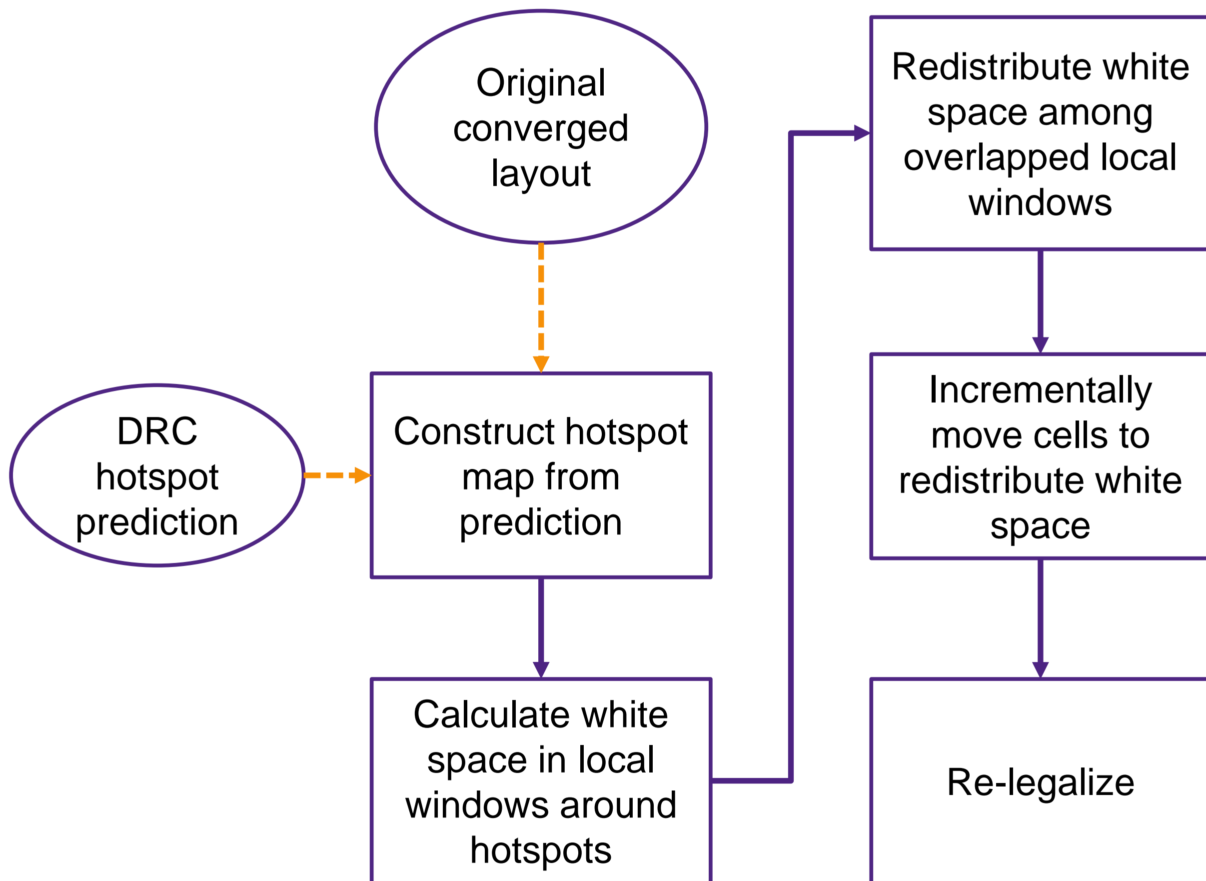


Figure 3.5: The predictor-guided routability optimization algorithm.

tally distribute this budget among the cells in a row-major scanning sequence by gradually adding the aforementioned keepout region instances within the windows until the space budget is used up.

Subsequent to this whitespace redistribution step, the detailed routing engine can use the newly introduced space between problematic cell instances to handle the complicated design rules. This approach of applying redistribution to only the potential DRV hotspots and avoiding the introduction of any new whitespace there, while leaving the rest of the layout untouched, minimizes the perturbation to the already-converged layout, which in turn minimizes the likelihood of incurring any significant timing penalty. Indeed, our experimental results (discussed in Section 3.4) demonstrate large routability improvements from the application of this algorithm, without hurting timing.

3.4 Experimental Setup and Results

In this section, we describe the experimental methodology that we have used to evaluate the effectiveness of our predictor-guided routability optimization algorithm. We then present the results of this evaluation.

Our experimental methodology is shown in Figure 3.6. Our routability optimization engine is implemented in *C++* as part of a state-of-the-art industrial physical implementation platform. The predictor model generation code is implemented using scripts in the *R* [248] statistical analysis package.[24] We evaluate our algorithm using an industrial benchmark design at a sub-14*nm* process node from a leading foundry. Given the difficulty of obtaining additional real-world sub-14*nm* benchmarks at this time, we generate multiple widely-differing netlists and layouts from our benchmark design by running the design through an industrial congestion-aware physical synthesis flow with different (but still realistic) placement and optimization settings.

[24]The extraction and training scripts are split across several machines to reduce turnaround time.

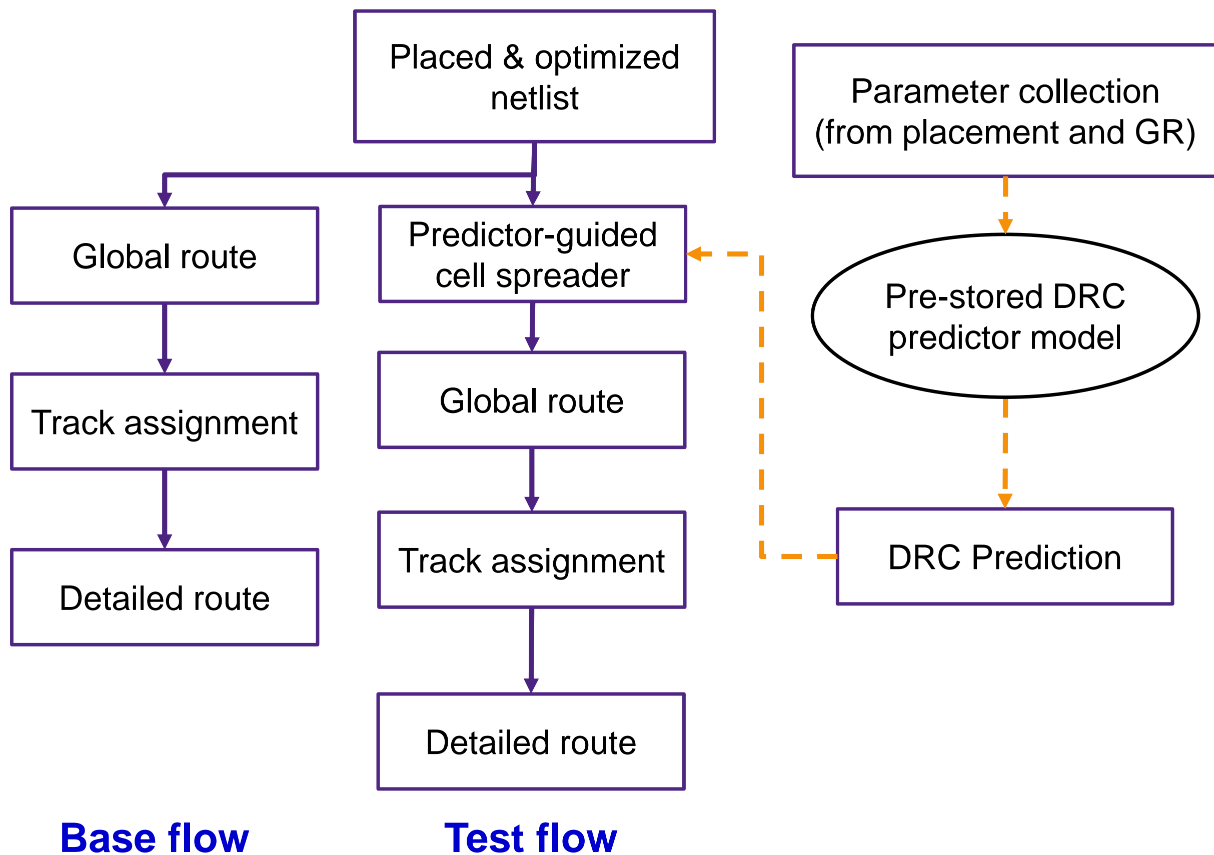


Figure 3.6: The experimental methodology used to evaluate the proposed predictor-guided routability optimization algorithm.

The input to our evaluation is an optimized netlist and a legalized placed layout obtained from the physical synthesis flow as described above. The layout has already been optimized using traditional congestion alleviation techniques during physical synthesis. Our base flow is the typical physical implementation flow that takes netlist and layout through standard global routing, track assignment and detailed routing using the state-of-the-art router embedded in our industrial physical implementation platform. For our test flow, we first use the pre-stored predictor to predict DRVs hotspots in our starting layout. This prediction is then used by our routability optimization engine for localized whitespace redistribution, followed by legalization. The resulting layout is fed to the same router as in the base flow.

We report the number of DRVs, total negative timing slack (TNS), wirelength, and number of

Table 3.3: Comparisons between the default and the learning-optimized layouts. There are under one million instances in this design.

	#DRVs			Wirelength (μm)			TNS (ns)			#FEPs		
	Base	**Test**	**%Change**	**Base**	**Test**	**%Change**	**Base**	**Test**	**%Change**	**Base**	**Test**	**%Change**
eg1	8478	1964	-76.8%	1742804	1747685	0.3%	-153.43	-158.4	3.2%	7289	7352	0.86%
eg2	1502	927	-38.3%	1750698	1753047	0.1%	-168.23	-163.5	-2.8%	7406	7374	-0.43%
eg3	2017	1819	-9.8%	1772889	1773701	0.0%	-215.75	-213.6	-1.0%	7817	7751	-0.84%
eg4	2026	1780	-12.1%	1735185	1735227	0.0%	-151.36	-149.6	-1.2%	7195	7143	-0.72%
eg5	4252	4255	0.1%	1831492	1836060	0.2%	-264.34	-275.6	4.3%	7865	7975	1.40%
eg6	3440	3891	13.1%	1790059	1794184	0.2%	-195.65	-203.5	4.0%	7587	7562	-0.33%
		Avg	-20.6%		**Avg**	0.2%		**Avg**	1.1%		**Avg**	-0.01%
		Max	13.1%		**Max**	0.3%		**Max**	4.3%		**Max**	1.40%
		Min	-76.8%		**Min**	0.0%		**Min**	-2.8%		**Min**	-0.84%

failing timing endpoints (#FEP) at the end of detailed routing in the base and test flows. The results are shown in Table 3.3.[25] In this table, negative values in the "%Change" columns refer to improvements achieved in the test flow (relative to the base flow), and positive values in these columns refer to degradations. As is evident from this data, we achieve significant reduction of the DRV count in the test flow. Moreover, these routability improvements are obtained without any significant impact on design closure: the timing and wirelength impacts are neutral, and the design area is unchanged. More specifically, we see that the number of DRVs reduces by an average of 20.6% and a maximum of 76.8%, with TNS degrading by an average of 1.1%, and the number of failing timing endpoints improving by an average of 0.01%, and wirelength degrading by an average of 0.2%.

3.5 Conclusions

In this chapter, we have addressed the route completion problem for designs at advanced process nodes. We make the case that traditional routability amelioration approaches that rely on GR-based congestion maps during physical synthesis are no longer effective at these advanced nodes due to the com-

[25] In our experiments, we do not retrain the model for these regenerated layouts.

plexity of the design rules. We quantify this difficulty by measuring the accuracy of a GR-based prediction of routability hotspots. We then propose a machine learning based algorithm to automatically improve the routability of these designs without hurting timing convergence. We evaluate the effectiveness of this algorithm on design layouts at a sub-14*nm* process node from a leading foundry, using an industrial physical implementation platform. Our experiments show that we are able to reduce the number of DRVs by an average of 20.6% and a maximum of 76.8%, with no adverse impact on design closure. Our future works include (i) improving the prediction accuracy and spreading results, (ii) guiding the routability optimization by applying the machine-learning model to coarse placement, and (iii) applying our methodology to other advanced node technologies.

3.6 Acknowledgments

Chapter 3 contains reprint of W.-T. J. Chan, P.-H. Ho, A. B. Kahng and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning", *Proc. ACM International Symposium on Physical Design*, 2017, pp. 15-21.

I would like to thank my coauthors Pei-Hsin Ho, Andrew B. Kahng, and Prashant Saxena. I also would like to express my gratitude to Thomas Andersen, Brent Gregory, Will Naylor, Siddhartha Nath and Jaime Wong for a number of valuable discussions, and to Jamie Wong for helping us set up the experimental framework.

Chapter 4

Design Space Exploration for 3DIC

This chapter presents a design space exploration methodology for monolithic 3DIC design. 3DIC is a promising design technology to extend the Moore's-Law scaling. However, due to the lack of golden 3DIC design flows, designers are unable to efficiently evaluate the achievable benefits of implementing existing designs in 3D. We propose our "infinite dimension" concept to evaluate 3D benefits of a given design. We observe that the 3D benefit correlates well with the designs' Rent exponents, which indicate the complexity of interconnect. Based on this observation, we heuristically modulate designs' Rent exponents during synthesis to improve the 3D benefit.

4.1 Background of 3DIC

Three-dimensional integrated circuits (3DIC) with multiple tiers is a promising technology in the "More-than-Moore" era to integrate more functionality with greater bandwidth and less power. Many previous works propose 3DIC optimization approaches to achieve better design quality over conventional planar implementations. Further, due to higher integration and reduced wirelengths, 3DICs with more

than two tiers are expected to offer larger benefits (e.g., less power). A recent work [204] shows that 3DICs with three tiers achieve 15% more power reduction as compared to corresponding two-tier 3DIC implementations and 36% power reduction as compared to 2D implementations. A much smaller body of work addresses the fundamental question of predicting 3DIC benefits over conventional 2D implementation, and upper-bounding these benefits. Chan et al. in [40] derive a 67% upper bound of wirelength reduction from a two-tier 3DIC over 2D designs. However, no previous works propose *upper bounds* on the power and total cell area reductions achievable by 3DICs over 2D designs.[26]

In this chapter, we revisit the 3DIC benefits in terms of power and area with multiple tiers. More specifically, we propose the concept of *implementation in infinite dimension* (that is, where all gates can be placed as close as possible – essentially, adjacent – to each other) to derive an upper bound on 3D power and area benefits for given design, technology, and tool/flow. Such implementation in infinite dimension is achieved by synthesis and netlist optimization with zero *wireload model* (0-WLM).[27] Our studies show that the power benefits can only be 18% for particular designs, even with an infinite-dimensional layout resource. Moreover, we study the maximum potential (performance, power, area/cost) benefits of 3DICs. None of our testcases is able to achieve more than (10%, 10%, 10%) improvement from 3D integration, a somewhat disappointing observation relative to the hoped-for benefits from 3DIC. We further evaluate design power across various dimensions (i.e., pseudo-1D, 2D, 3D with multiple tiers). We observe that design power sensitivity to different implementation dimensions correlates with Rent parameters of netlists, especially placement-based Rent parameters. Based on this observation, we suggest that netlist synthesis should be aware of the implementation dimension so as to minimize design power.

[26] 3D benefits refer to the power and total cell area reductions as well as frequency improvements achievable by 3DICs over 2D designs.

[27] We note that the upper bound on 3D benefits is specific to the given optimization tool/flow. However, since we use a leading-edge commercial P&R tool in our experiments, we believe our estimated upper bound is not far from the true upper bound.

We summarize our contributions as follows.

- We propose the concept of implementation with *infinite dimension* (i.e., netlist optimization with 0-WLM), based on which, we study the upper bound on power and area benefits of 3DICs.
- We show that upper bounds on 3DIC power and area benefits can be quite small – at most 39% power reduction and 10% area reduction even with infinite dimensions.
- We perform 3D benefit estimation across various technologies and compare the 3D benefits versus the benefits from an improved P&R tool/flow in 2D implementation.
- We study the maximum potential (performance, power, area/cost) benefits of 3DICs. Our results indicate that it is typically difficult to achieve a simultaneous (10%, 10%, 10%) improvement of (performance, power, area/cost) from pure logic-logic 3D integration.
- We study design power sensitivity to various implementation dimensions (i.e., pseudo-1D, 2D, 3D with different tier numbers and infinite dimension) and show the empirical evidence of a correlation between the power sensitivity and the Rent parameter of the netlist.
- We suggest that there is potential benefit from netlist synthesis optimizations being aware of the implementation dimension.
- Using a simple example with macros and/or blockages, we show that 3D benefits can be large for (block-based) SoC designs.

The remainder of this chapter is organized as follows. Section 4.2 reviews related works on 3DIC optimization and prediction of 3DIC benefits. Section 4.3 describes our implementation flows as well as design power and area estimation flows in different dimensions. In Section 4.4, we describe experimental

setup and results that quantify 3DIC power benefits and power sensitivity to dimensions. Section 4.5 concludes and gives directions for ongoing work.

4.2 Related Works

Various approaches have been proposed for implementation and optimization of 3DICs. Table 4.1 summarizes area, power and wirelength benefits reported in previous works. In the table, "—" indicates "not applicable", i.e., not addressed in the cited work. We note that although we show total cell area reported by previous works in the table, area reduction is typically not the major objective in 3DIC optimization. Instead, most of these works use wirelength reduction as their major design objective [18] [65] [66] [69] [70] [120] [122] [127] [180] [182].

We first review previous works for integration and optimization of 3DICs. Liu et al. [153] propose transistor-level 3D monolithic integration. Bobba et al. [22] propose a cell-mapping and placement flow for monolithic 3D. Thorolfsson et al. [209] propose a 3D placer based on mPL. Lim [149] obtains power benefit by studying the capacitance reduction in *through-silicon via based* (TSV-based) 3D implementations. Song et al. [204] propose a block-level folding approach and show corresponding power benefits of three-tier stacking. Chang et al. [48] apply the flow [179] to 7nm technology node. Nayak et al. [171], Athikulwongse et al. [15], and Panth et al. [178] study the power benefits from various 3D integrations (e.g., monolithic 3D, mini-TSV, and TSV-based integration). Song et al. [203] study power reduction with consideration of the power distribution network. Jung et al. [105] [106], Lee et al. [142], and Ok et al. [173] achieve power benefits by applying block-level integration to the *OpenSparcT2* processor, a multicore GPU, and a stereo image processor.

Estimation of 3D power benefits has also drawn much attention. Priyadarshi et al. [186] propose an architectural framework to estimate 3DIC power for design space exploration. Kim et al. [124] [123]

Table 4.1: Summary of 3D benefits studied in previous works.

Work	% Benefit (area)	% Benefit (power)	% Benefit (WL)	# tier
[15]	7%	9%	WL increases	2
[18]	—	—	50%	2
[22]	—	15%	13%	2
[40]	—	39%	—	2
[48]	—	17%	46%	2
[65]	—	—	54%	4
[66]	—	—	50%	2
[69]	—	—	30%	2
[70]	—	—	26%	3
[105]	8%	21%	26%	2
[106]	—	20%	9%	2
[120]	—	—	20%	2
[122]	—	—	4%	2
[124]	—	—	37%	2
[123]	—	28%	41%	4
[125]	—	23%	28%	4
[127]	—	—	32%	2
[142]	—	22%	—	2
[149]	—	3%	25%	2
[153]	—	7%	16%	2
[143]	—	37%	48%	2
[171]	—	37%	—	2
[173]	—	13%	14%	2
[178]	—	35%	33%	2
[180]	—	—	19%	2
[182]	—	—	40%	2
[186]	—	20%	—	2
[203]	—	19%	—	2
[204]	3%	36%	42%	3
[209]	—	13%	21%	2

[125] propose models to estimate wirelength and power reductions based on buffer insertion. Chan et al. [40] propose a machine learning-based methodology to estimate power benefits of (3DIC) design

flows, and apply their methodology to the flow proposed by Panth et al. [179]. However, no existing work studies potential *upper bounds* on 3DIC power and area benefits. To our knowledge, ours is the first work to address this gap.

In this chapter, we also examine the correlations between netlist properties and 3D power benefits. Our studies suggest that netlist synthesis could benefit from awareness of implementation dimensions. Previous works which study the relationship between netlist structures and placement and routing (P&R) quality of results (QoR) include those of Liu and Marek-Sadowska [154] [155], which propose metrics to predict 2D P&R wirelength from netlist structure. They also propose optimization during the technology mapping stage of logic synthesis to improve 2D P&R results. Rahman et al. [193] propose a low-power gate-sizing scheme using a rich library with complex and large-size cells for logic synthesis, and a library with only simple cells for P&R. Seo et al. [198] argue that the benefit of using complex cells in advanced nodes will diminish due to routing congestion. However, [198] does not consider how 3D integration might mitigate the routing congestion seen in a 2D design implementation. (We consider this aspect of 3D integration below.)

4.3 Methodology for 3D Benefit Estimation

We now describe our implementation and benefit estimation flows across various dimensions – pseudo-1D, 2D, 3D with multiple tiers, and infinite dimensions.

4.3.1 Pseudo-1D Implementation

To estimate the power penalty of design implementations in a limited dimension, we propose *pseudo-1D implementation*, that is, design implementation with a high aspect ratio layout. In this chapter, we refer to an implementation with layout aspect ratio 0.1 (block height equal to block width / 10) as a pseudo-1D implementation.

4.3.2 Optimal 2D Implementation

We pursue an "optimal" 2D implementation so as to quantify the true benefits from 3D integration with multiple tiers and from implementation in infinite dimension. To achieve this, we obtain multiple conventional planar implementations by sweeping several key parameters such as synthesis clock period, placement utilization and BEOL stack options; we then select the best (e.g., minimum power) outcome. Figure 4.1 shows an example where we vary the synthesis clock period and placement utilization for different 2D implementations. We observe that when the design has tight timing constraints, slightly smaller synthesis clock period eventually leads to smaller power after placement and routing. On the other hand, for a design with loose timing constraints, slightly larger synthesis clock period results in smaller design power after routing.

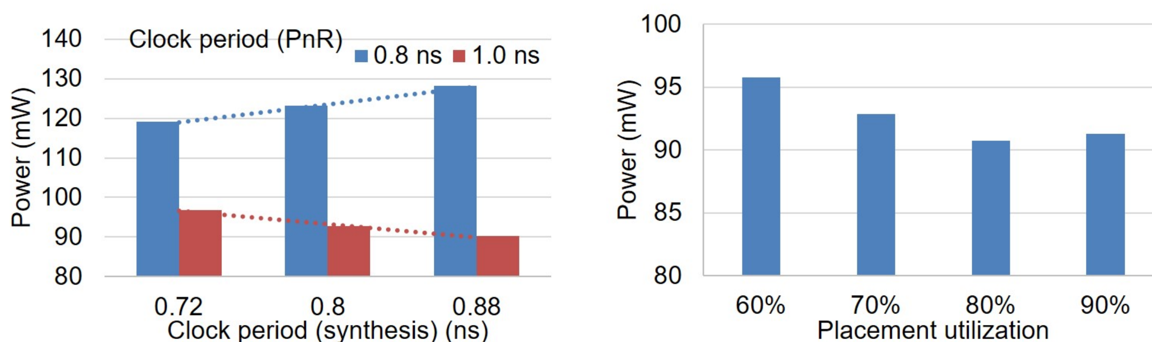


Figure 4.1: Design power with (a) varying synthesis clock period, and (b) placement utilization. Design: *JPEG*. Technology: 28*nm* FDSOI.

Furthermore, design power versus placement utilization exhibits a roughly unimodal behavior. In other words, if a placement is too compact, the power increases due to routing congestion (detouring, crosstalk, etc.). On the other hand, if the placement is too sparse, the power again increases due to longer wirelength (larger wire capacitance). Regarding BEOL stack options, we vary the number of layers from six to 11 in our experiments and compare the resultant design power after routing. However, we observe that the power variation is quite small (e.g., less than 3% for design *JPEG*). This is because the six-layer

stack is able to offer enough routing resources for the designs and technology used in our experiments. We therefore implement our design testcases by varying (i) synthesis clock period (e.g., 0.9×, 1.0× and 1.1× of clock period used in P&R) and (ii) placement utilization (e.g., 60%. 70%, 80%, 90%), then selecting the outcome with minimum power consumption.[28]

We also apply such implementation flows to other dimensions to obtain (as far as we are able) fair comparisons.

4.3.3 Power Benefit Estimation for 3DICs

Given that there is no "golden" 3DIC implementation flow, we propose an implementation flow, based on the conventional 2D implementation tools, to achieve an optimistic estimation of design quality of 3DICs with multiple tiers. With the Shrunk2D flow proposed in [179] as a starting point, we perform the flow described in Algorithm 2 to estimate 3DIC benefits with multiple tiers.[29] To estimate the reduction of wire parasitics from shrunk footprint area, we shrink the cell LEF by a factor of $1/\sqrt{T}$ in both height and width, where T is the number of tiers. We also apply the same scaling ratio to shrink the BEOL LEF to ensure that routing resources remain adequate. We implement designs based on the shrunk LEF (Line 1). With the shrunk LEF implementations, we then divide the die area into M x M grids. For each grid, we perform iterative min-cut partitioning to divide the cells within the grid into T clusters which are assumed to be placed on T tiers (Line 3). Details of our partitioning procedure are described in Algorithm 3. In the procedure, we iteratively apply min-cut partitioning based on Fiduccia-Mattheyses

[28] Our separate studies perform implementations with fine-grained choices of synthesis clock period (e.g., 0.8× to 1.2× with a step size of 0.05× of the P&R clock period) and placement utilization (e.g., 50% to 90% with a step size of 5%). Results for the *JPEG* testcase show that the optimal solution (i.e., solution with minimum power) found with fine-grained parameter sweeping is only 2% better than the solution found with our reported methods. We also attempt to vary the synthesis utilization (i.e., the utilization of floorplan as input to physical synthesis). However, experimental results show that for the technology and design testcases studied, sweeping of synthesis utilization does not lead to any power benefits as compared to a flow without physical synthesis.

[29] We have Shrunk2D flow from [179]. Since the flow runs on an old version of P&R tool (i.e., *Cadence SoC Encounter vEDI10.1*) where the results are even worse than the 2D implementation with *Cadence Innovus Implementation System v15.2*, we do not use the Shrunk2D flow for 3D benefits estimation.

(FM) algorithm to partition the cells into two parts with area ratio of $(T-k)$ where k is the iteration index. The min-cut bipartitioning is performed with MLPart [244]. After each partitioning, the cells from the smaller-area part will be assigned to a new tier. We then collapse the cells in the smaller-area part into one super cell, fix the super cell in the first partition, set its area to zero and continue with the partitioning procedure. The iterative partitioning optimization terminates when all cells are assigned to a tier. Based on the partitioning solution, we annotate parasitics to nets which have cells from different tiers. More specifically, we calculate the maximum delta in tier depth across all cells connected to the net, and for each unit of (delta in tier) depth, we annotate RC corresponding to one TSV and vias across six metal layers (Line 4). With the annotated RC, we perform incremental sizing, VT-swapping and buffer insertion optimization to fix timing violations.

Note that in our estimation flow, we can estimate the benefits from wire parasitic reduction in 3DICs. In addition, we ignore the potential performance and power penalties from placement legalization (when we allocate cells to different tiers), from routing congestion caused by cross-tier power delivery, from difficulties in clock tree synthesis in a 3DIC, and from additional die-to-die variability margin. Our proposed flow therefore provides an optimistic estimation of 3DIC design qualities.[30] In our study, we also attempt to back-annotate placement solution with shrunk LEF to physical synthesis optimization. However, our experimental results show that such a back-annotation loop does not lead to further power benefits in the routed design.[31] We therefore apply a one-pass flow in our experiments.

[30]Due to the lack of production-quality 3D design tool/flow, we are unable to precisely capture the penalties from routing congestion, cross-tier power delivery networks, clock tree synthesis, etc. In our estimation flow we do not consider area impact of vertical interconnects (e.g., through silicon vias), as in [160]. This results in an optimistic estimation of 3D *area* benefits, with degree of optimism dependent on area overheads of vertical interconnects.

[31]Based on our experience, physical synthesis typically improves maximum performance when the clock constraints are tight. However, due to the pessimism of wireload in the physical synthesis, we rarely observe power reduction from physical synthesis optimization.

Algorithm 2 Evaluation of 3DIC benefits with T tiers.

1: Perform 2D implementation with shrunk LEF (scaling ratio = $1/\sqrt{T}$ in cell height and cell width)
2: Divide die area into M x M grids uniformly
3: Apply FM-based min-cut partitioning for T tiers
4: Annotate RC according to tier assignment based on partitioning solution
5: Perform incremental optimization
6: Report design power

Algorithm 3 FM-based min-cut partitioning for T tiers.

Input: Netlist N, number of tiers T
Output: Subnetlists $Sol = \{N_1, N_2, ..., N_T\}$
1: $Sol \leftarrow \emptyset$
2: **for** $k = 1 : T-1$ **do**
3: $area_1 \leftarrow N.area/T$
4: $area_2 \leftarrow \frac{T-k}{T} \cdot N.area$
5: $\{N_k, N_{k+1}\}$ = 2WayMinCutPart(N, $area_1$, $area_2$)
// tolerance = 5%
6: $N \leftarrow$ collapse N_k into one super cell c_k
7: $c_k.area \leftarrow 0$
8: fix c_k in the first partition
9: $Sol \leftarrow Sol \cup \{N_k \setminus \{c_1, c_2, ..., c_k\}\}$
10: **end for**
11: **return** Sol

4.3.4 Implementation in Infinite Dimension

To estimate the upper bound on power and area benefits from 3DICs, we propose the concept of implementation with "infinite dimension", where we ignore wire parasitics during the implementation. To achieve this, we perform netlist optimization with zero wireload model (0-WLM).[32,33] Given that benefits from 3D integrations mainly come from the reduced wire parasitics in a shrunk footprint area, such implementation with infinite dimension is able to provide an upper bound on 3DIC benefits.

[32]To ensure a fair comparison to implementations at 2D and 3D, we perform netlist optimization with the same synthesis, placement and clock tree synthesis tool/flow but with 0-WLM and without any routing. Specifically, we use *Synopsys Design Compiler vH-2013.12-SP3* [251] to synthesize the netlist, and use 0-WLM during the synthesis; we use *Cadence Innovus Implementation System v15.2* [235] to perform placement and clock tree synthesis, and scale the interconnect RC by a very small number (i.e., 10^{-6}) during the placement and clock tree synthesis.

[33]Since our testcases are not hold-critical, the number of hold buffer insertions is negligible even when testcases are implemented with infinite dimension. To achieve an upper bound on 3DIC benefits, one might need to disable hold timing optimization during the implementation with infinite dimension.

4.4 Experimental Setup and Results

We perform experiments in a 28*nm*/12-track FDSOI foundry technology with dual-VT libraries, and 0.85V nominal supply voltage. We perform experiments on an *ARM CORTEX M0* core, three design blocks (*AES*, *JPEG*, *VGA*) from *OpenCores* [247], and *LEON3MP* from the *ISPD-12 benchmark* suite [175]. Parameters of these five testcases are shown in Table 4.2. For each design, we determine a range of clock periods starting from a clock period with relative loose timing constraint, up to the clock period which is close to the minimum clock period of the given design and technology. These designs are synthesized using *Synopsys Design Compiler vH-2013.12-SP3* [251] and then placed and routed using *Cadence Innovus Implementation System v15.2* [235]. We further use *Cadence Tempus Timing Signoff Solution v15.2* [237] for timing and power analysis, with wire parasitics (SPEF) obtained from Innovus.

Table 4.2: Summary of benchmarks used to evaluate 3D benefits.

Design	#Instances	#Flip-flops	Clock period range
CORTEX M0	∼9K	840	0.8*ns* - 1.0*ns*
AES	∼12K	530	0.6*ns* - 0.9*ns*
JPEG	∼43K	4712	0.8*ns* - 1.1*ns*
VGA	∼72K	17057	0.7*ns* - 1.0*ns*
LEON3MP	∼460K	108817	1.1*ns* - 1.3*ns*

4.4.1 Evaluation of 3D Benefits

We first compare design power and total cell area across various implementation dimensions and different clock periods. Figure 4.2 shows the power and area comparison. All the implemented designs have no hold violation and a setup violation less than 10*ps*. We observe that the maximum power benefits (i.e., the gap between the red curve versus the orange curve) from implementations in infinite dimension are respectively 36%, 39%, 20%, 18% and 26% for *CORTEX M0*, *AES*, *JPEG*, *VGA* and *LEON3MP*. The results show a large variation of 3D benefits across different designs. In addition, the power benefits from

3D integration with two, three and four tiers are less than 10% for designs *JPEG*, *VGA* and *LEON3MP*. Furthermore, we observe that the area benefits are small (i.e., $< 10\%$ for all designs, and $< 4\%$ for designs *JPEG* and *VGA*).

We further assess the upper bounds on potential PPAC (performance, power and area/cost) improvements from 3D integration based on our concept of infinite dimension.[34] Specifically, we implement designs in infinite dimension and sweep the clock period from a relatively large value (e.g., the maximum clock period shown in Figure 4.2) to the minimum achievable clock period, with a step size of 50*ps*. We also implement designs in infinite dimension at a higher supply voltage (i.e., 0.95V) to explore the power-area tradeoff (i.e., more area benefits at the cost of larger power). We then compare the performance, power and area of the implementations in infinite dimension versus those of our "best possible" 2D implementations, and obtain tuples of potential maximum (performance, power, area/cost) benefits.

Figure 4.3 shows the (performance, power, area/cost) tuples, where the results in the left column use low-frequency 2D implementations (i.e., clock periods = {1.1*ns*, 1.1*ns*, 0.9*ns*, 1.0*ns*, 1.4*ns*} for designs {*CORTEX M0*, *JPEG*, *AES*, *VGA*, *LEON3MP*}) as references; and the results in the right column use high-frequency 2D implementations (i.e., clock periods = {0.8*ns*, 0.8*ns*, 0.6*ns*, 0.7*ns*, 1.1*ns*} for designs {*CORTEX M0*, *JPEG*, *AES*, *VGA*, *LEON3MP*}) as references. Furthermore, we only show data points with all non-negative values in the tuple. Figure 4.3 shows that for a single metric, the maximum improvement of performance, power or area can be ~40%, ~40% or ~10%, respectively, without degradation on the other two metrics. However, no design is able to achieve (10%, 10%, 10%) benefits from 3D integration, and only one design (i.e., *CORTEX M0*) shows more than (10%, 10%, 5%) benefits. Moreover, we observe that the area benefits from 3D integration are typically small (i.e., $\leq$5% for most

[34] For "area/cost", we mean "area or cost". We assume that the cost is measured as area.

of the data points); this matches the results in Figure 4.2. These results may indicate a limited upside of pure logic-logic 3D integration.

