CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Reliable chip design from low powered unreliable components
Authors	Grandhi, Satish Kumar
Publication date	2019
Original Citation	Grandhi, S. K. 2019. Reliable chip design from low powered unreliable components. PhD Thesis, University College Cork.
Type of publication	Doctoral thesis
Rights	© 2019, Satish Kumar Grandhi. - http://creativecommons.org/licenses/by-nc-nd/3.0/
Download date	2026-11-04 01:50:06
Item downloaded from	https://hdl.handle.net/10468/8610



Reliable Chip Design from Low Powered Unreliable Components



Author: Satish Kumar Grandhi

Supervisor: Dr. Emanuel M. Popovici

Department of Electrical and Electronic Engineering, School of Engineering
Head of School: Prof. William Marnane

A Thesis Submitted to the National University of Ireland, Cork, in Fulfillment of the Requirements for the Degree of Doctor of Philosophy

September 2019

I would like to dedicate this thesis to my spiritual master his divine grace Bhakti Vedanta Swami Srila Prabhupada Founder of ISKCON . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements.

Declaration1: Part of the CPEP based reconvergent fanout analysis was done by my colleague Bo Yang [1] [2]. This is covered in Section 3.3.6.

Declaration2: Mathematical analysis of the CPE methodology was done with Dr. Savin Valentin, CEA LETI [3] [4]. This is covered in section 5.4.

Author: Satish Kumar Grandhi
September 2019

Abstract

The pace of technological improvement of the semiconductor market is driven by Moore's Law, enabling chip transistor density to double every two years. The transistors would continue to decline in cost and size but increase in power. The continuous transistor scaling and extremely lower power constraints in modern Very Large Scale Integrated(VLSI) chips can potentially supersede the benefits of the technology shrinking due to reliability issues. As VLSI technology scales into nanoscale regime, fundamental physical limits are approached, and higher levels of variability, performance degradation, and higher rates of manufacturing defects are experienced. Soft errors, which traditionally affected only the memories, are now also resulting in logic circuit reliability degradation. A solution to these limitations is to integrate reliability assessment techniques into the Integrated Circuit(IC) design flow. This thesis investigates four aspects of reliability driven circuit design: a)Reliability estimation; b) Reliability optimization; c) Fault-tolerant techniques, and d) Delay degradation analysis.

To guide the reliability driven synthesis and optimization of combinational circuits, highly accurate probability based reliability estimation methodology christened **Conditional Probabilistic Error Propagation(CPEP)** algorithm is developed to compute the impact of gate failures on the circuit output. CPEP guides the proposed rewriting based logic optimization algorithm employing local transformations. The main idea behind this methodology is to replace parts of the circuit with functionally equivalent but more reliable counterparts chosen from a precomputed subset of Negation-Permutation-Negation(NPN) classes of 4-variable functions. Cut enumeration and Boolean matching driven by reliability-aware optimization algorithm are used to identify the best possible replacement candidates. Experiments on a set of MCNC benchmark circuits and 8051 functional microcontroller units indicate that the proposed framework can achieve up to 75% reduction of output error probability. On average, about 14% SER reduction is obtained at the expense of very low area overhead of 6.57% that results in 13.52% higher power consumption.

The next contribution of the research describes a novel methodology to design fault tolerant circuitry by employing the error correction codes known as **Codeword Prediction Encoder(CPE)**. Traditional fault tolerant techniques analyze the circuit reliability issue from a static point of view neglecting the dynamic errors. In the context of communication and

storage, the study of novel methods for reliable data transmission under unreliable hardware is an increasing priority. The idea of CPE is adapted from the field of forward error correction for telecommunications focusing on both encoding aspects and error correction capabilities. The proposed Augmented Encoding solution consists of computing an augmented codeword that contains both the codeword to be transmitted on the channel and extra parity bits. A Computer Aided Development(CAD) framework known as CPE simulator is developed providing a unified platform that comprises a novel encoder and fault tolerant LDPC decoders. Experiments on a set of encoders with different coding rates and different decoders indicate that the proposed framework can correct all errors under specific scenarios. On average, about 1000 times improvement in Soft Error Rate(SER) reduction is achieved.

Last part of the research is the Inverse Gaussian Distribution(IGD) based delay model applicable to both combinational and sequential elements for sub-powered circuits. The Probability Density Function(PDF) based delay model accurately captures the delay behavior of all the basic gates in the library database. The IGD model employs these necessary parameters, and the delay estimation accuracy is demonstrated by evaluating multiple circuits. Experiments results indicate that the IGD based approach provides a high matching against HSPICE Monte Carlo simulation results, with an average error less than 1.9% and 1.2% for the 8-bit Ripple Carry Adder(RCA), and 8-bit De-Multiplexer(DEMUX) and Multiplexer(MUX) respectively.

Acknowledgements

Doing a Ph.D. is a journey of self-discovery which can be very rewarding and enlightening. It's a time to take charge of one's learning and to develop a broad set of valuable new skills. Like any long journey, though, it's not without hardship. My Ph.D. research started when I left Cypress Semiconductors, Bengaluru India back in 2013 and moved to Cork. In the intervening three years, my life has changed dramatically. I believe that every person one meets leaves signs on our road to improvement. I was lucky to meet many such beautiful people during my research. This work was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (i-RISC project).

First, I would like to thank my supervisor Dr. Emanuel Popovici for giving me the opportunity to pursue my doctoral studies on an intriguing subject such as VLSI EDA. Thanks for giving me the chance to travel around the world and visit many interesting places over the years and finally thanks for all the help, support and encouragement that he gave me during my Ph.D. studies.

To my i-RISC research group members: Prof Sorin Cotofona, TU Delft has been my second supervisor. His amazing command over the subject and the amazing sense of humor dazzles me. I was fortunate to have worked directly with Dr. Savin Valentin, CEA LETI and learn a bit about LDPC coding techniques from his decades of working experience. I was also very fortunate to have three post-docs supervising me at different time periods. Dr. Chen made me feel comfortable with the new place when I started my Ph.D. journey. Then, Dr. Christian Spagnol builds the foundation of my research and helped me in publishing quite a few papers. Last, but not the least, Dr. Elsa Dupraz came at the end of my Ph.D. and helped me to solve on some of the pending problems. I am very thankful for many discussions we had, about our research work and beyond. It takes a considerable amount of effort and perseverance, to come to the point where only acknowledgments are left to be included in the dissertation. They have helped me reach this point

My labmates need special mention for bearing me for three years. Bruno, Fiona, BO, Nasim, Alan, and David: Thank you, folks, for all the fun chats. Thanks to all the staff of the Electric and Electronic Engineering Department for all the help, assistance and kindness. I was also fortunate to make some good friends from my Indian community. Prasanna, Charles,

Umanath, and Preethi I must have bugged you a lot. It is not over, get ready for more for the rest of life. Time spent with my local Telugu batch was the best moment of this three years. Thanks, guys for the fun games and the spicy Indian food. To my family, I am grateful for the most important signs in my life that made it more comfortable when deciding which turns to take. Last but not the least, my beautiful wife Susheela who has been most supportive during the whole editing of the thesis.

Publications

The thesis is based in part on the following publications:

- **J2**: **Grandhi, S.**, Yang, B., Spagnol, C., Gupta, S. and Popovici, E., 2016. An EDA Framework for Reliability Estimation and Optimization of Combinational Circuits. Journal of Low Power Electronics, 12(3), pp.242-258.
- **J1**: Chen, J., Cotofana, S., **Grandhi, S.**, Spagnol, C. and Popovici, E., 2015. Inverse Gaussian distribution based timing analysis of Sub-threshold CMOS circuits. Microelectronics Reliability, 55(12), pp.2754-2761.
- **C7**: Yang, B., **Grandhi, S.**, Spagnol, C., Popovici, E. and Cotofana, S., 2016, June. An approach for digital Circuit Error/Reliability Propagation Analysis based on Conditional Probability. In 2016 27th Irish Signals and Systems Conference, pp.1-6.
- **C6**: Dupraz, E., Savin, V., **Grandhi, S.**, Popovici, E. and Declercq, D., 2016, May. Practical LDPC encoders robust to hardware errors. In 2016 IEEE International Conference on Communications, pp.1-6.
- **C5**: **Grandhi, S.**, Dupraz, E., Spagnol, C., Savin, V. and Popovici, E., 2016, May. CPE: Codeword Prediction Encoder. In 2016 21th IEEE European Test Symposium, pp.1-2.
- **C4**: **Grandhi, S.**, McCarthy, D., Spagnol, C., Popovici, E. and Cotofana, S., 2015, October. Rost-c: Reliability driven optimisation and synthesis techniques for combinational circuits. In 2015 33rd IEEE International Conference on Computer Design, pp.431-434.
- **C3**: **Grandhi, S.**, Spagnol, C., Chen, J., Popovici, E. and Cotafona, S., 2014, September. Reliability aware logic synthesis through rewriting. In 2014 27th IEEE International System-on-Chip Conference, pp.274-279.
- **C2**: **Grandhi, S.**, Spagnol, C. and Popovici, E., 2014, June. Reliability analysis of logic circuits using probabilistic techniques. In 2014 10th Conference on Ph. D. Research in Microelectronics and Electronics, pp.1-4.
- **C1**: Chen, J., Spagnol, C., **Grandhi, S.**, Popovici, E., Cotofana, S. and Amaricai, A., 2014, July. Linear compositional delay model for the timing analysis of sub-powered combinational circuits. In 2014 IEEE Computer Society Annual Symposium on VLSI, pp.380-385.

Table of contents

List of figures **xiii**

List of tables **xvi**

Nomenclature **xvii**

1 Introduction **1**

1.1 VLSI Design, Automation and Reliability 1

1.2 Objectives and Thesis Contributions 3

1.2.1 Research Objectives 3

1.2.2 Thesis Statement 3

1.2.3 Circuit Representation and Modification 4

1.2.4 Reliability Estimation and Analysis 5

1.2.5 Reliability Driven Logic Optimization 6

1.2.6 Fault Tolerant Graph Augmentation 7

1.2.7 PDF based Delay Degradation Analysis 8

1.3 The Research Framework 9

1.4 Conclusions 11

2 Reliability in Logic Circuit Design **13**

2.1 Introduction 13

2.1.1 Understanding Reliability 14

2.1.2 Transient faults in logic circuits 16

2.2 Reliability Estimation 17

2.3 Reliability Aware Logic Synthesis 19

2.4 Fault Tolerant Techniques 20

2.5 Static Timing Analysis 21

2.6 Conclusions 22

3 Reliability Estimation 24
3.1 Introduction . . . 24
3.1.1 Main Contributions and Outline . . . 24
3.2 Simulation Based Reliability Computation . . . 25
3.2.1 Mersenne Twister and Random Number Generation . . . 25
3.2.2 Gate Error Model . . . 26
3.2.3 The Methodology . . . 26
3.2.4 Limitations . . . 28
3.3 CPEP: Conditional Probabilistic Error Propagation . . . 29
3.3.1 The Data Structure . . . 29
3.3.2 Gate Error Models . . . 30
3.3.3 2-Input Ideal AND Gate . . . 31
3.3.4 Intrinsic Gate Error Effects . . . 32
3.3.5 Ideal inverter . . . 33
3.3.6 Re-convergent Fanout . . . 34
3.3.7 Bounding Node Error Probability . . . 35
3.3.8 CPEP based Analysis . . . 35
3.3.9 CPEP extension to other Gates . . . 37
3.3.10 Limitations . . . 38
3.4 CAD Tool: Reliability Evaluator . . . 38
3.4.1 Computation Algorithm . . . 39
3.4.2 Simulation Results . . . 40
3.5 Conclusions . . . 43

4 Reliability Aware Logic Synthesis 44
4.1 Introduction . . . 44
4.1.1 Main Contributions and Outline . . . 45
4.2 ABC : Open Source EDA tool . . . 46
4.2.1 AND Invert Graphs . . . 46
4.2.2 ABC Tool . . . 48
4.3 Rule based Rewriting . . . 49
4.3.1 Local Transformation Rules . . . 50
4.3.2 Exhaustive Analysis of Rules . . . 53
4.3.3 The CAD algorithm . . . 54
4.4 Experimental Results . . . 59
4.4.1 Case study . . . 59
4.4.2 Evaluation of MCNC Benchmark Circuits . . . 60

4.5 Cut Based AIG Rewriting . . . 61
4.5.1 The CAD Algorithm . . . 63
4.6 Experimental Results . . . 65
4.6.1 CM162a – A Case Study . . . 65
4.6.2 Evaluation of Benchmark Circuits . . . 70
4.7 Conclusions . . . 72

5 CPE : Codeword Prediction Encoder 73
5.1 Introduction . . . 73
5.1.1 Main Contributions and Outline . . . 74
5.2 LDPC Codes and Error Models . . . 76
5.2.1 LDPC codes . . . 77
5.2.2 LDPC Encoding and Decoding . . . 78
5.2.3 Gate Error Model . . . 79
5.3 Codeword Prediction Encoder (CPE) . . . 79
5.4 CPE Mathematical Analysis . . . 81
5.4.1 CPE Cost Analysis . . . 81
5.4.2 Notation and Conventions . . . 82
5.4.3 Cost analysis for Area/Power . . . 83
5.4.4 Error Correction Capacity . . . 85
5.4.5 CPE and Modular Redundancy . . . 86
5.5 CPE Simulator and CAD Automation . . . 90
5.5.1 CPE Core Architecture . . . 90
5.5.2 Criticality Threshold . . . 90
5.5.3 Pre-Processing . . . 91
5.5.4 Netlist Format . . . 92
5.5.5 CPE Simulator . . . 93
5.6 Experimental Results . . . 94
5.6.1 Critical Nodes . . . 95
5.6.2 Impact of Decoder Configuration . . . 96
5.7 CPE for Fault Prone Boolean Functions . . . 97
5.7.1 Encoding Mechanism . . . 97
5.8 Experimental Results . . . 98
5.8.1 Critical Nodes . . . 99
5.8.2 Area Overhead . . . 100
5.8.3 NMR Vs. CPE . . . 101
5.8.4 Impact of LDPC code sizes on Area . . . 101

5.8.5 Case Study 102
5.9 Conclusions 104

6 Delay Degradation Analysis 105
6.1 Introduction 105
6.1.1 Main Contributions and Outline 106
6.2 Library Cells Simulation Methodology 106
6.2.1 Library Characterization 107
6.2.2 Timing Verification 107
6.2.3 Simulation Methodology 108
6.3 Linear Compositional Delay Model 108
6.3.1 Flexibility (universality) of IGD model 109
6.4 IGD Based Delay Model for Combinational and Sequential circuits 112
6.4.1 Typical timing path in synchronous CMOS circuits 112
6.4.2 Combinational Gates - INV, NAND, NOR & XOR 113
6.4.3 Sequential Circuits- Master Slave DFF 114
6.4.4 Sequential Circuits- Sub-Threshold DFF 115
6.5 Fan-out Aware IGD Model 116
6.5.1 Fan-out effect estimation methodology 117
6.5.2 Transition time effect estimation methodology 118
6.5.3 The FOC and FOP effects 120
6.5.4 Model Validation for Synchronous Circuits 120
6.6 Conclusions 123

7 Summary and Future work 125
7.1 Contributions 125
7.1.1 Data Structure 126
7.1.2 Inverse Gaussian Distribution Based Timing Analysis 126
7.1.3 ROST-C: Reliability driven Optimization and Synthesis 127
7.1.4 Error Coding Driven Graph Augmentation 128
7.1.5 Boole-Shannon Limit of noisy combinational logic 128
7.2 Future Work 129

References 131

List of figures

1.1 Four significant areas of current research 4
1.2 Different Data-structures 5
1.3 Reliability Estimation Flow 6
1.4 Reliability Driven Circuit Optimization Example 7
1.5 The Codeword Prediction Encoder(CPE) flow 8
1.6 PDF propagation and Methodology 9
1.7 The IRISC Project Details 10
1.8 The Complete Flow 12

2.1 Classification of Reliability 14

3.1 Gate Error Model for GLS based technique. 26
3.2 Unreliable Data transmission through on Inverter and AND Gates. 27
3.3 And Inverter representation of Combinational Circuit 29
3.4 Unreliable AND Gate Model 30
3.5 Unreliable Inverter Model 33
3.6 Re-convergent Fanout Structures 34
3.7 Bounding Error 35
3.8 B9 Benchmark Circuit Error Bounds 36
3.9 Design Flow for Reliability Computation 41

4.1 And Inverter representation of Combinational Circuit 47
4.2 Cut Example 47
4.3 Boolean Equivalent 48
4.4 Logic Transformation Rule1 50
4.5 Logic Transformation Rule2 51
4.6 Logic Transformation Rule3 51
4.7 Logic Transformation Rule4 52
4.8 Logic Transformation Rule5 52

4.9 All simulation results for Rule1 55
4.10 All simulation results for Rule2 55
4.11 All simulation results for Rule3 56
4.12 All simulation results for Rule4 56
4.13 Reliability Aware Logic Synthesis Flow 58
4.14 Application of logic transformation rule-set on the original Circuit 60
4.15 Comparison of results 61
4.16 MCNC Benchmark CM162A – A Case Study 66
4.17 Comparison of results : Node count reduction 70
4.18 Comparison of results: Power reduction 71
4.19 Comparison of results : Reliability improvement 71