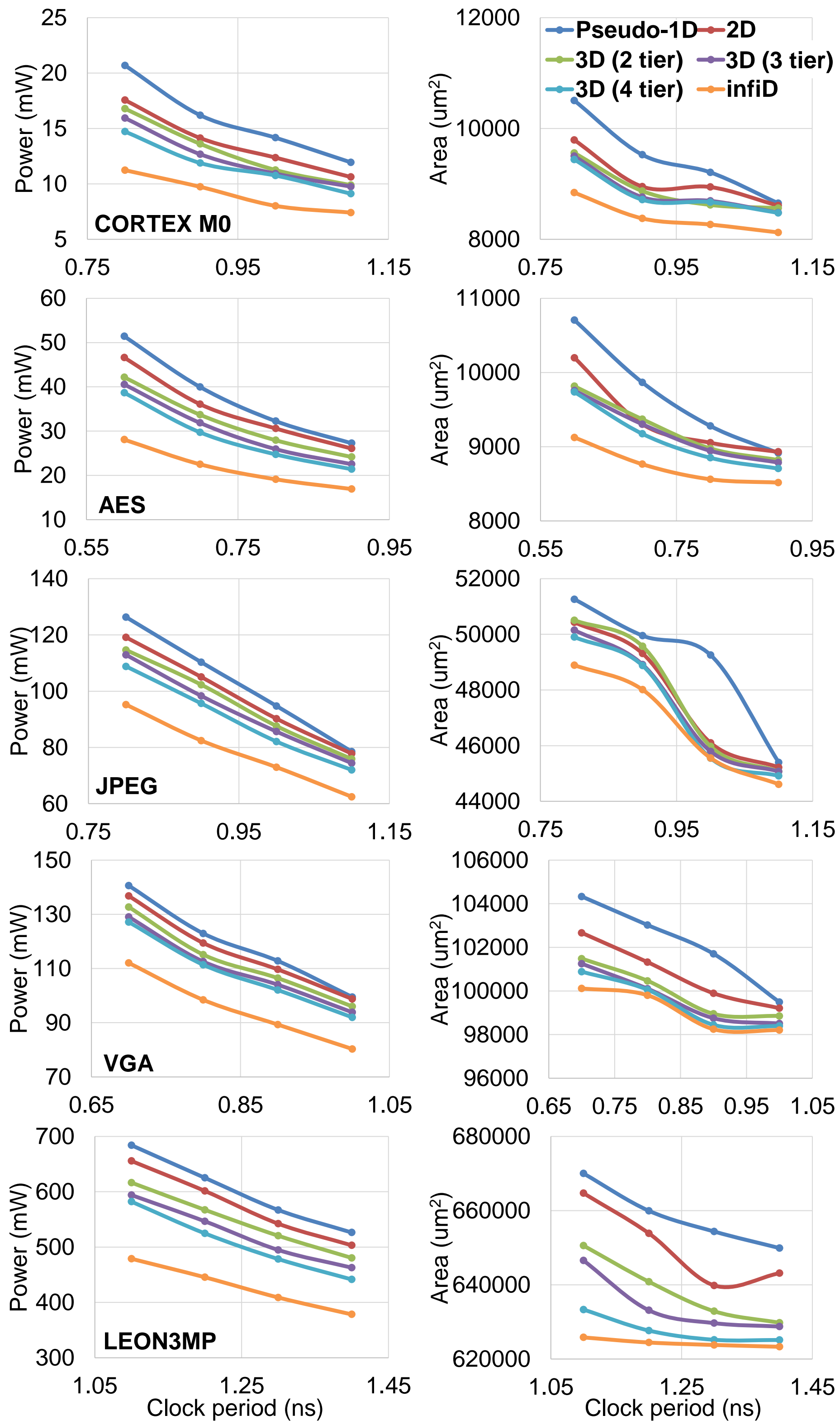


Figure 4.2: Design power and total cell area evaluated across various implementation dimensions.

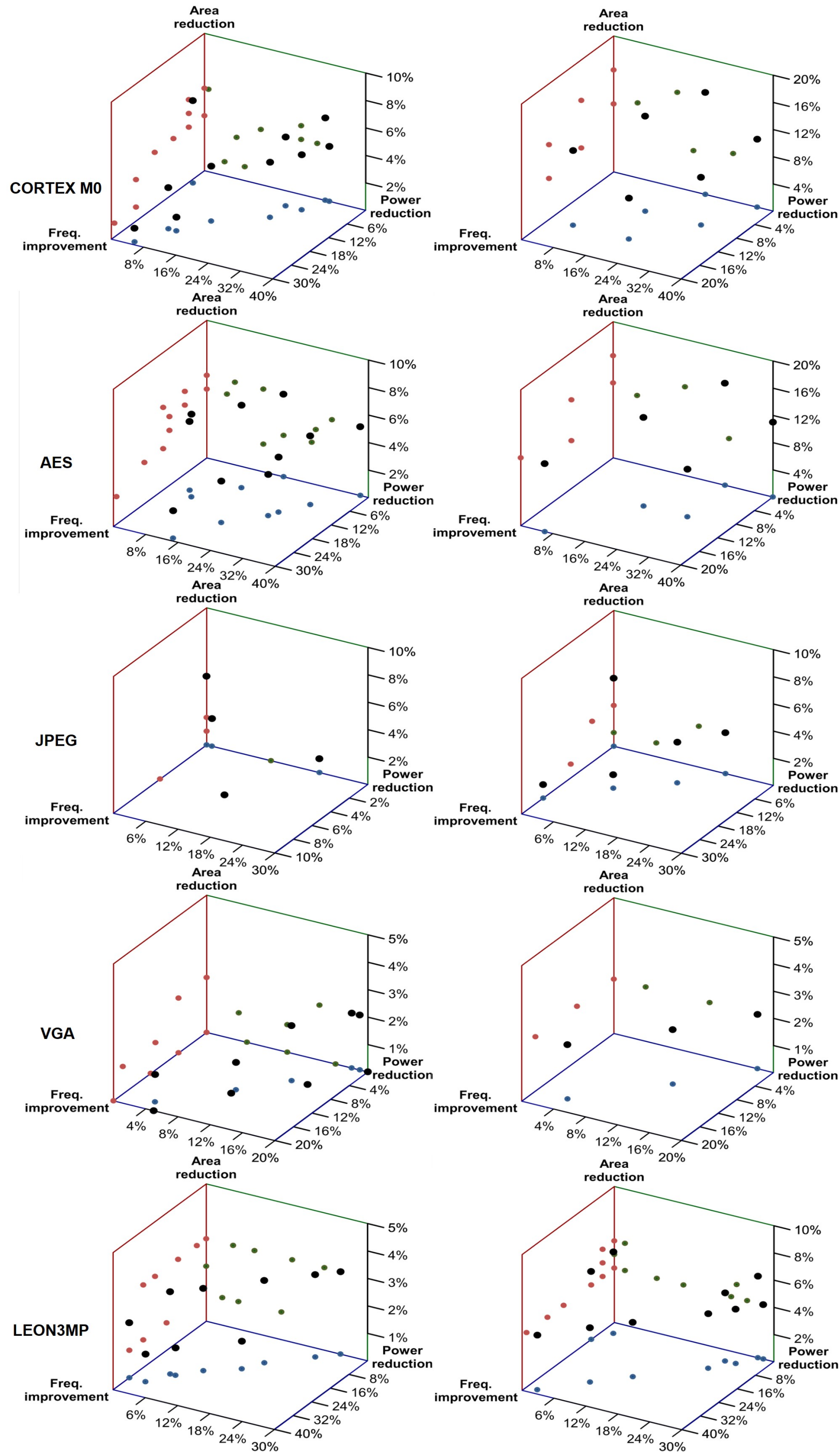


Figure 4.3: Maximum potential (performance, power, area/cost) improvements from 3D integration. Left: Using low-frequency 2D implementations as references. Right: Using high-frequency 2D implementations as references. Blue, red and green dots are respectively projections on performance-power, performance-area and power-area planes.

4.4.2 A More Realistic Evaluation

As discussed in Section 4.3, our 3D power estimation ignores potential larger clock skew due to inter-tier process variation [224]. To achieve a more realistic estimation of 3D benefits, we quantify the impact of clock skew on 3D power reduction. In our experiments, we enable multi-corner optimization by using both slow- and fast-corner libraries during the P&R stage. We further model potential clock skew increase due to difficulties in 3D *clock tree synthesis* (CTS) as well as inter-tier process variation by applying 0%, 5% and 10% clock uncertainties of the clock periods. The power benefits against the clock uncertainties are shown in Figure 4.4. The results show that the 3D power benefits diminish when the clock uncertainties increase from 0% to 10% even for two designs which originally have the largest 3D benefits among our benchmarks. More specifically, the power benefit of a two-tier 3D implementation decreases from 11% to 1% for *AES* and from 5% to -21% for *CORTEX M0*. Our observation indicates that it is critical for 3D clock tree optimization to minimize the impact of inter-tier variation on clock skew and latency.

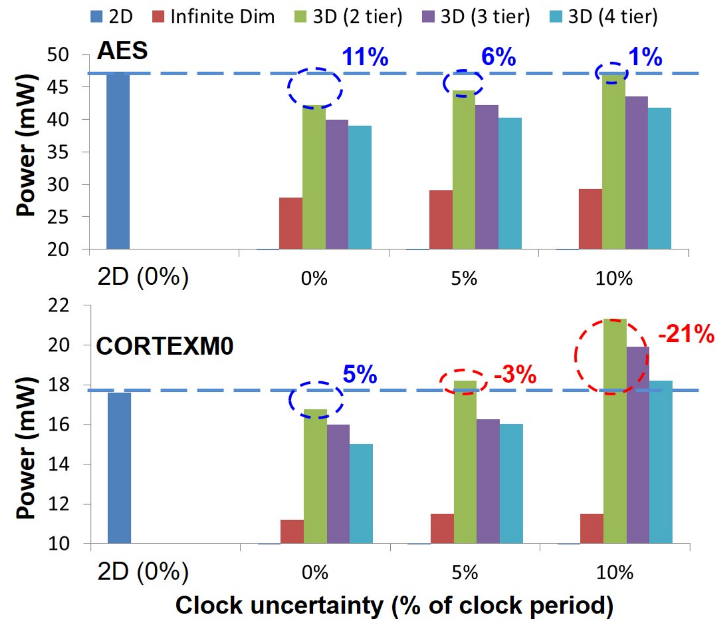


Figure 4.4: The impact of larger clock skew (due to complexity of 3D CTS as well as inter-tier variation) on 3D power benefits.

4.4.3 Comparison Between Improved 2D versus 3D

In this section, we study the power and area benefits from an improved P&R tool/flow in the conventional 2D implementation, and compare these benefits versus the estimated 3D benefits. Figure 4.5 shows the normalized power and area values of the 2D implementation using the latest version of a commercial P&R tool (i.e., *Cadence Innovus Implementation System v16.1*) (2D+) and the estimated 3D benefits with two tiers based on *Cadence Innovus Implementation System v15.2* (3D (2 tier)), with respect to 2D results using *Cadence Innovus Implementation System v15.2*. We observe similar power reductions from the improved 2D P&R tool/flow and 3D integration with two tiers. Quite interestingly, for particular designs (e.g., *CORTEX M0* and *LEON3MP*), the power benefits from the improved P&R tool/flow are even lower than the estimated 3D benefits. The results may indicate that even one EDA company's year-to-year improvement is of similar magnitude to 3D benefits for particular designs. Figure 4.5 further shows that the area benefits from both the improved P&R tool/flow and 3D integration are small in our results.

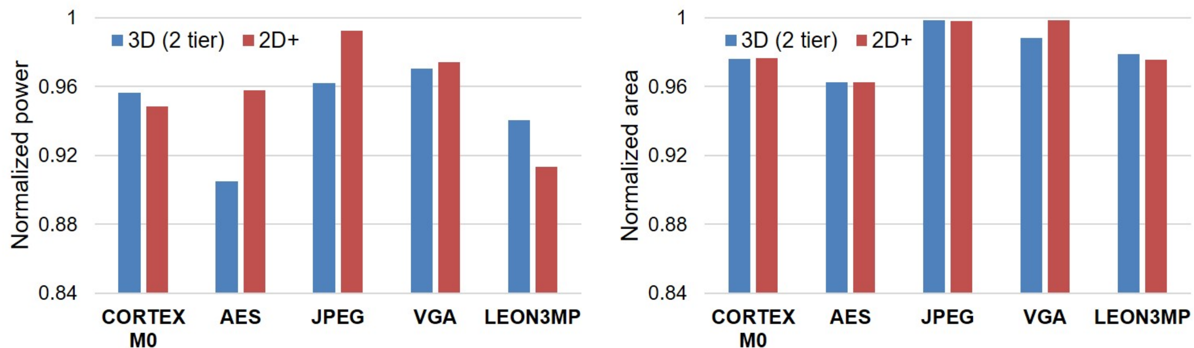


Figure 4.5: Comparison between estimated 3D power and area benefits with two tiers versus power and area reductions from the latest version of a commercial P&R tool (i.e., 2D+).

4.4.4 Assessment of 3D Benefit Across Technologies

We further assess the upper bound on 3D benefits across different technologies. Figure 4.6 shows the power and area benefits estimated in infinite dimension in 28*nm* FDSOI, 28*nm* LP (with 8T and 12T cells) and 45*nm* GS technologies. We use tight timing constraints with respect to each technology in our implementation. We observe consistent results across different technologies – (i) benefits on *CORTEX M0* and *AES* are relatively higher than those on other designs, and (ii) area benefits are typically smaller than power benefits (especially on *JPEG* and *VGA*). Moreover, we observe larger benefits in technologies with weaker driving strength (i.e., 28LP with 8T cells), where the wireload impact is larger on cell area and power.

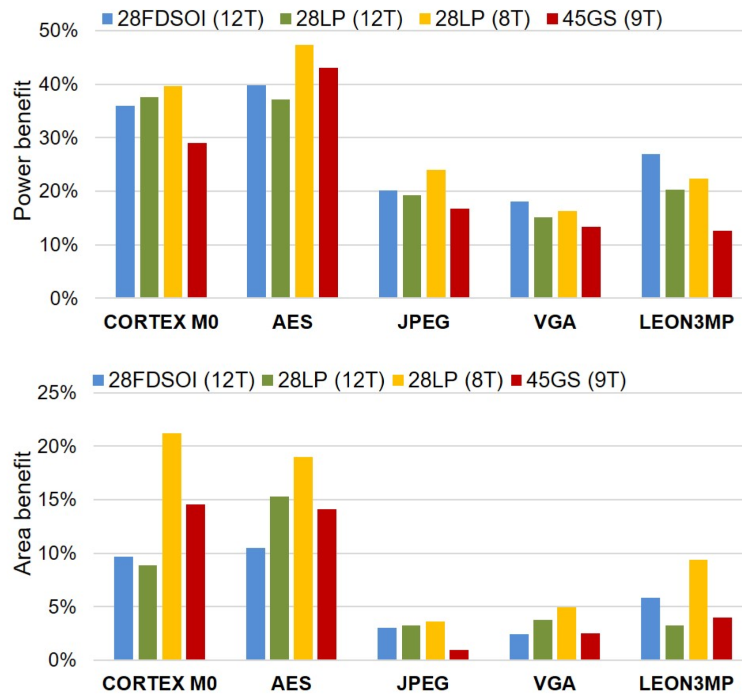


Figure 4.6: 3D benefits assessed in infinite dimension showing consistency across various technologies.

4.4.5 Netlist Study

We additionally study the possibility of correlations between (3D) power benefits and various netlist parameters (such as fanout distribution, slack distribution, sequential graph, Rent parameter, etc.) of designs. We observe that the power benefits are well correlated with Rent parameters.

We use the Rentian circuit generator *gnl* by Stroobandt et al. [207] to generate netlists with different Rent parameters, and we evaluate these netlists' power consumption across various implementation dimensions. The inputs to *gnl* are (i) number of cells, (ii) the target Rent parameter, (iii) the ratio between flip-flops and combinational cells, and (iv) the maximum path delay constraint. The *gnl* software starts with a set of standard cells and randomly inserts connections among the cells to form logic cones. The *gnl* software determines number of pins (of standard cells) and input/output terminals according to the user-specified Rent parameter. During the netlist generation, *gnl* recursively clusters logic cones to form larger ones. The number of terminals on the boundaries of the merged logic cones also follows the specified Rent parameter. The generated netlists thus have desired Rent parameters by construction.[35] Table 4.3 summarizes our generated testcases. We generate netlists using cells from the foundry 28*nm* 12-track FDSOI library and implement them using the flows described in Section 4.3.3. The initial generated netlists (with different Rent parameters) have similar power and area (i.e., within 3% difference) to help establish a fair comparison of power benefits across various Rent parameters. We define timing constraints such that the initial generated netlists have negative slacks, thus inducing non-trivial P&R optimizations.[36] Furthermore, to maintain a similar Rent parameter throughout the P&R flow (i.e., avoiding netlist restructuring), we apply a size-only restriction to all cell instances during the P&R optimization flow.

[35] We constrain *gnl* to instantiate equal numbers of DFFX8, INVX8, BUFX8, AND2X8, NAND2X7, OR2X8, NOR2X7, NAND3X12, NOR3X13 and XOR2X8 cells in the generated netlists.

[36] The *gnl* software constrains the maximum delays of the generated netlists by limiting the depths of the logic cones (i.e., inserting flip-flops at the boundary of the logic cones).

Table 4.3: Summary of Rentian testcases with different Rent parameters.[37]

Rent (input / actual)	Power (mW)	Area (μm^2)	Slack (ps)
0.50 / 0.63	46.4 (100%)	39552 (100%)	-72
0.55 / 0.66	46.8 (101%)	40262 (102%)	-74
0.60 / 0.69	46.7 (101%)	40404 (102%)	-68
0.65 / 0.71	47.4 (102%)	40532 (102%)	-110
0.70 / 0.74	46.9 (101%)	40607 (103%)	-73

Figure 4.7 shows the relationship between post-P&R power and netlist Rent parameter across various implementation dimensions. We observe that the power of the conventional 2D implementation increases with higher Rent parameters, and that the power increase (with Rent parameter) is smaller with 3D implementations. This suggests that implementations in higher dimensions can mitigate power penalties due to higher-degree topologies of interconnections, which are indicated by larger Rent parameter values. Accordingly, more 3D power benefits may be expected with netlists having larger Rent parameters. We also observe the existence of *thresholds of Rent parameters* beyond which 3D power benefits seem to increase more rapidly (e.g., 0.69 in Figure 4.7). Quantitative analysis of the relationship between power benefits and Rent parameter values will be one of our future works.

[37]The target clock period is 1*ns*. We show both input Rent parameters to the *gnl* software and actual Rent parameters of the generated netlists (placement-based) in the table.

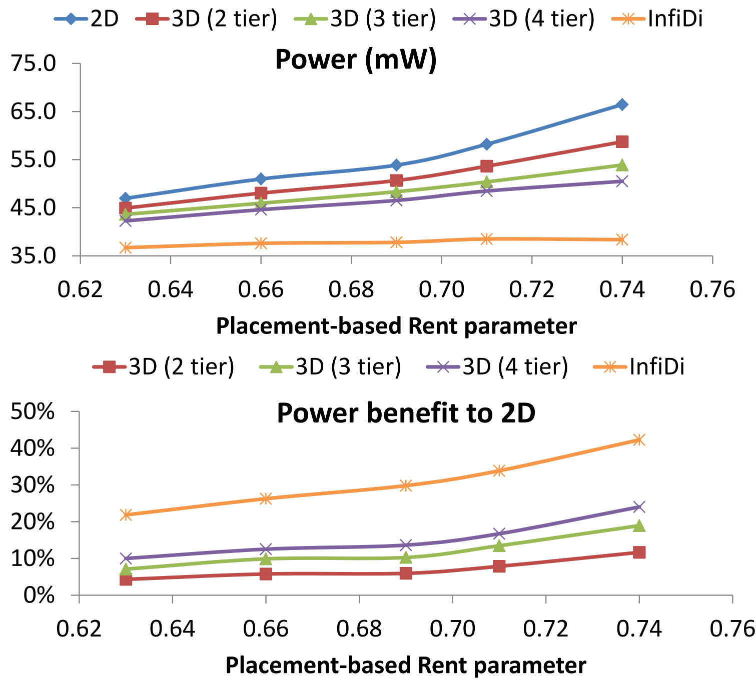


Figure 4.7: Power and power benefits versus Rent parameters for the 2D and the 3D implementations with different tiers.

We also perform similar studies with realistic designs. Figure 4.8 shows the correlation between the maximum 3DIC power benefits estimated in infinite dimension, and Rent parameter values. We extract Rent parameters of the netlists using both partitioning-based and placement-based methods, where we assume that one pin (terminal) is induced by each cut hyperedge. Partitioning-based Rent parameter values are extracted based on recursive bipartitioning using the min-cut hypergraph partitioner MLPart [244]. To calculate placement-based parameter values, we perform fast placement with a commercial P&R tool [235] without any sizing, VT-swapping or buffering optimizations. We then perform rectangle sampling based on the placement solutions to estimate Rent parameters.

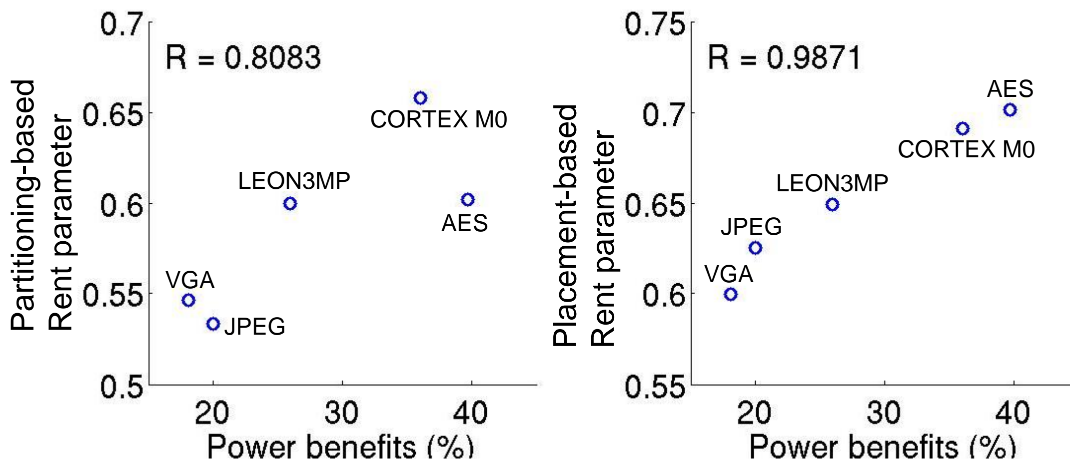


Figure 4.8: Power benefits correlate with Rent parameters.

Even for a larger testcase (*LEON3MP* with 436K instances), the runtime of the placement used to evaluate the placement-based Rent parameter is only 16 minutes. Our results show that the placement-based Rent parameter can possibly be a simple indicator of 3DIC power benefit for a given netlist.

In light of the correlation between power sensitivity to implementation dimensions and the netlist Rent parameter, we propose to modulate the cell usage during the synthesis stage to control the Rent parameters and achievable 3D power benefits. We categorize library cells in the 28*nm* FDSOI design enablement according to their input pin numbers – (i) one-input cells (buffers and inverters) (ii) two-input cells (NAND2, NOR2, etc.) (iii) three-input cells (NAND3, AOI21, etc.) (iv) four-input cells (NAND4, OAI22, etc.), and (v) >four input cells (AOI212, MUX41, etc.). We then scale cell area in Liberty files to modulate the cell usage during synthesis so as to achieve netlists with different Rent parameters. More specifically, we choose the *JPEG* design (which originally has a small Rent parameter) and scale down the area of complex cells; this induces the synthesis tool to use more complex cells and to increase the netlist Rent parameter.[38] We plot the placement-based Rent parameters against the portion of complex

[38]We synthesize *JPEG* with area scaled by $\{1\times, 2\times\}$ for 2-input cells, and by $\{1\times, 0.5\times\}$ for 3-input, 4-input and >4-input cells. An alternative way to modulate cell usage and Rent parameters during synthesis is to set a dont_use attribute for certain Liberty cells. However, the dont_use attribute cannot be assigned to NAND2, NOR2 cells in our EDA tooling. In addition, setting the dont_use attribute for a group of cells might degrade the synthesis solution quality due to limited available cell types.

cells (cells with more than three input pins) of various synthesized netlists in Figure 4.9. We observe that the Rent parameters are highly correlated to the incidence (proportion) of three-input cells. This demonstrates that we can modulate Rent parameters of the synthesized netlist. However, more precise control of Rent parameters during synthesis optimization remains as a direction for future work.

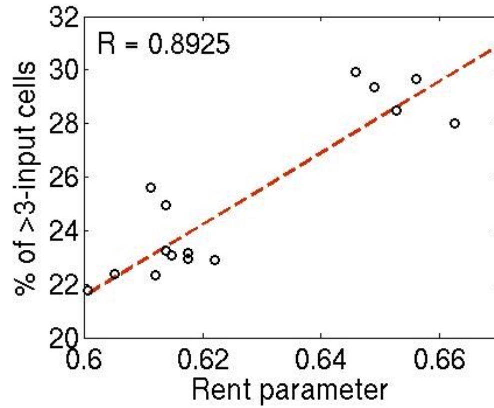


Figure 4.9: Correlation between incidence of cells with >3 inputs vs. Rent parameter.

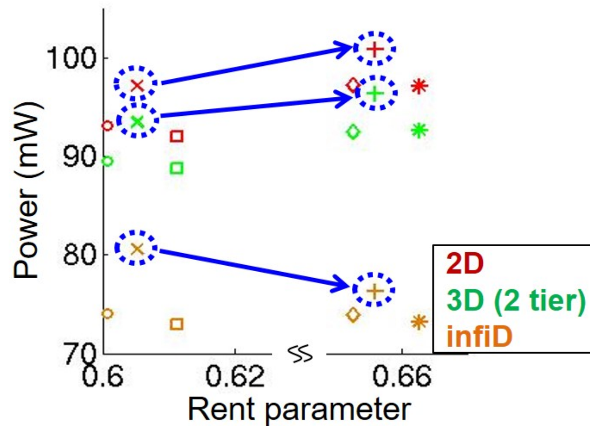


Figure 4.10: Power vs. Rent parameter with Rent Modulation.

In Figure 4.10, we further show power (after routing) of six synthesized netlists of design *JPEG* which have the maximum and minimum Rent parameters (Table 4.4). As highlighted in blue dotted circles in Figure 4.10, we observe that although a particular netlist shows small power after synthesis (indicated by infinite dimension), due to its large Rent parameter its power can be larger with a 2D implementation.

Table 4.4: Area scaling ratios of logic cells during synthesis in Figure 4.10.

Implementation	Rent	2-input	3-input	4-input	>4-input
O	0.600	1	0.5	1	1
X	0.605	2	0.5	1	1
□	0.611	1	1	1	1
◇	0.653	2	1	0.5	0.5
+	0.656	2	0.5	0.5	0.5
*	0.663	1	0.5	1	0.5

However, power penalty with a 3D implementation is smaller. This suggests that netlist synthesis should be aware of the implementation dimension. For instance, while a netlist with a small Rent parameter is desirable for a 2D implementation, there are fewer constraints on (or, sensitivities to) Rent parameters for a 3D implementation.

4.4.6 SoC-Level 3D Benefits

The above discussion, as well as many previous works on 3D wirelength benefits, all focus on blocks with only standard cells (i.e., pure logic-logic integration). For example, the work of [104] applies Rent's rule-based estimation to derive wirelength distribution in 3DICs. However, our P&R results indicate that their estimated benefits (e.g., 3.9× increase in frequency) might be optimistic. Mak et al. [160] compare 3D wirelength and 2D wirelength (where the 2D placement is simply assumed as side-by-side placement of tiers from the 3DIC) to estimate an upper bound on 3D wirelength benefits. Their results show an average of 18% wirelength reduction from 3D integration. Further, [40] uses an example to show that the maximum wirelength reduction from 3D integration is 66.7%. We show in Figure 4.11 that for a testcase with three blocks, the wirelength reduction can be close to 100%. In the example, there are connections between the north edge of Block A and the south edge of Block B (e.g., net a), the north edge of Block C and the south edge of Block A (e.g., nets b and c), as well as the south edge of Block C and the

north edge of Block B (e.g., nets d and e). As shown in Figure 4.11(a), the 2D wirelength is at least $4 \cdot H$. Figure 4.11(b) shows that the 3D wirelength can be zero if the vertical wirelength is ignored. Therefore, the example with ∼100% wirelength reduction indicates that there remain potential large 3DIC benefits versus 2DIC for (block-based) SoC designs. From the example in Figure 4.11, we see that there are possible gains from 3D integration for designs with memory cells. Since data access latency of a large memory system is typically limited by large interconnect delays, usage of die-to-die vertical interconnects and/or side-by-side integration of a 3D stacked memory and a processor on a silicon interposer to reduce memory access latency can improve the system performance [156] [187] [223] [229]. For example, the study of [187] shows a 47% latency reduction for a 4 MB 4-die stacked 3D SRAM array.

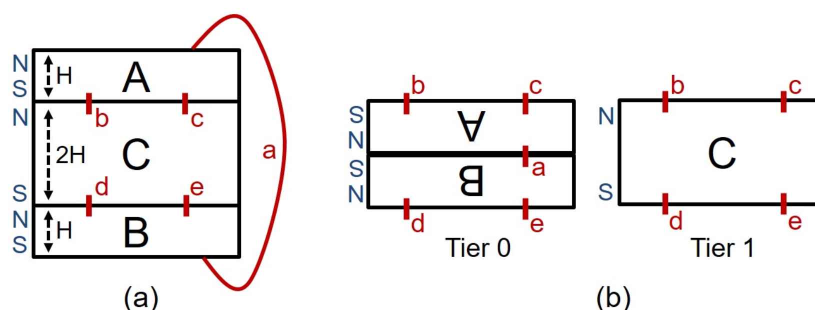


Figure 4.11: Potential 3D benefit in block-level planning. (a) 2D implementation with connections (shown in red) between the north edge of Block A and the south edge of Block B, the north edge of Block C and the south edge of Block A, the south edge of Block C and the north edge of Block B, where the wirelength is at least $4 \cdot H$. (b) 3D implementation with zero wirelength. a-e are five 2-pin nets.

4.5 Conclusions

In this chapter, we revisit previous assessments of the benefits of 3DIC implementation with respect to area, power and wirelength. Ours is the first work to estimate upper bounds on 3D power and area benefits based on the concept of *implementation with infinite dimensions*.

We examine several designs with our "infinite dimension" bounding methodology, and use the available area and power gaps between "best possible" 2D implementation to estimate upper bounds on

3D benefits. We further perform such 3D benefit estimation across various technologies. From our results, we observe that the 3D power and area benefits estimated in previous works (shown in Table 4.1) basically align with our proposed upper bounds. However, due to the design dependency of 3D benefits, benefits estimated in infinite-dimensional implementation can be small (e.g., power benefits as low as 18%) for some designs. Our study also indicates that although 3DIC might provide relatively large benefits in power or performance, it is typically difficult for pure logic-logic 3D integration to achieve a simultaneous (10%, 10%, 10%) improvement in (performance, power, area/cost) compared to the conventional 2D implementation. Such results indicate that 3D benefits are more likely to be achieved from the SoC-level and architectural-level optimizations instead of traditional P&R physical implementation optimizations. We use a simple example to show that there remains potential large 3DIC benefits versus 2DIC for (block-based) SoC designs. We also observe that inter-tier variation causes further significant reduction of available 3D power benefits.

In addition, we study design power across various dimensions and observe a correlation between design power and netlist Rent parameter. Modulation of the netlist Rent parameter during synthesis (that is, by changing the usage and distribution of fanins) suggests that a synthesis optimization that is aware of implementation dimensions may be helpful for reduced power in the final physical implementation. We also note that architecture-level improvements enabled by 3D integration (e.g., larger memory bandwidth) are still very promising, but are not addressed in our work.

Open directions for future research include (i) dimension-aware synthesis (i.e., synthesis for multi-tier 3D), (ii) quantitative analysis of the relationship between power benefits and Rent parameters, and (iii) architectural-level benefit exploration.

4.6 Acknowledgments

Chapter 4 contains reprints of W.-T. J. Chan, A. B. Kahng and J. Li, “Revisiting 3DIC Benefit with Multiple Tiers”, *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2016; and W.-T. J. Chan, A. B. Kahng and J. Li, “Revisiting 3DIC Benefit with Multiple Tiers”, *Integration, the VLSI Journal* 53, 2017.

I would like to thank my coauthors Andrew B. Kahng and Jiajia Li.

Chapter 5

Modeling for Approximate Computing

This chapter explores approximate computing. Approximate computing allows errors in computation in order to relax the design cost requirement. However, lack of error estimation blocks enabling of approximate computing. We propose a methodology to evaluate error metrics in approximate computing.

5.1 Background of Approximate Computing and Related Works

Modern VLSI designs face increasingly significant challenges in meeting the power and performance constraints demanded by present and future computing systems. Recently, approximate computing has been explored as a means of improving energy efficiency for noise-tolerant applications. While approximate computing circuits have been shown to be effective at improving energy efficiency at the expense of perfect functional correctness, modern CAD tools are ill-equipped to perform design automation for designs that contain approximate computing circuits. One key building block required for CAD tools that can create efficient approximate designs is the ability to quickly and accurately estimate the output quality of designs composed of approximate computing circuits. Such functionality is necessary for CAD

tools that would minimize the energy of a design during synthesis, optimization, etc. while maintaining acceptable output quality, as specified by system designers. In this chapter, we propose a flow that can analyze how errors originate and propagate in designs composed of approximate computing circuits to quickly and accurately estimate the output quality at nets in an approximate design. The following terminology is relevant to our treatment of approximate circuit design.

- *Error metric* (EMT): a unit of measure that quantifies the deviation between the outputs produced by a functionally correct design and an approximate design. We review several commonly-used EMTs from the existing literature in Section 5.2.1 below.
- *Approximate hardware module*: a hardware module that is functionally incorrect by design (e.g., approximate adders and multipliers).
- *Approximate circuit*: a circuit that contains one or more approximate hardware modules. Figure 5.1 compares an approximate circuit to its accurate counterpart.
- *Composed EMT* ($EMT_{composed}$): the estimated EMT value at an output or internal net in an approximate circuit.
- *Pre-characterized EMT* (EMT_{char}): a sampled EMT for an individual approximate hardware module that has been stored in a library. We measure EMT_{char} using Gaussian random variables as inputs and propose composition rules for different EMTs.
- *Composition function*: a function that maps EMT_{char} to $EMT_{composed}$.
- $D_{ref}(X)$: the output of a correct circuit (not approximate) for an input distribution X.
- $D_{appx}(X)$: the output of an approximate circuit for an input distribution X.

Given the above definitions, the *error metric composition* problem seeks to find a composition function for a composed EMT as described in Equation (5.1), where EMT^{i}_{char} denotes the EMT_{char} of the ith approximate module in an approximate circuit. Values of EMT_{char} for different approximate hardware modules may be stored in a library for quick reference during computation of $EMT_{composed}$.

$$EMT_{composed} = f\left(EMT^{1}_{char},\ EMT^{2}_{char},\ \ldots,\ EMT^{n}_{char}\right) \tag{5.1}$$

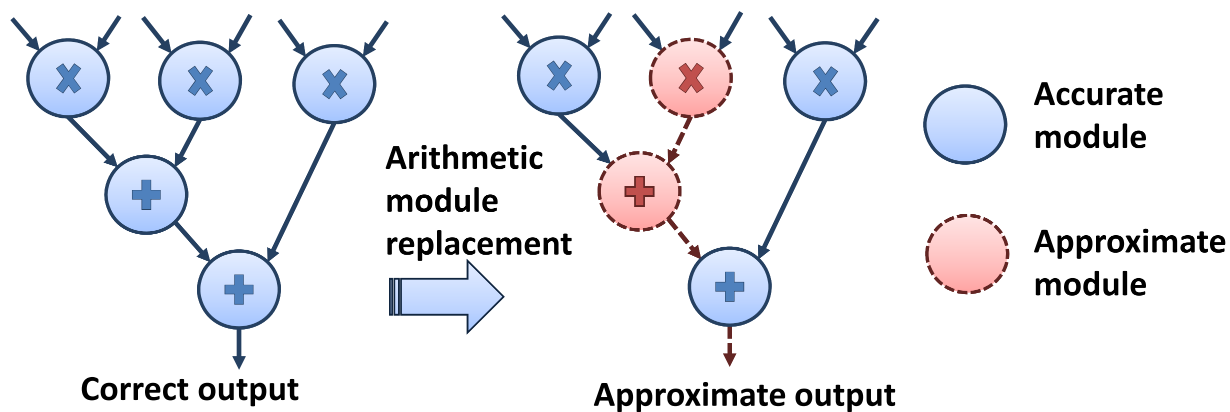


Figure 5.1: Illustration of approximation module replacement, in which accurate hardware modules are replaced by approximate ones.

In this chapter, we propose an automated methodology to estimate $EMT_{composed}$ at the outputs and internal nodes of an approximate circuit. Our approach accounts for the relationship between EMT behavior, input distribution statistics, and hardware characteristics of approximate hardware modules. We use lookup tables parameterized by approximate hardware module statistics to accelerate EMT composition; we further incorporate regression-based models that capture how errors propagate through the topology of an approximate circuit. We demonstrate and validate our methodology with several randomly generated benchmark circuits with varying complexity as well as designs evaluated in previous work [97] (e.g., FIR filter).

We make the following contributions.

- We analyze the interval-based approach in [97] [98], and explore the potential drawbacks of the approach.
- We propose composition rules for estimating the EMT observed at any net within an approximate circuit.
- We develop an approach to build pre-characterized libraries for individual approximate hardware modules and demonstrate how to accelerate the computation of composed EMTs using the libraries. Our approach reduces runtime for characterization and results in improved accuracy compared to previous works [97] [98].
- Compared to previous works, we improve the accuracy of EMT estimation by 3.75× for the same runtime. We achieve 1.36× and 8.4× runtime improvements, respectively, for library characterization and error composition.
- We demonstrate that our proposed approach achieves accurate estimates for approximate FIR circuits as well as random artificial circuits with various topologies.

The remainder of the chapter is organized as follows. Section 5.2 reviews related works according to different system abstraction levels and error sources. Section 5.3 describes the motivation of error metric composition and our search for rules that govern composition of error distributions. In Section 5.4, we show how to apply our composition rules and pre-characterized error libraries to analyze arbitrary circuit topologies. Section 5.5 concludes the chapter and gives directions for our future work.

5.2 Related Works

5.2.1 Error Metrics

Definitions of EMTs from the literature are given in Equations (5.2) to (5.7). Note that $E[\cdot]$ indicates the *expected value* of a random variable and $max[\cdot]$ indicates the *maximum value*.

$$ER = \sum_{X s.t. D_{appx}(X) != D_{ref}(X)} Pr(X) \tag{5.2}$$

$$ES = E[D_{appx}(X) - D_{ref}(X)] \tag{5.3}$$

$$ARES = E[(D_{appx}(X) - D_{ref}(X))/D_{ref}(X)] \tag{5.4}$$

$$MSE = E[|D_{appx}(X) - D_{ref}(X)|^2] \tag{5.5}$$

$$SNR = E[|D_{ref}(X)|^2/|D_{appx}(X) - D_{ref}(X)|^2] \tag{5.6}$$

$$MAXE = \max_X[|D_{appx}(X) - D_{ref}(X)|] \tag{5.7}$$

- *Error rate* (ER) [129] is used to evaluate the likelihood of correctness in arithmetic operations. An accurate estimation of ER is important in the case where approximate circuits spend additional cycles for error corrections.

- *Error significance* (ES) [213] addresses the magnitude of errors. We define *ES* as the signed average difference between correct and erroneous results.

- *Average relative error significance* (ARES) is used to measure the impacts of errors for image processing in [111] [232]. ARES is defined as the average absolute difference between correct and erroneous results, normalized to correct results. In *digital signal processing* (DSP) circuits, the magnitude of errors is important because small errors may be masked by other noise sources.

- *Mean squared error* (MSE) in [62] [216] and *signal-to-noise ratio* (SNR) in [62] [83] are common metrics to measure signal degradation in communication and image processing systems.

- *Maximum error* (MAXE) is defined as the maximum absolute value of produced errors. In [97], the MAXE metric is used to evaluate approximate circuits.

5.2.2 Approximate Arithmetic Modules

Various approximate arithmetic modules have been proposed in previous works, where aggressive timing and power benefits are obtained by breaking critical paths in the approximate module. To achieve a bounded error significance or configurable error rate, several techniques have been applied to reduce the severity of errors in these approximate hardware modules. ETAI [232] limits the maximum error by detecting a carry propagation and setting all lower sum bits to "1". A similar compensation approach is used in Shin's approximate adder [201], which detects a carry propagation using a specially designed truth table. By using error compensation approaches, the error can be reduced compared to simply breaking the carry chain. ETAIIM [232], ACA-SD [111], Lu's adder [158], and ACA-X [215] use a *carry-look-ahead* (CLA)-based approach to shorten the longest carry propagation path in the adder. These adders are composed of CLA submodules, and the numbers and sizes of the submodules can be configured at design time. The error significance and error rate can be configured by changing the length of carry propagation paths. Kahng et al. [111] also show that the errors can be detected and corrected in each CLA block, and that the accuracy can be configurable during runtime.

5.2.3 Analysis and Composition of Errors

We categorize existing works on hardware error analysis into four categories as shown in Table 5.1. In category (C1), the works focus on searching for useful approximations during logic synthesis. Venkataramani et al. [213] work with existing commercial synthesis tools and simplify logic based on *approximate don't-care* (ADC) information under a given error significance bound. Miao et al. [161]

Table 5.1: Categories of error analysis, propagation, and optimization works.

Category	(C1)	(C2)	(C3)	(C4)
Manipulated **E**lements	Logic cell	Arithmetic	Arithmetic	Multiple Levels
Error **S**ource	Appx. HW	Rounding	Appx. HW	Over-scaled VDD
Probabilistic **E**rrors	N	N	N	Y
Reference	[213] [161] [201]	[72] [206] [49] [50] [165] [132] [138] [128]	[96] [98] [97]	[118] [58] [59] [214]

focus on a methodology to design more efficient adders by combining logic components to reduce the maximum error. In [201], Shin et al. provide a heuristic to search for useful approximations based on a truth table to study the tradeoff between the error rate and literal terms (hardware cost). Previous works in category (C2) address rounding errors between floating-point and fixed-point conversions. In these works, the rounding errors are determined by the wordlength of hardware, and so are different from the errors induced by approximate hardware. In category (C3), [97] and [98] use an interval-based approach (*interval arithmetic* or *affine arithmetic*) to propagate errors. The interval-based approach uses pre-characterized libraries for error estimations, but the runtime of characterization can increase when more intervals are required for large ranges of signals. In category (C4), existing works assign overscaled supply voltages to achieve a graceful accuracy degradation. Kedem et al. [118] analyze propagations of errors induced by the degraded supply voltage, and they simplify the analysis by assuming that no error cancellations occur between multiple adders. Venkatesan et al. [214] propose the *MACACO* flow to evaluate propagations of errors induced by overscaled supply voltage. They also apply this approach to characterize errors for different approximate adders. Chippa et al. in [58] [59] propose methodologies to analyze and

optimize computing effort at different levels of abstraction, and also consider errors due to overscaled voltage supplies.

Compared to previous works, our work focuses on (i) the error propagation at arithmetic-level instead of gate-level computation, (ii) the errors induced by approximate hardware as opposed to the overscaled supply voltage, and (3) awareness of both ER and ES. Furthermore, we simplify the composition of errors with a pre-characterized library and regression coefficients.

5.3 Methodology for Error Estimation

5.3.1 Analysis of Existing Interval-based Approach

Huang et al. address the issue of error rate estimation for approximate circuits in [97] [98]. Their flow first characterizes approximate hardware modules by simulating the error probabilities for different input value intervals. Then, with given input operand distributions, they use *interval arithmetic* to estimate the *probability mass function* (PMF) of errors produced and propagated in an approximate arithmetic circuit. After propagating and composing errors with interval arithmetic, the error metrics are obtained from PMFs. Their interval-based approach samples the *probability distribution functions* (PDFs) or PMFs of errors to generate sampled PMFs. The height of each interval in the sampled PMF represents the probability of error.

Figure 5.2 shows an example PMF which is used in the interval-based approach. Due to the limited number of intervals used for characterization, the error magnitudes will be clamped to $\pm 2^{max}$ or $\pm 2^{min}$ if they are out of range. As a result, the accuracy of the interval-based approach will be impacted if the range of characterization does not match the inputs.

We observe two drawbacks in the interval-based approach. First, there is a quantization error, since the approach represents multiple error values with a single interval. If the actual error distribution

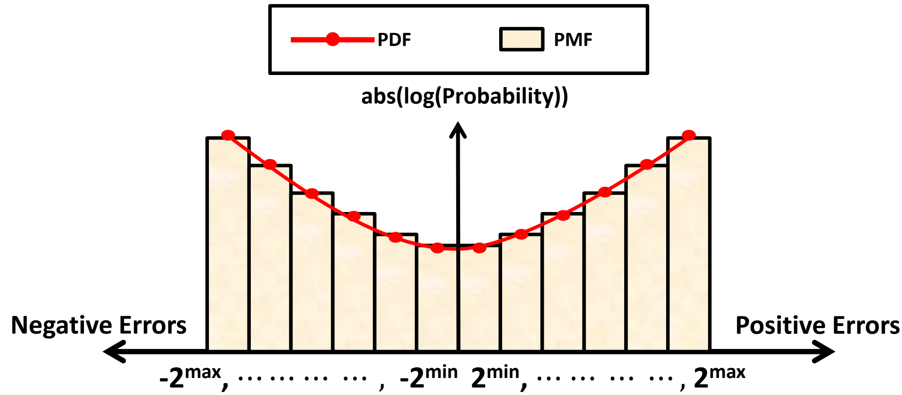


Figure 5.2: Probability mass function (PMF) used in the interval-based approach is shown. Note that the error magnitudes are in log scale. The error magnitudes will be clamped to $\pm 2^{max}$ or $\pm 2^{min}$ if they are out of range.

varies greatly within one interval, the estimation will be inaccurate. This may particularly be an issue for large intervals closer to $\pm 2^{max}$. For such large intervals, quantization error may be quite substantial. For example, an error value of $2^{max-1}+1$ is placed in the bin for 2^{max}, and the quantization error $(2^{max-1}-1)$ is essentially as large as the error value itself $(2^{max-1}+1)$. This example also illustrates another potential drawback. Unless the error value is an exact power of two, the quantization error for a large interval tends to be large. Perhaps counter-intuitively, error values that are very close to an interval value may cause very large errors. Second, the interval-based approach requires consecutive intervals to cover the range from maximum to minimum error magnitude ($\pm 2^{max}$ and $\pm 2^{min}$ in Figure 5.2). If the errors fall out of $\pm 2^{max}$ or $\pm 2^{min}$, the interval-based approach will clamp the estimated errors to the $\pm 2^{max}$ or $\pm 2^{min}$ values, and the estimation error will be saturated. If a large portion of errors or data experience this saturation issue, the estimation inaccuracy will be high. To address these drawbacks, the interval-based approach requires re-characterization of the libraries to increase the number of intervals, incurring significant runtime overhead. For better understanding of the strengths and weaknesses of the interval-based EMT composition, we evaluate the EMT estimation with a testcase shown in Figure 5.3(a). We vary the input distribution to evaluate accuracy for different input distributions and hardware configurations. We collect results from

100 combinations (10 Gaussian distributions with different standard deviations and 10 sets of ETAIIM configurations). Figure 5.3(b) shows the runtime of library characterization performed by the interval-based approach for different numbers of samples per interval. The accuracy results of the interval-based approach compared to Monte Carlo simulation are shown in the form of a correlation plot in Figure 5.3(c). From Figure 5.3(b) we notice that increasing the sample size to 18.5M requires 1.7 hours for library characterization, but estimation errors (offsets) are still observed in Figure 5.3(c). Possible reasons for the inaccuracy are (i) the use of discrete PMF and (ii) inaccurate propagation of EMTs from the pre-characterized library.

5.3.2 Analysis for Computation of Error Metrics

We analyze an ETAIIM adder to understand the error generation of approximate modules. ER of the ETAIIM adder is given in Equation (5.8). N is the total bit width of the adder; bits-per-block (BPB) is the size of *carry-lookahead* (CLA) blocks; and k is the number of connected CLA blocks, an architectural parameter used to control error magnitudes. From 5.4, we observe that the errors are related to the input values of CLA blocks because errors occur when all input bits of the CLA block are in carry-propagate state. For example, if most of the input values are small, then the probability of generating larger errors will be small. This observation regarding ETAIIM motivates us to study the sensitivity of EMTs to input distributions.

$$\begin{aligned} ER_{ETAIIM} = 1 - (1 - \frac{1}{2^{BPB}}\frac{2^{BPB}-1}{2^{BPB+1}})^{\frac{N}{BPB}-2-k} \\ \times(1 - \frac{1}{2^{(BPB\cdot k)}}\frac{2^{(BPB\cdot k)}-1}{2^{(BPB\cdot k)+1}}) \end{aligned} \tag{5.8}$$

-

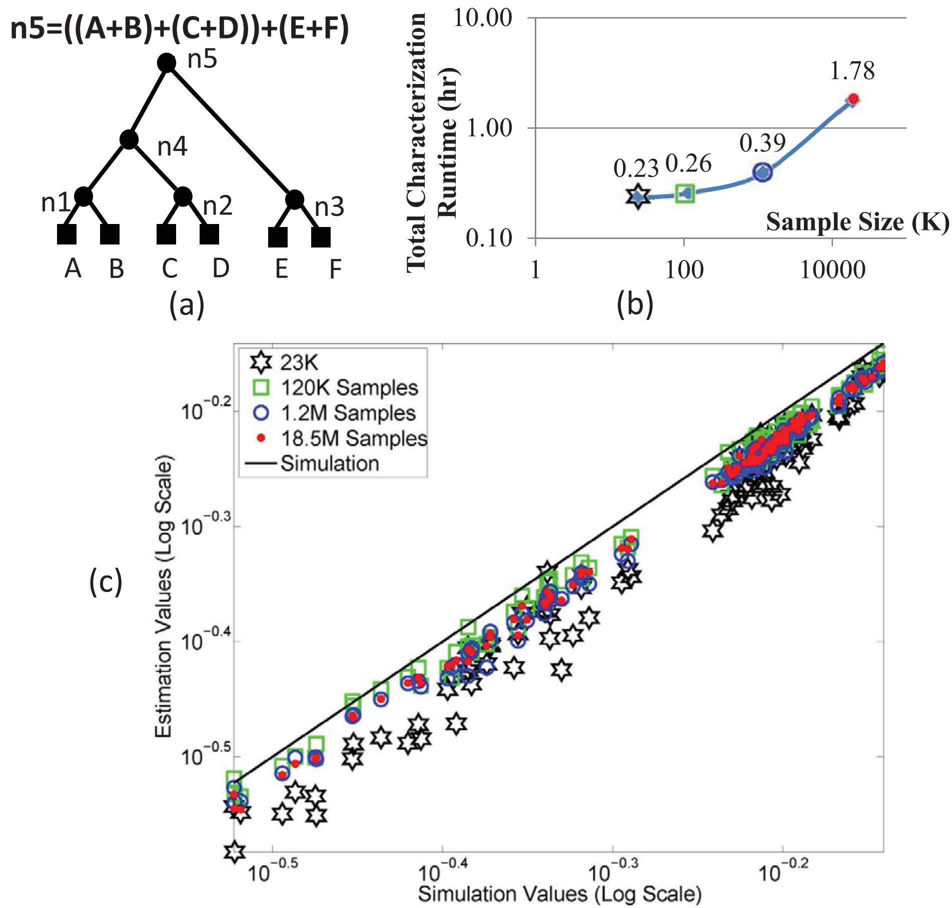


Figure 5.3: (a) Five-node testcase. (b) Runtime from interval-based approach for each sample size. (c) ER estimation results from interval-based approach. The results are generated from 100 testcases (10 hardware configurations and 10 combinations of input distributions).

5.3.3 Proposed Approach to Estimate EMTs

The analytical expression in Equation (5.8) is based on the assumption that distributions of the input values are uniform and the ranges cover from the *most significant bit* (MSB) to *least significant bit* (LSB). However, this is not always the case, and we need to consider input distributions for the accurately-estimated EMTs. To analyze the relationship between input distributions and EMTs, we use 24-bit ETAIIM adders and simulate the EMTs for different *BPB* and *k*. In this motivating experiment,

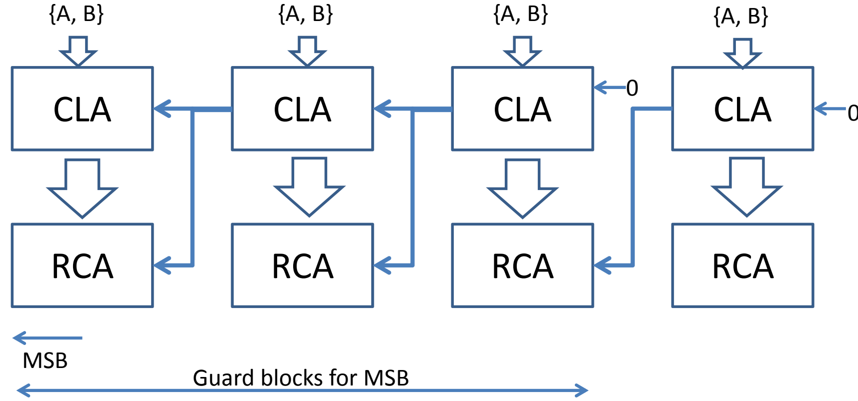


Figure 5.4: The structure of an ETAIIM approximate adder. CLAs are carry-lookahead sub-adders. RCAs are ripple-carry sub-adders.

both operands are assumed to have the same standard deviation for simplicity. Figure 5.5 shows each simulated EMT value (y-axis) with respect to the standard deviation of input data (x-axis).

Figure 5.5 shows that EMT values change with respect to both the standard deviations of input values and hardware configuration (k). Based on the results, we construct lookup tables to model the error metric of approximate modules instead of using analytical expressions. Modeling with lookup tables is preferred since it is difficult to derive an analytical expression if input values are not uniformly distributed.

Figure 5.6 illustrates our EMT formulation. To estimate the output EMT (EMT_Z), we consider *intrinsic* EMT values (EMT_{in}) which are generated by the approximate module itself, and *propagated* EMT values (EMT_A, EMT_B) which come from the previous stages. We propose a *lookup table* (LUT)-based approach to consider different input distributions. The lookup tables for different hardware configurations are merged to become the pre-characterized library. We construct two types of lookup tables, as illustrated in Figure 5.8(b). The EMT_{in} and STD_Z tables respectively contain (intrinsic) EMT values and output standard deviations with respect to the input standard deviations.

Our LUT-based approach can be divided into three steps as described in Figure 5.8(a). **Step 1: Value distribution propagation in the circuit topology.** We generate statistical properties with pre-characterized libraries. To obtain the statistical property of each node in the circuit, we traverse all the

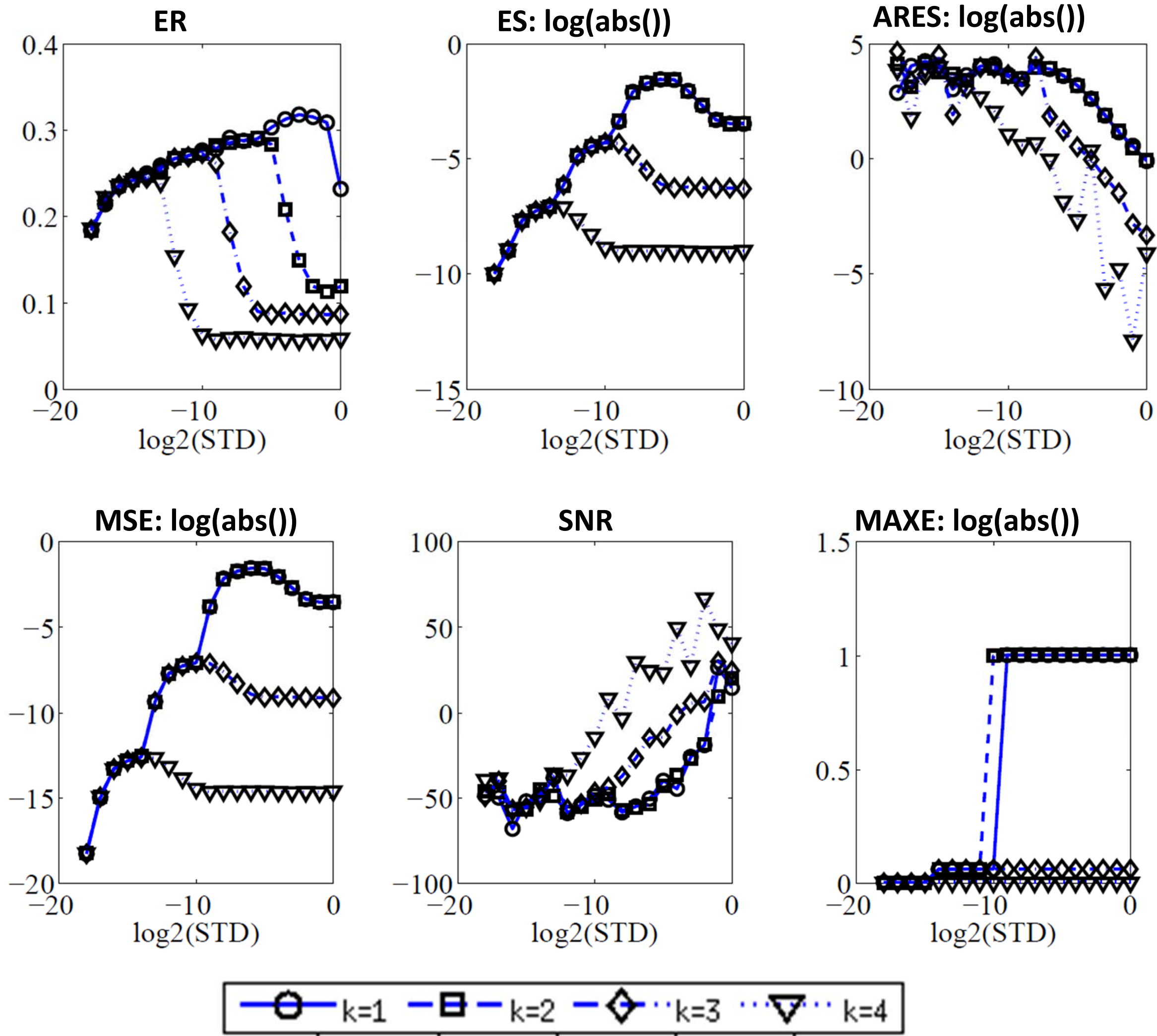


Figure 5.5: The simulated EMT results for input distributions. A 24-bit signed ETAIIM adder is simulated in the analysis. 20 bits are used for the fractional part, and the MSB guard block size k takes on values from one to four. For simplicity, both operands are assumed to have the same standard deviation.

nodes in the circuit in a topological order from primary inputs to a primary output. During the traversal, we look up the statistical property (standard deviation) from a pre-characterized table (STD_Z), and annotate the standard deviation values at all nodes. The results in the upper-left plot in Figure 5.7 demonstrate the feasibility of this approach.

Step 2: EMT estimation for approximate modules. With standard deviations of the internal nodes, we estimate EMT values using a pre-characterized table (EMT_{in}) for each internal node. The lookup table,

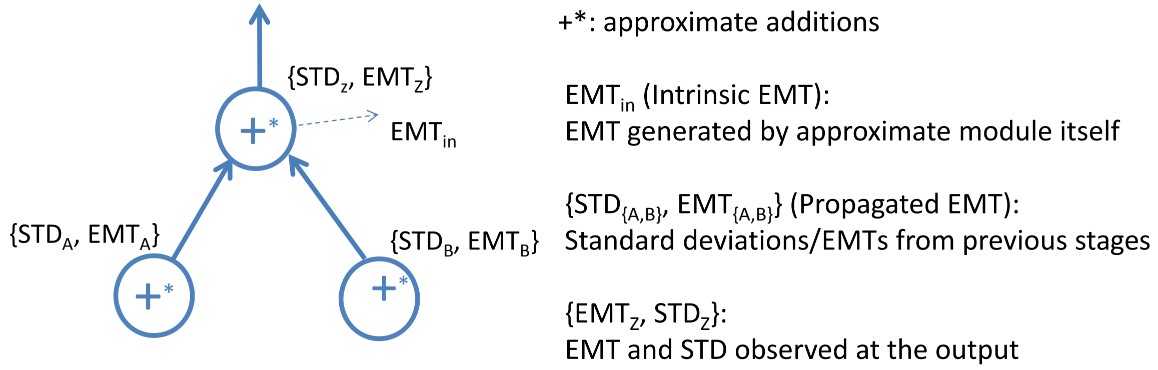


Figure 5.6: EMT estimation at a given node (approximate module) considering intrinsic and propagated EMTs.

EMT_{in}, is characterized by simulating EMT values as shown in Figure 5.5. We generate the LUTs for different approximate modules to estimate intrinsic error metric (EMT_{in}), which is generated by modules themselves without input errors. By combining Steps 1 and 2, we can estimate the EMT_{in} of each node in any circuit topology.

Step 3: Error composition with EMTs of each approximate module. With the generated EMTs (EMT_{in}) of each approximate module, we apply a regression approach to find the composed EMT values in the primary output. The error rate (ER) can be computed by multiplying pass rate (1-ER), and the composed ER is generated with Equation (5.9), where ER_Z is the composed ER, ER_A and ER_B are the propagated ERs to the inputs in Figure 5.6, ER_{in} is an intrinsic ER, and $\alpha_{\{in,P\}}$ are regression coefficients. Other EMTs (ES, ARES, MSE, SNR and MAXE) are amplitude-based error metrics, and we generate the composed EMT from Equation (5.10), where $\alpha_{\{in,P,C\}}$ are regression coefficients. These composition rules are developed for adders, and their generalization to multiplication and other arithmetic operations is a subject of future work.

$$ER_Z = 1 - 10^{\alpha_C} \cdot (1 - ER_{in})^{\alpha_{in}} \cdot ((1 - ER_A) \cdot (1 - ER_B))^{\alpha_P} \tag{5.9}$$

$$EMT_Z = \alpha_{in} EMT_{in} + \alpha_P (EMT_A + EMT_B) + \alpha_C. \tag{5.10}$$

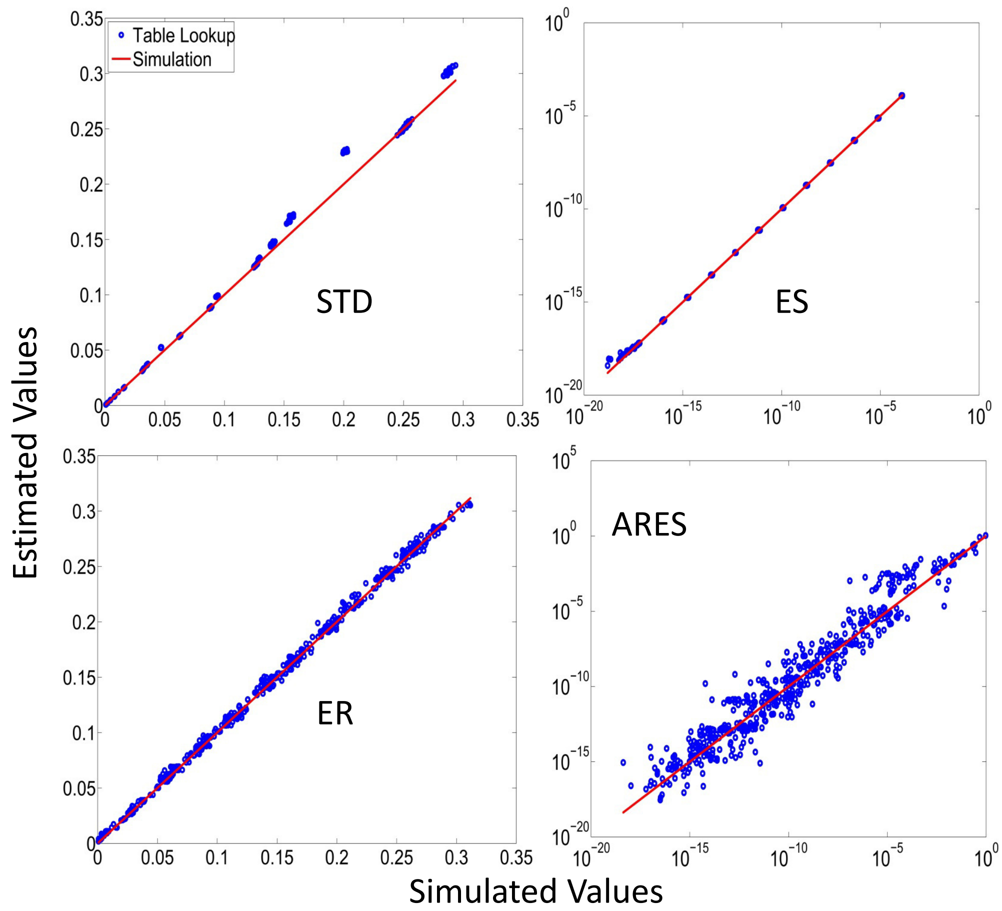


Figure 5.7: Estimated STD_A or STD_B and EMT_{in} values obtained from lookup tables. The *x*-axes are simulated values and *y*-axes are estimated values. The red lines show the ideal estimations and the blue dots show the estimated results from our proposed method.

To verify the correctness of our table lookup method in Step 2, we estimate the standard deviation (STD) and EMT_{in} as shown in Figure 5.7. We test with 10 combinations of hardware configurations and 10 combinations of different input distributions (Gaussian distribution with different standard deviations). Figure 5.7 shows the correlations between the estimated and simulated STD/EMT values from all internal nodes. The results show that we can obtain correct STD values from the lookup table with the topology traversal. With the estimated STD, we observe correct estimation for EMT_{in} (ER and ES). We find that ARES results are less accurate compared to the results of ER and ES. This is because ARES measures error relative to input data. If the magnitude of input data is small (near zero), the range of the ARES value will be large. In such a context, accurate estimations are difficult, given the limited number of grids in the lookup table.

Table 5.2 shows our regression results for improved EMT estimations. The upper part (a) of the table shows regression coefficients derived with different hardware configurations. The lower parts of the table show (b) the *estimation inaccuracy* and (c) the *absolute estimation inaccuracy*, as defined in Equation (5.11) and (5.12), where R_c and R_e are the simulated and estimated results, respectively. The two inaccuracy metrics are shown for both "without regression" and "with regression" cases.

$$Estimation\ inaccuracy \ = \ |R_c - R_e|/|R_c| \tag{5.11}$$

$$Absolute\ estimation\ inaccuracy \ = \ |R_c - R_e| \tag{5.12}$$

Without regression, we report the results with $\alpha_{IN} = \alpha_P = 1$ and $\alpha_C = 0$ for the coefficients in Equations (5.9) and (5.10); this is a pessimistic assumption (i.e., that there are no overlap effects from the composition). To obtain the coefficients in Equation (5.9) and (5.10) with regression, we simulate a single approximate adder with different operating conditions, which we model by changing the input

distributions (Gaussian distributions with zero mean and different standard deviations), and applying artificial errors. The artificial errors are also assumed to have Gaussian distributions with zero mean and different standard deviations. Using (i) the EMT_A, EMT_B, and EMT_Z measured from the simulation, as well as (ii) the EMT_{in} values obtained from lookup tables, we generate the regression coefficients using the linear regression toolbox in MATLAB [245].

After obtaining the regression coefficients, we apply them in EMT estimations and report the two inaccuracy metrics of EMT results. We observe from Table 5.2 that the regression coefficients help improve absolute inaccuracy of estimated ER, ES, ARES and SNR. However, the absolute inaccuracy slightly increases for MSE, and increases over 50% for MAXE. One possible reason could be that the data points that dominate MAXE are outliers for linear regression. Compared to absolute inaccuracy, the benefit of regression for estimation inaccuracy degrades for ES, MSE, and MAXE. This is because the linear regression applied to Equation (5.10) implies minimizing $|R_c - R_e|$, and $|R_c|$ in the denominator of Equation (5.11) is not considered. Improving the regression model to address both inaccuracy metrics mentioned above is one of our ongoing works.

5.4 Experimental Setup and Results

To evaluate the accuracy and performance of our EMT estimation approach, we perform several experiments. (For pessimistic evaluation, we use estimation inaccuracy in Equation (5.11) unless otherwise specified.) First, we demonstrate that our approach can be applied to a four-tap finite impulse response (FIR) filter. In the FIR experiment, the accuracies of six error metrics are evaluated. Second, we use *multiply-accumulator* (MAC) circuits with different sizes to compare the accuracy and runtime between our approach and the interval-based approach. Finally, we evaluate the accuracy of estimated

Table 5.2: (a) Regression coefficients derived with different hardware configurations, (b) estimation inaccuracy with and without regression, and (c) absolute estimation inaccuracy with and without regression.