5.1 Methodology of Codeword Prediction Encoder 75
5.2 Data transmission scheme 76
5.3 LDPC Codes 77
5.4 LDPC message computation 78
5.5 Gate Error Model. 79
5.6 Encoding error probability P_e with respect to gate error probability p_g. In the legend, the $(3,x)$-code represents the code with $d_v = 3$ and $d_c = x$. . . . 80
5.7 First encoding solution 80
5.8 The CPE approach 81
5.9 Delay vs. normalized area for MR and CPE schemes with various unrolling factors. 88
5.10 The CPE Tool Architecture 91
5.11 The CPE CAD flow 94
5.12 Critical Threshold impact on Output BER 95
5.13 Decoder Configuration impact on Output BER 96
5.14 CPE error free scenario employing faulty decoder 97
5.15 CPE error free scenario employing perfect decoder 97
5.16 CDF of the number of erroneous outputs generated by one single error injection 98
5.17 CDF plot of criticality degree 99
5.18 Critical node count for different Linear and Non-Linear circuits. 100
5.19 Area overhead due to parity augmentation. 100
5.20 Performance of NMR Vs CPE 101
5.21 Area and timing Analysis on IP cores using CPE methodology. 102
5.22 Output BER for various Criticality thresholds 103
5.23 Detailed plots for output error on F, P and decoder output nodes. 103

6.1 Library Characterization Flow 108
6.2 IGD and GD fittings for 2 Input AND gate 110
6.3 Seven AND gate chain 111
6.4 IGD vs GD fitting for 2-input AND gate with only supply variation. 112
6.5 IGD and GD fittings for basic gates charging and discharging events 114
6.6 IGD and GD fittings for DFF charging and discharging events 115
6.7 IGD and GD fittings for Sub-threshold based DFF architecture. 116
6.8 A sample circuit with FOC= 3 and FOP= 2. 119
6.9 DFFs +8-bit RCA CDFs. 121
6.10 Schematic of 8-bit DEMUX and MUX. 122
6.11 DFFs +8-bit DEMUX and MUX CDFs. 123

List of tables

3.1 Ideal AND Gate with Unreliable Inputs 32
3.2 Faulty AND Gate with Unreliable Inputs 33
3.3 MCNC Benchmark Circuits Based Accuracy and Performance Evaluation for different gate errors (ε) 42

4.1 Scenarios used for rules analysis 54
4.2 MCNC Benchmark Circuits Performance Evaluation (gate error: $\varepsilon = 0.05$.) 62
4.3 RWREL Performance Evaluation on different Benchmark Circuits (Gate error $\varepsilon = 0.001$) 68
4.4 Area, Delay and Power Analysis – A comparative Study 69

5.1 CPE Netlist Representation 92
5.2 Critical Gate count for different encoding schemes 95

6.1 μ and λ for NAND, INV and DFF 115
6.2 INV & NAND FOC key parameter values. 117
6.3 FOP effect on output transition time (all values in ps). 118
6.4 FOP effect on key parameters for INV. 119
6.5 FOP effect on key parameters for NAND. 120
6.6 DFFS +8-BIT RCA CDF deviations. 122
6.7 DFFS +8-BIT DEMUX and MUX CDF deviations. 123

Nomenclature

Acronyms / Abbreviations

AIG And Invert Graphs

ATPG Automatic Test Pattern Generation

BER Bit Error Rate

BLIF Berkeley Logic Interchange Format

BN Bayesian Networks

BSC Binary Symmetric Channel

CCS Composite Current Source

CDF Cumulative Distribution Functions

CNFET Carbon Nanotube Field-Effect Transistors

CPEP Conditional Probabilistic Error Propagation

CT Criticality Threshold

DRM Dynamic Reliability Management

DTA Dynamic Timing Analysis

DWAA Dynamic Weighted Average Algorithm

ECC Error Correcting Codes

EMI Electro-Magnetic Interference

FA Full Adder

FER Frame Error Rate

FO Fan-Out

GALB Gallager B

GD Gaussian Distribution

GLS Gate Level Simulation

HCI Hot Carrier Injection

IC Integrated Circuits

IEEE Institute of Electrical and Electronics Engineers

LDGM Low Density Generator Matrix

LDPC Low-Density Parity-Check

LDPC Low-Density Parity-Check

LER Line Edge Roughness

LUT Look Up Table

MC Monte-Carlo

MCNC Microelectronics Centre of North Carolina

MCSTA Monte Carlo Static Timing Analysis

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor

MP Modular Redundancy

MRF Markov Random Fields

MS Min-Sum

MT Mersenne Twister

NBTI Negative Bias Temperature Instability

NLDM Non Linear Delay Model

PDD Probabilistic Decision Diagrams

PGM Probabilistic Gate Model

PI Primary Inputs

PO Primary Outputs

PRNG Pseudo-Random Number Generator

PTM Predictive Transistor Models

PTM Probabilistic Transfer Matrices

PVT Process, Voltage, and Temperature

R2R Register to Register

RAR Redundancy Addition and Removal

RDF Random Dopant Fluctuations

RMR R-fold Modular Redundancy

RTL Register Transfer Language

SAIF Switching Activity Interchange Format

SCMS Self-Corrected Min-Sum

SER Soft Error Rate

SEU Single Event Upsets

SPRA Signal Probability Reliability Analysis

SSTA Statistical Static Timing Analysis

STA Static Timing Analysis

TGMS Transmission-Gate Master-Slave

TMR Triple Modular Redundancy

VHDL VHSIC Hardware Description Language

Chapter 1

Introduction

1.1 VLSI Design, Automation and Reliability

Computing technology is the cornerstone of modern day human progress. Digital systems permeate all areas of our lives, from personal computers, through automotive applications and medical systems, to common house appliances. Although these digital systems provide greater productivity and flexibility, they cannot be fault-free. In this regard, the semiconductor industry has driven more than five decades of improvements in its products mainly thanks to scaling trends. Technology scaling is leading to a decrease in device geometry and the increase in clock frequencies, resulting in a significant increase in the incidence of transient errors [5]. Thus, although technology trends allowed for easily obtained low-cost, high-performance microprocessors, the critical issue for these processors is to satisfy the requirements of dependable computing.

Traditional Very-Large-Scale Integration(VLSI) design methodologies and Electronic Design Automation(EDA) tools are centered around fulfilling timing, power, and area constraints or on achieving acceptable trade-offs among those [6] [7]. However, as the Complementary Metal-Oxide-Semiconductor(CMOS) technology entered the nanometer era, such an approach can no longer cover all the relevant design aspects. Technology scaling has precipitated higher operating speeds, lower operating voltages, and lower operating noise margins; all of which contribute to reduced switching energies, allowing legitimate logic signals to be readily overwhelmed by single-event-induced charge-collection transients [8]. Nanotechnology specific issues, e.g., power supply voltage reduction, the higher impact of the process parameter and temperature variations, resulting in increased device failure rates, making CMOS Integrated Circuits(IC) less reliable [9] [10].

As power usage is proportional to the square of voltage, operating at very low voltages offers a potential for significant power savings [11] [12]. However, this would mean the sup-

ply voltage is significantly below the transistor threshold voltage, and it is well known that in this weak inversion regime, Metal-Oxide-Semiconductor Field-Effect Transistor(MOSFET) transistors exhibit high voltage gain but very low currents [12]. There are many possible ways in which a sub-threshold circuit may become unreliable. The simplest is due to noise, made worse by leakage induced noise. Stuck-at or similar persistent faults can occur due to process variations [9] [10], either statically or dynamically (dependent on temperature and voltage). This behavior is inherent as silicon doping is a stochastic process, and in small process geometries, a tiny number of dopant atoms can be present in a MOSFET transistor channel. Thus, the stochastic process does not necessarily average out, leading to nearby MOSFET's having very different electrical properties and by implication switching behaviour [11].

The scaling of device feature sizes, operating voltages, and design margins raises a great concern about the susceptibility of circuits to transient faults [13] [14] [15] [16] [17] [18], which can be caused by different physical phenomena (e.g., energetic particle hits originating from cosmic rays, capacitive coupling, electromagnetic interference, power transients) [17]. As technology scales further, variations become prominent as well. The technology nodes below 90nm also referred to as deep sub-micron, experience higher levels of device parameter variations, which are changing the design problem from deterministic to probabilistic [19]. Reliable operation of digital systems is severely challenged, thus pointing to the use of fault tolerance driven design methodologies, not only for the mission or life-critical applications but also, for regular, mass-market applications. To allow for the efficient design of a system that can tolerate faults, a first natural step includes understanding the source of induced errors, but most importantly, their modeling and analysis for guiding the design processes.

When considering transient faults, it is important to note the following: (i)Not all transient faults lead to errors and (ii)Not all errors lead to system failures [5]. In this thesis, the focus is on the first claim estimating the likelihood that a transient fault at the output of an internal gate will lead to an error at primary outputs. The primary goal of this thesis is to allow for accurate modeling and efficient estimation of the susceptibility of combinational circuits to transient faults, including the impact of process parameter variation. Intensive research has been done so far around analysis of transient faults in combinational and sequential circuits [9] [10] [13] [14] [15] [16] [17] [18]. One obvious approach is to inject the fault into the given node of the circuit and simulate the circuit for different input vectors to find whether the fault propagates [20] [8] [21] [22]. However, this approach becomes intractable for larger circuits with a larger number of inputs and thus gives way to approximate approaches that use analytical methods to evaluate circuit susceptibility to transient faults. The proposed framework can be used to reduce the cost of applying various techniques for error detection and correction.

Another unreliability mechanism is unpredictable timing. In [23], it was indicated that sub-powered gate arrival times follow inverse Gaussian distributions, with a long calculation completion time tail. In a practical system, a gate chain has a certain allowed slack, and if individual gate completion times change this may be exceeded, and errors may occur. Even a small error probability at the level of individual gates might result in a significant error probability at the circuit final outputs [24]. It is noted that this tendency is not CMOS specific, as even the most promising post-silicon devices, e.g., Carbon Nanotube Field-Effect Transistors (CNFETs) that are considered to replace CMOS eventually suffer from various amounts of statistical variation in device behavior, potentially leading to a lack of reliability [10]. As a result, reliability is turning out into an important design metric sharing equal importance with the other existing design metrics. Consequently, design time reliability assessment and optimization is becoming a mandatory IC design flow step which targets the reliability improvement for circuits/systems built with unreliable components.

1.2 Objectives and Thesis Contributions

This section lists the research objectives and several important contributions of this work in modeling and analysis of transient faults.

1.2.1 Research Objectives

The first objective of this research is the achievement of systematic synthesis and optimisation of reliable circuits, culminating with a multi-objective circuit design optimization, with respect to its size, energy consumption, latency, and all driven by reliability. This includes various tasks like reliability estimation model, reliability optimisation algorithms and delay degradation timing analysis.

The second objective of this research is a fundamental study on the effectiveness of integrating error correcting codes into the structural (Boolean network) implementation of the circuit logical functionality.

1.2.2 Thesis Statement

This research presents a systematic Electronic Design Automation(EDA) methodology to model the propagation of errors through combinational circuits, optimization, and fault tolerant techniques to improve circuit reliability, and delay degradation analysis. The propagation of transient faults in combinational circuits can be efficiently and accurately modeled using probabilistic symbolic forms the foundation of a computational framework. This framework

can include (i) modeling of transient fault propagation, irrespective of the transient fault origin, (ii) the impact of variability sources on fault propagation and (iii) reliability driven logic optimization and fault tolerant techniques. The main research work presented as part of this thesis is divided into four significant areas as illustrated in Fig. 1.1. More details are provided in the next sub-section.

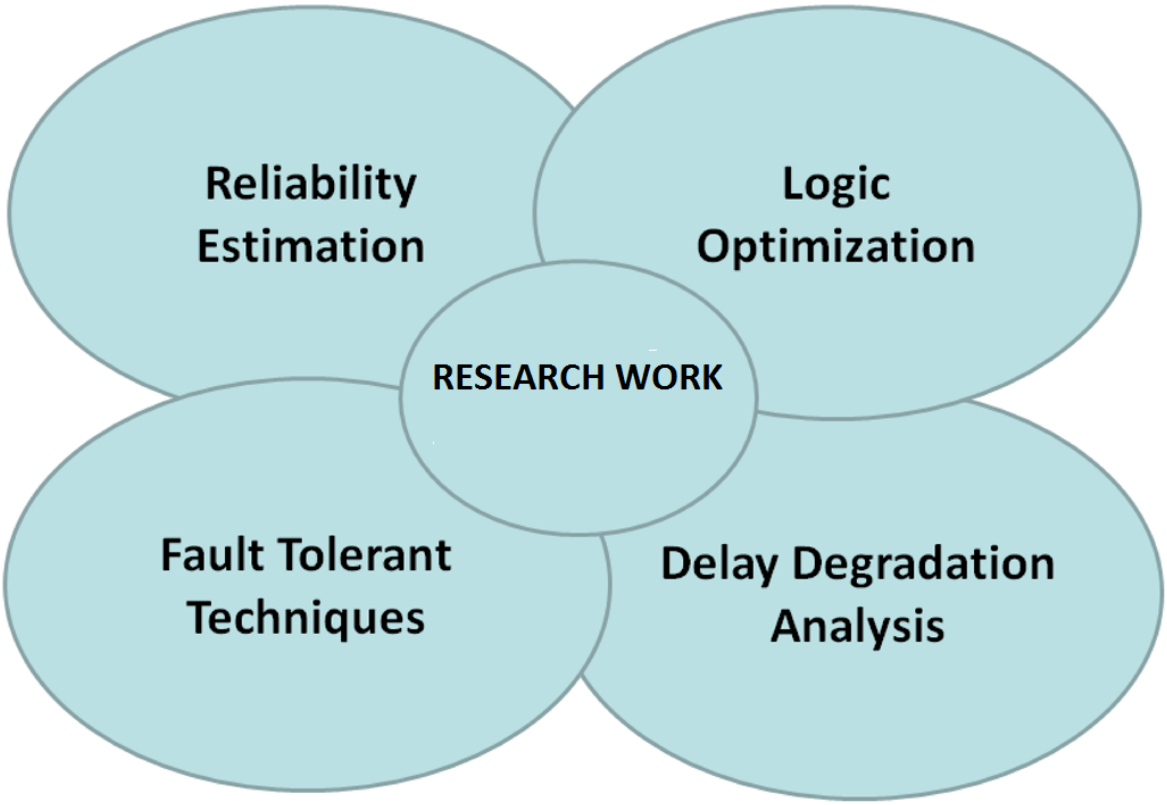


Fig. 1.1 Four significant areas of current research

1.2.3 Circuit Representation and Modification

Over the years, several academic EDA tools [25] [26] [27] have been proposed that provide a programming environment and a stable platform for research in logic synthesis, and power optimization as well as to implement new algorithms. In all these academic tools, data structures and algorithms largely determine the tool efficiency in providing support for implementing new features. AIG is employed as the data structure and ABC as the platform to develop and implement all the reliability related algorithms. ABC [28], which is a logic synthesis and verification tool that performs scalable logic optimization based on And-Invert-Graph(AIG) [29].

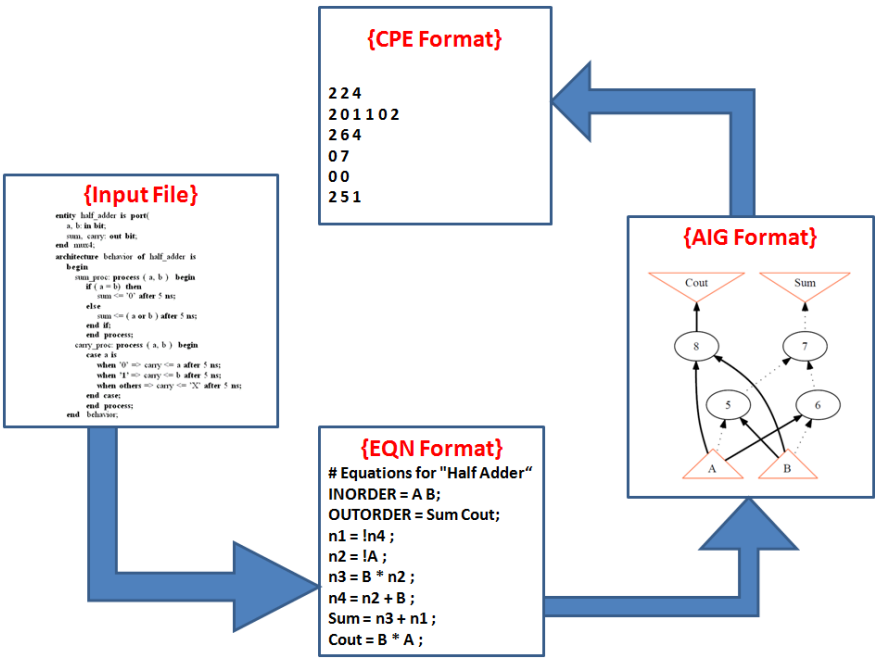


Fig. 1.2 Different Data-structures

As part of the research, A tool has been developed that converts the input Verilog file into multiple internal formats as described in Fig. 1.2.The tool accepts high-level description (e.g., Berkeley Logic Interchange Format(BLIF), VHSIC Hardware Description Language(VHDL), Verilog) of a generic function as an input. It then translates it into the ".eqn" intermediate format and then generates the corresponding AIG representation. During the process of modifying the circuit representation, the logical equivalence of the original and new circuit is guaranteed by adopting various formal verification techniques.

1.2.4 Reliability Estimation and Analysis

Reliability analysis of logic circuits deals with the computation of the impact that gate errors might have on the circuit Primary Outputs(PO). The reliability measures and models (error/energy/power/etc.) are central to designing tools and methodologies. Plain reliability analysis based on HSPICE Monte Carlo simulation is not feasible on real-time circuits due to its prohibitive computation time and computing resource requirements. Several analytical approaches were previously proposed [30]. In this research, the circuits are represented in the AIG format creating the need for novel reliability computation algorithm, with the prime focus being AND and INVERTER gates. Two different methodologies are devised based on simulation-based approach and the probabilistic error gate model-based approach. Fig. 1.3 depicts the complete flow of the reliability analysis tool.

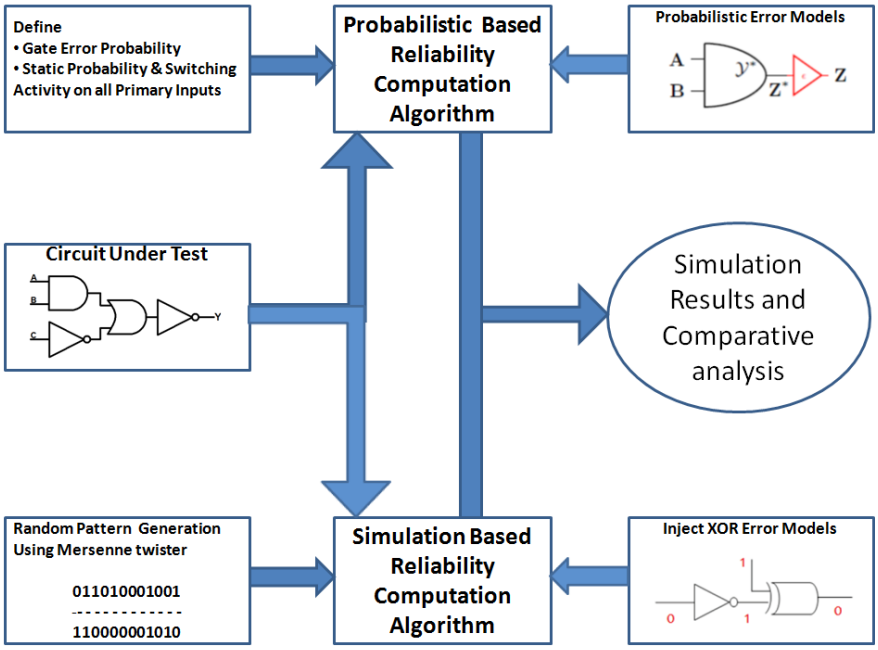


Fig. 1.3 Reliability Estimation Flow

Both the probabilistic and simulation-based reliability computation algorithms are applied on the circuit under test. The probabilistic based methodology emulated all the gates with the probabilistic error models and based on input switching activity, static probability, and gate error values, it computes the expected reliability of the output node. The simulation-based algorithm appends all the gates with an XOR gate to randomly toggle the output value to insert an error randomly. To overcome the default random generator function limitations, the Mersenne twister is used to generate highly random test patterns, which highly facilitates in computing the final output reliability [31]. The final output reports from both these models are compared to define the accuracy of the probabilistic gate error model. Though very accurate, the simulation methodology is very expensive regarding execution time, which might preclude its utilization on large circuits. Eventually, it is a trade-off between accuracy and speed when choosing one of these two algorithms.

1.2.5 Reliability Driven Logic Optimization

Logic optimization and synthesis is the process of taking in a higher-level circuit representation and translating it into real logic gates. The main focus is to develop reliability driven systematic synthesis and optimization methodologies, ideally culminating in multi-objective circuit design optimization. The introduction of reliability as circuit figure of merit leads to a 4-dimensional (area, delay, power, reliability) solution space and has a tremendous influence

on the complexity of the synthesis process. As part of the research, two optimisation techniques employing primitive rewriting techniques [32] are developed with improving circuit reliability as the end goal. A rewriting algorithm is a methodology where small sections of the circuit are identified and are replaced with logical functional equivalent's that improves given constraint(reliability in this case).

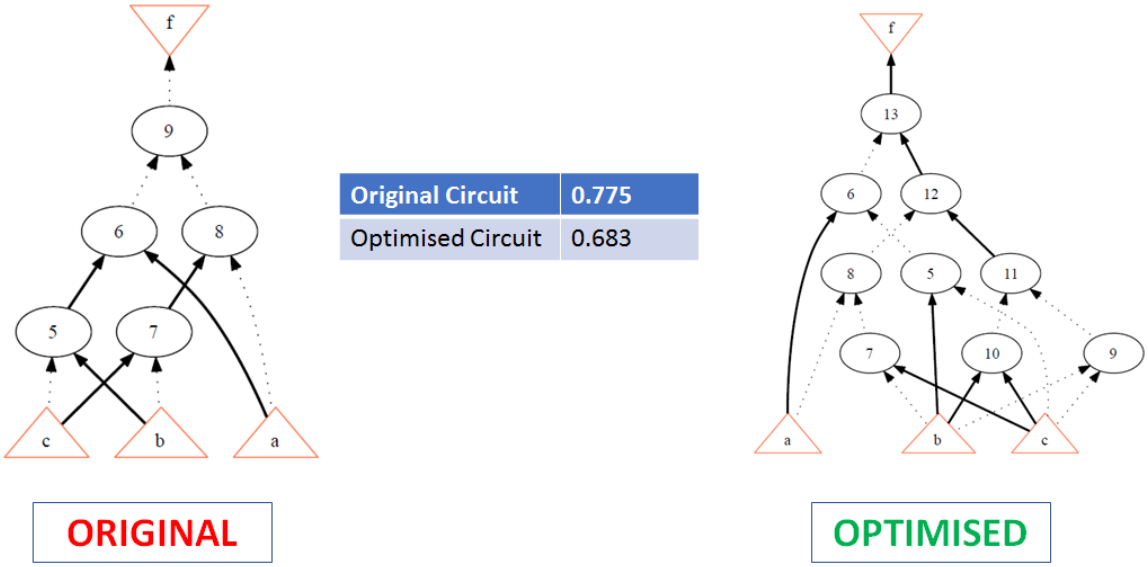


Fig. 1.4 Reliability Driven Circuit Optimization Example

Fig. 1.4 depicts the optimised circuit derived from the original circuit after employing a set of logic transformations. The reliability estimation algorithm is used to pick the logic transformation that provides the highest reliability improvement. This process continued until no more rules can be applied on a given node. The logic network describing the function is updated during each iteration of the optimization algorithm. The similar set of operations are employed on all the nodes in the circuit thereby obtaining the most optimised circuit.

1.2.6 Fault Tolerant Graph Augmentation

Fault tolerant technique is a fundamental study of integrating error-correcting codes into the structural (Boolean network) implementation of the circuit logical functionality that improves circuit reliability. The focus of this approach is not on changing the combinational logic but on augmenting it to enable the retrieval of the correct output even if errors have occurred inside the circuit. This work introduces new reliability driven fault tolerant methodology known as Codeword Prediction Encoder (CPE). Redundant logic is added by using Error Correcting Codes(ECC) based architectures thus enabling to retrieve the correct output and thereby

improving the circuit reliability. Some potential links between the logical representation of a digital circuit and error correcting codes to generate fault tolerant implementation of the logical functionality of the circuit are studied.

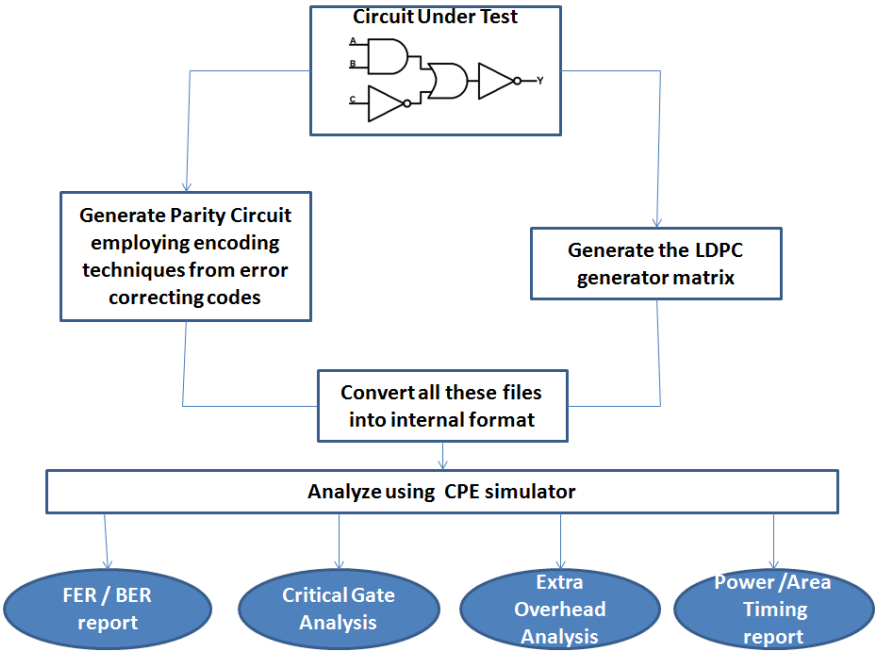


Fig. 1.5 The Codeword Prediction Encoder(CPE) flow

CPE methodology takes the input function netlist and translates it into an AND Invert set of equations for further manipulation and analysis of the number of gates and longest path modifications as shown in Fig. 1.5. An ECC scheme informs the logic network annotation. The two classes of logic functions identified in the research are linear or non-linear functions where linear functions can be described similarly as a linear code(vector-matrix multiplication). The resulting annotated logic network is then decoded using an additional logic network associated with the chosen ECC.

1.2.7 PDF based Delay Degradation Analysis

In semi-custom VLSI design flow, predefined standard logic cells are characterized to generate timing models that are extensively used in Static Timing Analysis(STA) [33]. STA computes timing delays on the critical paths that determines the maximum clock frequency at which the chip can safely operate. A comprehensive delay approximation methodology based on Inverse Gaussian Distribution(IGD) is proposed.

The main idea behind the proposal is first to gather the necessary gate key parameters utilizing Monte Carlo simulations and then linearly extrapolate (propagate) them through the

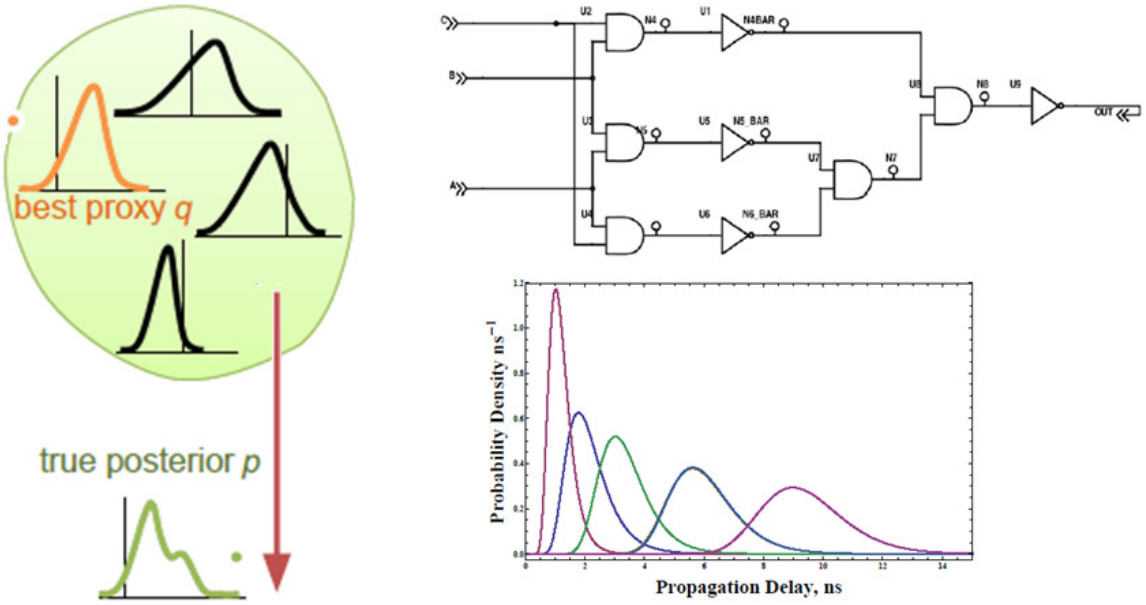


Fig. 1.6 PDF propagation and Methodology

logic network at the circuit level as shown in Fig. 1.6. A Look Up Table(LUT) for all the gates in the library is maintained to capture these critical parameters. The effect of fan-out value and input transition time on the gate delay is also taken into consideration and applies to the following components as well. The proposed analytical PDF delay model is highly accurate for both nominal and sub-threshold supply voltages. Experimental results indicate that the proposed method provides a very close match with Monte-Carlo(MC) simulations reporting the worst case average error of less than 3% while saving run-time by up-to 50 times order of magnitude.

1.3 The Research Framework

This research work is part of a bigger project called i-RISC(www.i-risc.eu). The i-RISC project targets a foundational breakthrough towards reliable, fault-tolerant chip design from unreliable components, which is a crucial issue for the long-term development of computing technology. The research novelty emerges from the synergistic utilization of (1) information theory and coding techniques, traditionally utilized to improve the communication systems reliability and (2) circuit and system theory and design techniques to create reliable/ predictable hardware. The aim is to enable the development of innovative fault-tolerant solutions at both circuit and system level that are fundamentally rooted in mathematical models, algorithms, and techniques from information and coding theory. Fig. 1.7 describes

the five major components involved in this project. This research strictly confines to reliable Boolean function synthesis.

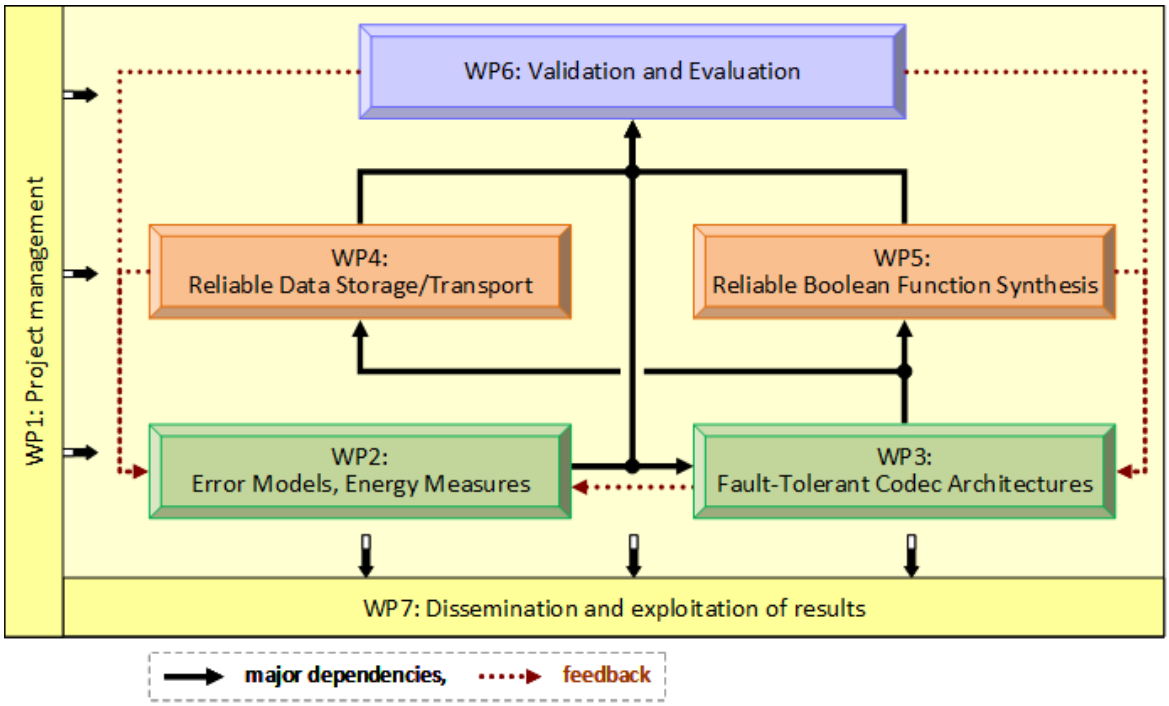


Fig. 1.7 The IRISC Project Details

The design flow that aims to connect different problems addressed through this research consists of several academic tools developed in-house, within the scope of the current research, that are integrated alongside several industrial tools. The complete design flow is presented in Fig. 1.8 connecting the four major contributions described earlier. The design flow outlines the path of a digital circuit from the Register Transfer Level(RTL) to the final error resilient technology mapped gate-level netlist followed by reliability, power, delay and area reports. Some of the important steps in this flow are as follows:

- Step 1: Converting the accepted form (BLIF, Verilog, VHDL) of input file format into AIG.
- Step 2: Run the Reliability driven logic optimization tools to synthesize gate level netlist.
- Step 3: Reliability analysis is performed to compute the improvement achieved regarding error resilience. Reliability details of every node in the network are dumped into an output file.