Regression Parameters						
	ER	**ES**	**ARES**	**MSE**	**SNR**	**MAXE**
α_{IN}	1.03E+00	1.00E+00	2.42E-02	1.00E+00	3.46E-01	9.40E-01
α_P	1.26E+00	9.98E-01	9.76E-01	1.00E+00	7.15E-02	7.98E-01
α_C	-5.85E-03	5.74E-08	-5.92E-03	-5.55E-09	-1.27E+00	8.65E-05
Estimation Inaccuracy						
w/o Reg.	4.15E-02	7.77E-02	8.38E+02	1.08E-01	1.35E+02	1.28E-01
with Reg.	7.40E-03	5.55E-01	2.09E+02	4.44E+04	4.04E-01	1.88E+01
Absolute Inaccuracy						
w/o Reg.	4.01E-02	2.90E-05	1.09E+01	2.24E-07	2.96E+03	9.37E-04
with Reg.	7.17E-03	2.71E-05	1.46E-01	2.90E-07	1.31E+01	1.52E-03

results for randomly-generated topologies. In the experiments, we use 64-bit ETAIIMs with different k parameters. The adders are assumed to have 60 fractional bits.

FIR filter. To demonstrate that our approach is applicable to realistic computing circuits, we estimate EMTs for the FIR filter design illustrated in Figure 5.9(a). Lookup table characterization for each error metric and standard deviation is performed for 12×12 different combinations of standard deviations (2^0, 2^{-2},... 2^{-22}). For each entry in the tables, we use 90K samples to obtain standard deviations and EMTs. The runtime for building this set of lookup tables is 1.37 hours on a 2.8GHz Intel Xeon E5-2640 Linux workstation with 128GB of memory. With our lookup tables, we implement the flow in Figure 5.8 with MATLAB [245].

Table 5.3 shows inaccuracy results of the estimations for each EMT. We assume that the constant multipliers are accurate, and the adders in the FIR filter are approximate modules. In the second column (error type), "IN" means an intrinsic EMT value generated by the approximate modules themselves, and

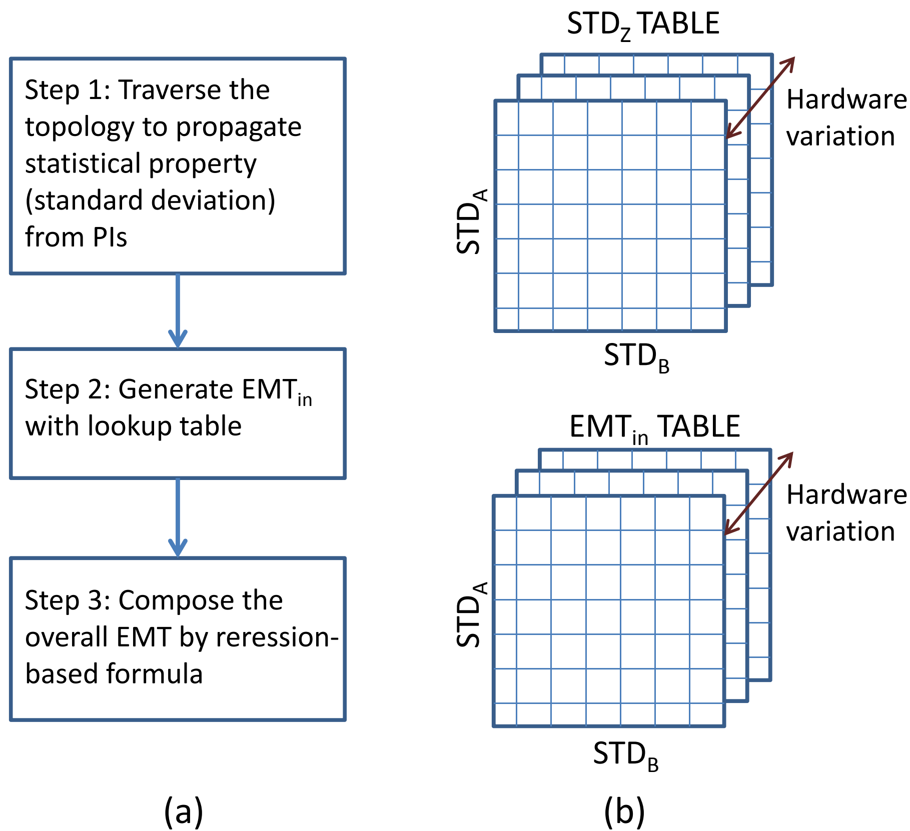


Figure 5.8: (a) Our proposed approach for error estimation, and (b) the lookup tables in the pre-characterized libraries for EMT_{in} and STD_Z.

"P" means a propagated EMT value composed from the EMTs in previous stages. Based on the results in Table 5.3, our approach provides accurate EMT estimations for ER, ES, MSE and MAXE metrics. For the same testcase, the inaccuracies of the interval-based approach are 17.6% and 60.2% for ER and ES, respectively.

Table 5.3: Estimation inaccuracy of a four-tap FIR filter shown in Figure 5.9(a).

		Estimation Inaccuracy					
Net	**Type**	**ER**	**ES**	**ARES**	**MSE**	**SNR**	**MAXE**
NET9	IN	0.3%	6.4%	17.0%	6.4%	19.1%	0.0%
NET10	IN	1.3%	2.6%	61.9%	3.3%	10.7%	0.0%
NET11	IN	1.0%	6.3%	419.6%	6.2%	6.1%	0.0%
NET11	P	13.4%	5.8%	692.3%	5.8%	436.4%	0.7%

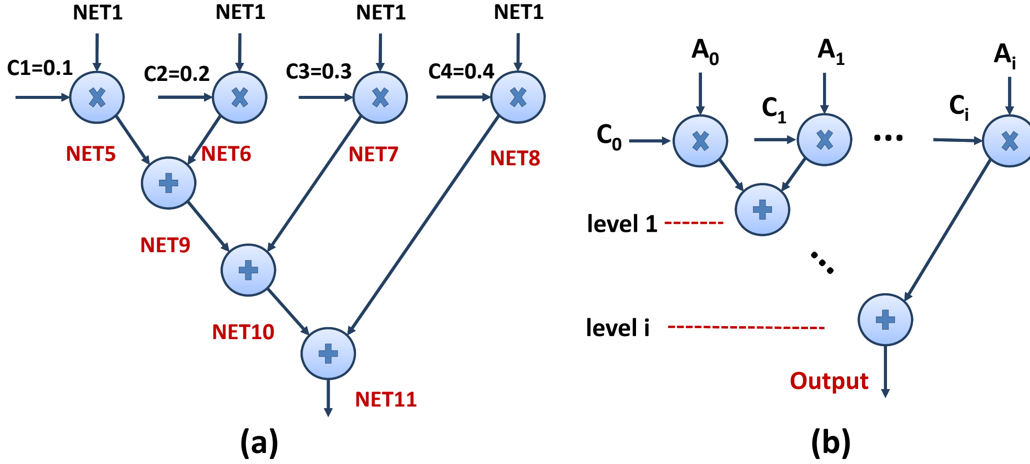


Figure 5.9: Configuration of (a) finite impulse response (FIR) filter and (b) multiply-accumulator (MAC) circuits used in the experiments.

MAC circuits. We test the accuracy and runtime of our approach against the interval-based approach for the MAC circuits shown in Figure 5.9(b), which are the general case of the FIR filter. We use 280 MAC circuits, having 14 different levels and 20 different configurations (parameters of each adder, constant values C_i, and input distributions).[39] We estimate EMTs for the MAC circuits using our approach and the interval-based approach. Figures 5.10 and 5.11 show correlation plots for ER and ES, respectively. For ER, we observe that our approach achieves 1.28× better accuracy than the interval-based approach with 8.4× faster runtime. For ES, we observe that the estimated results from the interval-based approach are clamped to -2^{-20} on the right end. This is due to the saturation issue mentioned in Section 5.3.1. For the same testcases, our approach is not affected by the saturation problem because the estimates of ES are interpolated or extrapolated from the lookup tables.

We evaluate runtime and accuracy for increasing circuit complexity by increasing the number of circuit levels in Figure 5.9(b). Figure 5.12(a) shows how runtime scales with circuit complexity. We observe that the runtime of error composition increases linearly for both our approach and the interval-based approach. Our approach is 8.4× faster than the interval-based approach. Figure 5.12(b) shows inaccuracy

[39]Note that the multipliers are assumed to be accurate, so they only change the distribution of data but do not increase the number of errors.

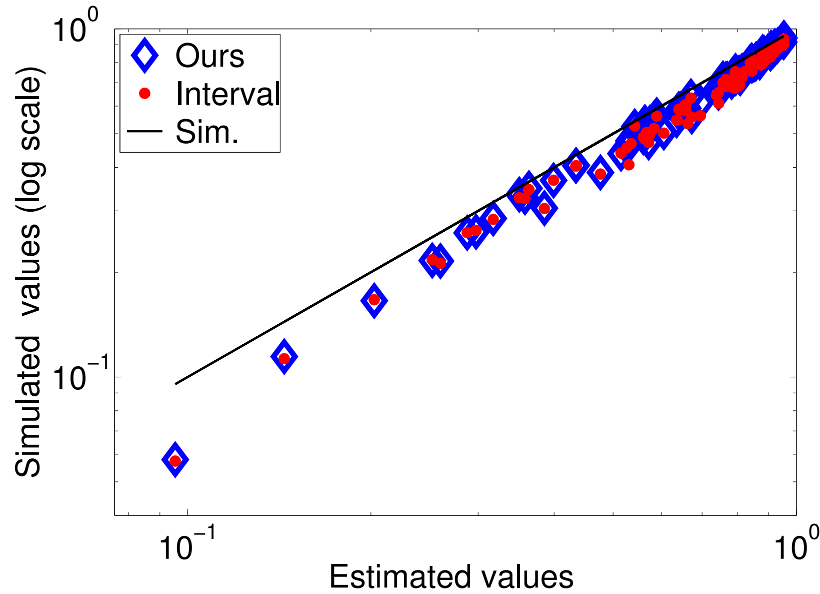


Figure 5.10: Comparison of ER metrics between our approach and the interval-based approach.

results. Our approach demonstrates improved accuracy compared to the interval-based approach, especially for the ES metric. The inaccuracy is reduced by 3.75× compared to the interval-based approach excluding saturation.[40]

Randomly generated topologies. To study the accuracy of EMT estimation with respect to the size and topology of testcases, we use randomly generated testcases as in [110]. We use the following three components to generate the random testcases; (i) primary inputs (PI) with different standard deviations, (ii) adders with different hardware configurations, and (3) arbitrary connections among adders and constant multipliers. We generate 50 artificial testcases with different numbers of nodes (adders or constant multipliers). The number of nodes ranges from 10 to 30 with a step size of five. The accuracy results for each EMT are plotted in Figure 5.13. We evaluate the estimated results from our approach with the regression coefficients generated from the model in Section 5.3.2. In the plot, inaccuracy results from 10 different topologies are averaged for each circuit size.

[40]Note that when the number of nodes is small (the left side of the figure), the magnitude of estimation errors tends to be large relative to the magnitude of data, and the inaccuracy of the interval-based approach is very high due to the saturation issue.

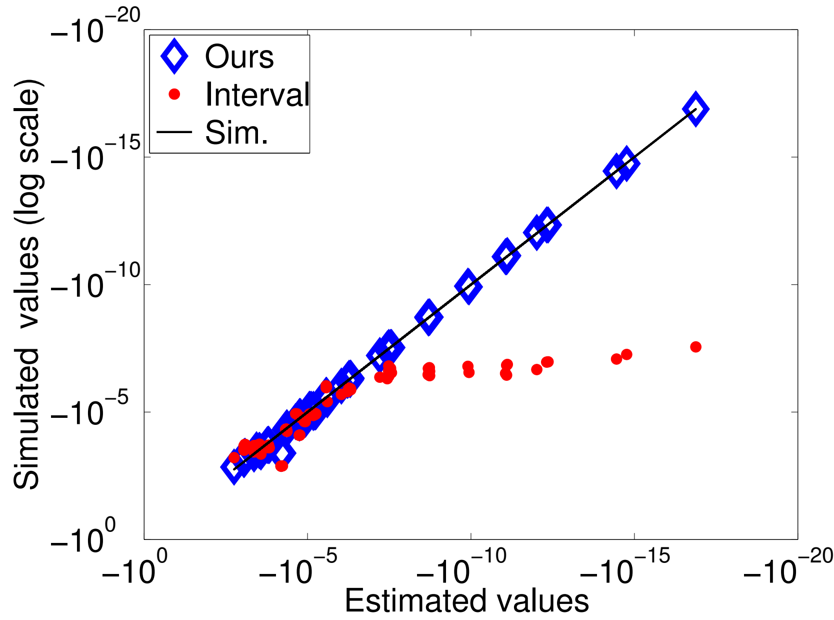


Figure 5.11: Comparison of ES metrics between our approach and the interval-based approach. The inaccuracy on the right side is (2×10^9).

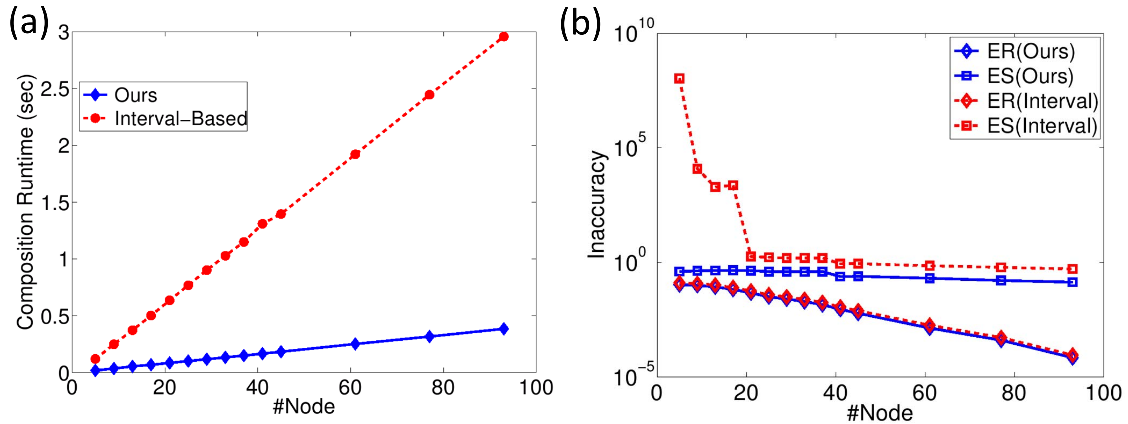
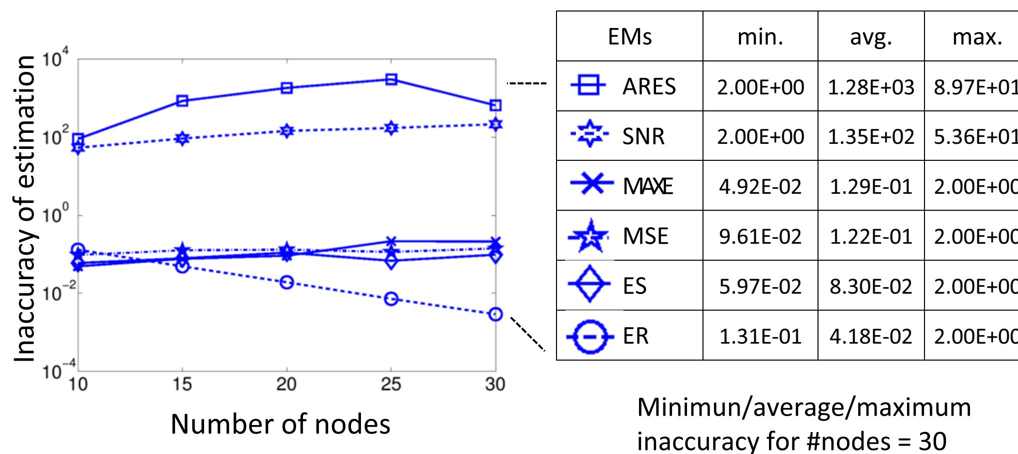


Figure 5.12: Comparison of (a) runtime for error composition and (b) inaccuracy of EMT estimation for the MAC circuits with different testcase sizes. Our average inaccuracy improvement against the interval-based approach is 3.75× excluding saturation.

For randomly generated circuits, we observe that ER, ES, MSE and MAXE show relatively accurate results with 4.18%, 8.30%, 12.2% and 12.9% inaccuracy, respectively. Moreover, the accuracy does not degrade as circuit complexity (number of nodes) increases. The estimates of ARES and SNR are inaccurate $(1.28\times10^3$ and $1.35\times10^2)$. Inaccuracy in these metrics arises because they measure error

relative to input data, and accurate estimation is difficult, as we have discussed in Section 5.3.3. Methods that would accurately handle their composition are obvious directions for our future work.



EMs	min.	avg.	max.
ARES	2.00E+00	1.28E+03	8.97E+01
SNR	2.00E+00	1.35E+02	5.36E+01
MAXE	4.92E-02	1.29E-01	2.00E+00
MSE	9.61E-02	1.22E-01	2.00E+00
ES	5.97E-02	8.30E-02	2.00E+00
ER	1.31E-01	4.18E-02	2.00E+00

Figure 5.13: Comparison of inaccuracy with respect to the number of nodes in randomly generated circuits.

5.5 Conclusions

We propose an approach for output quality estimation of approximate designs. Our LUT-based approach characterizes the statistical properties of approximate hardware modules, and a regression-based technique improves the accuracy of EMT estimation. With our composition approach, we achieve 1.36× and 8.4× runtime improvements for library characterization and error composition, respectively. We also achieve 3.75× accuracy improvement for ES compared to [97] [98] on a set of MAC circuits. We also demonstrate that our approach is applicable to general designs using the randomly generated testcases with up to 30 nodes in the configuration.

Our ongoing work seeks to improve the accuracy of EMT estimation for relative error metrics (e.g., ARES and SNR). We will also extend our approach to other approximate modules, including multipliers. In addition, we are working to develop a synthesis flow for approximate circuits using our EMT

estimation approach. We further anticipate broadening our current studies to include more approximate arithmetic units and different input distributions. Currently, we assume that the input distributions are given; however, different applications have different distributions. Our follow-on work will seek (i) approaches that track the change of input distributions and reconfigure the hardware during runtime, in order to adapt the distributions such that error metric requirements are maintained; and (ii) approaches that construct error metrics by decomposing arbitrary distributions into combinations of some basis distributions.

5.6 Acknowledgments

Chapter 5 contains a reprint of W.-T. J. Chan, A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Statistical Analysis and Modeling for Error Composition in Approximate Computation Circuits", *Proc. IEEE International Conference on Computer Design*, 2013.

I would like to thank my coauthors Andrew B. Kahng, Seokhyeong Kang, Rakesh Kumar and John Sartori. I also like to thank Jiawei Huang, John Lach and Gabriel Robins for providing the source code and scripts used in reference [98].

Chapter 6

Modeling and Optimization for Stochastic Computing

This chapter presents design methodologies stochastic computing. To enable more aggressive energy saving by exploiting stochastic computing, we exploit the timing-error resilience of stochastic computing to achieve aggressive voltage scaling. In the literature, we are among the first to introduce and exploit skew tolerance to save energy in stochastic circuit design, pointed out by Najafi et al. [169]. We demonstrate significant energy savings without severe output quality degradation compared to conventional binary computation paradigms.

6.1 Background of Stochastic Computing and Related Works

As we approach the limits of the traditional Moore's-Law scaling, energy and power constraints pose major challenges for IC designers. Many embedded systems such as wearable devices and medical implants have strict power and energy requirements due to battery capacity and physiological limita-

tions [136]. For example, body tissue may be damaged by excessive power dissipation in a poorly designed implantable circuit [64]. Various approaches have been proposed to overcome such energy/power problems. Notably, embedded systems are usually designed for specific applications; this allows designers to use dedicated hardware with more desirable physical and/or logical characteristics than conventional designs.

Stochastic computing (SC) [76] [185] has been proposed as an alternative low-power computing technique for several important embedded processing applications. SC circuits perform complex computations on (pseudo-)random bit-streams by means of simple logic gates. Figure 6.1 shows an SC circuit implementing the function $Z = \frac{1}{4} + \frac{1}{2}X_1X_2$. The number represented by each bit-stream is the probability of seeing a 1 in it. For example, the *stochastic numbers* (SNs) X_1, X_2, Z appearing at x_1, x_2, z represent $\frac{9}{12}, \frac{8}{12}, \frac{6}{12}$, respectively. The circuit has two primary inputs x_1 and x_2, and two auxiliary inputs r_1 and r_2. The auxiliary inputs are constant SNs of value $\frac{1}{2}$. The NAND gate of Figure 6.1 implements the stochastic function $Y_1 = 1 - X_1X_2$, which involves multiplication and subtraction. The OR gate implements $Y_2 = R_1 + R_2 - R_1R_2$, and since $R_1 = R_2 = \frac{1}{2}$, we have $Y_2 = \frac{3}{4}$. Finally, the XOR gate implements the function $Z = Y_1 + Y_2 - 2Y_1Y_2 = \frac{1}{4} + \frac{1}{2}X_1X_2$.

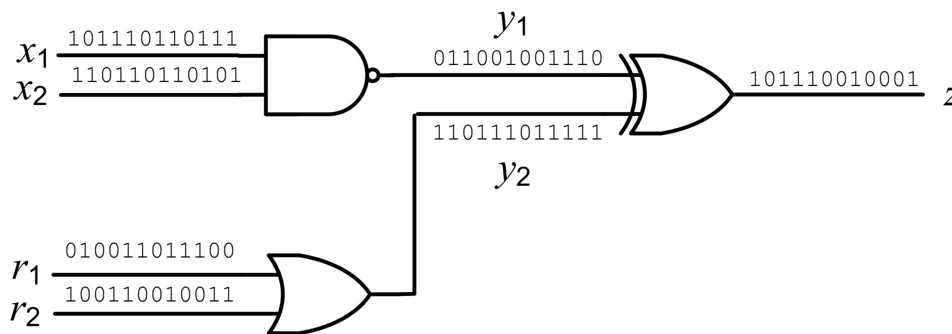


Figure 6.1: Stochastic computing circuit implementing the function $Z = \frac{1}{4} + \frac{1}{2}X_1X_2$. The stochastic number represented by each bit-stream is the probability of seeing a 1 in a randomly chosen position.

The main benefit of SC, as is evident from Figure 6.1, is that simple logic gates implement complicated arithmetic functions. For example, a single AND gate implements multiplication. Compared

to a conventional binary multiplier, the SC multiplier is orders of magnitudes smaller in size. This however comes at the cost of speed. SC circuits need to operate on bit-streams that grow exponentially as the precision increases, thus they take much more time to complete a computation. This has always been the main drawback of SC. The exponential loss in runtime not only hurts the performance, but also leads to excessive energy consumption when long bit-streams are used [166]. Consequently, SC circuits are mainly useful for low-precision computations [4]. Some recent work has focused on addressing this problem by reducing the runtime of SC circuits through the use of deterministic number sources [8], or by eliminating the power overhead of the clock distribution tree [168].

This chapter exploits SC's error tolerance in order to reduce the energy consumption of SC circuits via voltage/frequency scaling. SC circuits are error-tolerant because a single error on one of the bits has minimal effect on the numerical value of a long bit-stream, and multiple errors tend to cancel each other out. Finally, SC circuits provide a natural energy-accuracy tradeoff: the bit-stream length N, i.e., the number of clock cycles an SC circuit uses to perform a computation, directly affects its energy consumption and its accuracy.

SC, when first introduced in the 1960s, was attractive because it allowed simple implementation of arithmetic functions. However, it was dominated by conventional binary computing in the decades that followed, mainly because transistors became cheaper and performance became the primary design target. Throughout those decades, SC remained useful in certain applications, including efficient implementation of artificial neural networks [126] [26] [32].

After the turn of the century, SC regained attention because of its potential in low-power embedded processing applications. Some recent successful applications include low-density parity check (LDPC) decoding [77] [137] [81] and image processing [146] [12] [74]. Other recent applications of SC include data recognition and mining [60] [167], machine learning [82], and dynamical systems [217].

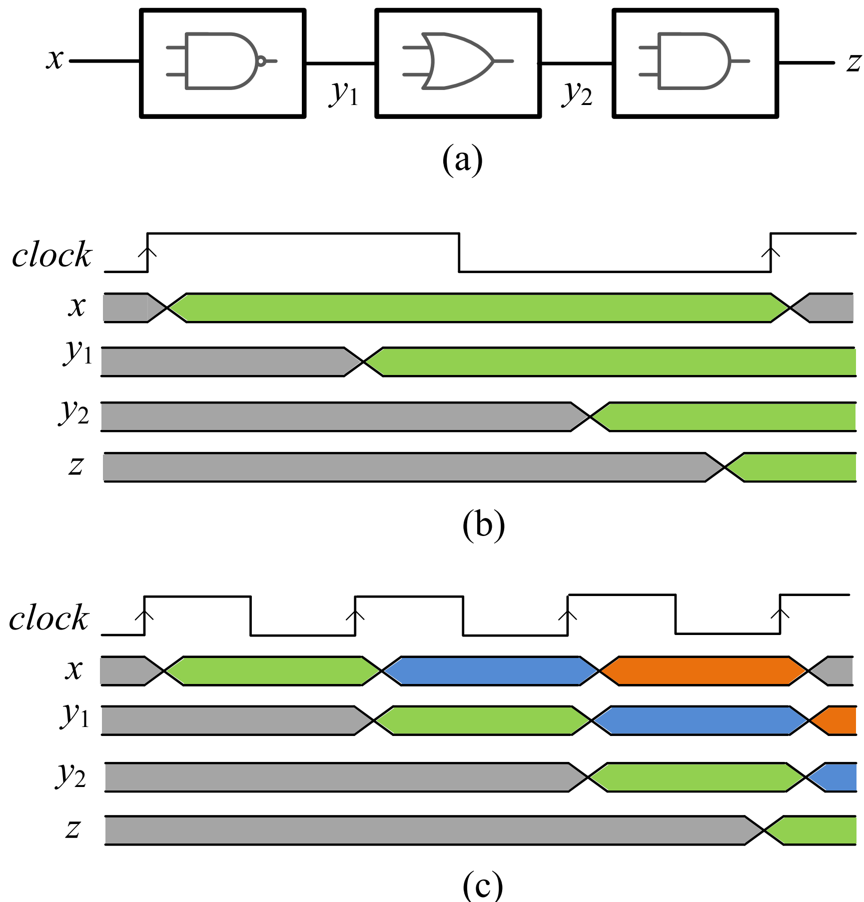


Figure 6.2: Example showing how a representative SC circuit (a) operates under aggressive voltage/frequency scaling and (b) shows the normal mode of operation, in which the period of the clock is the same as the propagation delay of the circuit. In this case, the input remains unchanged until the propagation to the output is complete. (c) shows a voltage/frequency scaled scenario where the clock period is approximately three times smaller than the propagation delay of the circuit. As soon as the input propagates through the first level, a new input is applied. The circuit levels therefore operate like a pipeline.

Note that [137] and [74] have silicon-validated SC designs that outperform their conventional binary counterparts.

As mentioned, we investigate the application of low-power techniques such as voltage scaling to SC with the goal of obtaining circuits with ultra-low energy needs. Voltage scaling, i.e., reducing the supply voltage of a circuit, reduces the circuit's energy consumption but increases its latency. If the application allows some latency overhead, aggressive voltage scaling can be employed at the cost of occasional erroneous outputs. Thus, voltage scaling allows designers to trade accuracy for energy. This approach has

been extensively studied in the non-SC literature [89] [92] [112], and methods of tolerating and/or correcting timing errors have been proposed. However, the probability of timing violations increases rapidly with voltage scaling, necessitating complicated error-correcting methods.

In this chapter, we show that representative SC circuits can tolerate up to 40% voltage reduction with no significant error. Figure 6.2 shows an example circuit and illustrates why, intuitively, we can aggressively scale the voltage/frequency of SC circuits. The idea is that we can apply new sets of inputs before the previous inputs have completely propagated through the circuit. In the ideal scenario (shown in Figure 6.2(c)), all the input signals propagate through different levels with the same speed; this scenario is very similar to the concept of wave-pipelining [28].

A major contribution of this chapter is an optimization method that improves the accuracy-energy tradeoff of SC circuits under voltage/frequency scaling. This is based on the observations of Figure 6.2. In order to achieve the ideal scenario shown in Figure 6.2(c), we need to modify the circuit to make sure signals propagate simultaneously. For this purpose, we employ synthesis and physical design techniques that keep the circuit's functionality intact, while effectively winning back any lost accuracy.

In the synthesis step, we employ circuit structures that are naturally balanced. Note that this is different from logic-level balancing of circuits because we look at circuits that are stochastically equivalent [53], meaning that their underlying logic function may not be the same, even though they implement the same stochastic function. Existing logic-level tools do not understand such equivalences.

Ideally, a conventional P&R (place and route) flow balances path delays as much as possible (e.g., trading the slacks of non-critical paths for power and area reductions). However, due to design constraints (e.g., maximum transition or maximum capacitance) and limited gate sizes, path delays are typically not perfectly balanced, especially when the paths have large differences in their depths. Figure 6.3 illustrates our proposed optimization. Assume that gate G_1 is already sized to its smallest size and is using high

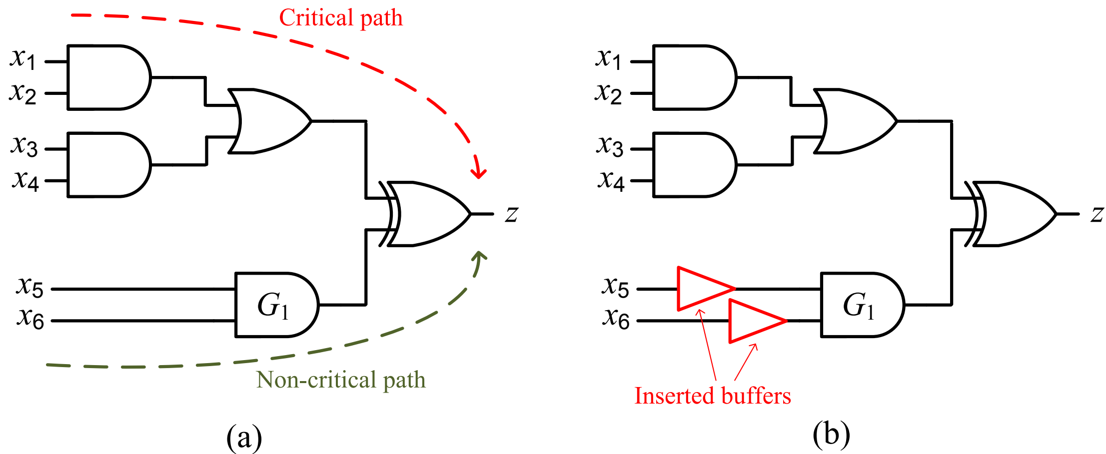


Figure 6.3: (a) SC circuit implemented via conventional P&R tools, where the main objectives are area and power reduction. (b) SC circuit implemented via the proposed optimization flow, where path delays are balanced to improve accuracy. Inserted buffers are shown in red.

V_{th} (threshold voltage), but due to the difference in the depths of the paths, delays from $x_{\{1,2,3,4\}}$ to z and those from $x_{\{5,6\}}$ to z are still not completely balanced. Our study shows (in Section 6.3) that such unbalanced path delays increase the computation error of voltage/frequency-scaled SC circuits. To improve the computation accuracy of SC circuits, we propose to further balance path delays using buffer insertion and wire detouring. As indicated in Figure 6.3(b), we insert buffers (shown in red) into the non-critical path. Although the inserted buffers incur a power penalty, they enable more frequency and/or voltage scaling and hence reduce the latency and the energy consumption of the circuit for a given accuracy requirement. To our knowledge, this is the first time such techniques have been employed in the context of SC. In addition, to guide the design space exploration, we demonstrate an improved Markov chain model of computation errors in Section 6.2.3 based on the model from [6]. We improve the modeling accuracy in [6] by applying least squares regression.

Note that the term "stochastic computing" has also been used recently to describe conventional circuits involving probabilistic behavior, including scenarios with voltage/frequency scaling [197] [199]. What we refer to as SC is the computation technique that was proposed in the 1960s [76] [185], and is unrelated to the concepts used in [197] [199].

In this chapter, we focus only on combinational SC circuits. Qian and Riedel [190] and Qian et al. [191] show a connection between combinational SC circuits and Bernstein polynomials [157] and prove that SC combinational circuits only implement certain types of polynomial functions. In this chapter, we use ReSC [191] as the main method of implementing most of our testcases. As we will discuss later in the chapter, the rival design method [7] is not as energy-efficient as ReSC.

We note that sequential SC circuits that implement a larger class of functions also exist in the literature [146] [196] [26]. Brown and Card [26] are among the first to develop the theory of sequential SC circuits in the context of neural networks. In particular, they implement a sigmoid function by using a simple finite-state machine. However, addressing sequential circuits is beyond the scope of this chapter; we leave it as a subject for future work. The *Sigmoid* testcase used in this chapter is a combinational implementation and is unrelated to the circuit designed by [26]. We also note that combinational circuits can implement non-polynomial functions by exploiting correlation [9], and that our optimization method is capable of handling them.

This chapter is organized as follows. Section 6.1 gives a brief review of SC, as well as an error analysis under voltage/frequency scaling conditions. It also illustrates the effect of stochastic number generation on the accuracy of SC circuits. Section 6.2 poses two related optimization problems, along with a straightforward way to find the minimum-energy operating point of an SC circuit for a desired accuracy level. It also discusses a fast error estimation method for SC circuits. Section 6.3 examines the opportunities for optimizing SC circuits at different voltage levels. Section 6.4 presents experimental results and conclusions are given in Section 6.5.

A stochastic circuit C is a logic circuit that operates on (pseudo-)random bit-streams, called *stochastic numbers* (SNs). Each wire x_i of C carries an SN X_i. The information conveyed by X_i, also conveniently denoted by X_i when no confusion is possible, is the rate or frequency of its 1-pulses and is

independent of bit-stream length. Formally, a bit-stream of length N with N_1 1's and $N - N_1$ 0's is called an SN with value or magnitude $X_i = N_1/N$. This is usually interpreted as the probability of seeing a 1 in a randomly chosen position of the bit-stream [76]. SN values range over the unit interval $[0,1]$, and their precision is determined by N.