- Step 4: Gate level simulations are performed to dump the switching activity details. These are saved in the standard Switching Activity Interchange Format(SAIF) format.
- Step 5: Using netlist from steps 1 and 2 and the switching reports from step 4, Synopsys design compiler tool is invoked to perform the power, area and timing analysis. Comparative studies are performed to report the savings or overhead because of the new netlist.
- Step 6: The LDPC encoding scheme is implemented on the resulting reliability optimised netlist. Its functionality drives the parity circuit augmented onto the circuit under test.
- Step 7: Convert all the netlist combinational circuit, parity, and the LDPC decoder into the internal proprietary format understood by the Codeword Prediction Encoder(CPE) simulator.
- Step 8: The CPE simulator is invoked to perform the encoding decoding simulations. All the reports comprising of the Frame Error Rate(FER)/Bit Error Rate(BER) analysis, critical node count, etc. are reported at this stage.
- Step 9: Perform delay degradation analysis on both the original netlist and the optimised netlist. This works as another yardstick to validate the synthesis tool.

1.4 Conclusions

In this chapter, the bigger picture of the challenges posed by miniaturization and reliability is presented. Some introductory topics which provide a backdrop for the research presented in subsequent chapters were covered. Reliability was introduced, followed by a discussion of different types of errors which helped in identifying several challenges associated with CMOS scaling. Also, the larger framework, namely iRISC FET-Open project, in which the current research is part of, is also presented.

Chapter 2 provides a literature survey covering aspects of the current research. Chapter 3 describes the reliability estimation techniques and the theoretical background involved in this thesis. Chapter 4 describes the reliability driven optimization techniques for combinational circuits. Chapter 5 describes the Low-Density Parity Check(LDPC) codes based fault tolerant technique called Codeword Prediction Encoder. Chapter 6 describes the Probability Density Function(PDF) based delay degradation analysis and the corresponding methodology. Finally, chapter 7 concludes this thesis by providing directions for future research.

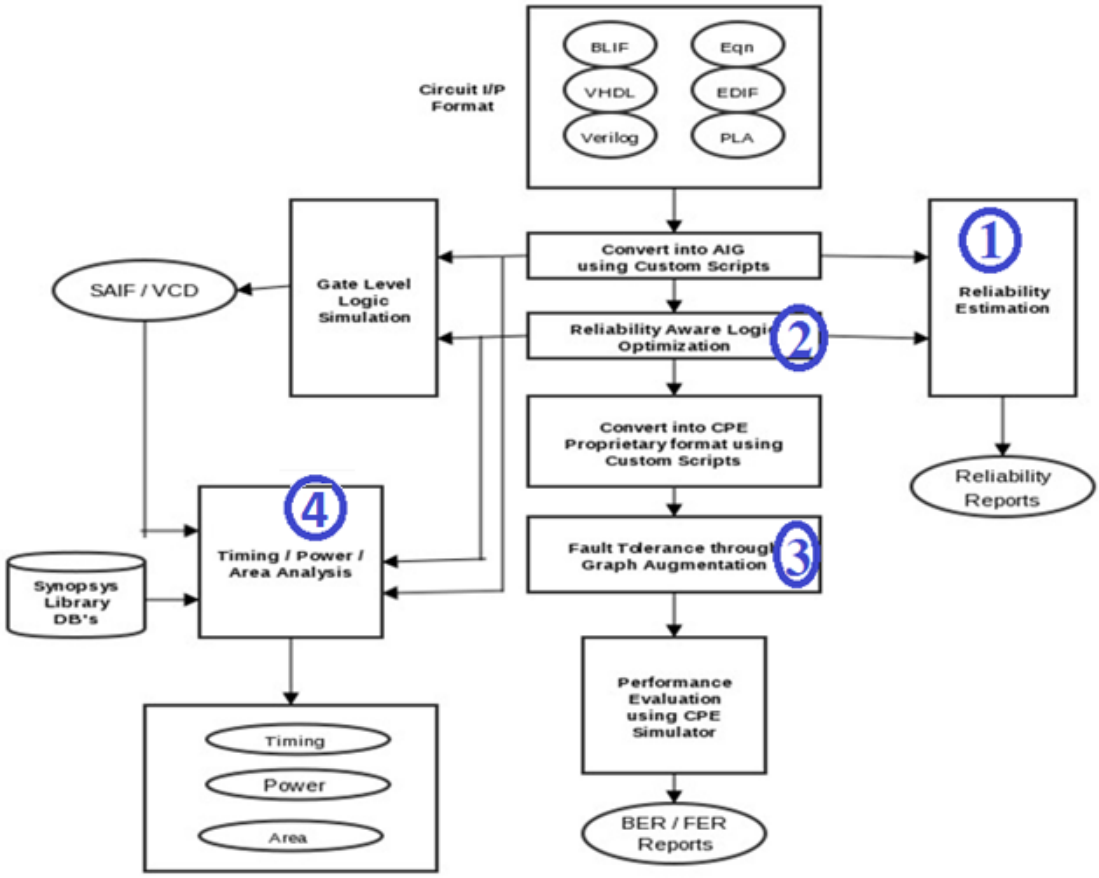


Fig. 1.8 The Complete Flow

Chapter 2

Reliability in Logic Circuit Design

The ongoing miniaturization of data processing and storage devices and the imperative of low-energy consumption can only be sustained through low-powered components. Lower supply voltages and variations in the technological process of emerging nanoelectronics devices make them inherently unreliable. Consequently, the nanoscale integration of chips built out of unreliable components has emerged as one of the most critical challenges for the next-generation electronic circuit design. To make such nanoscale integration economically viable, new solutions for efficient and fault-tolerant data processing and storage must now be invented.

The current research targets a foundational breakthrough towards reliable, fault-tolerant chip design from unreliable components, which is a crucial issue for the computing technology long-term development. The research novelty emerges from the synergistic utilization of (1) information theory and coding techniques, traditionally utilized to improve the communication systems reliability and (2) circuit and system theory and design techniques, in order to create reliable/ predictable hardware. The aim is to enable the development of innovative fault-tolerant solutions at both circuit- and system-level that are fundamentally rooted in mathematical models, algorithms, and techniques from information and coding theory.

2.1 Introduction

Device reliability was first studied in the early sixties when increasingly complex integrated systems were developed and fabricated. Conferences such as the first international reliability physics symposium (IRPS 1962, Chicago) was the first attempts to bring engineers and scientists together from all over the World to study the physics behind various failure effects. To overcome scaling limitations of devices fabricated in ultra-scaled CMOS processes, changes in device structures, processing materials, and processing conditions have been

introduced. These changes have drastically increased the complexity of nanometer CMOS technologies.

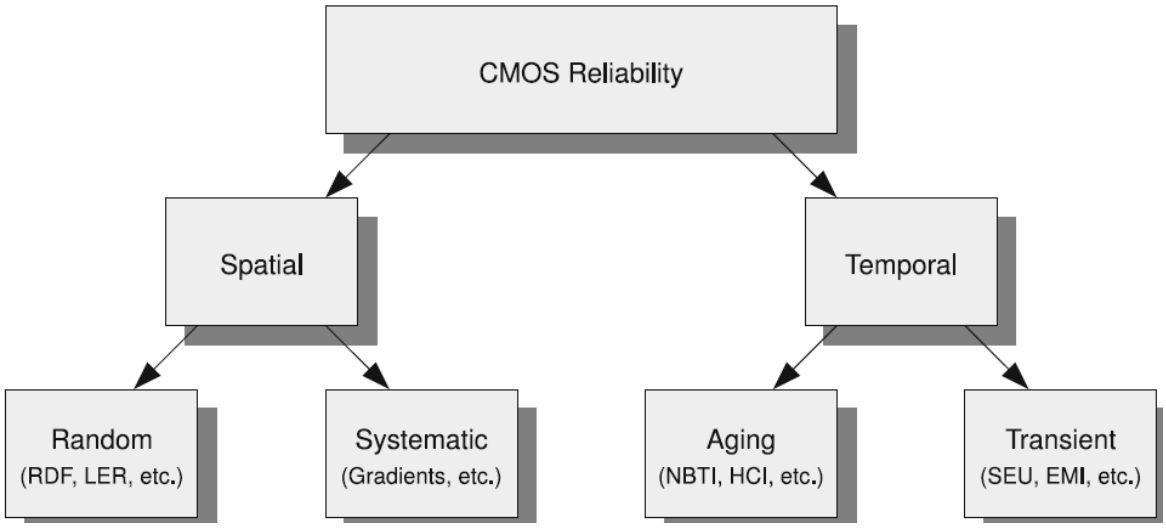


Fig. 2.1 Classification of Reliability

Fig. 2.1 illustrates how nanometer CMOS reliability issues can be categorized into spatial and temporal unreliability effects. Spatial unreliability effects are immediately visible right after production and are fixed in time. Spatial unreliability effects can be random (e.g., Random Dopant Fluctuations (RDF), Line Edge Roughness (LER), etc.) or systematic (e.g., gradient effects, etc.). The results depend on the circuit layout, the neighboring environment, process conditions, and the impact the geometry and structure of the circuit and can lead to yield loss. This yield loss can be functional or parametric, i.e., resulting in malfunctioning circuits or circuits with degraded performance respectively. Temporal unreliability effects, on the other hand, are time-varying and change depending on operating conditions such as the operating voltage, temperature, switching activity, presence and activity of neighboring circuits. A difference is made between wear out or aging effects (e.g., Hot Carrier Injection (HCI), Negative Bias Temperature Instability(NBTI), etc.) and transient effects (e.g., Electro-Magnetic Interference (EMI), Single Event Upsets (SEU), etc.) [9] [34].

2.1.1 Understanding Reliability

A clear understanding of several concepts and terminology related to reliability is needed to proceed with the understanding of the methodologies which are applied to guarantee optimal operation of VLSI systems, fault tolerance, and circuit architectures implementing them. Basic terms such as reliability, fault tolerance, faults, and fault modeling are introduced and explained in detail. For more in-depth details, refer to [35].

Reliability is defined according to IEEE [36] as the ability of a system or component to perform its required functions under stated conditions and for a specified period. The process yield of a manufacturing process is defined as the fraction, or percentage, of acceptable parts among all parts that are fabricated [35]. A system failure occurs or is present when the service provided by the system differs from the specified service or the service that should have been offered. The following three terms are crucial and related to system failure and thus need to be clearly defined, which are named defect, error, and fault.

A **defect** in an electronic system is the unintended difference between the implemented hardware and its intended design. Some typical defects of VLSI chips may be related to process, material, aging effects. The existence or emergence of defects reduces yield [35]. A wrong output signal produced by a defective system is called an **error**. An error is an effect whose cause is some defect. Errors can be classified into three main groups, namely permanent, intermittent, and transient errors, according to their stability and concurrence [35]. A **fault** is a representation of a defect at the abstracted functional level. A fault is present in the system when a physical difference is observed between the "good" or "correct" system and the actual system. The most common faults in a chip are spots and bridging faults caused by silicon impurities, lithography, and process variations [35].

Faults, errors, and failures operate according to a specific mechanism often known as the Fault-Error-Failure chain [37]. In other words, the failure occurs when the delivered service deviates from the specified function, because the system was erroneous, and the cause of an error is a fault. An error is a manifestation in the system of a fault, and a failure is a manifestation on the service of an error [38]. More specifically, one can classify hardware faults that occur during the system operation by their duration into [39]:

- Permanent faults, caused by irreversible device failures within a component due to damage, fatigue, or improper manufacturing. Once a permanent fault occurs, the faulty component can be restored only by replacement, or if possible, by repair.

- Transient faults, triggered by environmental conditions (e.g., voltage fluctuations, electromagnetic interference, radiation, etc.). These events typically have a short duration, and rarely do any lasting damage to the component affected, although they can induce an erroneous state in the system for an abbreviated period. Studies have shown that transient faults occur far more often than permanent ones and are also much harder to detect [5].

- Intermittent faults, caused by unstable hardware or different hardware states, tend to oscillate between periods of erroneous activity and dormancy. Replacement or redesign can repair them.

Since the focus of this research is on transient faults modeling, and the analysis of their impact on circuit reliability, the following sections describe in more details the sources of transient faults, their implications for logic circuit reliability.

2.1.2 Transient faults in logic circuits

Internal or external events can cause transient faults, and generally, manifest themselves as a transient pulse at the output of a logic cell. Externally induced transient faults, stemming from cosmic rays, reaction with Earth's atmosphere or radioactive decay of device packaging, have received most of the attention in recent years, and are claimed to be one of the major concerns for future technology nodes [14] [15].

If a transient fault is generated in a memory cell, or in a memory element (flip-flop, latch), an error resulting from this fault can immediately occur. Otherwise, the created pulse propagates through the circuit and causes an error once the memory cell or memory element latches it. An error caused by a transient fault is often called "soft", because if a failure results in the end effect of this fault, only the data is destroyed. In contrast to this, "hard" errors stem from permanent or intermittent faults that result from the damage in the internal structure of semiconductor material [5].

Once a transient fault occurs at the output of a gate within the circuit, it may propagate through the circuit on more than one path, resulting in re-convergent glitches at different inputs of the same gate in the fanout cone of original gate, or causing more than one soft error at the outputs of the circuit. A significant aspect of transient fault propagation through logic circuits is the set of masking factors that can prevent the fault from propagating to the outputs of the circuit. These masking factors, logical, electrical and latching-window masking [40], are described below.

Logical masking: Logical masking refers to the ability of a logic gate to tolerate faults occurring on its inputs on account of its inherent nature/functionality [40]. Consider a two-input AND gate. When the transient fault propagates to the input port of this gate, on its path through the circuit, if the other input port is tied low controlling value, it would mask the fault from propagating through the gate and, consequently, prevent it from propagating further on that path. It is important to note that different gate types have different logical masking strength. In other words, an inverter will always logically propagate a glitch, since there is only one input carrying the glitch. But, the probability of propagating the glitch through AND, OR, NAND and NOR gate is the same and depends on the number of inputs, while again, a glitch will always propagate through gates XOR and XNOR due to their logic function.

Electrical masking: Electrical masking happens when subsequent logic gates attenuate the voltage transient resulting from a particle strike because of the electrical property of the logic gate [40]. Due to the relation between electrical properties of gates and the size of the pulse representing the transient fault, the fault would be attenuated by the gates it propagates through. This may result in the disappearance of the fault before it reaches any or some of the outputs of the circuits, or it may decrease the duration and amplitude of the fault such that it is not large enough to cause a bit flip in a memory cell or memory element, once it arrives at their inputs. Gates that have more substantial delays, such as XOR and XNOR gates, will attenuate glitches more, while an inverter usually attenuates glitches much less. If the glitch is tiny, compared to the delays of gates it propagates through, it will always be attenuated. On the other hand, if the glitch is very large, compared to gate delays, it will still propagate to outputs.

Latching-window masking: When the transient fault arrives at the input of a memory cell or a memory element, it will be latched only if it arrives on time to satisfy setup and hold time conditions [40]. This depends on the time when the fault occurred inside the circuit, on the location where it occurred (that is, how far that location is from the memory cell or memory element), on the size of the pulse at the occurrence site, the clock cycle and the setup and hold time values.

2.2 Reliability Estimation

Logic circuit reliability analysis attempts to evaluate the impact that the gate errors could have on the circuit Primary Outputs (PO) correctness. Von Neumann pioneered the art of probabilistic error analysis and defined any system to be reliable only if the probability of its correct output is greater than a certain threshold [41]. The traditional approach to reliability analysis begins with simple SPICE simulations to estimate the circuit error probability. However, for most of the practically relevant circuits, simple reliability analysis based on SPICE Monte Carlo simulations is not feasible due to prohibitive computation time and excessive resource requirements. The impact of faults can be investigated through simulation. While faults can be simulated accurately only at the circuit level of abstraction, there are existing proposals that inject the fault at a high level of abstraction for early performance exploration [42] [43].

Complementing the simulation methods, various probabilistic analytical approaches to evaluate the circuit reliability have been proposed. The Probability Element (ProxEl) method was introduced in [44] to alleviate the typical problems encountered by Monte Carlo simulation and partial differential equations. In [45], the authors introduced the

Probabilistic Transfer Matrices (PTM) formalism, which relies on an exhaustive listing of the gate inputs/outputs, allowing simultaneous and exact reliability evaluation over all possible input combinations. Another analytical reliability estimation approach relies on the Probabilistic Gate Model (PGM) [46] [47] [48]. While it is applicable to potentially any gate and failure, the method assumes that the gate input/output signals are statistically independent, which leads to approximate reliability results. In [48], the correlations in the input signals or caused by re-convergent fan-outs are addressed by sequentially decomposing and recursively treating each fan-out, at the expense of increasing the computational time exponentially with the number of re-convergent fan-outs. In [30], the authors propose three scalable algorithms for reliability assessment. Notably, the single-pass reliability analysis algorithm can: (i) accurately evaluate the reliability of circuits without convergent fan-out and (ii) to assess approximately the reliability of circuits exhibiting spatial correlations, by computing pairwise correlation coefficients of dependent signals. The algorithm is based on expressing the error at a gate output as the cumulative effect of the intrinsic, local gate error component and an error component attributed to the failures of the gates in its fan-in cone. The single pass method is extended in [49] to multiple passes for reliability evaluation of sequential circuits. The Signal Probability Reliability Analysis (SPRA) method was proposed in [50] [51] embeds the cumulative effect of multiple, simultaneous errors in a circuit, in the form of a bit-flip error at the output of a faulty gate.