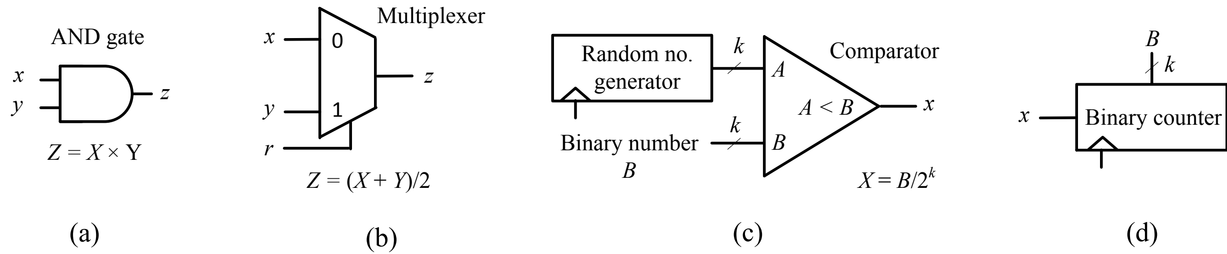


Figure 6.4: Basic SC components: (a) multiplier, (b) scaled adder, (c) stochastic number generator, and (d) stochastic-to-binary converter.

Figure 6.4 shows several basic SC components. As mentioned earlier, a single AND-gate (shown in Figure 6.4(a)) implements SC multiplication. Figure 6.4(b) shows a multiplexer that implements SC addition. Since SNs are in the unit interval, the sum of two SNs falls into the interval $[0,2]$ which cannot be represented by an SN. To mitigate this problem, a scaling factor of 1/2 is usually applied to bring the result back into the unit interval. Thus, the circuit of Figure 6.4(b) implements the scaled addition $Z = (X+Y)/2$.

When used along with conventional binary circuits, the inputs and outputs of stochastic circuits must go through a conversion process. Figure 6.4(c) shows a binary-to-stochastic converter which is usually referred to as *stochastic number generator* (SNG). At each clock cycle, the SNG compares its k-bit binary input with a uniformly distributed random number. As a result, the probability of seeing a 1 at the output of the comparator becomes proportional to the binary input. Several studies have shown that the random number generator of Figure 6.4(c) can sometimes be replaced by simple counters [8]. As we will show later in this section, the choice of random number generators can impact the power and

accuracy of an SC circuit. Converting SNs back to binary form can be done by counting the number of 1's, so the binary counter shown in Figure 6.4(d) suffices for this task.

6.1.1 Error Analysis

The inherent error tolerance of SC circuits stems from the fact that a single bit-flip in an SN of length N alters its magnitude by $1/N$, which is insignificant when N is sufficiently large. For example, the SN at the output of the circuit in Figure 6.1, where $N = 12$, represents $Z = \frac{6}{12}$. If one of the 1's or 0's of the bit-stream changes due to an error, the erroneous SN is $Z^* = Z \pm \frac{1}{12}$, a minimal change. Furthermore, multiple errors tend to cancel each other out if they occur in opposite directions, since it is the number of 1's, and not their positions, that determines the magnitude of an SN [51] [189] [191]. The probabilistic nature of SC circuits, along with the cancellation possibilities, makes it difficult to evaluate the accuracy of SC circuits.

Even though we will be dealing with the effect of timing errors in this chapter, it is worth mentioning how errors of bit-flip type affect SNs. The mean square error (MSE) of an SN in the presence of bit-flips can be calculated by the following equation [51]. Assuming Z is the error-free SN and Z^* is the erroneous SN, we have

$$MSE = \mathbb{E}[(Z - Z^*)^2] = p_e^2 \cdot (1 - 2 \cdot Z)^2 + \frac{1}{N} \cdot \big(Z \cdot (1 - Z) + p_e \cdot (1 - p_e) \cdot (1 - 4 \cdot Z \cdot (1 - Z))\big) \tag{6.1}$$

where $\mathbb{E}[*]$ denotes the expected value (averaging) operator, p_e is the probability of getting a bit-flip on each bit of the SN, and N is the SN's length. Equation (6.1) reflects the effect of error cancellation: when $Z = 1/2$, the error becomes zero for large N. However, the cancellation does not help much when $Z \neq 1/2$. In the extreme case of $Z = 0$ (or $Z = 1$), the error is maximized because no cancellation occurs.

This chapter investigates the application of voltage and frequency scaling to SC circuits. Voltage scaling refers to the systematic reduction of the power supply voltage (i.e., "undervolting"), which is

a standard technique used to reduce the power consumption of digital circuits. However, such scaling tends to produce timing violations that may cause output errors. Overly aggressive voltage scaling can induce many timing errors in conventional binary circuits and the resulting degradation of computational correctness can be catastrophic. By frequency scaling, we refer to the clocking of the circuit at higher than its nominal speed, at the cost of timing errors. It is possible to use design methods such as Razor [71] to make conventional circuits more resilient to timing errors that are induced by frequency scaling. However, these techniques are only effective when the error rate is relatively low. SC circuits, on the other hand, have the potential to achieve graceful degradation of computational correctness when the voltage (or frequency) scaling is extremely aggressive and the timing error rate is relatively high. In addition, we may also be able to retrieve lost accuracy by employing the optimization method proposed in this chapter.

The types of errors that affect an SC circuit are quite different than bit-flips when voltage (or frequency) scaling is applied. In general, SC circuits tolerate errors of the bit-flip type, so one would expect them to tolerate scaling-induced timing errors as well. As we show next, SC circuits are much better at tolerating such timing errors.

Timing errors may occur in an SN Z when a transition from 0 to 1 is delayed, in which case the 1 will not be captured in time, and the magnitude of Z will be reduced by $1/N$, where N is the bit-stream length. Similarly, on a 1-to-0 transition, the 0 may be missed because of a timing error, and the magnitude of Z will increase by $1/N$. Since the numbers of 0-to-1 and 1-to-0 transitions are almost the same for any bit-stream (the difference is at most one), these timing errors tend to cancel each other out. Figure 6.5 shows an example of an SN affected by transition errors. In this figure, Z has three 0-to-1 and three 1-to-0 transitions denoted by arrows. Due to delay errors, some of the transitions are missed in the erroneous case Z^*, but the resulting number value remains the same due to error cancellation. In a very recent work, Najafi et al. [168] show that SC circuits can tolerate timing variations caused by unsynchronized clocks.

Their observation is in agreement with the results of our work. Similarly, Perez-Andrade et al. [184] have shown that SC LDPC decoders can operate satisfactorily when clock-scaling-induced timing errors are present.

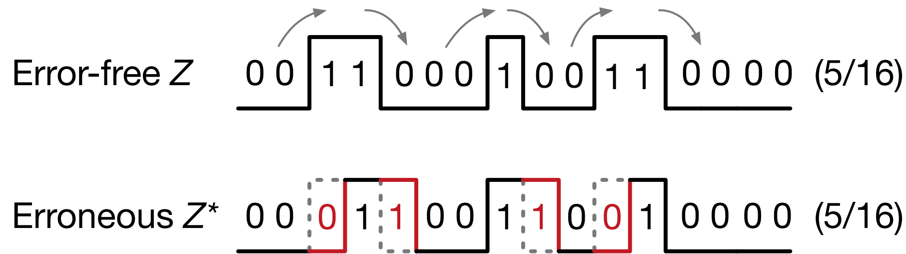


Figure 6.5: Effect of delay errors on an SN.

Now let us denote the 0-to-1 and 1-to-0 error rates by $e_{0\to1}$ and $e_{1\to0}$, respectively. We want to analyze their effect on an SN Z. We assume that the event of having a transition error at a certain clock cycle is an independent sample from a Bernoulli random variable with parameter $e_{0\to1}$ or $e_{1\to0}$, depending on the direction of the transition. This simplifying assumption is not a precise model, because after all, the underlying circuit is deterministic. However, there are several phenomena that produce seemingly random behavior in voltage/frequency-scaled SC circuits. First, since (pseudo-)random bit-streams are used in most cases, signals in two consecutive clock cycles will be statistically independent. This implies that an input signal does not always activate the same circuit paths; the activated paths depend on the signals of the previous clock cycles, which are pseudo-random. Second, the output of an SC circuit is usually collected at a flip-flop of a counter, and due to timing violations, the output flip-flop may become metastable and produce a seemingly random result. Third, when the supply voltage is reduced, the circuits become more susceptible to environment noise of a random nature. We do not consider the direct effect of each of these phenomena in this chapter; we model their collective effect by a Bernoulli random variable. We note that if deterministic bit-streams are used, as in our *EdgeDetection* testcase, the simplifying assumption of Bernoulli random variable fails to model the circuit behavior correctly.

For simplicity, we also assume that Z has equal numbers of 0-to-1 and 1-to-0 transitions. These numbers depend on two factors: the value of Z and its "activity". If $Z = 0$ or $Z = 1$, then the number of transitions will be zero. The maximum number of transitions usually occurs when $Z = 1/2$, if (pseudo-)random number sources are used in generating Z. But as we will show in Section 6.1.2, the choice of the random number source affects the number of transitions. We define the *activity factor* A as a number between 0 and 1 with the following properties. When $A = 1$ in Z, the number of transitions becomes maximum, and when it is 0, the number of transitions drops to the minimum. In the case of $Z = 1/2$, the maximum number of transitions will be N, where N is the length of the SN. This corresponds to an SN that transitions on every clock cycle, e.g., 01010101010... The SN $Z = 1/2$ with $A = 0$ corresponds to a bit-stream with the minimum number of transitions, e.g., 00000001111....

We are now ready to calculate the effect of transition errors. Following the analysis in [51], we observe that they cause output errors in two ways; they change the average value and the variance of Z^*. For the error-free number Z of length N and activity factor A we have

$$\mathbb{E}[Z^*] = \frac{1}{N} \cdot \mathbb{E}[Z \cdot N - N_{0\to1} \cdot e_{0\to1} + N_{1\to0} \cdot e_{1\to0}]$$

where $e_{i\to j}$ denotes the error rate on i-to-j transitions and $N_{i\to j}$ denotes the number of i-to-j transitions calculated by

$$N_{0\to1} = N_{1\to0} = 2 \cdot Z \cdot (1 - Z) \cdot A \cdot N$$

Hence

$$\mathbb{E}[Z^*] = Z + 2 \cdot Z \cdot (1 - Z) \cdot \mathbb{E}[e_{1\to0} - e_{0\to1}]) \cdot A \tag{6.2}$$

Equation (6.2) has two important implications. First, and more importantly, if $e_{0\to1} = e_{1\to0}$, then Z and Z^* will be equal on average. Second, the activity factor A has also a direct effect on the error. We will address impact of the activity factor in Section 6.1.2. For now, let us assume that $e_{0\to1} = e_{1\to0} = e$ is correct. Even though the expected value of Z^* is the same as the error-free Z, we may still see random

fluctuations due to the probabilistic nature of $e_{0\to1}$ and $e_{1\to0}$. Once again, if only deterministic signals are used, no random fluctuation will be seen. We compute the MSE using an approach similar to that in [51], i.e.,

$$MSE = \mathbb{E}[(Z-Z^*)^2] = \frac{1}{N^2}\cdot\mathbb{E}\left[\left(Z\cdot N-(Z\cdot N-N^*_{0\to1}+N^*_{1\to0})\right)^2\right]$$

in which $N^*_{i\to j}$ is a random variable denoting the number of i-to-j transition errors. We now have

$$MSE = \frac{1}{N^2}\cdot\mathbb{E}\left[(N^*_{0\to1}-N^*_{1\to0})^2\right] = \frac{1}{N^2}\cdot\left(\mathbb{E}[(N^*_{0\to1})^2]+\mathbb{E}[(N^*_{1\to0})^2]-2\mathbb{E}[N^*_{0\to1}\cdot N^*_{1\to0}]\right)$$

Since $N^*_{0\to1}$ is a binomial random variable with parameters $N_{0\to1}$ and e, we can evaluate the first term of the above equation by finding the second moment of $N^*_{0\to1}$.

$$\mathbb{E}[(N^*_{0\to1})^2] = e^2\cdot N_{0\to1}\cdot(N_{0\to1}-1)+e\cdot N_{0\to1}$$

and, since $\mathbb{E}[(N^*_{0\to1})^2] = \mathbb{E}[(N^*_{1\to0})^2]$, we get

$$MSE = \frac{1}{N}\cdot\left(4\cdot A\cdot Z\cdot(1-Z)\cdot e\cdot(1-e)\right) \tag{6.3}$$

Equation (6.3) has several important and counterintuitive implications:

- The errors due to voltage scaling can be reduced by increasing the number length N. While it is well known that increasing N reduces the random fluctuation errors in SC [76], Chen et al. have observed that when bit-flips are present, increasing N will not help [51].
- The activity factor A can also play an important role, since reducing A decreases the MSE.
- If the transition error rate $e = 1/2$, then the error is maximized. Obviously, if we set $e = 0$, we reduce the error to zero, but somewhat surprisingly, if we increase the error rate to $e = 1$, we also get $MSE = 0$. This is a scenario in which every transition of Z is erroneous, and since the numbers of transitions are equal, the errors all cancel each other out. Note that when $e = 1$, no random

fluctuation is seen in the circuit, and the circuit behaves deterministically. This scenario is similar to the desirable behavior shown in Figure 6.2(c).

Among the three preceding implications, increasing N is the least desirable because it increases the runtime of the circuit, and hence its energy consumption. The activity factor can be controlled by generating suitable SNs at the input (see Section 6.1.2). Balancing the transition errors $e_{0\to1}$ and $e_{1\to0}$ is the main target of this chapter, and will be addressed in the following sections.

We emphasize that Equations (6.2) and (6.3) are based on several simplifying assumptions and only reflect the result of delay errors on a single signal Z. In reality, the assumptions may not hold due to circuit complications. For example, the assumption of having $e_{0\to1} = e_{1\to0}$, which is based on Equation (6.2), leads to desirable error reduction, and cannot be easily achieved in real-world examples. We use Equation (6.2) as a guideline or ideal case that we want to approach. Similarly, Equation (6.3) sets guidelines for error reduction, some of which are not easily achieved. For instance, the $e = 1$ scenario where the final error becomes zero is never seen in our experiments. Also, $A = 0$ is another unachievable case that reduces the final error to zero. However, as we show next, reducing A decreases the final error.

6.1.2 Stochastic Number Generation

Stochastic number generation is an important step of a stochastic computation and directly affects parameters such as accuracy and area cost. It has been studied fairly extensively in the SC literature [101] [192]. While most of the early work on SC assumed that SNs must be (pseudo-)random, several recent studies suggest that deterministic SNs can also be successfully employed in SC [74] [8].

Equation (6.3) shows that the activity factor A of an SN directly affects its error. Interestingly, the activity factor also affects dynamic power consumption [183], so reducing A leads to both power and error reduction. While A cannot be completely fixed in a general SC circuit, it is possible to control it

to some extent by a careful choice of stochastic number generators. Figure 6.6 shows three SNs with different activity factors representing 1/2. The most commonly used method of SN generation, i.e., using (pseudo-)random number generators [101], yields numbers with medium activity factor. The SN generation method of [8] yields SNs with high activity factors, and hence is not suitable for the purposes of this chapter. The low activity factor SN shown in Figure 6.6 is generated by the method of [74]. This number has only one transition, so it is very tolerant of transition errors and consumes very little dynamic power.

Figure 6.7 compares the impact of different SN generation methods on a voltage-overscaled SC circuit. The supply voltage of the circuit under test has been reduced from the nominal value of V_{dd} = $1.0V$ to $V_{dd} = 0.72V$. As a result, the output Z^* deviates from the correct output Z. As seen in the figure, the deviation is higher when a high-activity stochastic number generator (SNG) is used. Based on this observation, we employ the SN generation method of [74] to the extent allowed by the SC design. Note that SC circuits usually require independence (zero correlation) among their inputs, so in many cases it is not possible to generate all the inputs using the same method. In other words, we do not have complete control over the activity factors of all the signals.

Low activity factor $A \simeq 0$	0000000011111111
Medium activity factor $A = 1/2$	0010011010111100
High activity factor $A = 1$	0101010101010101

Figure 6.6: Three SNs with different activity factors representing $Z = 1/2$.

6.2 Problem Description and Proposed Solutions

This section defines the two main problems that are addressed in this chapter. An error estimation method, which allows quick evaluation of SC circuits under voltage/frequency scaling, is also discussed.

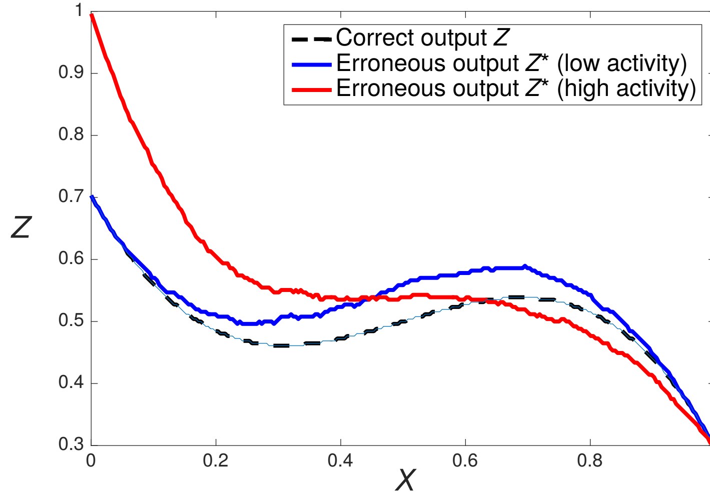


Figure 6.7: The effect of SN generation on output errors; they are lower when a low-activity SNG is used.

6.2.1 Finding the Minimum Energy Point

First, we pose and answer the following question: "Given an SC circuit, what is the lowest energy required for computation with a given required accuracy (err_{goal})?" To quantify the accuracy of a circuit, several error metrics, such as maximum error, MSE, etc. can be employed. From this point forward, we use the "average error" metric

$$err = \mathbb{E}[|Z - Z^*|]$$

where Z is the correct or "golden" value of the circuit output and Z^* is the erroneous output. The difference between Z and Z^* is averaged over all possible inputs. This average-error metric will be used to measure the accuracy of both SC and conventional binary circuits. It is important to note that our choice of average error as the accuracy metric is because of its simplicity. The general approach that we propose below is not limited to a specific error metric. However, due to the probabilistic nature of SC circuits, all of the deduced error bounds will be probabilistic.

The length N of the SNs used in a stochastic computation controls the accuracy and the total energy consumed by the circuit. Thus, by decreasing N, one can trade away accuracy for energy or

power savings. This natural tradeoff has been successfully used in the past [12]. Our work here shows that voltage/frequency scaling adds new dimensions to the accuracy-power tradeoff possibilities for SC circuits. In effect, SC circuits have three control knobs – (i) supply voltage V_{dd}, (ii) clock frequency f, and (iii) bit-stream length N (or, equivalently, clock cycle count) – that determine their accuracy and energy/power consumption. Finding the best operating point for a circuit is thus a new and challenging problem.

Previous methods search for the minimum N for which the average error is less than the given err_{goal}. However, as discussed, it is possible to adjust all three parameters (supply voltage V_{dd}, clock frequency f, and SN length N) concurrently in order to find the best solution. We will refer to the triplet (V_{dd}, f, N) as an *operating point* of an SC circuit. We now formalize the above question in the following problem statement.

Minimum-Energy Operating Point (MEOP) Problem. Given an SC circuit, find the operating point (V_{dd}, f, N) that has minimum energy consumption while satisfying the accuracy requirement of average error $\leq err_{goal}$.[41]

In addition to providing a solution to the MEOP problem, which we do in Section 6.2.2 below, we also consider optimization to improve error behavior under voltage scaling conditions. Excessive supply-voltage downscaling and/or increase of the operating frequency can result in the misalignment of signal *actual arrival times* (AAT) at output z with respect to the clock capture phase. Without loss of generality, for any pair of timing paths from inputs x_i and x_j to output z in an SC circuit, we assume the corresponding arrival times at z are AAT_i and AAT_j, respectively, such that $k_i \cdot T \leq AAT_i \leq (k_i + 1) \cdot T$ and $k_j \cdot T \leq AAT_j \leq (k_j + 1) \cdot T$, where T is the clock period. We say that these two timing paths exhibit

[41] It is practically impossible to search the entire solution space given that f and V_{dd} are continuous, and N is an arbitrary integer. Here "minimum energy" refers to the minimum energy point among a given set of (V_{dd}, f, N) combinations. Since varying N has been extensively studied in the literature, we only consider one choice of N in our search space. This means that the energy savings reported in this chapter are obtained only from voltage/frequency scaling.

arrival time misalignment if $k_i \neq k_j$. In other words, the two signals cannot be captured in the same clock cycle. We will show that the arrival time misalignment has significant impact on computation accuracy for SC circuits. Figure 6.8 shows one example of two timing paths (arcs $x_1 - z$, $x_2 - z$) to illustrate that arrival time alignment matters. In the example, Case (a) assumes no timing violation for both paths. This case generates the correct output sequence 1, 0, 1. Case (b) has timing violations on both timing paths. However, the two arrival times are captured within the same clock cycle (i.e., T_2). Therefore, there is no arrival time misalignment. Although both signals are delayed by one cycle, the output sequence at z is still correct, i.e., it is 1, 0, 1.[42] In Case (c), due to unbalanced path delay, signals from x_1 and x_2 arrive at z in two different cycles. Thus, Case (c) has an arrival time misalignment which leads to a computation error, as shown by the red-dotted oval in Figure 6.8. Moreover, the output sequence cannot be recovered by adjusting the capture phase.

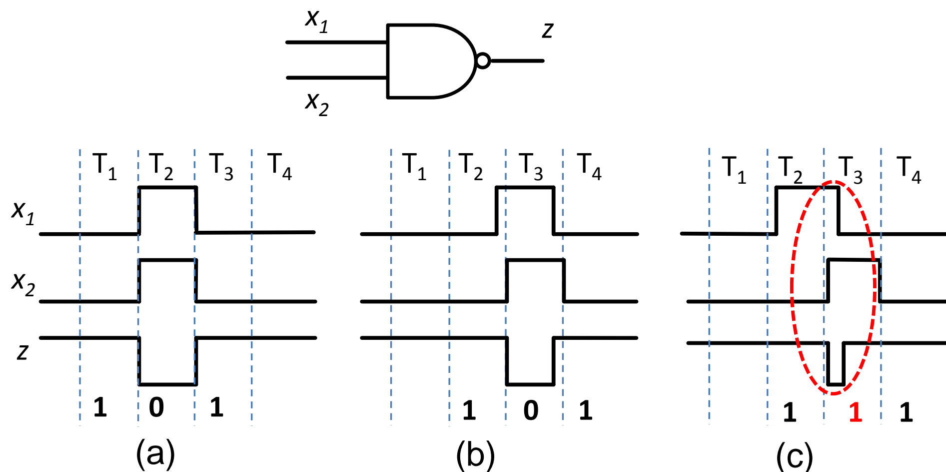


Figure 6.8: Misalignment of arrival times at z with respect to the clock capture phase can lead to a computation error.

Motivated by the discussion above, we propose to employ logical and physical design techniques to align the arrival times at the output of an SC circuit. As observed in Equation (6.2), it is desirable to have equal error rates on 0-to-1 and 1-to-0 transitions ($e_{0\to1}$ and $e_{1\to0}$), because balanced errors reduce

[42]In Case (b), the output at the end of T_2 (derived from propagated signals in T_1 of Case (a)), can be incorrect. However, the corresponding impact on computational accuracy is negligible given that N is typically large, e.g., $N = 4{,}096$.

the average error. Accordingly, we define the following problem statement, whose solution is discussed in Section 6.3 below.

SC Circuit Optimization (SCOpt) Problem. Given a stochastic function and a range of supply voltages, find a circuit implementation that has minimum average error across the given supply voltage range.

6.2.2 Solution of the MEOP Problem

We now present our solution to the MEOP problem defined in the previous section. Briefly, given an SC circuit, we want to find the most energy-efficient operating point (V_{dd}, f, N) for a given accuracy metric. Our approach to this problem is a straightforward search within the operating-point space. In other words, we try different operating points and, for each, evaluate the accuracy and energy of the corresponding circuit. We then choose the point that has the lowest energy while satisfying the accuracy requirements.

Unlike conventional binary circuits, errors in SC circuits tend to cancel each other out. In addition, SC circuits have a non-deterministic nature, i.e., their behavior can be described by probabilities. For these reasons, evaluating the accuracy of SC circuits is *not* trivial. Exhaustive simulation can be used to evaluate the accuracy of small stochastic circuits. However, for larger circuits it is impractical to perform exhaustive simulation for every operating point. With this in mind, we propose a method for fast error estimation. We note that our search strategy is not novel, but we are proposing a new model that enables fast exploration of the solution space.

6.2.3 Error Estimation using a Markov Chain

We propose a Markov chain (MC) model [80] to estimate errors. This model assumes that an SC circuit involving timing errors can be in *correct* or *incorrect* states. In a correct state, the circuit is

producing the same output as the circuit with no timing errors. Since there are two possible output values, we have two correct states: C_0 in which the output is 0, and C_1 in which the output is 1 (Figure 6.9). In addition to the correct states, there are four incorrect ones. In an incorrect state, the SC circuit is producing an incorrect result due to a timing violation. Timing violations occur in two forms: (i) delay errors that appear when a 0-to-1 or 1-to-0 transition is missed at the output, and (ii) glitches that appear when the output was not supposed to have a transition. We distinguish between these two error types and allocate different states to them. State D_i ($i \in \{0,1\}$) is a state in which the output is the incorrect value i due to a delay error, and G_i is a state caused by a glitch in the output signal. Table 6.1 summarizes the MC model states.

Table 6.1: Description of each state in the MC model.

Term	Meaning
C_0	Output is 0 and is correct
C_1	Output is 1 and is correct
D_0	Output is 0 and is incorrect due to a delay error
D_1	Output is 1 and is incorrect due to a delay error
G_0	Output is 0 and is incorrect due to a glitch
G_1	Output is 1 and is incorrect due to a glitch

The edges of the MC model indicate the transition probabilities between the states. For simplicity, we only show edges for the error cases and assume that the output magnitude is 0.5. In general, the magnitude of the output also affects the transition probabilities. Furthermore, there are implicit edges that are the complements of the shown edges and land on correct states. For instance, the implicit edge that goes from C_0 to C_1 is the complement of the edge that goes from C_0 to D_0, and so has transition probability $1 - p_{e1}$. As an example, let us assume that $p_{e1} = 0.1$. This means that if the circuit is in state C_0, and the next output is going to be 1, there is a 10% chance that the output transition is not captured

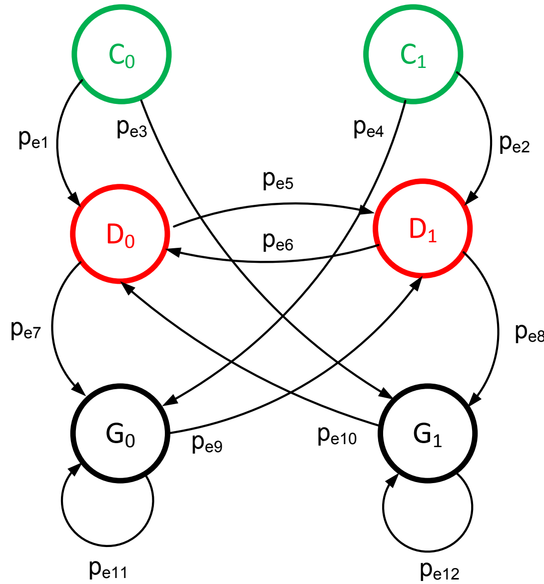


Figure 6.9: Markov chain (MC) model for the proposed error estimation approach. The states are described in Table 6.1.

due to a delay error, and hence the circuit lands in D_0 with probability p_{e1}. The other 90% of the time, the transition is successfully made and the circuit goes to the correct state C_1.

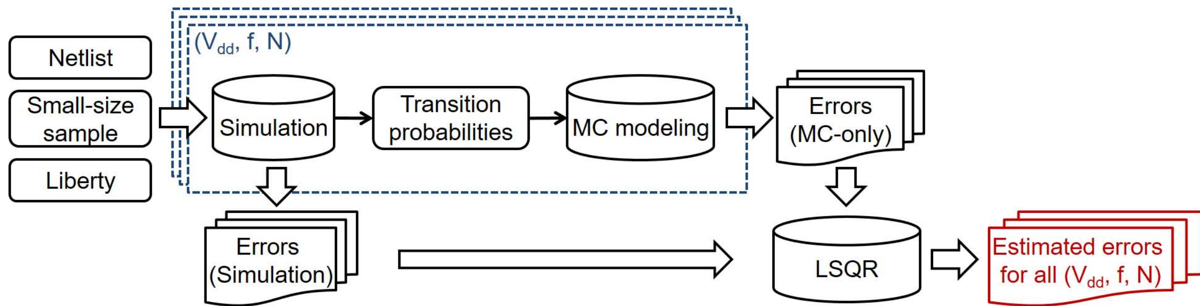


Figure 6.10: Flow to construct MC model and estimate computation errors across various operating points.

If the transition probabilities are known, we can find the equilibrium probability, i.e., the stationary distribution, of the MC and then we can evaluate the accuracy of the circuit in question. We do so by calculating the probability of seeing a 1 at the output of the circuit, i.e., the probability of being in states C_1, D_1, or G_1, and comparing it with the correct output probability. Figure 6.10 illustrates our MC-based error estimation flow. We obtain the transition probabilities by generating a small input sample set

(uniformly selected from the input space) and simulating the circuit. We perform logic simulation based on gate-level netlists. We then gather statistics for the transition rates between different states of the MC model. The size of the sample set determines the tradeoff between simulation runtime and the accuracy of the constructed MC model. We gradually increase the input sample size and find that in our testcases, the transition probabilities always converge when the input sample size is less than or equal to 20 (increasing the sample size to 21 does not lead to a significant change in the collected data). We therefore use 20 input samples in our experiments. Once the transition probabilities are estimated, we plug them into the MC model of Figure 6.9 and evaluate the accuracy of the circuit.

Our experimental results show that although the estimated errors from our MC model correlate well with the actual errors from simulation, they are typically pessimistic. We therefore apply a *least squares regression* (LSQR) technique to improve the estimation accuracy. The LSQR step uses the final error values observed during the simulations. To further clarify, we simulate the circuits for 20 evenly distributed input samples from the big space of possible inputs. We collect two data sets from the simulations: (i) transition probabilities that are used to construct the MC model and (ii) final error values that are used to correct the MC model. Figure 6.11 shows an example where the LSQR technique improves the estimation accuracy. The MC model enables fast design space exploration by avoiding exhaustive simulation. Note that the MC model is constructed only once for each (design, operating point) combination.

We verify our modeling flow by comparing the average error values predicted by the MC model (i.e., MC + LSQR) and by post-layout simulation. We use six representative testcases throughout this chapter. These testcases are mostly from image processing and artificial neural network applications. A list of the testcases appears in Section 6.4. Although the MC model can be employed in all cases, it is mostly useful for the bigger testcases (*GammaCorrection* [191], *Neuron* [26], *Sigmoid* and *BilateralFilter*) since exhaustive simulation would be time consuming in these cases.

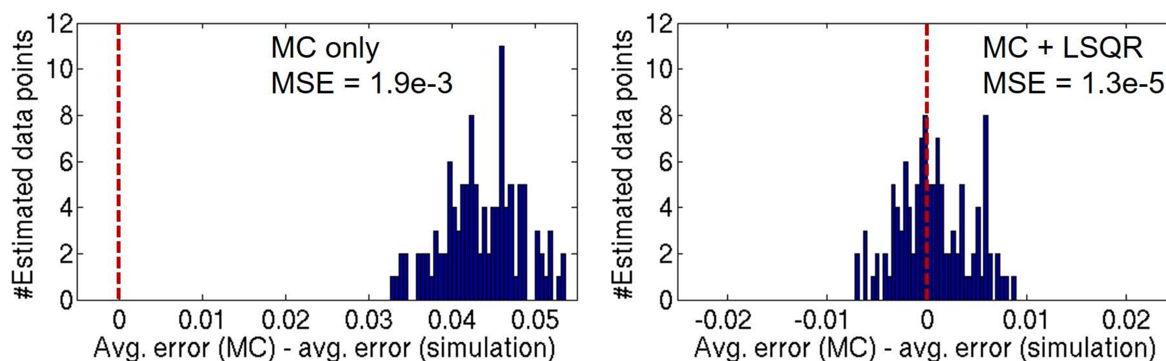


Figure 6.11: Distribution of the estimation errors across different operating points. Left: Estimation from the MC model without LSQR (i.e., the method proposed earlier in [6]) is pessimistic. Right: Application of the LSQR technique significantly improves the estimation accuracy.

After logic synthesis, placement and routing (SP&R), each testcase is simulated in *Cadence NC-Verilog* [236] with delays that are annotated from the SP&R flow results. To show the ability of the MC model to predict errors under aggressive voltage and frequency scaling, the circuit is signed off at V_{dd} = $1.0V$, worst process corner, 125°C; it is then operated at lower voltages (V_{dd} = {0.6, 0.64, ...0.96}V) and boosted clock frequencies (e.g., up to 5× of the signoff frequency).

Figure 6.12 shows that the predicted average errors are well-correlated with the post-layout simulation results of the majority of the testcases. The MC model does not perform well for small testcases (*EdgeDetection* and *PolySmall*) mainly because they exhibit deterministic behavior (especially the *EdgeDetection* testcase). But as noted, the MC model may not be useful in small testcases, since exhaustive simulation is feasible and fast.

The estimation error is relatively larger in the low-error cases (e.g., *GammaCorrection*) compared to the high-error cases (e.g., *Neuron*). The low-error cases happen when the transition errors $e_{0\to1}$ and $e_{1\to0}$ are either very small or very large. In such cases, the behavior of the circuit becomes mostly deterministic. For example, as discussed along with Equation (6.3), when $e_{0\to1} = e_{1\to0} = 1$, all the

transitions acquire an error, yielding a zero error for the SN. Our MC model's main limitation is that it cannot model such deterministic scenarios.