Also, Bayesian Networks (BNs) have been applied in the context of circuit reliability evaluation [52] [53]. BNs, whose underlying semantics are based on directed graphs, allows one to capture both the temporal and spatial circuit dependencies comprehensively, providing an exact and minimal probabilistic model for reasoning and inference in causal logic networks. The Markov Random Fields (MRFs) based reliability evaluation approach presented in [54], employs the Gibbs distribution to characterize the reliability in terms of entropy and the noise in terms of thermal energy. While being suitable for reliability assessment of small circuits or of conventional redundant architectures, such as NAND multiplexing and triple modular redundancy [55], for arbitrary multilevel logic circuits, the MRF-based approach becomes computationally intensive because the minimization technique in the Gibbs distribution function has a substantial number of variables.

In general, the reliability estimation techniques are based on either simulation method or probabilistic methodology. The complexity and accuracy of simulation-based techniques strongly depend on the scale of circuit and test vector selection [52]. As discussed, several analytical methods such as those using Probabilistic Transfer Matrices(PTMs) [56], Probabilistic Gate Models(PGMs) [48] and Probabilistic Decision Diagrams(PDDs) [57] have already been proposed to investigate behavior of circuits under faults. PTM suffers from

massive matrix storage and manipulation overhead that results in its inapplicability to large circuits. Another approach based on Bayesian networks was proposed in [53]. Though they apply to medium scale circuits, manipulating Bayesian networks for large circuits has been seen to be intractable. One of the best approaches to date was described in [30], with minor limitations. For example, the observability based algorithm efficiency is directly linked to the ability to observe internal nodes. From VLSI testing principles, it is understood that as the circuits grow larger and larger, the complexity of observing a node turns out to be more time-consuming. In the same work, the single pass reliability analysis algorithm is proposed which provides excellent results for circuits without re-convergent fan-out. But, the computational complexity increases as the number of correlation coefficients increases.

2.3 Reliability Aware Logic Synthesis

Traditional logic synthesis methodologies and EDA tools are centered on fulfilling timing, power, and area constraints or on achieving acceptable trade-offs among those [Pedram96] [Mehrotra11]. However, as the Complementary Metal-Oxide-Semiconductor (CMOS) technology entered the nanometer era, such an approach cannot cover any longer all the relevant design aspects. Technology scaling has precipitated higher operating speeds, lower operating voltages, and lower operating noise margins; all of which contribute to reduced switching energies, allowing legitimate logic signals to be readily overwhelmed by single-event-induced charge-collection transients [Dodd03]. Nanotechnology specific issues, e.g., power supply voltage (Vdd) reduction, the higher impact of the process parameter and temperature variations, resulting in increased device failure rates, making CMOS Integrated Circuits (ICs) less reliable [Borkar05] [Constantinescu03].

In the era of deep sub-micron CMOS technology, spatial and temporal variability is resulting in less predictable device behavior [9]. Stochastic logic delay variation on par with the nominal delay can occur due to the local (or intra-die or within-die) variations in transistor V_{th} [58]. Further, circuits operated in the near or sub-threshold region to achieve substantial power savings results in an increased amount of output Soft Error Rate (SER). Given the combined effect of variability and aging mechanisms, it is evident that the state of the art CMOS gates is highly unreliable. A variety of system/circuit level techniques like Dynamic Reliability Management (DRM), and inexact computing [59], [60], [61] have already been proposed to overcome reliability related concerns. These techniques are at a higher abstract level and do not leverage the significant gains that are achieved by employing graph modification/altering techniques. Reliability driven logic synthesis is one area that has not received much attention but is gaining a lot of importance

in the last few years. In [62], the circuit output Soft Error Rate (SER) is reduced through localized circuit restructuring by taking advantage of don't care based re-synthesis and local rewriting. In [63], a technique to improve the circuit robustness to soft errors based on Redundancy Addition and Removal (RAR) by eliminating gates with a large contribution to the overall SER is proposed. Efficient algorithms for synthesizing approximate circuits for simultaneous masking of logical and timing errors were proposed in [64]. Automatic Test Pattern Generation(ATPG) based rewiring method, yet structurally-different implementations to reduce the SER was implemented in [65].

2.4 Fault Tolerant Techniques

Fault tolerant techniques for improving the reliability of digital circuitry have been of interest for a long time. Von Neumann [41] first introduced a classification of the error type and proposed solutions based on multiplexing techniques as early as 1956. From a hardware implementation point of view, many schemes have been presented to deal with faulty gates. One of the most fundamental and useful fault-tolerant mechanisms introduced in that work was the well-known R-fold Modular Redundancy (RMR) [66], where R replicas (R = 3, 5, 7, . . .) of a computing subsystem present their outputs to a voter block that generates a reliable output based on a majority criterion. The RMR in its R = 3 version, Triple Modular Redundancy(TMR), has been widely used in the design of systems where reliability is considered a vital issue. This approach came directly from Von Neumann work and had been used in many scenarios [67], [68], [69]. The flip side of this technique is that it can lead to large hardware consumption due to the need for implementing the logic unit three times and then majority voting on the result.

Despite the full spectrum of research works based on the RMR technique, almost all of them analyze the reliability issue from a static point of view. In other words, given a set of error-prone data replicas generated by R independent subsystem replicas, the majority of studies seek to determine the reliability characteristics of the RMR structure without any temporal consideration. Other approaches focus on detection of the error, and some examples consist of Berger Codes [70] for unidirectional errors, m-out-of-n codes [71], 2-rail checker [72], where input and logic are modified to force two outputs to have complementary values if no fault encored. Attempt to introduce error control coding techniques in the IC design flow in an automated fashion have started and some relative information can be found in [73] [74]. Another approach to detecting an error and that is general enough is the concept of Check Symbols Generation or Parity Prediction Functions [75], [76] where circuitry is added to generate extra bit to ensure parity.

A different approach to improve fault tolerance is based on the use of methods derived from Error Control Coding (ECC) theory to protect the combinational logic that implements a Boolean Function. The focus of this approach is not on changing the combinational logic but on augmenting it to enable the retrieval of the correct output even if errors have occurred. Taylor [77] has done important work to cross the field of the circuit design with the knowledge of error correction theory, [78] that used LDPC codes to build fault-tolerant storage and computation architectures on unreliable systems. Taylor's approach has been the base for numerous other works [79], [80], [81]. Following Taylor's approach, LDPC decoders on unreliable hardware have been widely investigated [81], [82]. As the main result, it was shown that when some of the decoder parameters (number of quantization levels, channel value, etc.) are carefully chosen, LDPC decoders are naturally robust to faulty hardware, with no need for additional circuitry. Unfortunately, it was also shown that LDPC encoders fail entirely when they are built from unreliable gates [83]. The focus of the current work is thus on constructing reliable LDPC encoders constructed from unreliable gates.

2.5 Static Timing Analysis

The traditional approach is to perform highly accurate SPICE simulations with the downside being the inevitable long runtime. To overcome this, corner analysis has been widely adopted and deals with multiple Process, Voltage, and Temperature(PVT) corners. However, the high sensitivity and unpredictability of deep sub-micron CMOS devices turns this approach to being either overly pessimistic or optimistic [84]. Several improved methodologies have been proposed to achieve better accuracy within acceptable computing time. Statistical Static Timing Analysis(SSTA) [85] was proposed to determine the distribution of propagation delays and signal timing violation on digital CMOS circuit critical paths. Nonetheless, SSTA requires burdensome efforts to automate the approach while disregarding the input pattern delay dependence. To overcome these obstacles, Monte Carlo Static Timing Analysis(MCSTA) [85] and Dynamic Timing Analysis(DTA) [86] were proposed. The MCSTA is the one-off generation of a Variation Cell Library for standard cells, which is used to carry out static timing analysis to create thousands of randomized gate-level netlist. MCSTA can be considered as a trade-off between the time-consuming Monte Carlo(MC) SPICE simulation and the relatively inaccurate SSTA. A statistical DTA approach that employs the normal Gaussian approximation to model the propagation delay based on distinct input patterns [87]. While reasonably accurate, the approach can be costly concerning processing time, as its accuracy directly depends on the number of considered input vectors.

A propagation delay estimation algorithm [87] based on GD was employed to compute delay, where a close match was found between the measured propagation delay profile and the Gaussian Probability Density Function(PDF). However, the choice for approximating the delay PDF with a normal distribution was based on fitting only two Monte Carlo simulations with no scientific explanation to support the model. Although the GD delay model efficiently estimates the delay under nominal voltage supply, several GD characteristics hint its inability to capture delay data distributions in the general case. By definition, GD is characterized by a function with the field that stretches from $-\infty$ to $+\infty$, which indicates that it assumes non-zero value also for negative time values. This is a clear mismatch with the circuit physical reality since no signal propagation delay can be negative. Furthermore, the normal (Gaussian) distribution is symmetric around its mean value, which may not always be the case when considering gate delay data. The simulation results described in the next section demonstrate that the GD symmetric property does not hold true for the cases of interest.

In this line of thought, several other improved methodologies have been proposed to achieve better accuracy within acceptable computing time. The practice of using Probability Distribution Function(PDF) to model propagation delays known as Statistical Static Timing Analysis(SSTA) is most popular [88]. One typical example of SSTA application is the use of the Gaussian distribution function [87]. This is appropriate when the CMOS component is in the strong inversion regime. However, for low supply voltages, the propagation delay profile is highly non-Gaussian [58]. Various distributions have been successfully used to model the propagation delay profile of a CMOS component for these regimes which include the weak and moderate inversion regimes [89], and the Log-Normal Distribution [90]. Nonetheless, SSTA requires burdensome efforts to automate the approach while disregarding the input pattern delay dependence. To overcome these obstacles, Monte Carlo Static Timing Analysis(MCSTA) [85] and Dynamic Timing Analysis(DTA) [86], linear composition methodology [23] were also proposed.

2.6 Conclusions

The initial sections of this Chapter present the reliability and its theoretical understanding. Different types of faults are discussed with special emphasis on transient faults. Based on this, approaches for reliability estimation and optimization employing multiple techniques were considered. In the second half of the Chapter, a review of reliability estimation and optimization techniques was presented.

The reliability estimation review covered traditional probability based as well as simulation-based techniques such as PTM, PGM, Bayesian Networks based techniques. Further, the

efficiency of these techniques with reconvergent fanout as the constraint was also discussed. Reliability aware logic synthesis section lists different logic optimization techniques that employ localized circuit restructuring to reduce the soft error rate of the circuits.

The fault tolerant techniques literature survey identified a range of different techniques with special emphasis on LDPC based applications. The main conclusion of these papers was while the LDPC based decoders are robust to handling errors, the encoders are not. The final topic of interest is static timing analysis. Techniques like Statistical Static Timing Analysis(SSTA), Monte Carlo Static Timing Analysis(MCSTA), Dynamic Timing Analysis(DTA) were briefly discussed. Further, the inverse Gaussian-based Probability Density Function(PDF) captures the timing data better compared to Gaussian is also discussed.

In the next chapter, some novel reliability estimation techniques using probability and simulation-based techniques.

Chapter 3

Reliability Estimation

3.1 Introduction

The main factors driving the design of digital systems for a long time were cost, performance, and, more recently, power consumption. However, with continuous technology scaling, the diminishing value of reliability of deep sub-micron technologies is a fact thereby the reliable operation of digital systems is severely challenged. Moreover, it is widely accepted that nanoelectronic based systems will rely on a significantly lower reliability rate than what was known so far. Thus, fault-tolerant reliability driven design methodologies are the need of the hour. To allow for the efficient design of a system that can tolerate faults, a first natural step includes understanding the source of induced errors, but most importantly, their modeling and analysis for guiding the overall design process. Hence, the first step in building a reliability-aware design is to develop an efficient algorithm that computes circuit reliability.

3.1.1 Main Contributions and Outline

In this research, the problem of computing circuit reliability for combinational circuits is investigated employing two different approaches. The first approach consists of simulating the combinatorial circuit under test by injecting faults with specific characteristics to gather statistical information on how these errors propagate through the circuit. The novelty lies in the fact that Mersenne twister is used over the traditional 'RAND' function to generate random vectors [31]. The second method called Conditional Probabilistic Error Propagation (CPEP) [1] [91] [2] builds a probabilistic error model of the combinatorial circuit and calculates the reliability of the circuit based on mathematical analysis of the model. This is quite generic and can be applied to any error scenario and for any logic gate. But, in the synthesis algorithms presented in the subsequent chapter, the circuits are represented in the

AIG format, hence the prime focus is placed on AND and Inverter gates. The algorithm uses the Dynamic Weighted Average Algorithm (DWAA) [92] approach to account for the impact of re-convergent fanout on the overall results.

Both these techniques have trade-offs WRT each other, and one supersedes the other based on specific application. Though the probabilistic-based method is high-speed, it can be slightly less accurate. The simulation-based methodology is very accurate but can be time-consuming especially for large circuits. As this tool would be a part of a new logic synthesis framework, the analytical methodology is used in intermediate steps where it is required to compare two configurations to make a fast decision of choosing which is better. Exact reliability numbers are not required at this stage which enables the algorithm to do with less accuracy. The simulation-based methodology is used during sign-off stages when accurate reliability numbers are needed.

3.2 Simulation Based Reliability Computation

The most accurate and straightforward approach to reliability estimation is by circuit simulation using a set of input vectors. The transitions that occur can be easily observed for a gate and can be averaged out for the set of input vectors to give an estimate of circuit reliability. Simulation-based reliability estimation technique can be employed at various levels of abstraction, namely: switch-level, gate level, and block-level. Only gate-level simulation is of interest since it provides very accurate activity estimates and is significantly faster than switch-level simulation, which simulates transitions at the transistor level. Block-level simulation, which considers larger blocks such as registers, adders, multipliers, memories, and state machines, is also not considered. Commonly known as Gate Level Simulation(GLS) technique employed in power estimation involves the use of logic components like AND/INVERTER gates, latches, flip-flops and interconnection nets. The most common analysis method involves an event-driven simulation [93]. When a transition or event occurs at an input of a gate, it may trigger an output event after a certain time delay. Power consumption is estimated by calculating the switching capacitance at the node of the gate, and by the number of events that occur at that node.

3.2.1 Mersenne Twister and Random Number Generation

One of the most important constraints related to the gate level simulation technique is the availability of significantly random input vectors. The traditional random function *rand* from the 'C' library provides random numbers with a small period of length 2^{32} - 1. Since the

number of required simulation steps runs into millions, there is a high possibility that patterns are repeated and highly correlated with each other. To overcome this issue, a pseudo-random number generator called Mersenne twister is employed. The Mersenne Twister(MT) is a Pseudo-Random Number Generator (PRNG) developed by Makoto Matsumoto and Takuji Nishimura in the 90's [31]. The most commonly used version of the Mersenne Twister algorithm is based on the Mersenne prime 2^{19937} - 1. The standard implementation of that, MT19937, uses a 32-bit word length. There is another implementation that uses a 64-bit word length, MT19937-64; it generates a different sequence. Its name derives from the fact that its period length is chosen to be a Mersenne prime. In mathematics, a Mersenne prime is a prime number of the form $M_n = 2^n$ - 1 and has a very long period of 2^{19937} - 1. Thus, it helps in overcoming all the simulation limitations.

3.2.2 Gate Error Model

Fig. 3.1 graphically presents the unreliable gate model employed to inject errors onto the gate outputs with a pre-defined gate error probability. An unreliable AND gate is modeled as an ideal (error-free) AND gate followed by a faulty XOR that determines the stochastic error behavior by toggling the gate output with a pre-defined probability. While the input nodes A & B in Fig. 3.1 can be faulty based on a level of traversal of the signal, the input node 'E' toggles between '0' & '1' based on the pre-fixed gate error probability value. To explain, if the gate error probability of an AND gate is 0.005, then for every thousand iterations, the input node 'E' of the XOR gate would be set to '1' five times.

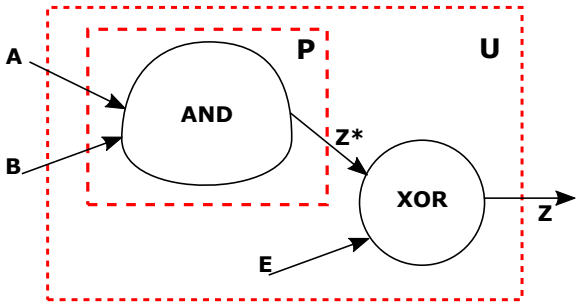


Fig. 3.1 Gate Error Model for GLS based technique.

3.2.3 The Methodology

Gate-level simulation-based reliability estimation involves simulating a Boolean network consisting of logic gates while keeping track of transitions, both error-free and error-prone conditions, to determine error probability for each node in the network. During a simulation,

the value at the output of a gate is determined from the values at the input of the gate each time an input changes. Fig. 3.2 depicts the unreliable data transmission. While the AND gate output is correctly transmitted as '1', the inverter output is toggled to '0' based on gate unreliability. Two specific commands are used to manipulate the default random number generator.

- mt_{seed}(void): Choose a seed from random input;
- mt_{lrand}(void): Generates 32-bit random value.

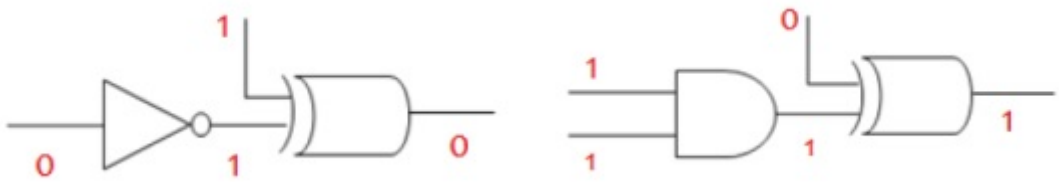


Fig. 3.2 Unreliable Data transmission through on Inverter and AND Gates.