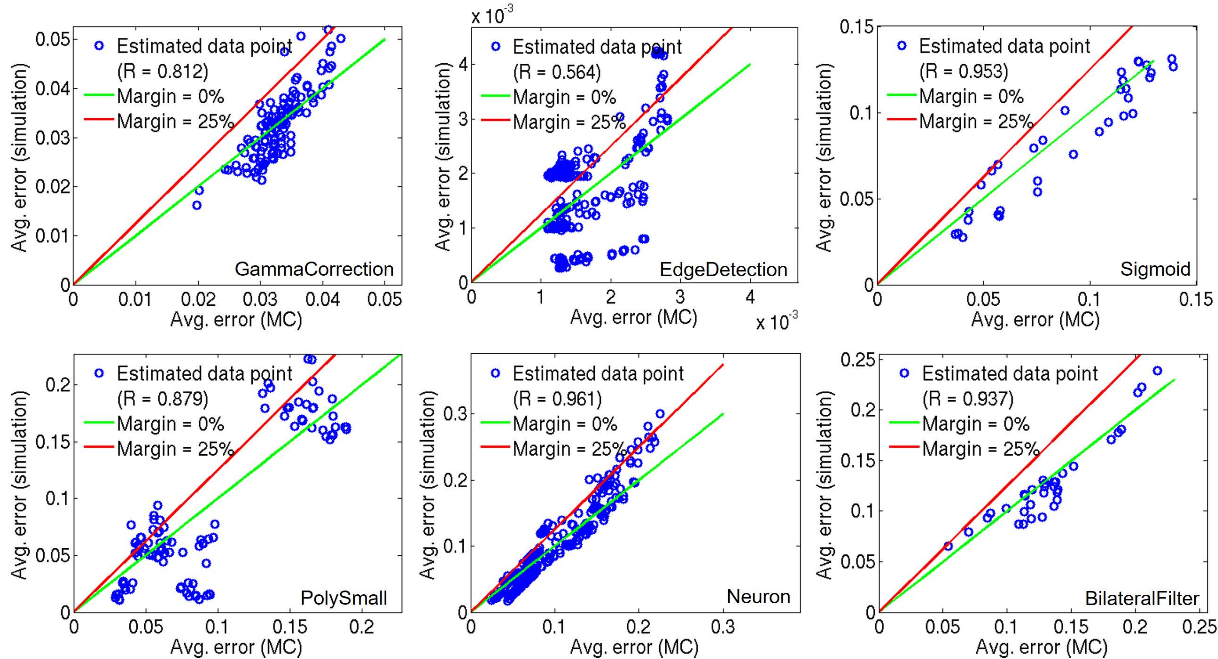


Figure 6.12: Plots showing correlation between the errors estimated by our proposed Markov chain (MC) model and the errors obtained from post-layout simulations. A 25% margin added to the MC estimated errors is sufficient to guard against small discrepancies for most of the testcases.

Furthermore, the MC error estimation involves some discrepancies, as seen in Figure 6.12. We observe that in most of our testcases, a 25% margin is sufficient to guard against the discrepancies of the larger testcases. However, this makes the MC model pessimistic; we may discard acceptable operating points. As we discuss in Section 6.4, using the MC model does not always find the optimum operating point that is found via exhaustive simulation. Nevertheless, we believe that the MC model is still useful for the following scenarios: (i) estimating the error of a big circuit for which exhaustive simulation is impossible, and (ii) exploring the huge space of operating points.

To further clarify the second scenario, consider an MEOP problem with $err_{goal} = 0.01$ on a relatively large circuit. Our approach is to search the space of possible operating points and choose one

that meets the err_{goal} with the least amount of energy consumption. For each operating point, we first run the MC model to quickly assess its error behavior. If the MC model shows a high error, say 0.1, then we can safely dismiss the current operating point and move on to the next one. Otherwise, we perform exhaustive simulation on the operating point to precisely assess its error behavior. Thus, the MC model saves valuable simulation time for many operating points.

6.3 Methodology for Circuit-Level Optimization

The previous section dealt with a scenario in which an SC circuit is already implemented and we can only choose an operating point for it, i.e., the MEOP problem defined in Section 6.1. In this section, we consider how to optimize the circuit using logic synthesis and physical design techniques to improve its energy efficiency (the SCOpt problem). We first discuss the timing behavior of SC circuits and highlight the main causes of errors, as well as the opportunities to eliminate them in Section 6.3.1. We then discuss the proposed optimization methods in Section 6.3.2.

6.3.1 Arrival Time Misalignment Matters

To examine the impact of arrival time misalignment on computation accuracy in an SC circuit, we insert and gradually increase the delays at the circuit's inputs; in the example shown in Figure 6.13, we sweep the delay from $0ps$ to $450ps$, i.e., $3\times$ the clock period, with a step size of $15ps$. We record the change in the average computation error. We perform this experiment on two implementations of testcase *PolySmall*, where one implementation uses the conventional P&R flow and the other is optimized to have more balanced path delays. Figure 6.13(a) shows the path delay distribution of the two implementations. Note that the initial designs have the maximum path delay around $140ps$. Therefore, the designs will have timing violations due to the inserted input delays.

The results in Figure 6.13(b) show that changing the input delay results in periodic fluctuation of computation accuracy, which indicates the impact of arrival time misalignment with respect to the capture phase. More specifically, when a large number of paths exhibit arrival time misalignment, e.g., when the delay ranges between 15*ps* to 65*ps* for the balanced case, the corresponding computation error is large. On the other hand, when there is no arrival time misalignment, e.g., when the delay ranges between 60*ps* to 150*ps* for the balanced case, although the design has larger timing violations, the computation error is small. Further, due to a wider range of path delays in the unbalanced case, the unbalanced implementation shows more data points with non-minimum average error (as seen in Figure 6.13(b)). Therefore, to reduce the likelihood of the misalignment of arrival times and to minimize the computation error, we propose buffer insertion and route detouring to minimize input-output path delay differences in SC circuits.

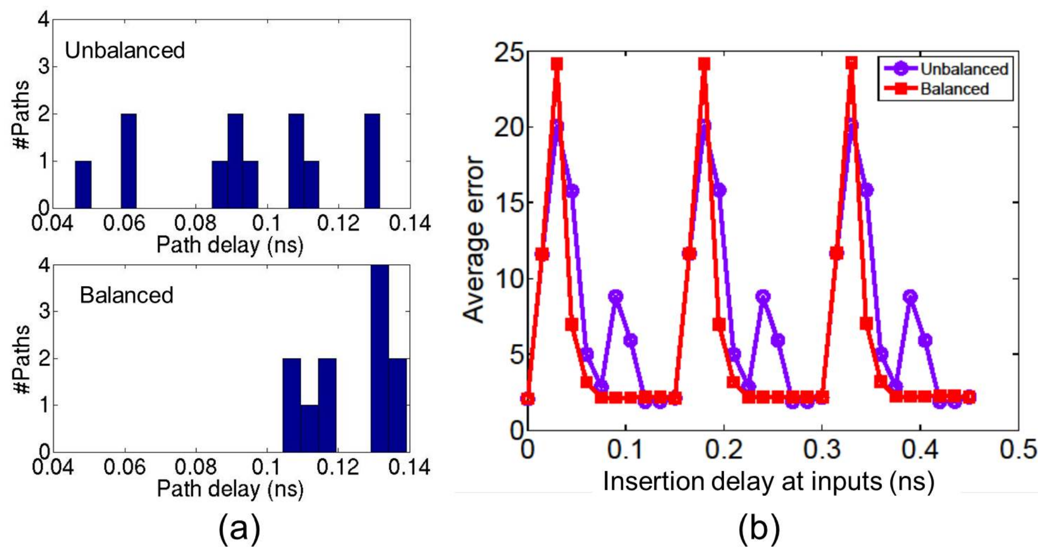


Figure 6.13: Simulation results of the *PolySmall* testcase synthesized in 28*nm* FDSOI technology and clocked at 6.7*GHz*: (a) path delay distributions of two implementations, and (b) computation error for different input delays.

6.3.2 Optimization Methodologies

To resolve the arrival time misalignment issue and reduce the computation errors at a low supply voltage or with an overscaled frequency, we perform optimization during SC circuit implementation (i.e., SP&R) to balance the circuit's path delays.

First, we examine two major SC design styles: STRAUSS (Spectral TRAnsform Use in Stochastic circuit Synthesis) [7] and ReSC (Reconfigurable Stochastic Computing) [191]. We then compare their path delays and computation errors for a given range of supply voltages. Figure 6.14 compares STRAUSS and ReSC for testcase *PolySmall* in 28*nm* FDSOI technology. We observe that the SC circuit implemented with ReSC tends to have more balanced path delays and smaller errors than that designed by STRAUSS. The ReSC architecture, which consists of an adder and a multiplexer, is very symmetric with respect to the primary inputs of the circuit. The STRAUSS-based circuits, on the other hand, have an asymmetric structure, which makes them smaller than the ReSC circuits, but leads to unbalanced path delays, and hence greater sensitivity to timing errors. We therefore only implement SC circuit designs based on ReSC in the experiments reported in the rest of this chapter.

To optimize the circuit, we perform buffer insertion and/or route detouring at the post-routing stage to balance path delays.[43] Various mathematical programming methods have been applied in the previous literature to guide the buffer insertion and wire sizing/route detouring for minimization of clock skew or data path delay [63] [86]. Given that SC circuits typically have small sizes[44], we formulate a *Mixed Integer-Linear Program* (MILP) to search for the optimal solution based on a given set of buffering candidates.

[43]We note that buffer insertion and route detouring techniques are not novel. They have been used to balance clock paths for skew minimization [86], and to balance signal paths to prevent power side-channel attacks in smartcards [210]. However, we appear to be the first to apply such optimization techniques to balance path (datapath) delays in SC circuits in order to improve their accuracy.

[44]Typical image-processing SC circuits have only around 20 gates [12], while the largest known SC circuits have no more than 1,250 gate instances [146].

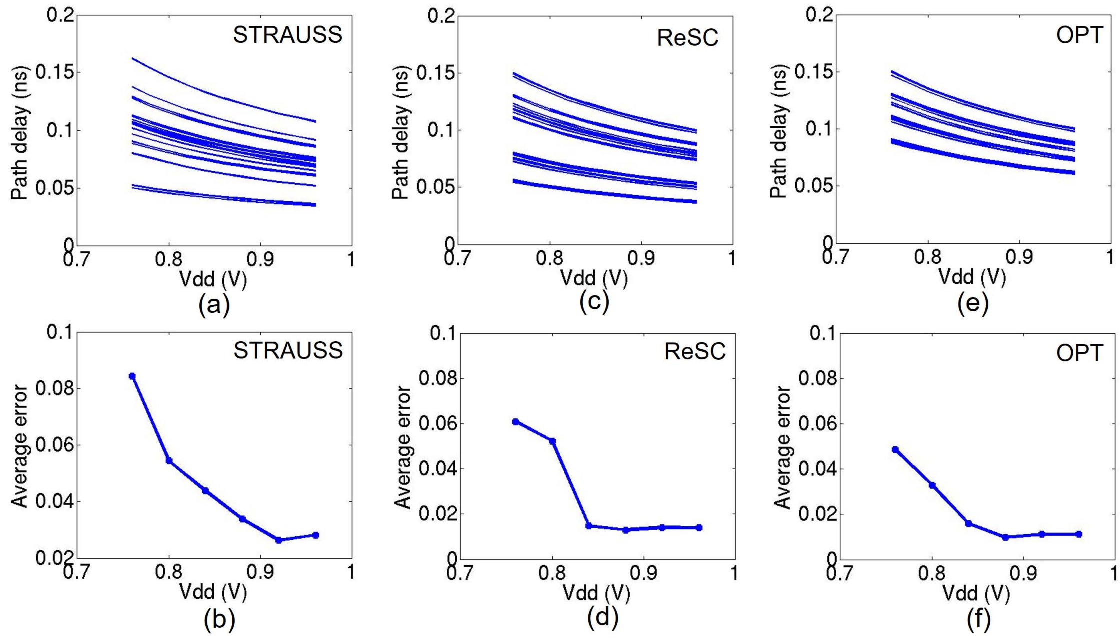


Figure 6.14: Path delays (top) and average computation errors (bottom) at supply voltages ranging from $0.72V$ to $0.98V$ for testcase *PolySmall* at 28*nm* FDSOI. Each trace in (a)(c)(e) denotes a timing path with a unique combination of rise/fall transitions. (a-b) STRAUSS [11]; (c-d) ReSC [191]; and (e-f) optimized circuit using our proposed MILP-based method.

We formulate our MILP as follows. The objective of the optimization is to minimize the normalized maximum delay difference (denoted by U) among timing paths of a design across a given range of supply voltages. Constraints are upper bounds on the maximum path delay and design leakage power. The notation used in the formulation is given in Table 6.2.

Table 6.2: Description of the notation used in the MILP.

Term	**M**eaning
V_k	Supply voltage, ($1 \leq k \leq K$; V_K is the highest voltage)
P_i	Timing path, ($1 \leq i \leq M$)
D_i^k	Path delay of P_i at V_k
U	Upper bound on maximum normalized delay difference
G^k	Leakage power of the design at V_k
n_r	Wiring net ($1 \leq r \leq R$)
d_j^k	Delay increase due to buffer insertion and/or routing at V_k, ($1 \leq j \leq Q$)
g_j^k	Leakage power penalty of buffer insertion choice at V_k, ($1 \leq j \leq Q$)
c_{rj}	Indicator of buffer insertion and/or routing detour on n_r
α	Normalized upper bound on delay increase
β	Normalized upper bound on leakage power penalty

$$\text{Minimize} \quad U \tag{6.4}$$

$$\text{subject to} \;\; D_i'^k = D_i^k + \sum_{1 \leq i \leq M, 1 \leq j \leq Q} c_{rj} \cdot d_j^k \tag{6.5}$$

$$\sum_{1 \leq j \leq Q} c_{rj} \leq 1, \qquad \forall\, 1 \leq r \leq R \tag{6.6}$$

$$D_{max}^k = \max_{1 \leq i \leq M} D_i^k,\ \forall\, 1 \leq k \leq K \tag{6.7}$$

$$\alpha \cdot D_{max}^k \geq D_i'^k, \quad \forall\, 1 \leq i \leq M,\ 1 \leq k \leq K \tag{6.8}$$

$$D_{max}'^k \geq D_i'^k, \quad \forall\, 1 \leq i \leq M,\ 1 \leq k \leq J \tag{6.9}$$

$$D_{min}'^k \leq D_i'^k, \quad \forall\, 1 \leq i \leq M,\ 1 \leq k \leq K \tag{6.10}$$

$$U \geq \frac{D_{max}^K}{D_{max}^k} \cdot (D_{max}'^k - D_{min}'^k) \tag{6.11}$$

$$\beta \cdot G^k \geq \sum_{1 \leq r \leq R, 1 \leq j \leq Q} c_{rj} \cdot g_j^k,\ 1 \leq k \leq K \tag{6.12}$$

where $D_i'^k$ is the optimized path delay of path P_i, with the buffer insertion and/or routing detour solu-

tion indicated by c_{rj}. D'^{k}_{max} and D'^{k}_{min} are, respectively, the maximum and minimum path delays at supply voltage V_k with buffer insertion and routing detour. U is the upper bound on the normalized path delay difference at all supply voltages. The MILP model minimizes U, thus minimizing the maximum normalized path delay difference at all supply voltages. In addition, G^k is the leakage power of the original design. The parameter g^k_j is the leakage power penalty of buffer insertion at supply voltage V_k. Our formulation constrains the optimization to not result in more than α times the original maximum path delay, or more than β times the original leakage power at each supply voltage.

Our initial studies attempted to include gate sizing and V_{th} swapping in the optimization process. However, this leads to a significant runtime and complexity increase in our MILP optimization, where each gate instance can have six to 22 candidate library cells for gate sizing and V_{th} swapping in the technology used. Furthermore, small sizing and/or V_{th}-swapping moves do not have a large impact on path delay, and so are not helpful in balancing path delays. On the other hand, large sizing and/or V_{th}-swapping moves might cause maximum-capacitance and maximum-transition violations due to weak drive strength for downsizing and/or swapping to a higher V_{th}, or large input pin capacitance for upsizing. We therefore only apply buffer insertion and/or route detouring in our optimization.

Figures 6.14(e)-(f) show the resultant path delays and computation errors of the optimized SC circuit. They indicate significant improvement over both the unoptimized STRAUSS and ReSC implementations.

There is a tradeoff between power overhead due to inserted buffers and wire segments versus the energy benefits from improved accuracy with more balanced path delays (e.g., greater supply voltage downscaling is enabled by more balanced path delays). Our optimization reduces energy only when the power benefits from voltage downscaling outweigh the power overhead due to inserted buffers and wire segments. A small design with simple netlist structure might already have relatively balanced path

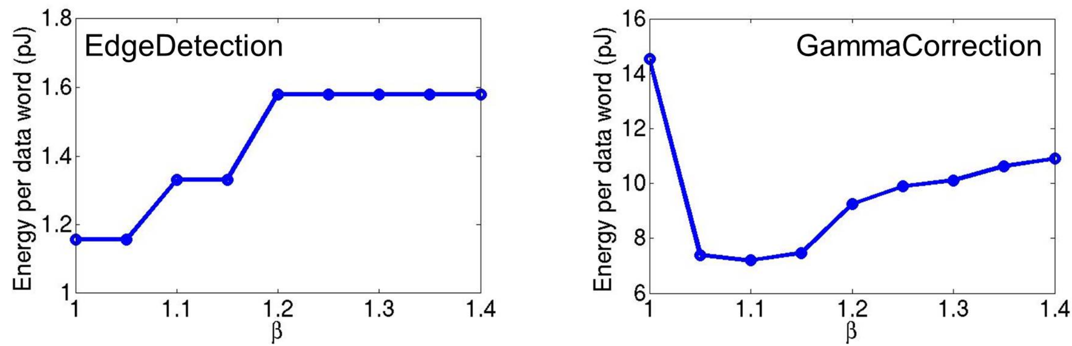


Figure 6.15: Energy of designs optimized with different β values. The target average error rate is 0.02. Operating points are selected based on exhaustive simulation so as to minimize energy.

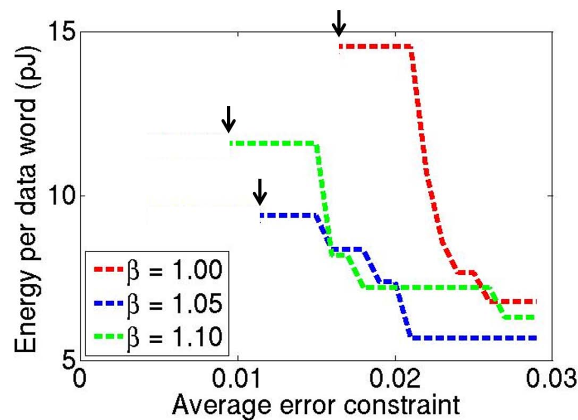


Figure 6.16: Energy comparison of three *GammaCorrection* circuits optimized with different β values. The optimum operating point for a given accuracy requirement (err_{goal}) is selected via exhaustive simulation. Black arrows indicate the minimum achievable error of each circuit.

delays and be sensitive to the power overhead of buffer insertion (i.e., the relative power overhead of buffer insertion is large), where the potential benefit from our optimization is small. On the other hand, power overhead due to buffer insertion is relatively small with respect to the total design power in a large design. Therefore, a large design is more likely to benefit from our optimization. Figure 6.15 shows the design energy optimized with different β values for a given accuracy requirement (i.e., err_{goal} = 0.02). We observe that for the small testcase *EdgeDetection* (with ∼5 instances), a larger β value always increases design energy.[45] On the other hand, for the relatively large testcase *GammaCorrection* (with ∼100 instances), the change of design energy with various β values shows a unimodal behavior due to the

[45]Given that the *EdgeDetection* testcase only has ∼5 instances, even insertion of the minimum-size buffer leads to relatively large power overhead.

tradeoff between the power overhead of buffer insertion and the energy benefits from improved accuracy. These results support our intuition that large designs are more likely to benefit from optimization.

Figure 6.16 compares energy use across different accuracy requirements for testcase *GammaCorrection*, which is optimized with different β values. The black arrows in Figure 6.16 indicate the minimum achievable error for each optimized circuit. We observe that a larger β value leads to higher accuracy. However, due to the tradeoff between power overhead of buffer insertion and route detouring versus energy benefits from improved accuracy, a higher β value does not necessarily provide smaller design energy. We therefore sweep the value of β to explore such a tradeoff and to minimize design energy. This optimization procedure is illustrated in Algorithm 4, in which we iteratively increase the value of β by δ (in our experiments $\delta = 0.05$) until there is no further energy reduction.

Algorithm 4 SC circuit optimization.

```
 1: β ← 1; Energy ← inf; is_improved ← true
 2: while is_improved do
 3:     Solve MILP; Perform buffer insertion and/or route detouring as ECOs
 4:     Perform exhaustive simulation to search for min-energy operating point for each err_goal
 5:     Calculate average energy over all err_goal; Update Energy
 6:     β ← β + δ
 7:     if Energy reduces then
 8:         is_improved ← true
 9:     else
10:         is_improved ← false
11:     end if
12: end while
```

To evaluate the influence of process corners and temperature variation, we characterize standard cell libraries in different corner cases (worst corner and best corner) and temperatures (125°C and 25°C) using *Synopsys SiliconSmart* [255]. We choose the same *GammaCorrection* testcase and simulate the circuits over different supply voltages. The result is shown in Figure 6.17. The differences among the corners are within 60% (normalized to the largest error at the same voltage) when the supply voltage decreases from $1.2V$ to $0.72V$. This results in a maximum of 5% change in the output error.

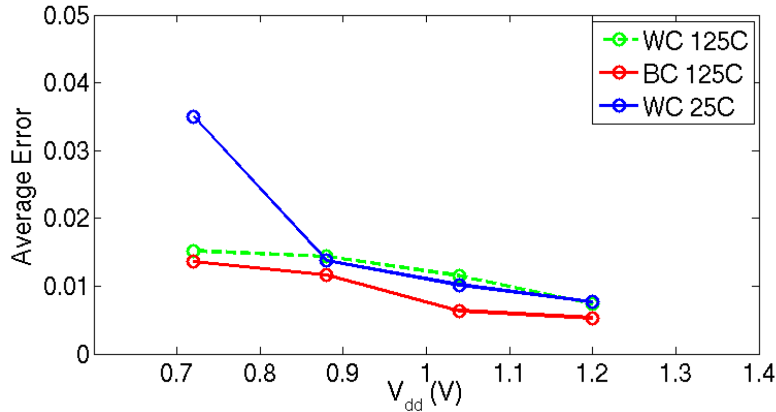


Figure 6.17: Evaluation of the impact of process and temperature variations on the average error. The testcase *GammaCorrection* is optimized at 125°C and simulated at different corners and temperatures.

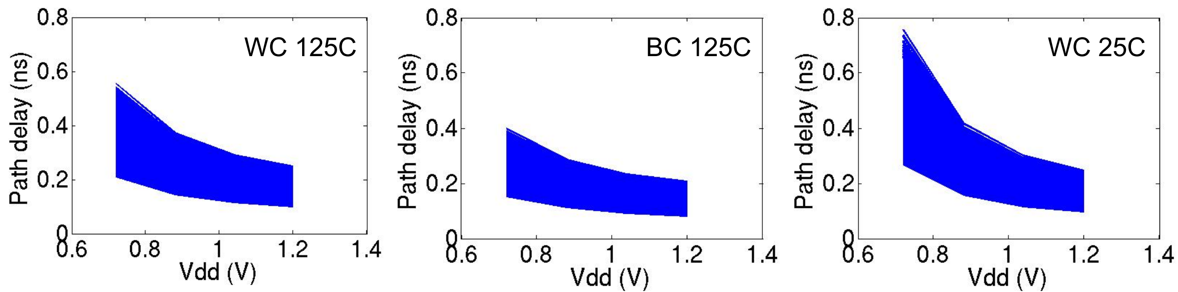


Figure 6.18: Path delay for the four different corners in Figure 6.17.

Figure 6.18 further shows the path delays at different supply voltages for various corners. We observe that due to smaller gate delays, the maximum path delay difference reduces in the best-corner cases, leading to smaller errors. In addition, lower temperature increases the maximum path delay difference (temperature inversion effect), especially at low supply voltages, which leads to larger errors as compared to the default (worst corner, 125°C) case.

To ensure the feasibility of ECOs (engineering change orders), we characterize lookup tables (LUTs) based on buffer insertion and/or route detouring candidates with different input slew and load capacitance values. We then formulate our MILP and optimize circuits based on the characterized LUTs. The approach is similar to that of [86]. To balance path delays at a range of supply voltages and reduce the MILP runtime, we select buffer insertion and/or routing detour candidates that cover a wide range of delay-voltage tradeoffs, but with a small set of choices. We study the delay-voltage tradeoffs with various

gate types, gate sizes, threshold voltages, and wirelengths. We observe that the delay-voltage tradeoff is greatly affected by threshold voltage, gate size and wirelength, which matches the observations made in [42]. Therefore, we apply two buffering styles—a single-stage non-inverting buffer, and an inverter pair with routing detour in between—as shown in Figure 6.19. Our approach selects from buffers and inverters of various sizes based on the delay requirements. We use both low V_{th} (LVT) and regular V_{th} (RVT) cells. The detoured wirelength, L, ranges from 10μm to 50μm with a step size of 10μm. Based on the LUTs, we further extend the buffering candidates with multiple cell stages (e.g., five stages of X100 buffers) to cover a wide range of delays. However, a large number of buffering candidates can significantly increase the MILP runtime. We therefore prune the candidates such that for a range of delay and delay-voltage tradeoffs, we uniformly divide the solution space into 4×4 sub-regions. We then select the buffering solution with minimum leakage power from each sub-region. Figure 6.20(a) shows the solution space with up to five stages of buffering candidates. Figure 6.20(b) shows the pruned buffering candidates with delay ranges from $20ps$ to $120ps$. Our experiments show that the pruning significantly reduces the runtime, while leading to negligible degradation in solution quality.[46]

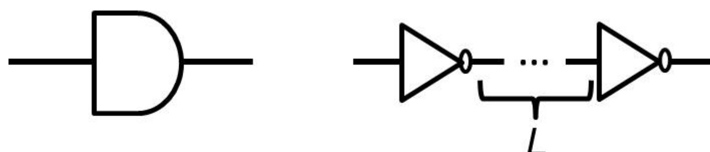


Figure 6.19: Applied buffering styles: a single-stage non-inverting buffer, and an inverter pair with routing detour.

Using the MILP solution, we perform buffer insertion and routing detour as ECO steps. Given that single-stage non-inverting buffer insertion is trivial, we use ECO commands from the P&R tools to perform buffer insertion and placement legalization. For insertion of an inverter pair with routing detour, we perform the ECO steps described in Algorithm 5. In the design flow, we start by inserting the first

[46]For the largest design with $\sim$500 gate instances, the MILP runtime is less than 20 seconds on a 24-core $2.5GHz$ Intel Xeon server.

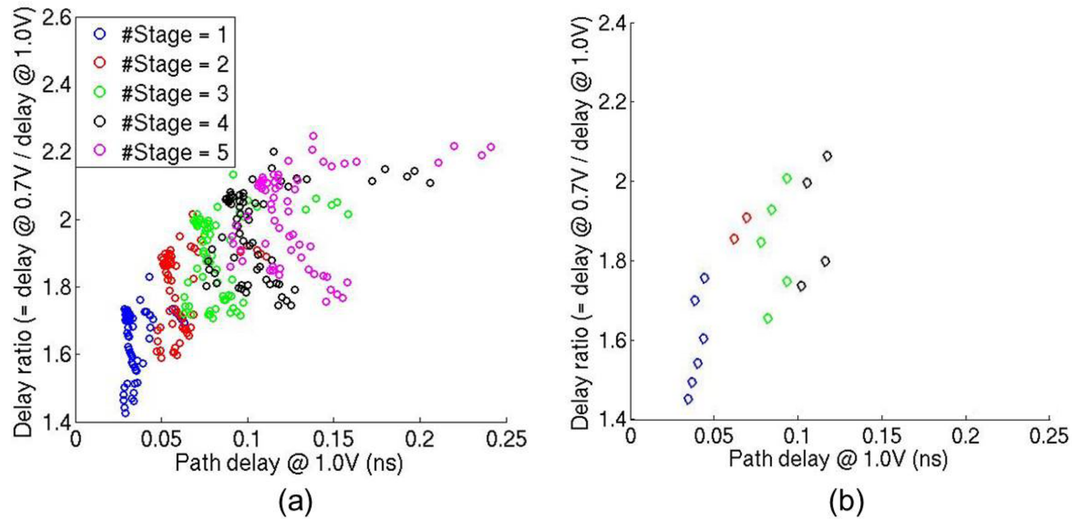


Figure 6.20: (a) Buffering solution space, i.e., delay range and delay-voltage tradeoff range, with multiple stages of buffers/inverter pairs. The circle colors denote different numbers of stages of buffers/inverter pairs. (b) Pruned buffering candidates.

inverter. We then legalize the location of the inserted inverter so that there is enough space for wire detour, e.g., by moving the inverter away from the block boundary, and to ensure there is no overlap with previous routing detours. We then insert the second inverter such that the distance is 25 sites in the horizontal direction and two rows in the vertical direction with respect to the first inverter. Last, we perform routing detour with the 1W2S (single-width double-spacing) routing rule on layers M3 and M4, between two inverters. An example of detoured routing is shown in Figure 6.21. Our current optimization method does not comprehend switching activity information. However, function-aware and input-pattern-aware optimization will be one of our future directions.

Algorithm 5 Insertion flow of inverter pairs.

1: Place first inverter
2: Legalize the location of the first inverter
3: Insert second inverter such that its distance to the first inverter is 25 sites and two rows in horizontal and vertical directions
4: Perform routing detour with 1W2S

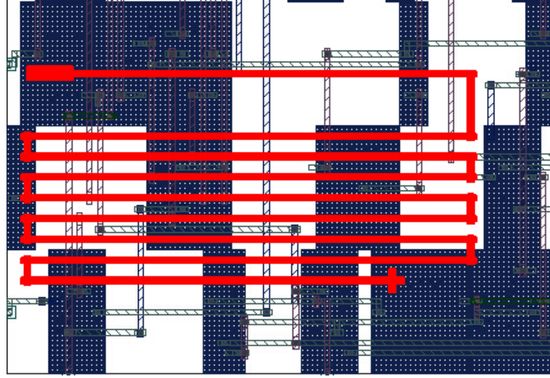


Figure 6.21: Layout of routing detour (in red). The detoured wirelength is 40μm. Shaded blocks are standard cells.

6.4 Experimental Setup and Results

The experiments are implemented in 28*nm* FDSOI technology. We synthesize the testcases using *Synopsys Design Compiler vH2013.03-SP3* [251], and place and route them using *Synopsys IC Compiler vI-2013.12-SP1* [252]. We use *Synopsys PrimeTime vH-2013.06-SP2* [253] and *Synopsys PT-PX vH-2013.06-SP2* for timing and power analyses, respectively. We perform gate-level simulation using *Cadence NC-Verilog v8.2* [236]. We construct the Markov chain model using *MATLAB R2013a* [245]. The MILP solver used in our optimization flow is *CPLEX v12.5* [241]. Our testcases (see Table 6.3) are representative circuits obtained from the SC literature and employed in typical applications such as image processing and neural network design. For input generation, we convert binary input vectors to pseudo-random bit-streams via SNGs.

6.4.1 Circuit Optimization Results

To evaluate the effectiveness of our optimization methods, we apply them to the testcases of Table 6.3 and compare the results with those of the unoptimized circuits. Figure 6.22, for example, shows how our optimization method changes the 0-to-1 and 1-to-0 error rates of the *GammaCorrection* testcase. It also shows that reducing the difference of the two error rates leads to output error reduction, even

Table 6.3: Summary of stochastic circuit testcases.

Testcase	#cells	Description
GammaCorrection	∼100	A common image processing task [191]
EdgeDetection	∼5	A common image processing task [12]
PolySmall	∼20	A simple polynomial of degree 3 implemented using methods of [7] and [191]
Neuron	∼500	A 128-input neuron [26]
Sigmoid	∼120	A common function used in artificial neural networks
BilateralFilter	∼300	An edge-preserving smoothing filter; used in image processing

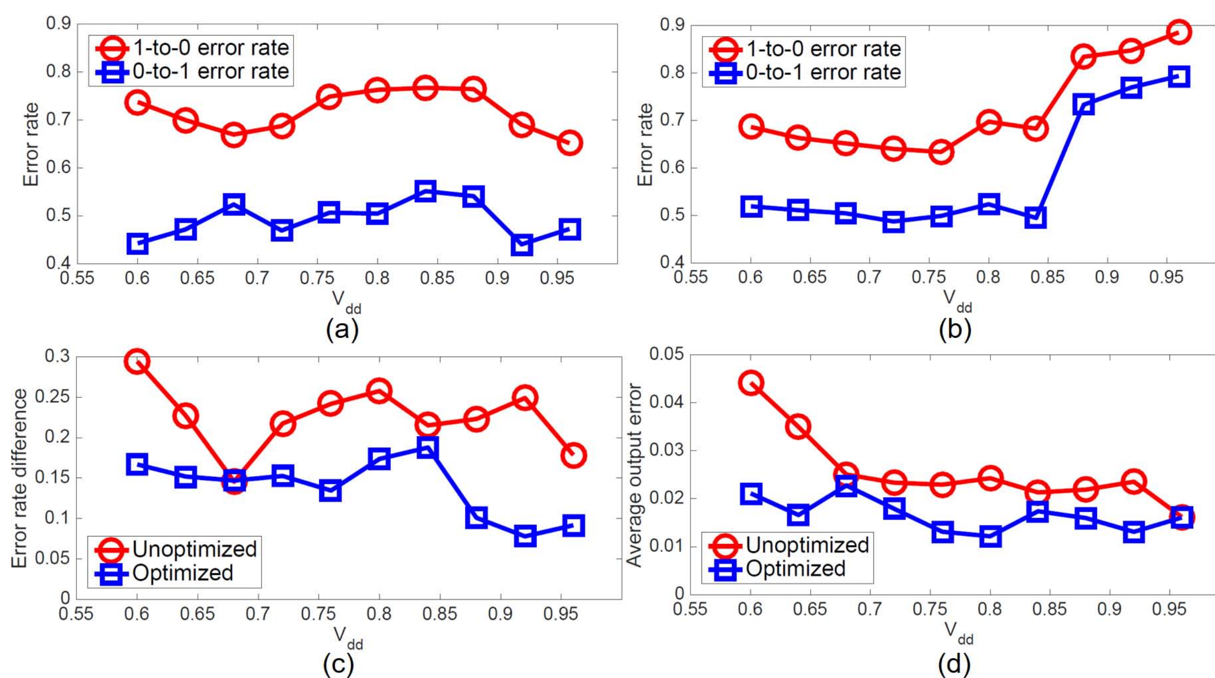


Figure 6.22: Optimization results for the *GammaCorrection* testcase: (a) 0-to-1 and 1-to-0 error rates of the unoptimized circuit; (b) 0-to-1 and 1-to-0 error rates of the optimized circuits; (c) error rate difference for both circuits, where the optimized circuit has a lower error rate difference even though the error rates are higher for some cases; and (d) average output error for both circuits, where the optimized circuit has a lower error.

though the error rates are increased. For example, for the voltage range V_{dd} = 0.88–0.96V, we see that the optimized circuit has more 0-to-1 and 1-to-0 errors. The final results shown in Figure 6.23 also confirm

the error reductions of the optimized circuits, i.e., optimized circuits have less energy per data word for a given fixed accuracy constraint. However, because the difference between the error rates is lower than that of the unoptimized circuit, we see a better output error behavior for the optimized circuit.