The tool generates random vectors and applies them on the Primary Inputs(PI) of the AIG based circuit representation. The tool can process multiple simulations runs simultaneously, analyzing how reliability varies with input vectors. Extra XOR gates are employed to model the error on each gate to compute the numbers under the ideal and faulty scenario. First, the following four important variables are defined:

- pNtk: It is a pointer to the network (circuit) under consideration;
- pNode: It is a pointer to the node under consideration in the circuit (all the Primary inputs, Internal Nodes and primary outputs will be treated as Node);
- pLeft_{Child}: It is a pointer to the Left Child of the node under consideration;
- pRight_{Child}: It is a pointer to the Right Child of the node under consideration.

The generated random patterns are applied unto the primary inputs of the circuits, and the output value at each internal node is calculated by considering the circuit to be ideal. The computation is carried out as follows: traverse through the node in sequential order and for every node derive the right and left child using the built-in function.

After the calculation of the left and right child, the next check is to figure out if there is an inverter at the child or not. A particular item within the data structure $\text{pNode}_{UC} \rightarrow$ fCompl0 is used to denote this feature. If this value is equal to zero, then it indicates that there is no

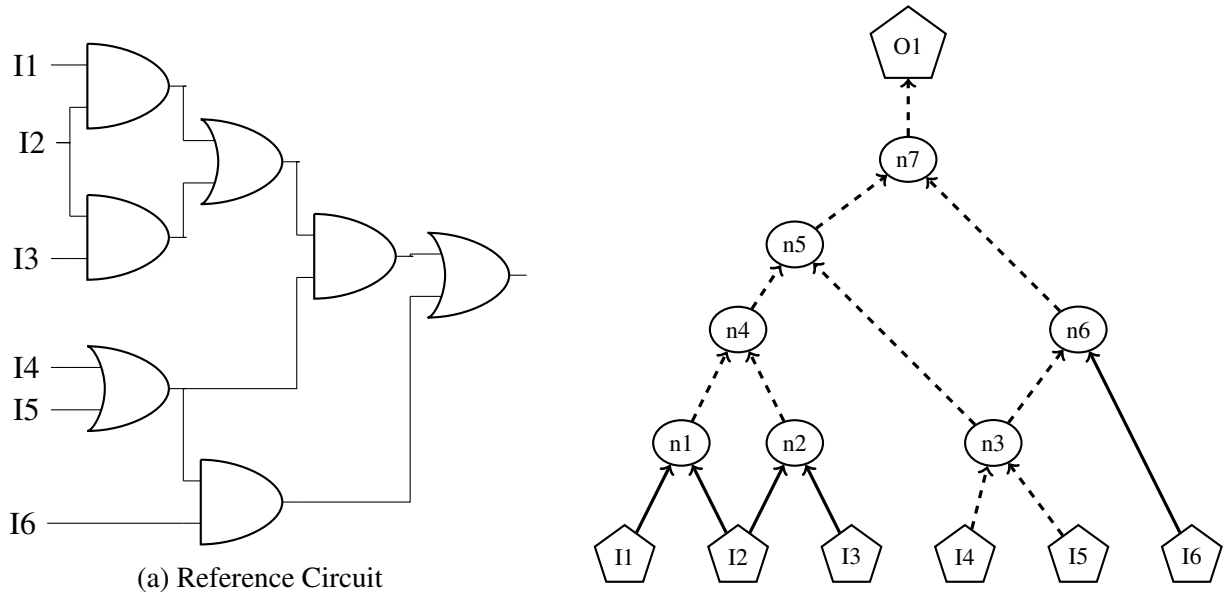
inverter at the left child and if the value is 1, then it suggests the presence of an inverter at the left child. Similarly, $\text{pNode}_{UC} \rightarrow \text{fCompl1}$ if this value is equal to zero then it indicates there is no inverter at the right child, and if the value is one then it suggests the presence of an inverter at the right child.

Until now, all the AND gates, as well as inverters, has been fault free but to compute the circuit reliability faulty gate error models must be employed. This is achieved by using XOR gates at each ideal gate output. An XOR gate is applied just after each gate with one input from the gate and other from a randomly generated vector which defines the gate error probability. Now, the output of the XOR gate will be random in accordance with the inputs and hence the gates can be regarded as faulty. Once, both ideal and flawed values on all the nodes are captured, the error probability and in turn the reliability is efficiently computed. To compute the error probability, traverse through each node and check whether the Ideal value is equal to faulty value. If yes, then increase the error probability count for that node.

The procedure is repeated till all the node have been traversed, and respective error probability count values are updated after each iteration. After the completion of all the iterations, for each node divides its error probability count by the number of iterations to calculate the error probability.

3.2.4 Limitations

Gate-level simulation is a well-studied problem, and much effort has been placed on improving its speed [94] [95] [96]. The advantage of this technique is its accuracy compared to probabilistic methodologies and the fact that it can be used irrespective of circuit, technology or design style. However, it is highly pattern dependent and hence suffers from two major drawbacks. First, it requires extensive use of computing time, especially for large circuits. Moreover, simulation requires input vectors, which are often not available when designing a new system. Thus, the results may be erroneous as some of the input patterns used for estimation may never occur during normal operation. Keeping this criterion in mind, this methodology is used only for final estimation limiting ourselves to use the probabilistic methodology most of the time during the intermediate computations. In this line of thought, a novel analytical model called Conditional Probabilistic Error Propagation (CPEP) is presented next. The biggest advantage of the probabilistic model when compared to simulation-based methodology is the computation time with little compromise in accuracy.



(a) Reference Circuit

(b) AND Invert representation

Fig. 3.3 And Inverter representation of Combinational Circuit

3.3 CPEP: Conditional Probabilistic Error Propagation

Probabilistic methods involve modeling transitions occurring at a gate as probability functions. The probabilities of the nodes to change their logic state are propagated through the circuit. Since probabilities are used, no input vectors are required, resulting in a reduction in computational effort. Thus, these techniques are considered as pattern independent [97]. Issues like signal independence, spatial and temporal signal correlations determine the accuracy and complexity of the technique [98].

3.3.1 The Data Structure

In this thesis, And-Inverter Graphs (AIGs) is used as the underlying data structure to model logic circuits. It becomes a natural choice mainly due to the following three reasons. Firstly, using AIG makes it simple to demonstrate the implementation of this algorithm due to traversing an AIG can be directly modeled as traversing interleaved binary trees. Secondly, it is well known that all logic circuits can be constructed using AND and Inverter gates. Thirdly, it will be much easier to integrate this algorithm to some open source EDA tools such as (ABC) [28] which use AIG as the data structure to model logic circuits as well. As stated in graph theory, each graph is composed of two basic types of elements, *i.e.,* vertices (or nodes) and edges [99].

Therefore, it is institutional to use nodes and edges representing logic gates and interconnections between gates respectively. A simple AIG in Fig.3.3 illustrates essential elements in the AIG model. As shown in the figure, AND gates are denoted by egg-shaped nodes and INVERTERS are denoted by the dashed line respectively; the trapezium-shaped leaves denote the primary inputs with the label character, primary inputs share the same label indicates a fanout and the root node represents the primary output.

3.3.2 Gate Error Models

In this approach, a Von Neumann erroneous gate is modeled as an ideal logic gate cascaded with a faulty buffer as shown in Fig.3.4. The two nodes Z* and Z are named as internal and the external output node. As AIG's are used to represent combinational circuits, the analysis is limited to probabilistic gate error models for AND and Inverter logic gates only.

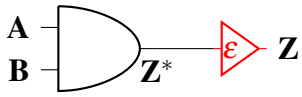


Fig. 3.4 Unreliable AND Gate Model

Some of the common conventions used throughout this section are listed below:

- $P(x_0)$ - Probability of node 'x' to be 0;
- $P(x_1)$ - Probability of node 'x' to be 1;
- $P(x_\varepsilon)$ - Probability of error on node 'x';
- $P(x_c)$ - Probability that an error is masked;
- EP_y - Error probability of Logic Gate 'y';
- R_y - Reliability of Logic Gate 'y'.

Consider an unreliable AND gate, such a gate can be modeled as an ideal (error-free) AND gate followed by a faulty buffer that accounts for the stochastic behavior of the errors. This model moves the entire error statistic on the output, and so it implicitly assumes a symmetrical error behavior concerning the inputs. This assumption holds true for many physical gate implementations and simplifies analysis. This is an assumption made by this algorithm so that an error injection gate can be equated to a normal logic gate when traversing the graph which dramatically simplifies the algorithm. For the unreliable AND gate shown

in Fig.3.4, the static probability as defined in Eq.3.1 is the union probability of SP_A and SP_B assuming input A and B are independent.

$$SP_{Z^*} = SP_A \times SP_B \tag{3.1}$$

To analyze this model behavior, let's consider that errors on the output of a combinatorial gate can be due to two reasons: (i) Errors on the gate input nodes and (i) intrinsic errors within the gate. In this work, the primary inputs of the complete circuit are assumed to be error-free and statistically independent.

3.3.3 2-Input Ideal AND Gate

As AIG's comprises only 2-input AND gates, the analysis is restricted to 2 input AND gates and note that its extension to multi-input AND gates is straightforward. The static probability values of the internal output node Z* can be expressed as:

$$\begin{aligned} P_z^*(1) &= P[A = 1, B = 1] \\ P_z^*(0) &= 1 - P_z^*(1) \end{aligned} \tag{3.2}$$

First, the error on the internal node (Z*) due to the errors on the input nodes of an ideal AND gate is computed. It must be noted that error on the inputs need not necessarily result in wrong output value. Consider error free '0' on one of the input pin. This would mask the error on the other input node from being propagated onto the output. Similarly, consider the scenario where the input pins are set to '0' & '1' and both are in error. This event of double error mutually negates each other and will still result in a correct output state. Hence, due to the masking and double error events, there are no simple rules to predict the state of the output. Tab. 3.1 presents an exhaustive enumeration on all the possible cases. To explain the table, consider the case of input A=0 and B=0. Then, the internal output Z* is ideally 0. Now, each of the inputs can assume an error (ε) or a correct (c) state. The state of the inputs determines if Z*=0 is correct or not. For these inputs values, the internal output node is in error if and only if both the inputs are in error. It is evident from Tab. 3.1 that error on one or both inputs need not necessarily translate into output error. Only 6 of the possible 16 cases result in an error on the internal output. The probabilities for each of these events to occur are presented in the last column.

To arrive at a closed form representation of AND gate output node error probability, it is assumed that the inputs of the gates are independent (i.e., there is no re-convergence) to

Reliable Condtions			Unreliable Condtions			Error Probability
A	B	Z*	A	B	Z*	
0	0	0	c/0	c/0	c/0	
			c/0	ε/1	c/0	
			ε/1	c/0	c/0	
			ε/1	**ε/1**	**ε/1**	**P[A=0,B=0,A_ε,B_ε]**
0	1	0	c/0	ε/0	c/0	
			c/0	c/1	c/0	
			ε/1	ε/0	c/0	
			ε/1	**c/1**	**ε/1**	**P[A=0,B=1,A_ε,B_c]**
1	0	0	ε/0	c/0	c/0	
			ε/0	ε/1	c/0	
			c/1	c/0	c/0	
			c/1	**ε/1**	**ε/1**	**P[A=1,B=0,A_c,B_ε]**
1	1	1	**ε/0**	**ε/0**	**ε/0**	**P[A=1,B=1,A_ε,B_ε]**
			ε/0	**c/1**	**ε/0**	**P[A=1,B=1,A_ε,B_c]**
			c/1	**ε/0**	**ε/0**	**P[A=1,B=1,A_c,B_ε]**
			c/1	c/1	c/1	

Table 3.1 Ideal AND Gate with Unreliable Inputs

simplify analysis. This allows for the utilization of simple formulas to compute reliability and reduce the overall algorithm running time. Under such assumption, the static probability of the AND gate is

$$P_z^*(1) = P_A(1)P_B(1) \tag{3.3}$$

The internal node error probability can be expressed as the sum of all the six terms in Tab. 3.1 that result in an erroneous output and evaluates to:

$$\begin{aligned} P_\varepsilon(\mathbf{Z}^*) = & P_\varepsilon(\mathbf{A})P_\varepsilon(\mathbf{B})P_{\mathbf{A}}(0)P_{\mathbf{B}}(0)+ \\ & P_\varepsilon(\mathbf{A})(1-P_\varepsilon(\mathbf{B}))P_{\mathbf{A}}(0)P_{\mathbf{B}}(1)+ \\ & (1-P_\varepsilon(\mathbf{A}))P_\varepsilon(\mathbf{B})P_{\mathbf{A}}(1)P_{\mathbf{B}}(0)+ \\ & [P_\varepsilon(\mathbf{A})+P_\varepsilon(\mathbf{B})-P_\varepsilon(\mathbf{A})P_\varepsilon(\mathbf{B})]P_{\mathbf{A}}(1)P_{\mathbf{B}}(1) \end{aligned} \tag{3.4}$$

3.3.4 Intrinsic Gate Error Effects

The effect of the intrinsic gate error is analyzed now. Tab. 3.2 presents the AND gate output behavior in the presence of both input and internal errors. Binary Symmetric Channel (BSC) [100] technique is employed to model the wrong buffer behavior. The gate output static and the error probability can be defined as:

$$
\begin{aligned}
P_z(1) &= P_z^*(1)(1-P_F) + P_Z^*(0)P_F \\
P_z(0) &= P_z^*(0)(1-P_F) + P_Z^*(1)P_F \\
P_\varepsilon(Z) &= P_F + P_\varepsilon(Z^*) - 2 * P_F * P_\varepsilon(Z^*)
\end{aligned} \tag{3.5}
$$

Z*		Gate	Z		Error Probability
Ideal	State	Fault	Value	State	
0	c	c	0	c	
	c	**f**	**1**	$\boldsymbol{\varepsilon}$	**P[Z*=0,Z_c,P_f]**
	$\boldsymbol{\varepsilon}$	**c**	**0**	$\boldsymbol{\varepsilon}$	**P[Z*=0,Z_ε,P_c]**
	ε	f	1	c	
1	c	c	1	c	
	c	**f**	**1**	$\boldsymbol{\varepsilon}$	**P[Z*=1,Z_c,P_f]**
	$\boldsymbol{\varepsilon}$	**c**	**1**	$\boldsymbol{\varepsilon}$	**P[Z*=1,Z_ε,P_c]**
	ε	f	0	c	

Table 3.2 Faulty AND Gate with Unreliable Inputs

3.3.5 Ideal inverter

Following similar lines, the unreliable inverter is modeled as an ideal inverter followed by a faulty buffer as depicted in Fig. 3.5.

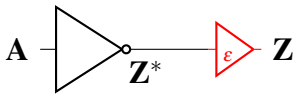
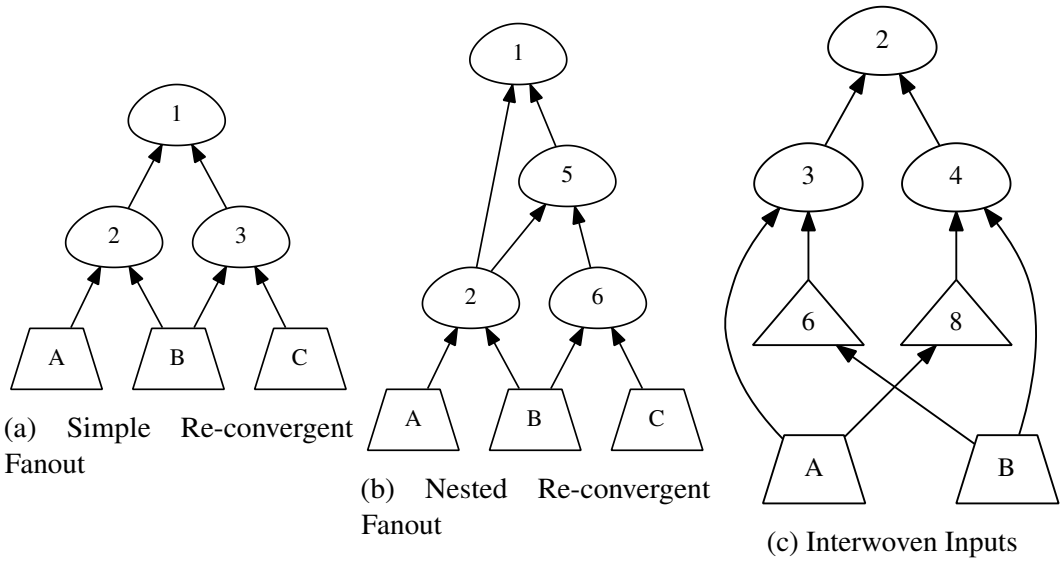


Fig. 3.5 Unreliable Inverter Model

An exhaustive analysis results in Eq. 3.6 to represent the output node error probability of an unreliable inverter.

$$
\begin{aligned}
P_z^*(1) &= 1 - P_A(1) \\
P_\varepsilon(Z^*) &= P_\varepsilon(A) \\
P_z^*(1) &= P_z^*(1)(1-P_F) + P_Z^*(0)P_F \\
P_z(0) &= P_z^*(0)(1-P_F) + P_Z^*(1)P_F \\
P_\varepsilon(Z) &= P_F + P_\varepsilon(Z^*) - 2 * P_F * P_\varepsilon(Z^*)
\end{aligned} \tag{3.6}
$$



(a) Simple Re-convergent Fanout

(b) Nested Re-convergent Fanout

(c) Interwoven Inputs

Fig. 3.6 Re-convergent Fanout Structures

3.3.6 Re-convergent Fanout

The methodology developed in the previous section does not take into consideration the impact of the re-convergent fanout gate error on the ultimate reliability of the output node. The statistical dependence among signals in a combinational circuit is possible due to re-convergent fanout if the primary inputs are independent. When re-convergent fanout gates are present in a circuit, the signals at the inputs of re-convergent gates are correlated ignoring which can result in erroneous results. Computing the node error probability in the presence of re-convergent fanout is complex because Eq. 3.4 does not hold true. This is because each of the terms in Eq. 3.4 cannot be factorized due to dependencies between the four probabilities. There is no closed form solution to solve the terms in Eq. 3.4. Iterative approaches do exist, but their complexity grows exponentially for each of the six terms and their running time is not acceptable for application in synthesis tools.

Fanout source represents the gate output nodes that drive more than one inputs of other gates, and re-convergent fanout nodes are the set of gates which has more than one path to its inputs from any fanout source [101]. A simplest re-convergent fanout is shown in Fig. 3.6(a). As stated in [48], there are two types of re-convergent fanouts which are disjoint re-convergent fanouts as Fig. 3.6(b) and dependent re-convergent fanouts, which can be further divided into nested re-convergent fanout and interwoven branches of inputs. The dependent re-convergent fanouts are depicted in Fig. 3.6(c).



Fig. 3.7 Bounding Error

3.3.7 Bounding Node Error Probability

The methodology of Bound and Propagate is now introduced. It accounts for the re-convergent fanout gate error while not increasing the overall simulation time. The algorithm computes the upper and lower bounds for the error probability on all the gates with the re-convergent fanout. Eq. 3.4 accounts for the six possible scenarios that would result in an output error. Bounding each of these terms singularly results in loose bounds that would quickly converge to the upper/lower bound of 1/0, respectively. To avoid this scenario, only the total gate output error probability is bounded. As depicted in Fig. 3.7, an error on any of the input nodes can be propagated onto the output if the other input node is set to '1'. If the other node is at '0', the error would be masked. Now to obtain the bounds, two cases are presented; a pessimistic and an optimistic scenario. The plot in Fig. 3.8 illustrates the accuracy of the error bounds when applied to MCNC benchmark circuit 'B9'.

Pessimistic Rule: The maximum error bound on the output node of the 'AND' gate is defined as the summation of the error probabilities on each of the input nodes when the other node is set to '1'. Eq. 3.7 represents this. From Tab. 3.1, it is proven that this rule is pessimistic since many cases exist where a single error gets "masked".

$$P_\varepsilon(Z) \leq P_\varepsilon^{Max}(A)P_1(B) + P_\varepsilon^{Max}(B)P_1(A) \triangleq P_\varepsilon^{Max}(Z) \tag{3.7}$$

Optimistic Rule: The minimum error bound on the output node of the 'AND' gate is defined as the maximum of the product of the error on the input nodes when the probability of other node set to '1'. Eq. 3.8 represents this.

$$P_\varepsilon(Z) \geq Max\{P_\varepsilon(A)P_1(B), P_\varepsilon(B)P_1(A)\} \tag{3.8}$$

3.3.8 CPEP based Analysis

This section extends the probabilistic based methodology to deal with the impact of re-convergent fanout gate error on the overall circuit error probability [1]. All re-convergent fanouts originated from the PI's are considered regardless of the type of re-convergent fanout. In other words, when applying the total probability law for a re-convergent fanout point node,

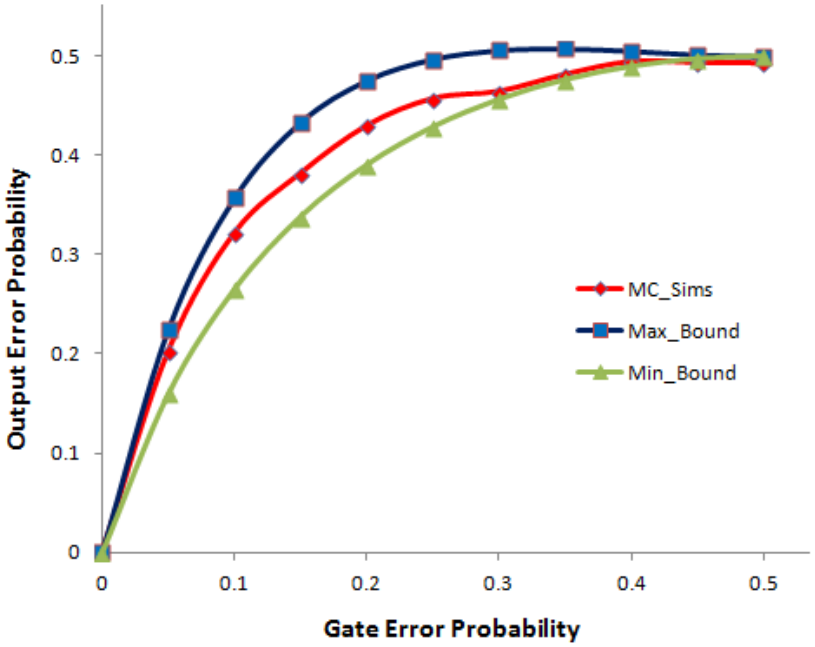


Fig. 3.8 B9 Benchmark Circuit Error Bounds

only the total probability conditioning corresponding to its fanout PIs is considered instead of deriving closed form propagation expressions for every single re-convergent point node.

Specifically, two lists are used to identify whether a node is re-convergent fanout point and the corresponding fanout PIs, they are All PI (API) list and Common PI (CPI) list. For an AND gate node Z, node LX and node RX are denoted as its left-hand side sub-node and right-hand side sub-node respectively, obtaining API and CPI of node Z using Eq. 3.9 and Eq.3.10.

$$API(Z) = API(LX) \cup API(RX) \tag{3.9}$$

$$CPI(Z) = API(LX) \cap API(RX) \tag{3.10}$$

For an inverter node Z, the right hand side sub-node is treated as NULL, and obtain Eq.3.11 and Eq.3.12.

$$API(Z) = API(LX) \cup \text{NULL} \tag{3.11}$$

$$CPI(Z) = API(RX) \cap \text{NULL} \tag{3.12}$$

where "$\cup$" is the union operator, for which particularly $X \cup \text{NULL} = X$ and "$\cap$" is the intersection operator, for which particularly $X \cap \text{NULL} = \text{NULL}$. In addition, the API and CPI of a primary input are defined as itself and NULL respectively. For example, considering

the circuit in Fig. 3.6(a) as example, it can be easily derived that API($Node1$) = $\{A,B,C\}$, CPI($Node1$) = $\{B\}$. For the circuit in Fig.3.6(b), an extra PI "A" have to be taken into conditioning as well due to the fanout introduced by node 2, thus resulting in API($Node1$) = $\{A,B,C\}$, CPI($Node1$) = $\{A,B\}$. The CPI list of particular node represents it's all fanout source primary inputs, thus Eq.3.13, can be used to represent the probability of this node.

$$P(Z) = \sum_{j=0,1} P(X_j)P(Z|X_j) \tag{3.13}$$

When CPI is not NULL, we set each primary input to "1" and "0" respectively to obtain new circuits under each condition, until there are no more re-convergent fanout in the new circuits (called simplest circuits).After calculating the probability for the conditioned circuits, we obtain the conditional probabilities which can be substituted in the Eq.3.13 and obtain the total probability. It is obvious that the efficiency of this conditioning algorithm depends on the conditioning order of primary inputs in the CPI list.

If we condition the common PIs in both LX's CPI and RX's CPI at first, the circuit will be conditioned to the simplest circuit more rapidly. For instance, the logic circuit in Fig.3.6(b) becomes simpler if we condition PI "B" first than conditioning "A" . To achieve this improvement, we introduce another list to order the PIs to be conditioned, say the Order PI (OPI) list.

$$\begin{aligned} OPI(Z) &= CPI(LX) \cap CPI(RX) \\ &\quad + [CPI(Z) - CPI(LX) \cap CPI(RX)] \end{aligned} \tag{3.14}$$

Where "+" is the appending operator, for which $X + \text{NULL} = X$ and "$-$" are defined as the excluding operator, for which particularly we define $X - \text{NULL} = X$ and $\text{NULL} - X = \text{NULL}$. For example, the OPI of node 1 of the logic circuit in Fig.3.6(c) is $\{B,A\}$.

3.3.9 CPEP extension to other Gates

CPEP algorithm has been developed only considering And & Inverter gates as the current research deals with only AIG based circuit representations. But, the CPEP model can be easily extended to any other generic as well. For example, let us consider the 2 input XOR gate.

The static probability as defined in Eq.3.15 is the union probability of SP_A and $SP_{B'}$ plus the union probability of $SP_{A'}$ and SP_B assuming input A and B are independent.

$$SP_{Z^*} = SP_A \times SP'_B + SP'_A \times SP_B \tag{3.15}$$

Employing a table similar to that of Tab. 3.1, the error on the internal node (Z*) due to the errors on the input nodes of an ideal XOR gate is computed into a closed form equation as shown in Eq. 3.16.

$$\begin{aligned} P_{\varepsilon}(\mathbf{Z}^{*}) =& [P_{\varepsilon}(\mathbf{A})(1-P_{\varepsilon}(\mathbf{B}))+(1-P_{\varepsilon}(\mathbf{A}))P_{\varepsilon}(\mathbf{B})][P_{\mathbf{A}}(0)P_{\mathbf{B}}(0)+ \\ & P_{\mathbf{A}}(0)P_{\mathbf{B}}(1)+P_{\mathbf{A}}(1)P_{\mathbf{B}}(0)+P_{\mathbf{A}}(1)P_{\mathbf{B}}(1)] \end{aligned} \tag{3.16}$$

3.3.10 Limitations

Probabilistic based methods are well-studied from power perspective and similar methodologies are being replicated for reliability estimation as well. The advantage of this technique is its quick computation time compared to simulation based methodologies. CPEP methodology developed as part of this research also falls in this category. Two of the major limitations of the proposed CPEP technique are:

- CPEP methodology has been developed with the sole purpose of catering to AIG's. But with other gates with more than two inputs, the mathematics involved becomes more complicated which will result in increased computation time.
- As with most probabilistic based techniques, CPEP too will run into accuracy issues with increased re-convergent fanout nodes.

3.4 CAD Tool: Reliability Evaluator

In this section, the core algorithm and the experimental setup used to demonstrate the accuracy of the proposed approach against Monte-Carlo simulations are presented. Further, the accuracy and simulation time savings compared to Monte-Carlo are reported.

Algorithm 1 Algorithm of Computation Core

Require: netlist file

```
1: generate data structure according to netlist file
2: while first!=NULL do                          ▷ the first element in output_list
3:     UPDATE_LISTS(first)
4:     COMPUTE_EP(first)
5:     first=first→next                          ▷ move to next output
6: end while
```

3.4.1 Computation Algorithm

The algorithm is implemented by introducing deep-left-first binary tree traversal method, due to the recursive definition of binary tree.Recursion is suitable for the complete algorithm design. The algorithm of the error probability computation core is shown in Alg. 1. The input of this algorithm is a netlist file containing the hardware description and parameters (error rate of gates and the static probability of PIs) of the logic circuit to be estimated. The first step is to generate the data structure according to the netlist file and save all output nodes to a list called "output_list". Then for each element in the output_list, the calculation will be performed until all outputs have been computed, therefore a "while" loop is used. The implementation details of the computing procedure are shown in Alg. 2.

Algorithm 2 Compute_EP Methodology

Require: OPI list

```
 1: procedure COMPUTE_EP(Z)                          ▷ Z is the root node
 2:     if Z is PI then
 3:         Read default values of SP, EP, CP of PI
 4:         return
 5:     end if
 6:     if Z is INV then
 7:         COMPUTE_EP(LX)
 8:         Calculate SP, EP, CP with independent model
 9:         return
10:     end if
11:     if OPI is NULL then                             ▷ Independent
12:         COMPUTE_EP(LX)
13:         COMPUTE_EP(RX)
14:         Calculate SP, EP, CP with independent model
15:     else                                            ▷ Reconvergent Fanout
16:         New0=CONDITION(Z, cond_PI, 0)
17:         COMPUTE_EP(new0)
18:         New1=CONDITION(Z, cond_PI, 1)
19:         COMPUTE_EP(new1)
20:         Calculate total probability employing Eq. 3.13
21:     end if
22: end procedure
```

The algorithm conditions the type of the current node, it is quite simple as the case of the current node is a PI, an Inverter or an AND gate without re-convergent fanout. However, the handling of re-convergent fanouts is much complicated. It starts when the OPI is not NULL, after conditioning the first element in OPI list to"0" in line 16, a new circuit "new0" is

obtained (the OPI of new0 has been updated in condition procedure), then call "compute_EP" procedure recursively to compute the conditional probability $P[Z_\varepsilon|X_0]$ as line 17. Similarly, $P[Z_\varepsilon|X_1]$ is computed in line 18, 19. Finally, the total probability is calculated using Eq. 3.13. Fig. 3.9 presents the methodology employed within the tool to compute circuit reliability. The algorithm is implemented by introducing deep-left-first binary tree traversal method, due to the recursive definition of the binary tree. It accounts for the error introduced by both Inverter & AND gates. Using Eq. 3.4 and 3.5, the error due to the AND gate is computed both on the internal and external output nodes. This flow has been integrated into the open source tool 'ABC' [102] which automates the error probability computation. As shown in the flowchart, the handling of re-convergent fanouts starts if the OPI is not NULL in decision (5), after conditioning the first element in OPI list to "0", we obtain a new circuit "new0" (the OPI of new0 has been updated in condition procedure), then call "compute_EP" procedure recursively to compute the conditional probability $P[Z_\varepsilon|X_0]$. Similarly, $P[Z_\varepsilon|X_1]$ is computed in procedure (10), (11). Typically, a tree ought to not have any cycles (except for PI fanouts) such as the circuit in Fig.3.6(a). However, when traversing a "tree" with cycles (a graph) such as the one in Fig.3.6(b), a very straightforward method is used to keep the implementation simple, although it is not runtime optimal method at this moment. To be more specific, we treat a cycle is formed by one particular node X and its multiple parent nodes $Y_0, \ldots, Y_n$, therefore, each time we visit node X from different parent nodes, we traverse the sub-tree whose "root" node is X once again. For example, the visiting order of the latter circuit is $1 \to 2 \to A \to B \to 5 \to 2 \to A \to B \to 6 \to B \to C$.

3.4.2 Simulation Results

Fig. 3.9 depicts the complete experimental setup developed to compare the algorithm results with fault inserted gate level simulations. The sample size used for reliability analysis is 10^5, 5×10^4, and 10^4 for 0.001, 0.01 and 0.05 error scenario's respectively. All the primary inputs are assumed to be independent and set the switching activity on all the pins to 0.5. The switching activity numbers are flexible and can be set to any other value. In Tab. 3.3, columns 1 and 2 give the name and number of gates in the benchmark circuit. Column 3 captures the accuracy of the method when compared with Monte-Carlo Simulations while Column 4 highlights the significant time savings the proposed algorithms achieved when compared with Monte-Carlo simulations. From the table, the proposed algorithm maintains accuracy within 10% while significantly speeding up on the computation time.