Figure 6.23 shows the minimum energy required for each design to meet a given average error constraint err_{goal} (within the space of possible operation points). A cross sign ($\times$) indicates that no suitable operating point was found for the given err_{goal}. Note that a circuit with a large number of inserted buffers (more balanced path delays) might have a small energy consumption when the error constraint is high because of voltage scaling, but it can have a large energy consumption when the error constraint is low due to the power penalty of the inserted buffers. Thus, it is difficult to find an optimized circuit that achieves minimum energy for all error constraints. To address this, we consider multiple optimized circuits, each optimized with a different β value. Note that we show different optimized circuits in Figure 6.23 to illustrate our optimization performance. Designers can choose their own accuracy requirements and use our method to find an optimized circuit tailored for their requirements. Furthermore, multiple accuracy-energy requirements are also supported in our optimization. A key observation here is that the optimized circuits are able to achieve lower err_{goal} values than the unoptimized circuits, especially for large testcases and/or tight error constraints..

In spite of the power overhead of added buffers and wires, the improved accuracy of the optimized circuits enables more aggressive voltage scaling, yielding lower power. We observe that when the error constraints are tight, the optimized circuits can meet the constraints at a lower V_{dd}, leading to significant energy savings. For example, up to 49% energy reduction occurs in the *GammaCorrection* testcase with $err_{goal} = 0.02$. In addition, the results show that tighter error constraints require larger β values (i.e., more balanced path delays) for circuit optimization (especially in the large testcases). This indicates that SC circuits with more balanced path delays are able to achieve higher accuracy. On the other hand, when

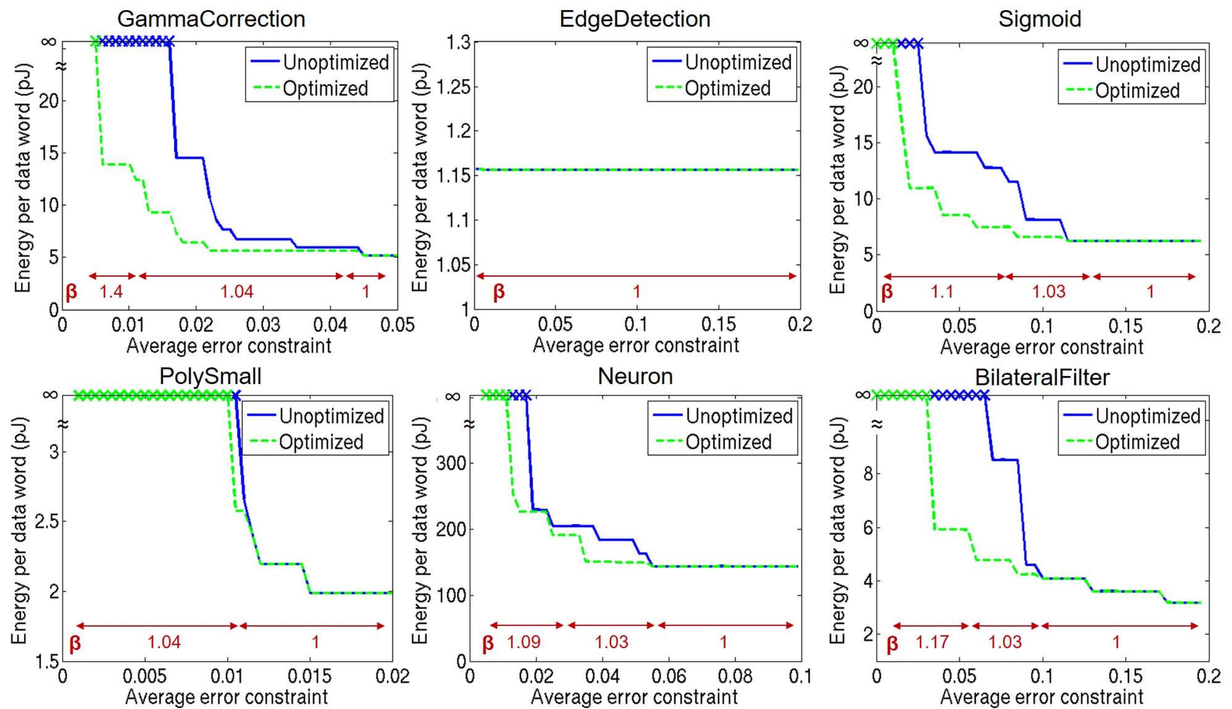


Figure 6.23: Energy comparison for different accuracy requirements (err_{goal}) between unoptimized implementations (blue solid line) and optimized circuits (green dashed line). Operating points are selected based on exhaustive simulation. The V_{dd} range is $0.6V$ to $1.0V$ with a step size of $4mV$. A cross sign ($\times$) indicates that no suitable operating point was found for the given err_{goal}. β values corresponding to the optimized circuits are shown in red font.

the error constraints are loose, or when the initial design is fairly well-balanced, e.g., the *EdgeDetection* testcase, the unoptimized circuits (i.e., with $\beta = 1$) can also perform satisfactorily at low supply voltages. In such cases, especially for circuits with a small number of instances, where the relative power overhead of buffer insertion is large, optimizations with buffer insertion and route detouring are not efficient because the inserted buffers and wires increase the overall energy consumption. Therefore, our optimization flow (illustrated in Algorithm 4) chooses not to insert any buffer or wire for the small testcases *EdgeDetection* and *PolySmall*, and the optimized circuits are the same as the unoptimized circuits.

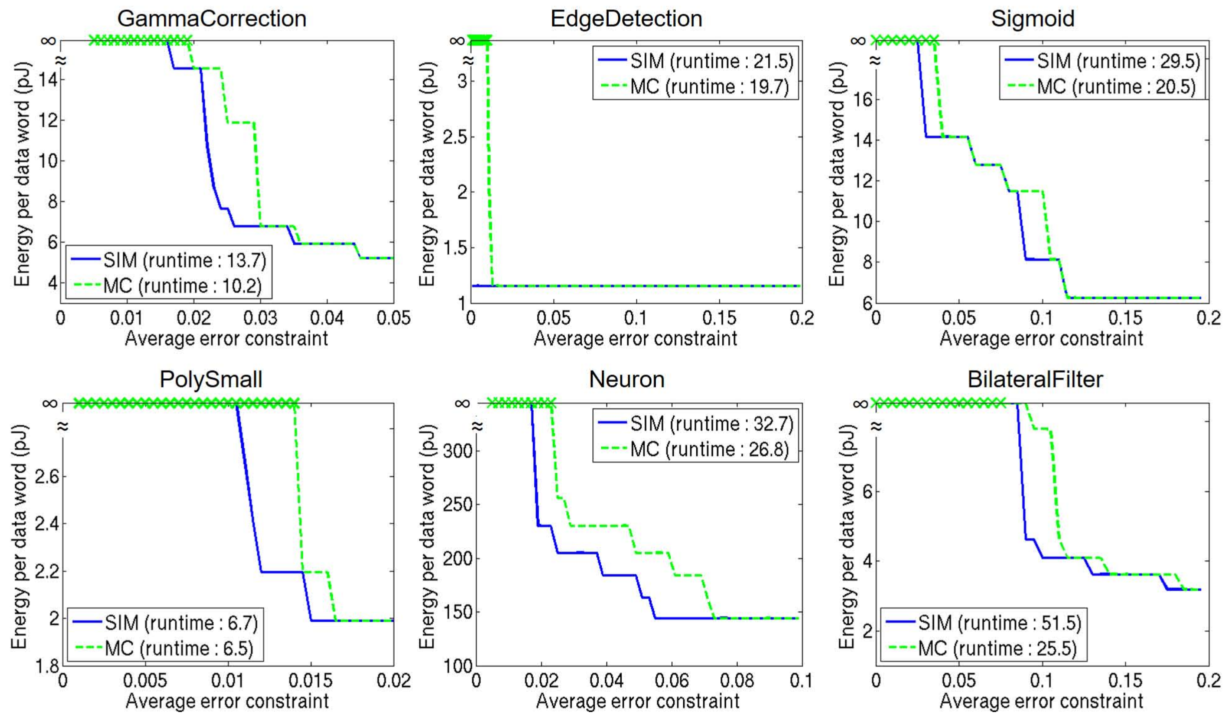


Figure 6.24: Energy comparison between operating points from MC-based search (green dashed line) and exhaustive search (blue solid line) for different accuracy goals (err_{goal}). A cross sign ($\times$) indicates that no suitable operating point was found. The MC-based approach uses far fewer samples for circuit simulation, thus significantly reducing the simulation runtime. (Runtime unit: minutes.)

6.4.2 Validation of MC Model

To gauge the effectiveness of our MC model, we perform an energy-accuracy comparison between the operating points selected by the MC-based flow and the ones selected via exhaustive simulation; see Figure 6.24. In this experiment, we only consider the unoptimized implementation of the testcases. The results show that our MC-based flow finds operating points that are similar in terms of energy to those selected via exhaustive simulations for most of the designs and error constraints. Moreover, Figure 6.24 shows that the MC-based flow reduces runtime significantly especially for large design, e.g., ~50% for *BilateralFilter*. We observe that the energy penalty of using the MC-based flow is relatively high for small designs (e.g., *PolySmall*) where the computation errors are typically small and the MC-based estimation is less accurate.

6.4.3 Comparison with Conventional Circuits

We now perform a comparison between the optimized SC *GammaCorrection* and *EdgeDetection* and their binary counterparts. Figure 6.25 shows images generated by conventional binary and SC circuits at different supply voltage levels. We use both the SNR (signal to noise ratio) and MS-SSIM (multi-scale structure similarity) [218] error metrics to quantify the quality of the output images.[47] The results show that while the SC circuits tolerate aggressive voltage scaling, the binary circuits' output quality quickly drops, even with modest voltage changes.[48] This leads to significant energy savings in the SC case. Figure 6.25 shows that for the *GammaCorrection* testcase, the SC circuit consumes more energy than the conventional binary circuit at $V_{dd} = 1$V. However at $V_{dd} = 0.6$V, the SC circuit achieves the same accuracy as the conventional circuit at $V_{dd} = 1$V, making it more energy efficient (about 44% less energy). Similarly, for the *EdgeDetection* testcase, the SC circuit at $V_{dd} = 0.6$V achieves the same accuracy as the conventional circuit at $V_{dd} = 0.9$V, thus achieving 95% energy reduction. In Figure 6.25, we also report the area and the runtime of the circuits. As expected, the SC circuits have lower area than their binary counterparts, but longer runtime.

6.5 Conclusions

We present several optimization and modeling methodologies to exploit voltage/frequency scaling in SC circuits for reduced energy consumption at the cost of timing errors. We also demonstrate that

[47] MS-SSIM evaluates the similarities of luminance, contrast and structure components between original image and processed image. We use the MATLAB function from [54] in our experiments to calculate MS-SSIM.

[48] SNR and MS-SSIM do not change monotonically with the supply voltage. There are several explanations for this. First, voltage reduction increases the rate of 0-to-1 and 1-to-0 errors ($e_{0\to1}$ and $e_{1\to0}$), but the changes are not necessarily linear, meaning that the difference $|e_{0\to1} - e_{1\to0}|$ may decrease due to voltage reduction, thus yielding a lower error. Second, based on Equation (6.3), the MSE decreases when e increases beyond 0.5. So a non-monotonic error behavior is not surprising. We also note that both *GammaCorrection* circuits (SC and binary) approximate the original gamma correction function (i.e., $Z = X^{0.45}$ [191]), and that the SC circuits have inherent random fluctuation errors. So even at a high supply voltage, some error exists. When timing errors are introduced via voltage reduction, they can cancel out the other errors and cause non-monotonic error behavior similar to that seen in Figure 6.25.

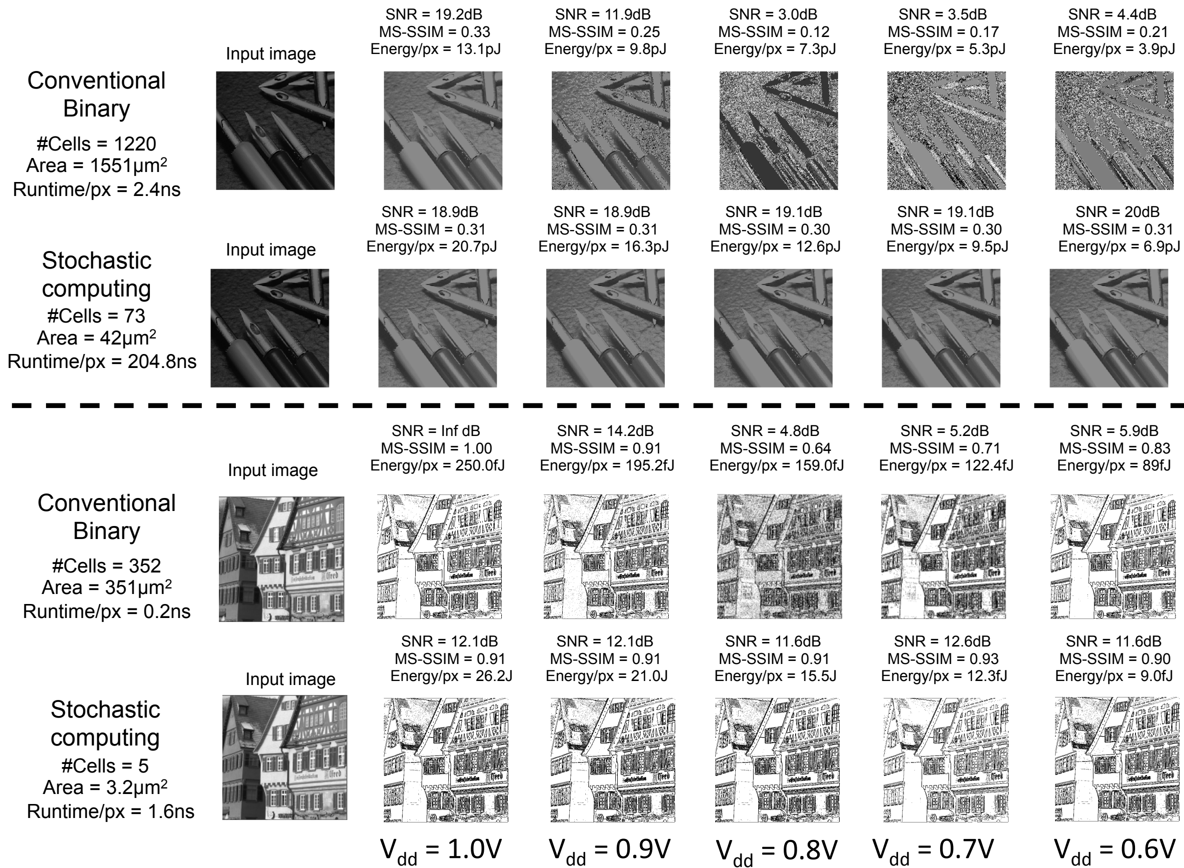


Figure 6.25: Voltage scaling results of *GammaCorrection* (top, design obtained from [191]) and *EdgeDetection* (bottom, design obtained from [12]) executed by conventional and stochastic circuits, both implemented in 28*nm* FDSOI technology. After applying our proposed optimization techniques, the stochastic circuits show better tolerance against aggressive voltage scaling. Test images are from [240] and [239].

SC circuits are extremely tolerant of timing errors. Hence, they can operate successfully under highly aggressive voltage/frequency scaling with very little loss of accuracy, unlike almost all conventional logic circuits. Based on these results, we define and solve the problem of finding the minimum-energy operating point of an SC circuit for a desired accuracy level. To enable rapid exploration of the operating-point space, we introduce a Markov chain-based technique for error estimation. We further observe that the accuracy of an SC circuit under scaled conditions can be improved by balancing its path delays. Accordingly, we perform optimizations during the logical and physical design phases to balance path delays.

These methods have been successfully applied to several representative SC circuits, achieving substantial energy reduction without significant accuracy loss. To determine the robustness of our optimization approach, we also demonstrate that process and temperature variation have little impact on the error behavior of the optimized SC circuits, even under voltage scaling.

6.6 Acknowledgments

Chapter 6 contains reprints of A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Optimizing Stochastic Circuits for Accuracy-Energy Tradeoffs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015; and A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Trading Accuracy for Energy in Stochastic Circuit Design", *ACM Journal on Emerging Technologies in Computing Systems* 13(3), 2017.

I would like to thank my coauthors Armin Alaghi, John P. Hayes, Andrew B. Kahng and Jiajia Li.

Chapter 7

Conclusion and Future Directions

This thesis presents several analyses and promising techniques for emerging physical design challenges. The identified and analyzed challenges are consistent with the latest scaling challenges projected by the EDA research community and semiconductor industries [159] [108] [243].

The first challenge discussed in Chapter 2 is the complication among active voltage scaling, signoff, and BEOL degradation. Before this thesis, the three issues are discussed separately, and we have shown the connections among them. To quantify the impact of the complication and provide guidance for signoff, Chapter 2 indicates that for signal wires, large drivers should be avoided as they suffer from more delay degradation due to EM-induced resistance increase. For P/G mesh, we quantify the impact of resistance increase and conclude that the area-power curve formed by different BTI signoff corners is shifted due to EM. We empirically analyze the cost of fixing EM at signoff with different guardbands against BTI. In our studies, this cost is up to 1.6% increase in area and 6% increase in power.

The second challenge discussed in Chapter 2 is the yield loss during process learning. We compensate this yield loss with opportunistic, redundant logic insertion. Our methodology for redundant logic insertion extracts candidate clusters with small terminal count using recursive bipartitioning, then maxi-

mizes duplicated logic by selecting clusters via integer-linear programming. Experimental results on our benchmark circuits show that for large design areas, logic redundancy can improve defect-limited yield by up to 1.62×.

Chapter 3 demonstrates the potential to tackle physical design problems by using machine learning techniques. We first present the miscorrelation of prediction against detailed routing results with conventional congestion estimation. To tackle the miscorrelation, the thesis demonstrates a learning-based model and improves the layout routability on top of the model. Our experiments show that we are able to reduce the number of DRC violations by an average of 20.6% and a maximum of 76.8%, with no adverse impact on design closure.

Chapter 4 proposes a methodology to estimate 3D benefits of given designs. We examine several designs with our "infinite dimension" bounding methodology, and demonstrate the area and power gaps between "best possible" 2D implementations and estimated upper bounds of 3D benefits. We further perform such 3D benefit estimation across various technologies. Our study also indicates that although 3DIC might provide relatively large benefits in power or performance, it is typically difficult for pure logic-logic 3D integration to achieve a simultaneous (10%, 10%, 10%) improvement in (performance, power, area/cost) compared to the conventional 2D implementation. This suggests that SoC-level and architectural-level optimizations instead of traditional P&R physical implementation optimizations are more essential for 3DIC.

Chapter 5 explores approximate computing. We propose an approach for output quality estimation of approximate designs. Our LUT-based approach characterizes the statistical properties of approximate hardware modules and a regression-based technique improves the error metric estimation. With our composition approach, we achieve 1.36× and 8.4× runtime improvements for library characterization

and error composition, respectively. We also achieve 3.75$\times$ accuracy improvement for error significance (ES) compared to previous works.

Chapter 6 explores stochastic computing. We present several optimization and modeling methodologies to exploit voltage/frequency scaling in stochastic circuits. We also demonstrate that SC circuits are extremely tolerant of timing errors. Hence, they can operate successfully under highly aggressive voltage/frequency scaling with very little loss of accuracy. We also introduce a Markov chain-based technique for error estimation. We further observe that the accuracy of an SC circuit under scaled conditions can be improved by balancing its path delays. Accordingly, we perform optimizations during logical and physical design to balance path delays. These methods achieve substantial energy reduction without significant accuracy loss.

Looking beyond this thesis, future directions and ongoing works include the following.

- Machine learning in design automation is not only a promising solution but also a new perspective to formulate problems. In the short term, there are many miscorrelations between the successive stages of the current design flow. Such miscorrelations have increased the difficulty of meeting design constraints, and the resulting uncertainties in the design process have increased risk in project management. As shown in previous works and this thesis, machine learning is a promising direction to compensate such miscorrelations.

  Many design tasks are still difficult to solve by automation, and rely heavily on experienced engineers' knowledge. Examples of these problems are floorplan and design of *power distribution network* (PDN). Recently, artificial intelligence has proved its success in many applications. With the help of artificial intelligence [68], design automation may enable the design framework of "no human in the loop", where design software can start from existing design database, generate representative training data, and improve quality of results by learning from data.

- 3DIC integration is a promising technology to continue area scaling. In this thesis, we have demonstrated its benefit bound. However, the 3D benefit considering memory architecture still remains unexplored. Furthermore, close integration of CMOS logic, emerging memory devices, and high-density interconnects has been demonstrated by recent work [202]. Pathfinding for such 3DIC integration, navigating a path from system specification to implementation while exploring a large range of design choices along the way, is still unaddressed.

- Inexact hardware's applications still remain open. Recent research strongly motivates the use of inexact hardware for neuromorphic computing. Determining system-level tradeoffs between energy efficiencies and accuracies of neuromorphic computing platforms is challenging due to a very large design space that includes such parameters as static/dynamic bit-width optimization, mapping to approximate or stochastic circuits, and truncation of arithmetic circuits in critical neurons.

Bibliography

[1] T. Adler and E. Barke, "Single Step Current Driven Routing of Multiterminal Signal Nets for Analog Applications", *Proc. Design, Automation and Test in Europe*, 2000, pp. 446-450.

[2] T. Adler, H. Brocke, L. Hedrich and E. Barke, "A Current Driven Routing and Verification Methodology for Analog Applications, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2000, pp. 385-389.

[3] S. N. Adya, I. L. Markov and P. G. Villarrubia, "On Whitespace and Stability in Mixed-Size Placement and Physical Synthesis" *Integration, the VLSI Journal* 39(4) (2008), pp. 340-362.

[4] A. M. Aguiar and S. P. Khatri, "Exploring the Viability of Stochastic Computing", *Proc. IEEE International Conference on Computer Design*, 2015, pp. 420-423.

[5] A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Trading Accuracy for Energy in Stochastic Circuit Design", *ACM Journal on Emerging Technologies in Computing Systems* 13(3) (2017), pp. 47:1-47:30.

[6] A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng and J. Li, "Optimizing Stochastic Circuits for Accuracy-Energy Tradeoffs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 178-185.

[7] A. Alaghi and J. P. Hayes, "STRAUSS: Spectral Transform Use in Stochastic Circuit Synthesis", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(11) (2015), pp. 1770-1783.

[8] A. Alaghi and J. P. Hayes, "Fast and Accurate Computation Using Stochastic Circuits", *Proc. Design, Automation and Test in Europe*, 2014, pp. 76:1-76:4.

[9] A. Alaghi and J. P. Hayes, "Exploiting Correlation in Stochastic Circuit Design", *Proc. IEEE International Conference on Computer Design*, 2013, pp. 39-46.

[10] A. Alaghi and J. P. Hayes, "Survey of Stochastic Computing", *ACM Transactions on Embedded Computing Systems* 12(2) (2013), pp. 92:1-92:12.

[11] A. Alaghi and J. P. Hayes, "A Spectral Transform Approach to Stochastic Circuits", *Proc. IEEE International Conference on Computer Design*, 2012, pp. 315-312.

[12] A. Alaghi, C. Li and J. P. Hayes, "Stochastic Circuits for Real-Time Image-Processing Applications", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 136:1-136:6.

[13] M. A. Alam, K. Roy and C. Augustine, "Reliability- and Process-Variation Aware Design of Integrated Circuits - A Broader Perspective", *Proc. IEEE International Reliability Physics Symposium*, 2011, pp. 4A.1.1-4A.1.11.

[14] L. Arnaud, P. Lamontagne, R. Galand, E. Petitprez, D. Ney and P. Waltz, "Electromigration Induced Void Kinetics in Cu Interconnects for Advanced CMOS Nodes", *Proc. IEEE International Reliability Physics Symposium*, 2011, pp. 3E.1.1-3E.1.10.

[15] K. Athikulwongse, D. H. Kim, M. Jung and S. K. Lim, "Block-Level Designs of Die-to-Wafer Bonded 3D ICs and Their Design Quality Tradeoffs", *Proc. Asia and South Pacific Design Automation Conference*, 2013, pp. 687-692.

[16] C. Bamji and E. Malavasi, "Enhanced Network Flow Algorithm for Yield Optimization", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 1996, pp. 746-751.

[17] M. Basoglu, M. Orshansky and M. Erez, "NBTI-Aware DVFS: A New Approach to Saving Energy and Increasing Processor Lifetime", *Proc. International Symposium on Low Power Electronic Design*, 2010, pp. 253-258.

[18] K. Bernstein, P. Andry, J. Cann and P. Emma, "Interconnects in the Third Dimension: Design Challenges for 3D ICs", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2007, pp. 562-567.

[19] A. Biddle, "Design Solutions for 20nm and Beyond", *Synopsys White Paper*, 2012.

[20] J. R. Black, "Electromigration Failure Modes in Aluminum Metallization for Semiconductor Devices", *Proceeding of the IEEE* 57(9) (1969), pp. 1578-1594.

[21] I. A. Blech, "Electromigration in Thin Aluminum Films on Titanium Nitride", *Journal of Applied Physics* 47(4) (1976), pp. 1203-1208.

[22] S. Bobba, A. Chakraborty, O. Thomas, P. Batude, V. F. Pavlidis and G. De Micheli, "CELONCEL: Effective Design Technique for 3-D Monolithic Integration Targeting High Performance Integrated Circuits", *Proc. Asia and South Pacific Design Automation Conference*, 2011, pp. 336-343.

[23] Y. Bourai and C.-J. R. Shi, "Layout Compaction for Yield Optimization via Critical Area Minimization", *Proc. Design, Automation and Test in Europe*, 2000, pp. 122-127.

[24] U. Brenner and A. Rohe, "An Effective Congestion Driven Placement Framework", *Proc. ACM International Symposium on Physical Design*, 2002, pp. 6-11.

[25] L. C. Briand, Y. Labiche and Z. Bawar, "Using Machine Learning to Refine Black-Box Test Specifications and Test Suites", *Proc. International Conference on Quality Software*, 2008, pp. 135-144.

[26] B. D. Brown and H. C. Card, "Stochastic Neural Computation. I. Computational Elements", *IEEE Transactions on Computers* 50(9) (2001), pp. 891-905.

[27] M. T. Buehler, J. M. Cohn, D. J. Hathaway, J. D. Hibbeler and J. Koehl, "Use of Redundant Routes to Increase the Yield and Reliability of a VLSI Layout", US patent 20060265684, 2007.

[28] W. P. Burleson, M. Ciesielski, F. Klass and W. Liu, “Wave-Pipelining: A Tutorial and Research Survey”, *IEEE Transactions on Very Large Scale Integration Systems* 6(3) (1998), pp. 464-474.

[29] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov and A. Zelikovsky, “On Wirelength Estimations for Row-based Placement” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18(9) (1999), pp. 1265-1278.

[30] A. E. Caldwell, A. B. Kahng and I. L. Markov, “Improved Algorithms for Hypergraph Bipartitioning”, *Proc. Asia and South Pacific Design Automation Conference*, 2000, pp. 661-666.

[31] A. E. Caldwell, A. B. Kahng and I. L. Markov, “Hierarchical Whitespace Allocation in Top-down Placement” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(11) (2003), pp. 1550-1556.

[32] V. Canals, A. Morro, A. Oliver, M. L. Alomar and J. L. Rosselló, “A New Stochastic Computing Methodology for Efficient Neural Network Implementation”, *IEEE Transactions on Neural Networks and Learning Systems* 27(3) (2015), pp. 551-564.

[33] J.-A. Carballo, W.-T. J. Chan, P. A. Gargini, A. B. Kahng and S. Nath, “ITRS 2.0: Toward a Re-Framing of the Semiconductor Technology Roadmap”, *Proc. IEEE International Conference on Computer Design*, 2014, pp. 139-146.

[34] A. Ceyhan, M. Jung, S. Panth, S. K. Lim and A. Naeemi, “Impact of Size Effects in Local Interconnects for Future Technology Nodes: A Study Based on Full-Chip Layouts”, *Proc. Interconnect Technology Conference/Advanced Metallization Conference*, 2014, pp. 345-348.

[35] T.-B. Chan, W.-T. J. Chan and A. B. Kahng, “ILP-Based Identification of Opportunistic Redundant Logic Insertions for Opportunistic Yield Improvement During Early Process Learning”, *Proc. IEEE International Conference on Computer Design*, 2017, pp. 269-272.

[36] T.-B. Chan, W.-T. J. Chan and A. B. Kahng, “On Aging-Aware Signoff for Circuits with Adaptive Voltage Scaling”, *IEEE Transactions on Circuits and Systems I* 61(10) (2014), pp. 2920-2930.

[37] T.-B. Chan, W.-T. J. Chan and A. B. Kahng, “Impact of Adaptive Voltage Scaling on Aging-Aware Signoff”, *Proc. Design, Automation and Test in Europe*, 2013, pp. 1683-1688.

[38] W.-T. J. Chan, K. Y. Chung, A. B. Kahng, N. D. MacDonald and S. Nath, “Learning-Based Prediction of Embedded Memory Timing Failures During Initial Floorplan Design”, *Proc. Asia and South Pacific Design Automation Conference*, 2016, pp. 178-185.

[39] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, “BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques”, *Proc. IEEE International Conference on Computer Design*, 2016, pp. 41-48.

[40] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, “3D-IC Benefit Estimation and Implementation Guidance from 2D-IC Implementation”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 30:1-30:6.

[41] W.-T. J. Chan, P.-H. Ho, A. B. Kahng and P. Saxena, “Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning”, *Proc. ACM International Symposium on Physical Design*, 2017, pp. 15-21.

[42] T.-B. Chan and A. B. Kahng, "Tunable Sensors for Process-Aware Voltage Scaling", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 7-14.

[43] W.-T. J. Chan, A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Statistical Analysis and Modeling for Error Composition in Approximate Computation Circuits", *Proc. IEEE International Conference on Computer Design*, 2013, pp. 47-53.

[44] W.-T. J. Chan, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Integration, the VLSI Journal* 53 (2017), pp. 226-235.

[45] W.-T. J. Chan, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2016, pp. 6:1-6:8.

[46] W.-T. J. Chan, A. B. Kahng and S. Nath, "Methodology for Electromigration Signoff in the Presence of Adaptive Voltage Scaling", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2014, pp. 1-7.

[47] W.-T. J. Chan, A. B. Kahng, S. Nath and I. Yamamoto, "The ITRS MPU and SOC System Drivers: Calibration and Implications for Design-Based Equivalent Scaling in the Roadmap", *Proc. IEEE International Conference on Computer Design*, 2014, pp. 153-160.

[48] K. Chang, K. Acharya, S. Sinha, B. Cline, G. Yeric and S. K. Lim, "Power Benefit Study of Monolithic 3D IC at the 7nm Technology Node", *Proc. International Symposium on Low Power Electronic Design*, 2015, pp. 201-206.

[49] M. L. Chang and S. Hauck, "Precis: A Design-Time Precision Analysis Tool", *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002, pp. 229-238.

[50] M. L. Chang and S. Hauck, "Variable Precision Analysis for FPGA Synthesis", *Proc. NASA Earth Science Technology Conference*, 2003.

[51] T. H. Chen, A. Alaghi and J. P. Hayes, "Behavior of Stochastic Circuits under Severe Error Conditions", *Information Technology* 56(4) (2014), pp. 182-191.

[52] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen and B. Han, "Full-Chip Routing Considering Double-Via Insertion", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27(5) (2008), pp. 844-857.

[53] T. H. Chen and J. P. Hayes, "Equivalence Among Stochastic Logic Circuits and Its Application", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 131:1-131:6.

[54] M.-J. Chen, C.-C. Su, D.-K. Kwon, L. K. Cormack and A. C. Bovik, "Full-reference Quality Assessment of Stereopairs Accounting for Rivalry", *Signal Processing: Image Communication* 28(9) (2013), pp. 1143-1155.

[55] X. Chen, Y. Wang, Y. Cao, Y. Ma and H. Yang, "Variation-Aware Supply Voltage Assignment for Simultaneous Power and Aging Optimization", *IEEE Transactions on Very Large Scale Integration Systems* 20(11) (2012), pp. 2143-2147.

[56] T.-Y. Chiang, B, Shieh and K. C. Saraswat, "Impact of Joule Heating on Deep Sub-Micron Cu/low-k Interconnects", *Proc. Symposium on VLSI Technology*, 2002, pp. 38-39.

[57] V. K. R. Chiluvuri and I. Koren, “Layout-Synthesis Techniques for Yield Enhancement”, *IEEE Transactions on Semiconductor Manufacturing* 8(2) (1995), pp. 178-187.

[58] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy and S. T. Chakradhar, “Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2010, pp. 555-560.

[59] V. K. Chippa, A. Raghunathan, K. Roy and S. Chakradhar, “Dynamic Effort Scaling: Managing the Quality-Efficiency Tradeoff”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2011, pp. 603-608.

[60] V. K. Chippa, S. Venkataramani, K. Roy and A. Raghunathan, “StoRM: A Stochastic Recognition and Mining Processor”, *Proc. International Symposium on Low Power Electronic Design*, 2014, pp. 39-44.

[61] M. Cho, H. Xiang, R. Puri and D. Z. Pan, “TROY: Track Router with Yield-Driven Wire Planning”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2007, pp. 55-58.

[62] I. S. Chong, H.-Y. Cheong and A. Ortega, “New Quality Metrics for Multimedia Compression Using Faulty Hardware”, *Proc. International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2006.

[63] C. C. N. Chu and D. F. Wong, “A Quadratic Programming Approach to Simultaneous Buffer Insertion/Sizing and Wire Sizing”, *IEEE Transactions on Very Large Scale Integration Systems* 18(6) (1999), pp. 787-798.

[64] H. Chun, Y. Yang and T. Lehmann, “Safety Ensuring Retinal Prosthesis with Precise Charge Balance and Low Power Consumption”, *IEEE Transactions on Biomedical Circuits and Systems* 8(1) (2014), pp. 108-118.

[65] J. Cong, C. Liu and G. Luo, “Quantitative Studies of Impact of 3D IC Design on Repeater Usage”, *Proc. International VLSI/ULSI Multilevel Interconnection Conference*, 2008, pp. 344-348.

[66] J. Cong, G. Luo, J. Wei and Y. Zhang, “Thermal-Aware 3D IC Placement via Transformation”, *Proc. Asia and South Pacific Design Automation Conference*, 2007, pp. 780-785.

[67] Y.-Y. Dai and R. K. Brayton, “Circuit Recognition with Deep Learning”, *Proc. International Symposium on Hardware Oriented Security and Trust*, 2017, pp. 162-168.

[68] Intelligent Design of Electronic Assets (IDEA), *DARPA Electronics Resurgence Initiative*, https://www.darpa.mil/work-with-us/electronics-resurgence-initiative

[69] S. Das, A. Chandrakasan and R. Reif, “Three-Dimensional Integrated Circuits: Performance, Design Methodology, and CAD Tools”, *Proc. IEEE Computer Society Annual Symposium on VLSI*, 2003, pp. 13-18.

[70] J. Deguchi, T. Sugimura, Y. Nakatani, T. Fukushima and M. Koyanagi, “Quantitative Derivation and Evaluation of Wire Length Distribution in Three-Dimensional Integrated Circuits Using Simulated Quenching”, *Japanese Journal of Applied Physics* 45(4B) (2006), pp. 3260-3265.

[71] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", *Proc. IEEE/ACM International Symposium on Microarchitecture*, 2003, pp. 7-18.

[72] C. F. Fang, R. A. Rutenbar and T. Chen, "Effcient Static Analysis of Fixed-Point Error in DSP Applications via Affine Arithmetic Modeling", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2003, pp. 275-282.

[73] X. Federspiel, L. Doyen and S. Courtas, "Use of Resistance-Evolution Dynamics During Electromigration to Determine Activation Energy on Single Samples", *IEEE Transactions on Device and Materials Reliability* 7(2) (2007), pp. 236-241.

[74] D. Fick, G. Kim, A. Wang, D. Blaauw and D. Sylvester, "Mixed-Signal Stochastic Computation Demonstrated in an Image Sensor with Integrated 2D Edge Detection and Noise Filtering", *Proc. IEEE Custom Integrated Circuits Conference*, 2014, pp. 1-4.

[75] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 1982, pp. 175-181.

[76] B. R. Gaines, "Stochastic Computing Systems", *Advances in Information Systems Science*, 1969, pp. 37-172.

[77] V. C. Gaudet and A. C. Rapley, "Iterative Decoding Using Stochastic Computation", *Electronics Letters* 39(3) (2003), pp. 299-301.

[78] B. Geden, "Understand and Avoid Electromigration (EM) and IR-drop in Custom IP Blocks", *Synopsys White Paper*, 2011.

[79] R. S. Ghaida, K. Doniger and P. Zarkesh-Ha, "Random Yield Prediction Based on a Stochastic Layout Sensitivity Model", *IEEE Trans. on Semiconductor Manufacturing* 22(3) (2009), pp. 329-337.

[80] C. M. Grinstead and L. J. Snell, *Introduction to Probability, 2nd ed.*, American Math. Soc., 2003.

[81] W. J. Gross, V. C. Gaudet and A. Milner, "Stochastic Implementation of LDPC Decoders", *Proc. Asilomar Conference on Signals, Systems and Computers*, 2005, pp. 713-717.

[82] S. Gupta and K. Gopalakrishnan, "Revisiting Stochastic Computation: Approximate Estimation of Machine Learning Kernels", *Proc. Workshop on Approximate Computing Across the System Stack*, 2014, pp. 1-2.

[83] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan and K. Roy, "IMPACT: Imprecise Adders for Low-Power Approximate Computing", *Proc. International Symposium on Low Power Electronic Design*, 2011, pp. 409-414.

[84] L. Hagen, A. B. Kahng, F. J. Kurdahi and C. Ramachandran, "On the Intrinsic Rent Parameter and Spectra-Based Partitioning Methodologies", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13(1) (1994), pp. 27-37.

[85] K. Han, A. B. Kahng and H. Lee, "Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-Based Detailed Router", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 68:1-68:6.

[86] K. Han, A. B. Kahng, J. Lee, J. Li and S. Nath, "A Global-Local Optimization Framework for Simultaneous Multi-Mode Multi-Corner Skew Variation Reduction", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 26:1-26:6.

[87] S. S. Han, A. B. Kahng, S. Nath and A. Vydyanathan, "A Deep Learning Methodology to Proliferate Golden Signoff Timing", *Proc. Design, Automation and Test in Europe*, 2014, pp: 260:1-260:6.

[88] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[89] K. He, A. Gerstlauer and M. Orshansky, "Low-Energy Signal Processing Using Circuit-Level Timing-Error Acceptance", *Proc. IEEE International Conference on Integrated Circuit Design and Technology*, 2012, pp. 1-4.

[90] X. He, T. Huang, W.-K. Chow, J. Kuang, K.-C. Lam, W. Cai and E. F. Y. Young, "Ripple 2.0: High Quality Routability-Driven Placement via Global Router Integration", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 1-6.

[91] X. He, T. Huang, L. Xiao, H. Tian, G. Cui and E. F. Young, "Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2011, pp. 74-79.

[92] R. Hegde and N. R. Shanbhag, "Soft Digital Signal Processing", *IEEE Transactions on Very Large Scale Integration Systems* 9(6) (2001), pp. 813-823.

[93] W. Hou, H. Yu, X. Hong, Y. Cai, W. Wu, J. Gu and W. H. Kao, "A New Congestion-Driven Placement Algorithm Based on Cell Inflation", *Proc. Asia and South Pacific Design Automation Conference*, 2001, pp. 606-608.

[94] M.-K. Hsu, S. Chou, T.-H. Lin and Y.-W. Chang, "Routability-Driven Analytical Placement for Mixed-Size Circuit Designs", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2011, pp. 80-84.

[95] B. Hu and M. Marek-Sadowska, "Congestion Minimization During Placement without Estimation", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2002, pp. 739-745.

[96] J. Huang and J. Lach, "Exploring the Fidelity-Efficiency Design Space Using Imprecise Arithmetic", *Proc. Asia and South Pacific Design Automation Conference*, 2011, pp. 579-584.

[97] J. Huang, J. Lach and G. Robins, "Analytic Error Modeling for Imprecise Arithmetic Circuits", *Proc. Silicon Errors in Logic - System Effects (SELSE)*, 2011.

[98] J. Huang, J. Lach and G. Robins, "A Methodology for Energy-Quality Tradeoffs Using Imprecise Hardware", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2012, pp. 504-509.

[99] J. Hwang and A. El-Gamal, "Optimal Replication for Min-Cut Partitioning", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1992, pp. 432-435.

[100] T. Iizuka, M. Ikeda and K. Asada, "Timing-Driven Cell Layout De-Compaction for Yield Optimization by Critical Area Minimization", *Proc. Design, Automation and Test in Europe*, 2006, pp. 1-6.

[101] P. Jeavons, D. A. Cohen and J. Shawe-Taylor, "Generating Binary Sequences for Stochastic Computing", *IEEE Transactions on Information Theory* 40(3) (1994), pp. 716-720.

[102] I. H.-R. Jiang, H.-Y. Chang and C.-L. Chang, "WiT: Optimal Wiring Topology for Electromigration Avoidance", *IEEE Transactions on Very Large Scale Integration Systems* 20(4) (2012), pp. 581-592.

[103] Z.-W. Jiang, B.-Y. Su and Y.-W. Chang, "Routability-Driven Analytical Placement by Net Overlapping Removal for Large-scale Mixed-Size Designs", *Proc. ACM/IEEE Design Automation Conference*, 2008, pp. 167-172.

[104] J. W. Joyner, R. Venkatesan, P. Zarkesh-Ha, J. A. Davis and J. D. Meindl, "Impact of Three-Dimensional Architectures on Interconnects in Gigascale Integration", *IEEE Transactions on Very Large Scale Integration Systems* 9(6) (2001), pp. 922-928.

[105] M. Jung, T. Song, Y. Wan, Y.-J. Lee, D. Mohapatra, H. Wang, G. Taylor, D. Jariwala, V. Pitchumani, P. Morrow, C. Webb, P. Fischer and S. K. Lim, "How to Reduce Power in 3D IC Designs: A Case Study with OpenSPARC T2 Core", *Proc. IEEE Custom Integrated Circuits Conference*, 2013, pp. 1-4.

[106] M. Jung, T. Song, Y. Wan, Y. Peng and S. K. Lim, "On Enhancing Power Benefits in 3D ICs: Block Folding and Bonding Styles Perspective", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2014, pp. 1-6.

[107] A. B. Kahng, "Design-Based Equivalent Scaling to the Rescue of Moore's Law", *UCI ECE Colloquium*, 2012.

[108] A. B. Kahng, "In Search of Lost Time", *Keynote in ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2016.

[109] A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proc. ACM International Symposium on Physical Design*, 2018.

[110] A. B. Kahng and S. Kang, "Construction of Realistic Gate Sizing Benchmarks With Known Optimal Solutions", *Proc. ACM International Symposium on Physical Design*, 2012, pp. 153-160.

[111] A. B. Kahng and S. Kang, "Accuracy-Configurable Adder for Approximate Arithmetic Designs", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2012, pp. 820-825.

[112] A. B. Kahng, S. Kang, R. Kumar and J. Sartori, "Slack Redistribution for Graceful Degradation Under Voltage Overscaling", *Proc. Asia and South Pacific Design Automation Conference*, 2010, pp. 825-831.

[113] A. B. Kahng, B. Liu and I. I. Măndoiu, "Nontree Routing for Reliability and Yield Improvement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23(1) (2004), pp. 148-156.

[114] A. B. Kahng, M. Luo and S. Nath, "SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects", *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2015, pp. 1-8.

[115] A. B. Kahng and S. Nath, “Optimal Reliability-Constrained Overdrive Frequencies Selection in Multicore Systems”, *Proc. International Symposium on Quality Electronic Design*, 2014, pp. 1-8.

[116] A. B. Kahng, S. Nath and T. S. Rosing, “On Potential Design Impacts of Electromigration Awareness”, *Proc. Asia and South Pacific Design Automation Conference*, 2013, pp. 527-532.

[117] A. B. Kahng and X. Xu, “Accurate Pseudo-Constructive Wirelength and Congestion Estimation”, *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2003, pp. 61-68.

[118] Z. Kedem, V. J. Mooney, K. K. Muntimadugu, K. V. Palem, A. Devarasetty and P. D. Parasuramuni, “Optimizing Energy to Minimize Errors in Dataflow Graphs Using Approximate Adders”, *Proc. International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2010, pp. 177-186.

[119] D. R. Kelly, B. J. Phillipsy and S. Al-Sarawi, “Approximate Signed Binary Integer Multipliers for Arithmetic Data Value Speculation”, *Proc. Conference on Design and Architectures for Signal and Image Processing*, 2009, pp. 97-104.

[120] D. H. Kim, K. Athikulwongse and S. K. Lim, “A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2009, pp. 674-680.

[121] M.-C. Kim, J. Hu, D.-J. Lee and I. L. Markov, “A SimPLR Method for Routability-Driven Placement”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2011, pp. 67-73.

[122] T.-Y. Kim and T. Kim, “Clock Tree Embedding for 3D ICs”, *Proc. Asia and South Pacific Design Automation Conference*, 2010, pp. 486-491.

[123] D. H. Kim and S. K. Lim, “Through-Silicon-Via-aware Delay and Power Prediction Model for Buffered Interconnects in 3D ICs”, *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2010, pp. 25-31.

[124] D. H. Kim, S. Mukhopadhyay and S. K. Lim, “Through-Silicon-Via Aware Interconnect Prediction and Optimization for 3D Stacked ICs”, *Proc. ACM International Workshop on System-Level Interconnect Prediction*, 2009, pp. 85-92.

[125] D. H. Kim, S. Mukhopadhyay and S. K. Lim, “TSV-Aware Interconnect Distribution Models for Prediction of Delay and Power Consumption of 3-D Stacked ICs”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(9) (2014), pp. 1384-1395.

[126] Y. C. Kim and M. A. Shanblatt, “Architecture and Statistical Model of A Pulse-Mode Digital Multilayer Neural Network”, *IEEE Transactions on Neural Networks* 6(5) (1995), pp. 1109-1118.

[127] D. H. Kim, R. O. Topaloglu and S. K. Lim, “Block-Level 3D IC Design with Through-Silicon-Via Planning”, *Proc. Asia and South Pacific Design Automation Conference*, 2012, pp. 335-340.

[128] A. B. Kinsman and N. Nicolici, “Finite Precision Bit-width Allocation Using SAT-Modulo Theory”, *Proc. Design, Automation and Test in Europe*, 2009, pp. 1106-1111.

[129] D. Koes and T. Chelcea, “Adding Faster With Application Specific Early Termination”, *CMU Research Showcase*, 2005.

[130] C. Kring and A. R. Newton, "A Cell-Replicating Approach to Min-Cut Based Circuit Partitioning", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1991, pp. 2-5.

[131] P. Kulkarni, P. Gupta and M. D. Ercegovac, "Trading Accuracy for Power With an Underdesigned Multiplier Architecture", *Proc. IEEE/ACM International Conference on VLSI Design*, 2011, pp. 346-351.

[132] K. Kum and W. Sung, "Combined Word-length Optimization and High-level Synthesis of Digital Signal Processing Systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20(8) (2001), pp. 921-930.

[133] S. V. Kumar, C. H. Kim and S. S. Sapatnekar, "Adaptive Techniques for Overcoming Performance Degradation Due to Aging in Digital Circuits", *Proc. Asia and South Pacific Design Automation Conference*, 2009, pp. 284-289.

[134] S. V. Kumar, C. H. Kim and S. S. Sapatnekar, "Adaptive Techniques for Overcoming Performance Degradation Due to Aging in CMOS Circuits", *IEEE Transactions on Very Large Scale Integration Systems* 19(4) (2011), pp. 603-614.

[135] S.-Y. Kuo, "YOR: A Yield-Optimizing Routing Algorithm by Minimizing Critical Areas and Vias", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12(9) (1993), pp. 1303-1311.

[136] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester and D. Blaauw, "A Modular $1mm^3$ Die-Stacked Sensing Platform with Optical Communication and Multi-Modal Energy Harvesting", *IEEE Journal of Solid State Circuits* 48(1) (2013), pp. 229-243.

[137] X. R. Lee, C. L. Chen, H. C. Chang and C. Y. Lee, "A 7.92 Gb/s 437.2 mW Stochastic LDPC Decoder Chip for IEEE 802.15.3c Applications", *IEEE Transactions on Circuits and Systems I* 62(2) (2015), pp. 507-516.

[138] D.-U. Lee, A. A. Gaffar, O. Mencer and W. Luk, "MiniBit: Bitwidth Optimization via Affine Arithmetic", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2005, pp. 837-840.

[139] Y. Lee and T. Kim, "A Fine-Grained Technique of NBTI-Aware Voltage Scaling and Body Biasing for Standard Cell Based Designs", *Proc. Asia and South Pacific Design Automation Conference*, 2011, pp. 603-608.

[140] K.-Y. Lee, C.-K. Koh, T.-C. Wang and K.-Y. Chao "Optimal Post-Routing Redundant Via Insertion", *Proc. ACM International Symposium on Physical Design*, 2008, pp. 111-117.

[141] K.-Y. Lee, C.-K. Koh, T.-C. Wang and K.-Y. Chao, "Fast and Optimal Redundant Via Insertion", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27(12) (2008), pp. 2197-2208.

[142] Y. J. Lee and S. K. Lim, "On GPU Bus Power Reduction with 3D IC Technologies", *Proc. Design, Automation and Test in Europe*, 2014, pp. 1-6.

[143] Y. J. Lee, D. Limbrick and S. K. Lim, "Power Benefit Study for Ultra-High Density Transistor-Level Monolithic 3D ICs", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 1-10.

[144] Y.-W. Lee, Y.-H. Lin and Y.-L. Li, “Minimizing Critical Area on Gridless Wire Ordering, Sizing and Spacing”, *Journal of Information Science and Engineering* 30(1) (2014), pp. 157-177.

[145] B. Li, C. Christiansen, C. Burke, N. Hogle and D. Badami, “Short Line Electromigration Characteristics and Their Applications for Circuit Design”, *Proc. IEEE International Reliability Physics Symposium*, 2013, pp. 3F.2.1-3F.2.5.

[146] P. Li and D. J. Lilja, “Using Stochastic Computing to Implement Digital Image Processing Algorithms”, *Proc. IEEE International Conference on Computer Design*, 2011, pp. 154-161.

[147] C. Li, M. Xie, C.-K. Koh, J. Cong and P. H. Madden, “Routability-Driven Placement and White Space Allocation”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2004, pp. 394-401.

[148] J. Lienig and G. Jerke, “Current-driven Wire Planning for Electromigration Avoidance in Analog Circuits”, *Proc. Asia and South Pacific Design Automation Conference*, 2003, pp. 783-788.

[149] S. K. Lim, *Design for High Performance, Low Power, and Reliable 3D Integrated Circuits*, New York, Springer, 2012.

[150] W.-H. Liu, T.-K. Chien and T.-C. Wang, “A Study on Unroutable Placement Recognition”, *Proc. ACM International Symposium on Physical Design*, 2014, pp. 19-26.

[151] W.-H. Liu, T.-K. Chien and T.-C. Wang, “Region-Based and Panel-Based Algorithms for Unroutable Placement Recognition” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(4) (2015), pp. 502-514.

[152] W.-H. Liu, Y.-L. Li and C.-K. Koh, “A Fast Maze-Free Routing Congestion Estimator with Hybrid Unilateral Monotonic Routing”, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 713-719.

[153] C. Liu and S. K. Lim, “A Design Tradeoff Study with Monolithic 3D Integration”, *Proc. International Symposium on Quality Electronic Design*, 2012, pp. 529-536.

[154] Q. Liu and M. Marek-Sadowska, “Wire Length Prediction-Based Technology Mapping and Fanout Optimization”, *Proc. ACM International Symposium on Physical Design*, 2005, pp. 145-151.

[155] Q. Liu and M. Marek-Sadowska, “Semi-Individual Wire-Length Prediction with Application to Logic Synthesis”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(4) (2006), pp. 611-624.

[156] G. H. Loh, Y. Xie and B. Black, “Processor Design in 3D Die-Stacking Technologies”, *Proc. IEEE/ACM International Symposium on Microarchitecture* 27(3) (2007), pp. 31-48.

[157] G. G. Lorentz, *Bernstein Polynomials 2nd Ed.*, Chelsea, New York, 1986.

[158] S.-L. Lu, “Speeding Up Processing With Approximation Circuits”, *IEEE Computer* 37(3) (2004), pp. 67-73.

[159] L.-C. Lu, “Physical Design Challenges and Innovations to Meet Power, Speed, and Area Scaling Trend”, *Keynote in ACM International Symposium on Physical Design*, 2017.

[160] W. K. Mak and C. Chu, "Rethinking the Wirelength Benefit of 3-D Integration", *IEEE Transactions on Very Large Scale Integration Systems* 20(12) (2012), pp. 2346-2351.

[161] J. Miao, K. He, A. Gerstlauer and M. Orshansky, "Modeling and Synthesis of Quality-Energy Optimal Approximate Adders", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 728-735.

[162] E. Mintarno, J. Skaf, R. Zheng, J. B. Velamala, Y. Cao, S. Boyd, R. W. Dutton and S. Mitra, "Self-Tuning for Maximized Lifetime Energy-Efficiency in the Presence of Circuit Aging", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30(5) (2011), pp. 760-773.

[163] V. Mishra, University of Minnesota, *personal communication*, November 2013.

[164] V. Mishra and S. S. Sapatnekar, "The Impact of Electromigration in Copper Interconnects on Power Grid Integrity", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 1-6.

[165] A. Nayak, M. Haldar, A. Choudhary and P. Banerjee, "Precision and Error Analysis of MATLAB Applications during Automated Hardware Synthesis for FPGAs", *Proc. Design, Automation and Test in Europe*, 2001, pp. 722-728.

[166] B. Moons and M. Verhelst, "Energy-Efficiency and Accuracy of Stochastic Computing Circuits in Emerging Technologies", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 4(4) (2014), pp. 475-486.

[167] A. Morro, V. Canals, A. Oliver, M. L. Alomar and J, L. Rossello, "Ultra-Fast Data-Mining Hardware Architecture Based on Stochastic Computing", *PLoS ONE* 10(5) (2015).

[168] M. H. Najafi, D. J. Lilja, M. Riedel and K. Bazargan, "Polysynchronous Stochastic Circuits", *Proc. Asia and South Pacific Design Automation Conference*, 2016, pp. 492-498.

[169] M. H. Najafi, D. J. Lilja, M. D. Riedel and K. Bazargan, "Polysynchronous Clocking: Exploiting the Skew Tolerance of Stochastic Circuits", *IEEE Transactions on Computers* PP(99) (2017), pp.1-13.

[170] A. Nardi and A. L. Sangiovanni-Vincentelli, "Synthesis for Manufacturability: a Sanity Check", *Proc. Design, Automation and Test in Europe*, 2004, pp. 796-801.

[171] D. K. Nayak, S. Banna, S. K. Samal and S. K. Lim, "Power, Performance, and Cost Comparisons of Monolithic 3D ICs and TSV-Based 3D ICs", *Proc. SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2015, pp. 1-2.

[172] T. K. Ng, J. Oldfield and V. Pitchumani, "Improvements of a Mincut Partition Algorithm", *Proc. ICCAD*, 1987, pp. 470-473.

[173] S. H. Ok, K. R. Bae, S. K. Lim and B. Moon, "Design and Analysis of 3D IC-Based Low Power Stereo Matching Processors", *Proc. International Symposium on Low Power Electronic Design*, 2013, pp. 15-20.

[174] B. Ouattara, L. Doyen, D. Ney, H. Mehrez, P. Bazargan-Sabet and F. L. Bana, "Redundancy Method to Assess Electromigration Lifetime in Power Grid Design", *Proc. IEEE International Interconnect Technology Conference*, 2013, pp. 1-3.

[175] M. M. Ozdal, C. Amin, A. Ayupov, S. M. Burns, G. R. Wilke and C. Zhuo, "ISPD-2012 Discrete Cell Sizing Contest and Benchmark Suite", *Proc. ACM International Symposium on Physical Design*, 2012, pp. 161-164.

[176] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2007, pp. 59-62.

[177] M. Pandey, "Machine Learning in Formal Verification", *Keynote in International Workshop on Logic and Synthesis*, 2017.

[178] S. Panth, K. Samadi, Y. Du and S. K. Lim, "High-Density Integration of Functional Modules Using Monolithic 3D-IC Technology", *Proc. Asia and South Pacific Design Automation Conference*, 2013, pp. 681-686.

[179] S. Panth, K. Samadi, Y. Du and S. K. Lim, "Design and CAD Methodologies for Low Power Gate-Level Monolithic 3D ICs", *Proc. International Symposium on Low Power Electronic Design*, 2014, pp. 171-176.

[180] S. Panth, K. Samadi, Y. Du and S. K. Lim, "Placement-Driven Partitioning for Congestion Mitigation in Monolithic 3D IC Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(4) (2014), pp. 540-553.

[181] E. Papadopoulou, "Net-aware Critical Area Extraction for Opens in VLSI Circuits via Higher-Order Voronoi Diagrams", *IEEE TCAD* 20(5) (2011), pp. 583-597.

[182] M. Pathak, Y. J. Lee, T. Moon and S. K. Lim, "Through-Silicon-Via Management during 3D Physical Design: When to Add and How Many?", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2010, pp. 387-394.

[183] M. Pedram, "Power Estimation and Optimization at the Logic Level", *Journal of High Speed Electronics and Systems* 5(2) (1994), pp. 179-202.

[184] I. Perez-Andrade, X. Zuo, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo, "Analysis of Voltage- and Clock-Scaling-Induced Timing Errors in Stochastic LDPC Decoders", *Proc. IEEE Wireless Communications and Networking Conference*, 2013, pp. 4293-4298.

[185] W. J. Poppelbaum, C. Afuso and J. W. Esch, "Stochastic Computing Elements and Systems", *Proc. AFIPS Fall Joint Computer Conference*, 1967, pp. 635-644.

[186] S. Priyadarshi, W. R. Davis and P. D. Franzon, "Pathfinder3D: A Framework for Exploring Early Thermal Tradeoffs in 3DIC", *Proc. IEEE International Conference on Integrated Circuit Design and Technology*, 2014, pp. 1-6.

[187] K. Puttaswamy and G. H. Loh, "3D-Integrated SRAM Components for High-Performance Microprocessors", *IEEE Transactions on Computers* 58(10) (2009), pp. 1369-1381.

[188] Z. Qi, Y. Cai and Q. Zhou, "Accurate Prediction of Detailed Routing Congestion Using Supervised Data Learning", *Proc. IEEE International Conference on Computer Design*, 2014, pp. 97-103.

[189] W. Qian, *Digital Yet Deliberately Random: Synthesizing Logical Computation on Stochastic Bit Streams*, Ph.D. Dissertation, University of Minnesota, 2011.

[190] W. Qian and M. D, Riedel, "The Synthesis of Robust Polynomial Arithmetic with Stochastic Logic", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2008, pp. 648-653.

[191] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic", *IEEE Transactions on Computers* 60(1) (2011), pp. 93-105.

[192] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "The Synthesis of Combinational Logic to Generate Probabilities", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2009, pp. 367-374.

[193] M. Rahman, R. Afonso, H. Tennakoon and C. Sechen, "Power Reduction via Separate Synthesis and Physical Libraries", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2011, pp. 627-632.

[194] J. A. Roy and I. L. Markov, "Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23(4) (2007), pp. 632-644.

[195] T. Sadakane, H. Shirota, K. Takahashi, M. Terai and K. Okazaki, "A Congestion-Driven Placement Improvement Algorithm for Large Scale Sea-of-gates Arrays", *Proc. IEEE Custom Integrated Circuits Conference*, 1997, pp. 573-576.

[196] N. Saraf, K. Bazargan, D. J. Lilja and M. D. Riedel, "Stochastic Functions Using Sequential Logic", *Proc. IEEE International Conference on Computer Design*, 2013, pp. 507-510.

[197] J. Sartori, J. Sloan and R. Kumar, "Stochastic Computing: Embracing Errors in Architecture and Design of Processors and Applications", *Proc. International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2011, pp. 135-144.

[198] J.-S. Seo, I. L. Markov, D. Sylvester and D. Blaauw, "On the Decreasing Significance of Large Standard Cells in Technology Mapping", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 116-121.

[199] N. R. Shanbhag, R. A. Abdallah, R. Kumar and D. L. Jones, "Stochastic Computation", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2010, pp. 859-864.

[200] K.-N. Shim, J. Hu and J. Silva-Martinez, "A Dual-Level Adaptive Supply Voltage System for Variation Resilience", *Proc. International Symposium on Quality Electronic Design*, 2010, pp. 38-43.

[201] D. Shin and S. K. Gupta, "Approximate Logic Synthesis for Error Tolerant Applications", *Proc. Design, Automation and Test in Europe*, 2010, pp. 957-960.

[202] M. M. Shulaker, G. Hills, R. S. Park, R. T. Howe, K. Saraswat, H. S. P. Wong and S. Mitra, "Three-dimensional Integration of Nanotechnologies for Computing and Data Storage on a Single Chip", *Nature* 547(7661) (2017), pp. 74-78.

[203] T. Song, M. Jung, Y. Wan, Y. Peng and S. K. Lim, "3D IC Power Benefit Study Under Practical Design Considerations", *Proc. IEEE International Interconnect Technology Conference*, 2015, pp. 335-338.

[204] T. Song, S. Panth, Y. J. Chae and S. K. Lim, "Three-Tier 3D ICs for More Power Reduction: Strategies in CAD, Design, and Bonding Selection", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 752-757.

[205] P. Spindler and F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-Driven Placement", *Proc. Design, Automation and Test in Europe*, 2007, pp. 1226-1231.

[206] M. Stephenson, J. Babb and S. Amarasinghe, "Bitwidth Analysis With Application to Silicon Compilation", *Proc. ACM Conference on Programming Language Design and Implementation*, 2000, pp. 108-120.

[207] D. Stroobandt, P. Verplaetse and J. van Campenhout, "Generating Synthetic Benchmark Circuits for Evaluating CAD Tools", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19(9) (2000), pp. 1011-1022.

[208] T. Taghavi, C. J. Alpert, A. Huber, Z. Li, G.-J. Nam and S. Ramji, "New Placement Prediction and Mitigation Techniques for Local Routing Congestion", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2010, pp. 621-624.

[209] T. Thorolfsson, G. Luo, J. Cong and P. D. Franzon, "Logic-on-Logic 3D Integration and Placement", *Proc. International 3D Systems Integration Conference (3DIC)*, 2010, pp. 1-4.

[210] K. Tiri and I. Verbauwhede, "A Digital Design Flow for Secure Integrated Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(7) (2006), pp. 1197-1208.

[211] K. Tsota, C.-K. Koh and V. Balakrishnan, "Guiding Global Placement with Wire Density", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 212-217.

[212] R. Vattikonda, W. Wang and Y. Cao, "Modeling and Minimization of PMOS NBTI Effect for Robust Nanometer Design", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2006, pp. 1047-1052.

[213] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy and A. Raghunathan, "SALSA: Systematic Logic Synthesis of Approximate Circuits", *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2012, pp. 796-801.

[214] R. Venkatesan, A. Agarwal, K. Roy and A. Raghunathan, "MACACO: Modeling and Analysis of Circuits for Approximate Computing", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2011, pp. 667-673.

[215] A. K. Verma, P. Brisk. and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design", *Proc. Design, Automation and Test in Europe*, 2009, pp. 1250-1255.

[216] Z. Wang and A. C. Bovik, "Mean Squared Error: Love it or Leave it? A New Look at Signal Fidelity Measures", *IEEE Signal Processing Magazine* 26(1) (2009), pp. 98-117.

[217] Z. Wang, N. Saraf, K. Bazargan and A. Scheel, "Randomness Meets Feedback: Stochastic Implementation of Logistic Map Dynamical System", *Proc. ACM/EDAC/IEEE Design Automation Conference*, pp. 132:1-132:7, 2015.

[218] Z. Wang, E. P. Simoncelli and A. C. Bovik, “Multi-scale Structural Similarity for Image Quality Assessment”, *Proc. Asilomar Conference on Signals, Systems and Computers*, 2003, pp. 9-12.

[219] M. Wang, X. Yang, K. Eguro and M. Sarrafzadeh, “Multicenter Congestion Estimation and Minimization during Placement”, *Proc. ACM International Symposium on Physical Design*, 2000, pp. 147-152.

[220] S. Ward, D. Ding and D. Z. Pan, “PADE: A High-Performance Placer with Automatic Datapath Extraction and Evaluation through High-Dimensional Data Learning”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2012, pp. 756-761.

[221] J. Westra, C. Bartels and P. Groeneveld, “Probabilistic Congestion Prediction”, *Proc. ACM International Symposium on Physical Design*, 2004, pp. 204-209.

[222] Z. Wo, I. Koren and M. Ciesielski, “An ILP Formulation for Yield-driven Architectural Synthesis”, *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005, pp. 12-20.

[223] Y. Xie, G. H. Loh, B. Black and K. Bernstein, “Design Space Exploration for 3D Architectures”, *ACM Journal on Emerging Technologies in Computing Systems* 2(2) (2006), pp. 65-103.

[224] H. Xu, V. F. Pavlidis and G. De Micheli, “Effect of Process Variations in 3D Global Clock Distribution Networks” *ACM Journal on Emerging Technologies in Computing Systems* 8(3) (2012), pp. 20:1-20:25.

[225] J.-T. Yan and Z.-W. Chen, “Electromigration-aware Rectilinear Steiner Tree Construction for Analog Circuits”, *Proc. Asia Pacific Conference on Circuits and Systems*, 2008, pp. 1692-1695.

[226] X. Yang, B.-K. Choi and M. Sarrafzadeh, “Routability-Driven White Space Allocation for Fixed-Die Standard-Cell Placement” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(4) (2003), pp. 410-419.

[227] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang and C. Chiang, “Accurate Process-Hotspot Detection Using Critical Design Rule Extraction”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2012, pp. 1167-1172.

[228] S. Zafar, Y. H. Kim, V. Narayanan, C. Cabral Jr., V. Paruchuri, B. Doris, J. Stathis, A. Callegari and M. Chudzik, “A Comparative Study of NBTI and PBTI (Charge Trapping) in SiO2/HfO2 Stacks with FUSI, TiN, Re Gates”, *Proc. Symposium on VLSI Technology*, 2006, pp. 23-25.

[229] J. Zhao, G. Sun, G. H. Loh and Y. Xie, “Optimizing GPU Energy Efficiency with 3D Die-stacking Graphics Memory and Reconfigurable Memory Interface”, *ACM Transactions on Architecture and Code Optimization* 10(4) (2013), pp. 24:1-24:25.

[230] K. Zhong and S. Dutt, “Algorithms for Simultaneous Satisfaction of Multiple Constraints and Objective Optimization in a Placement Flow with Application to Congestion Control”, *Proc. ACM/EDAC/IEEE Design Automation Conference*, 2002, pp. 854-859.

[231] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou and Y. Cai, “An Accurate Detailed Routing Routability Prediction Model in Placement”, *Proc. Asia Symposium on Quality Electronic Design*, 2015, pp. 119-122.

[232] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo and Z. H. Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing", *IEEE Transactions on Very Large Scale Integration Systems* 18(8) (2010), pp. 1225-1229.

[233] *Apache RedHawk User's Manual.*
http://www.apache-da.com

[234] *Pegasus Flies to the Clouds.*
https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/pegasus

[235] *Cadence Innovus User's Manual.*
http://www.cadence.com

[236] *Cadence NC-Verilog User's Manual.*
http://www.cadence.com/

[237] *Cadence Tempus User's Manual.*
http://www.cadence.com

[238] *Cadence Virtuoso User's Manual.*
http://www.cadence.com

[239] *Canon Image Test Set.*
Available at http://www.cipr.rpi.edu/resource/stills/canon.html

[240] *Kodak Image Test Set.*
Available at http://www.cipr.rpi.edu/resource/stills/kodak.html

[241] *IBM ILOG CPLEX.*
http://www.ilog.com/products/cplex/

[242] *International Technology Roadmap for Semiconductors Interconnect Chapter.*, 2011
http://www.itrs2.net/itrs-reports.html

[243] *International Technology Roadmap for Semiconductors (ITRS) 2.0.*
http://www.itrs2.net/

[244] *MLPart.*
http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Partitioning/MLPart

[245] *MathWorks MATLAB.*
http://www.mathworks.com

[246] *Mentor Calibre User's Manual.*
https://www.mentor.com/

[247] *OpenCores.*
http://opencores.org

[248] *The R Project for Statistical Computing*,
https://www.r-project.org/

[249] *Leaps Package in R Language*,
https://cran.r-project.org/web/packages/leaps/

[250] *RentCon.*
http://vlsicad.ucsd.edu/WLD/RentCon.pdf

[251] *Synopsys Design Compiler User's Manual.*
http://www.synopsys.com

[252] *Synopsys IC Compiler User's Manual.*
http://www.synopsys.com

[253] *Synopsys PrimeTime User's Manual.*
http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeTime.aspx

[254] *Synopsys PowerRail User's Manual.*
http://www.synopsys.com

[255] *Synopsys Siliconsmart User's Manual.*
http://www.synopsys.com