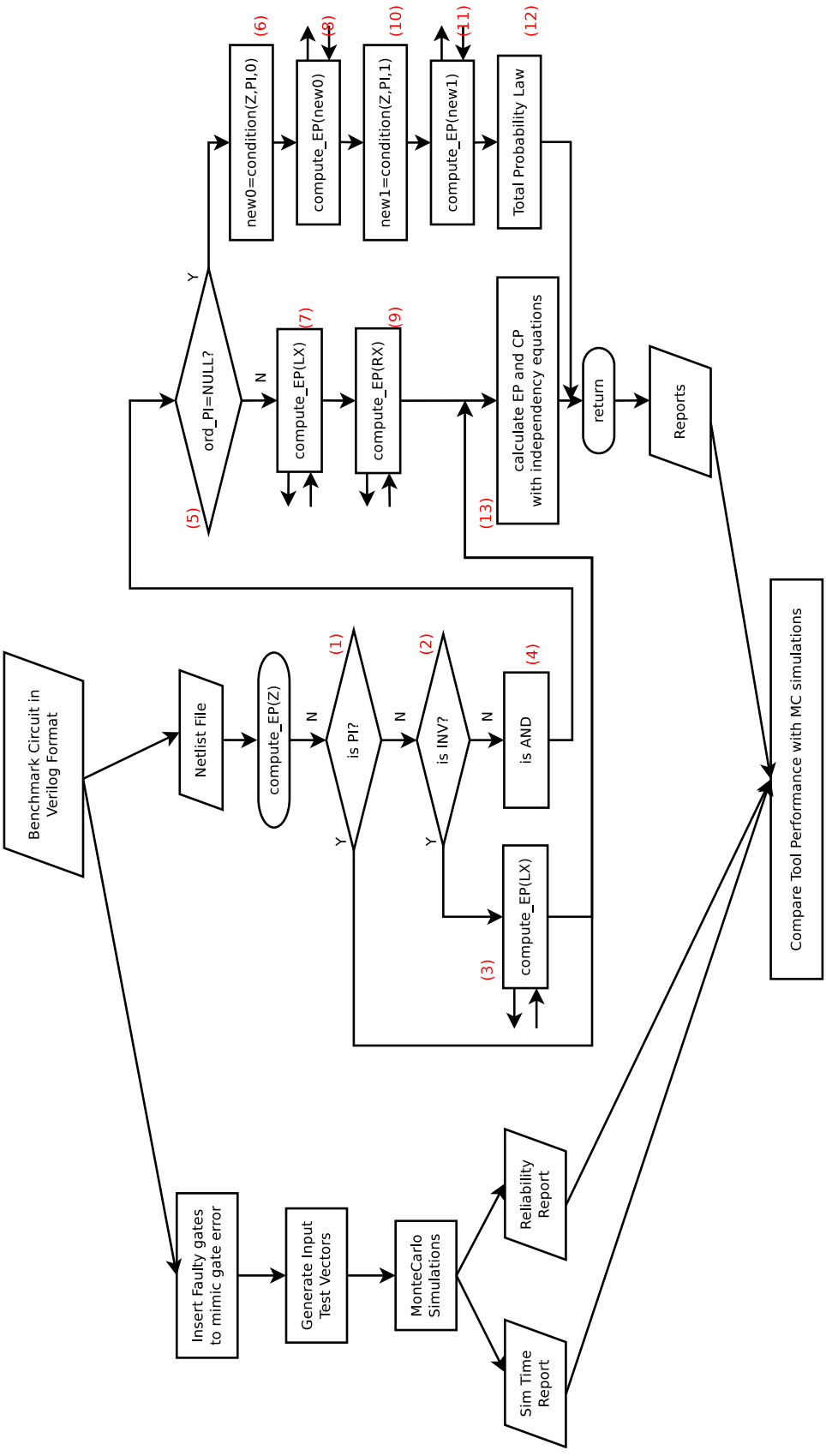


Fig. 3.9 Design Flow for Reliability Computation

Table 3.3 MCNC Benchmark Circuits Based Accuracy and Performance Evaluation for different gate errors (ε)

Benchmark	PI Count	Gate Count		Avg Error Deviation on all outputs %			Runtime{s}	
		AND	Inverter	ε = 0.001	ε = 0.01	ε = 0.05	GLS	CPEP
apex6	135	659	544	0.384	1.079	0.243	2889.96	**0.012013**
b9	41	101	77	0.247	0.696	0.138	520	**2.515699**
cht	47	187	189	0.237	0.702	0.308	789.97	**0.003880**
cm138a	6	16	13	0.130	0.389	0.129	120.03	**0.000154**
cm42a	4	18	14	0.133	0.267	0.518	140.07	**0.000101**
cm82a	5	20	25	0.514	1.132	2.549	129.98	**0.015272**
cm85a	11	36	30	0.412	1.200	0.908	209.94	**12.991886**
cmb	16	47	29	0.153	0.710	0.853	220.08	**0.000597**
count	35	127	130	0.282	0.416	1.004	530.07	**0.015586**
cu	14	55	29	0.120	0.290	0.150	260.08	**0.096305**
decod	5	30	4	0.080	0.250	0.090	159.94	**0.000138**
des	256	4092	2650	0.738	2.148	1.703	16070.07	**3.038316**
frg2	143	1120	611	0.347	0.998	1.104	4889.96	**0.039363**
pair	173	1474	1198	0.385	0.743	1.065	6449.99	**0.387360**
x3	135	811	512	0.258	0.666	0.306	3739.96	**0.009839**

3.5 Conclusions

Reliability evaluation is the most fundamental and crucial step towards developing a reliability-aware logic synthesizer and also fault tolerant circuit design. The increasing demand for reliability analysis calls for automated tools that could analyze the circuit reliability in quick time without compromising on the accuracy. A new gate error model has been described that enables the algorithm to study the impact of the logical error on individual gates on the overall circuit. A complete mathematical analysis was presented in a close formed equation that describes the gate error probability. The proposed gate error model is employed to propagate the gate error probability on generic circuits.

The problem of computing reliability information for combinational circuits has been addressed by investigating two different methodologies. Both the methods described are accurate, scalable, and efficient techniques for reliability analysis of logic circuits. The first approach employs simulation techniques by injecting faults with specific characteristics to gather statistical information on how these errors propagate through the circuit. The main novelty lies in the fact that Mersenne twister is used over the traditional ŔANDfunction to generate random vectors. The second method called Conditional Probabilistic Error Propagation(CPEP) builds a probabilistic error model of the combinatorial circuit and calculates the reliability of the circuit based on a mathematical analysis of the model. This is quite generic and can be applied to any error scenario and for any logic gate.

Further, two different approaches to study the impact of re-convergent fanout on reliability estimation was discussed. First, the bounding approach where the idea was to estimate the minimum and maximum possible error probability instead of an exact number. While this is a fast technique, further investigation in greater detail is required to develop tighter bounds and eventually integrate it into the synthesis tool. Second, CPEP based algorithm has been extended to address re-convergence issues wherein separate data structures are created to maintain the re-convergent fan-outs. An analysis in line with the dynamic weighted average algorithm is employed to estimate the reliability considering these re-convergent fanout nodes. Experimental results obtained with the proposed CPEP framework is within 2% average error and up to 1000 times faster when compared to Monte Carlo simulations. This error is within acceptable limits as the focus of this approach is to quickly compare hundreds of logically equivalent realizations and select higher reliability circuit configuration.

In this chapter, probabilistic and simulation based error model of the combinational circuit to study the impact of logical error on individual gates on the overall circuit is presented. In the next chapter, reliability driven 4-cut enumeration and Boolean matching technique that improves circuit reliability will be presented.

Chapter 4

Reliability Aware Logic Synthesis

4.1 Introduction

Logic synthesis is the process of translating a higher level description of a circuit into gate level netlist [103], which generally involves three steps: a) compiling the high-level representation into an intermediate representation b) optimizing the intermediate representation c) technology mapping. Traditional logic synthesis methodologies and tools were centered on fulfilling timing, power, and area constraints or on achieving acceptable trade-offs among those [6] [7]. However, as the CMOS technology entered the nanometer era, these specifications alone can no longer cover all the relevant aspects and reliability comes into consideration.

Logic synthesis traditionally is classified into two broad categories, local rule-based transformations (or rewriting) and technology independent/dependent algorithms [32]. Rewriting is based on employing a set of local transformation techniques on a small sub-section of the graph to improve area, power or timing. Algorithmic-based approaches work on the observation that there exists a certain set of operations, which are inherently good irrespective of technology. This work explores the possibility of reducing output error probability by employing local transformation techniques [104] [105]. Though reliability driven logic optimization is in its infancy when compared to power and delay driven optimization, the method presented here is still based on the popular and successful concept of local transformations [106] [107]. Rewriting of 4-input cuts by employing Negation-Permutation-Negation (NPN) equivalent logic configurations are employed to improve circuit reliability. Key differences in the cut enumeration and Boolean matching techniques when constraint optimization is circuit reliability, instead of area, are discussed in detail.

4.1.1 Main Contributions and Outline

This work introduces a new reliability driven circuit optimization and synthesis flow based on the standard rewriting technique to improve the reliability of circuits built out of unreliable gates [108] [109] [2]. ABC [28], a logic synthesis and verification tool which performs scalable logic optimization based has been used to accommodate all the algorithms. Two different methodologies are proposed both based on the rewriting technique. The first is rule-based re-synthesis and the second one has adopted from cut based rewriting. Rule-based re-synthesis approaches rely on searching the graph for specific substructures and replacing these with alternatives. The search and replacement of substructures are hand derived and hard-coded. A selected subset of NPN-equivalent (Negation-Permutation-Negation equivalent) local transformation rules for logic optimisation is employed to improve circuit reliability. The biggest advantage is that there is no extra area overhead as node count does not increase. These set of rules along with an algorithm to compute the impact of gate errors on the circuit output(s) are integrated unto a reliability-aware logic synthesis tool that applies the transformation rules in a guided fashion on complex combinational circuits. Evaluation of the tool on a set of MCNC benchmark circuits and results show a reliability improvement up to 7.5%.

The second approach is based on cut rewriting. A cut is a self-contained subgraph of the main graph having a set number of inputs and a single output. In cut based rewriting, the cuts that are rooted at a particular node are identified, and alternative cuts that compute the same function are evaluated as replacements for that node. A precomputed forest of possible alternatives is provided from which possible options are selected. This methodology is superior concerning its performance compared to the original methods proposed. To improve circuit reliability, other redundant nodes can help in masking gate errors thereby reducing output error. Throughout this work, the exact nature of the unreliability is ignored, and it is assumed that each gate has a constant, independent probability of flipping its output from the correct value to the opposite. For this work, a probability of error of $1e^{-4}$ is chosen.

Evaluation of the tool on a set of standard MCNC benchmark circuits and the conventional arithmetic circuits is performed. Results show an average reduction in output error of up to 14% while a peak improvement of up to 75.5% is observed. Performance analysis using Synopsys Design Compiler shows that very low area overhead of 6.57% that results in 13.52% higher power consumption is the extra cost incurred.

4.2 ABC : Open Source EDA tool

Several academic open-source tools are available that provide a programming environment and a solid platform for research in logic synthesis, technology mapping, power and delay estimation and optimization. These academic tools represent the Boolean functionality of any digital circuit using a data structure. Manipulation for logic synthesis, optimization, and technology mapping is done on these data structures. In this section, first the data structure AIG is discussed, and the details of the ABC tool are described.

4.2.1 AND Invert Graphs

And Invert Graphs (AIGs) are common circuit representation where each graph node corresponds to a two-input AND gate and the edges, representing the node interconnections, optionally having an inverter. An example circuit in both schematic and AIG form is presented in Fig. 4.1(a) and Fig. 4.1(b) respectively. Circles represent AND nodes, solid lines represent direct gate connections and dashed lines represent connections through an inverter. The following properties of AIGs facilitate the development of robust applications in synthesis, mapping, and formal verification:

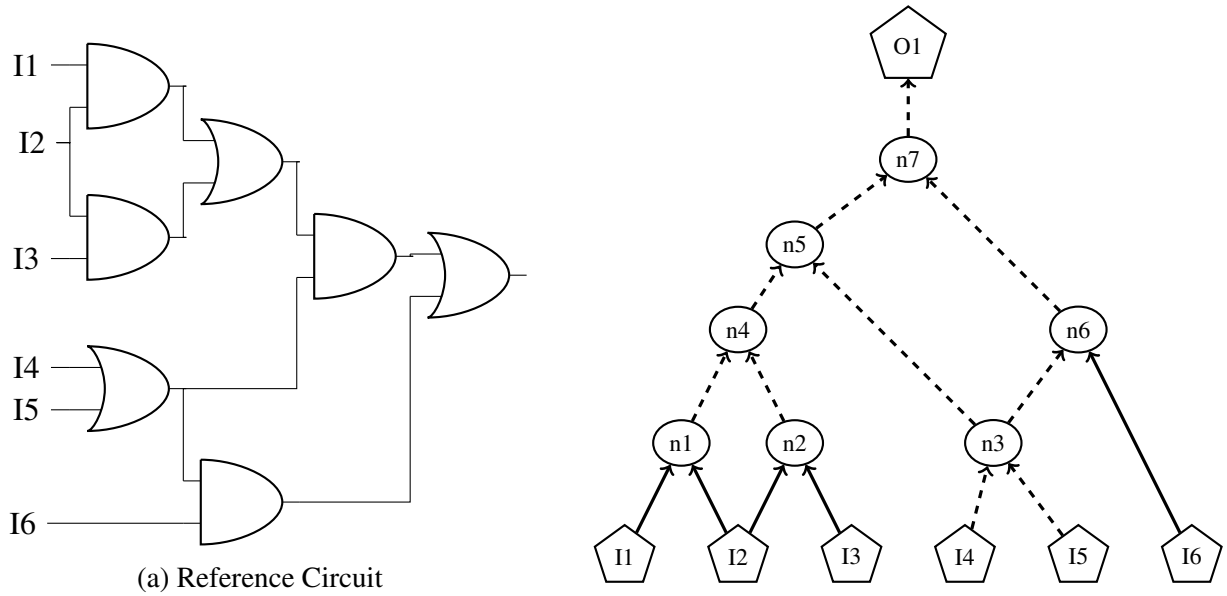
- AIGs unify the synthesis/mapping/verification by representing logic compactly and uniformly. During technology mapping, the AIG is used as a subject graph annotated with cuts that are matched with LUTs or gates. At any time, verification can be performed by constructing a miter of the two synthesis snapshots represented as one AIG, handled by a complex AIG-based verification flow.

- Although AIG transformations are local, they are performed with a global view afforded by the structural hashing table. Because these computations are memory/runtime efficient, they can be iterated leading to superior results unmatched by a single application of a more global transform.

- An AIG can be efficiently duplicated, stored, and passed between calling applications as a memory buffer or compactly stored on disk in the AIGER format [99].

Next, some of the standard terminology used throughout the rest of the chapter is defined.

– **Cut**: A cut of a node N in a network is a set C of nodes such that any path from Primary Inputs (PI) to N passes through one of the nodes from the set [110]. Node N itself forms a trivial cut. The size of a cut refers to the number of leaves, rather than to the number of interior nodes. Cuts of up to a specific size k are called k-cuts or k- feasible cuts. In Fig. 4.2, the node set {I6, n1, n2, n3} defines a 4-cut C on the node n7. C is a 4-cut since it has four



(a) Reference Circuit

(b) AND Invert representation

Fig. 4.1 And Inverter representation of Combinational Circuit

leaves: n1, n2, n3, and I6. Every path from a PI to n7 passes through at least one of them. Nodes n4, n5, and n6 are the internal nodes of the cut.

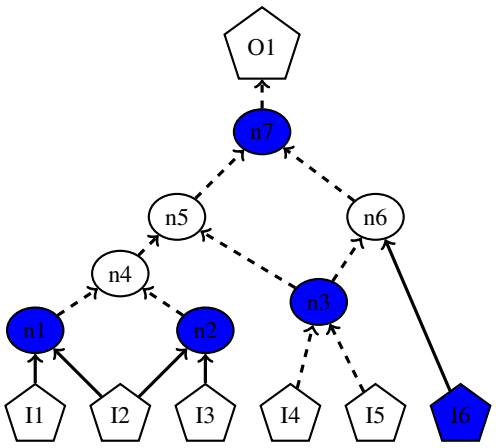


Fig. 4.2 Cut Example

– **NPN equivalence**: Two Boolean functions, F and G, are NPN-equivalent, i.e., belong to the same NPN equivalence class, if F can be transformed into G through the negation of inputs (N), permutation of inputs (P), and negation of the output (N) [111]. For example, the subcircuit comprising nodes {n1, n2, n4} in Fig. 4.1(b) and Fig. 4.2, evaluating the function G and F, are NPN-equivalent. G can be transformed to F by employing the following

procedure: (1) negate I3 and I1, (2) swap the position of I3 and I2, and (3) negate the output node.

– **Boolean matching**: It is a technique widely used in technology mapping [112]. It is the process of replacing a subgraph with another functionally equivalent sub-graph. It is typically done by calculating the canonical form representation of functions by employing simple methods like negation and swapping of the nodes.

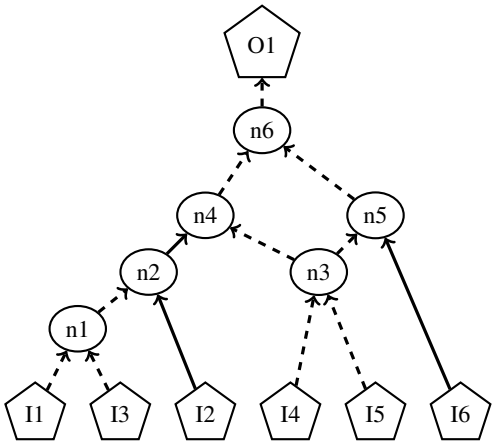


Fig. 4.3 Boolean Equivalent

4.2.2 ABC Tool

ABC is a growing software system for synthesis and verification of binary sequential logic circuits appearing in synchronous hardware designs developed at University of California Berkeley. In this research, the algorithms developed are build as a wrapper on top of the existing ABC tool. ABC combines scalable logic optimization based on And-Inverter Graphs (AIGs), optimal-delay DAG-based technology mapping for look-up tables and standard cells, and innovative algorithms for sequential synthesis and verification. ABC provides an experimental implementation of these algorithms and a programming environment for building similar applications. The goal of the ABC project is to provide a public-domain implementation of the state-of-the-art combinational and sequential synthesis algorithms and, at the same time, create an open-source environment, in which such applications can be developed and compared. The current version of ABC can optimise/map/retime industrial gate-level designs with 10^5 gates and 10^4 sequential elements for an optimal delay and heuristically minimized area in about one minute of CPU time on a modern computer. The runtime of the combinational synthesis, mapping, and verification is typically faster. Future development will focus on improving the algorithms and making most of the packages

stand-alone. This will allow the user to customize ABC for their needs as if it were a tool-box rather than a complete tool. Most of the research work is based on AND-Inverter graphs, manipulated in ABC.

ABC is derived from a tool called SIS [26]. Regarding logic representation, the main difference between SIS and ABC is that SIS works on a logic network whose nodes are represented using SOPs, while ABC works on an AIG whose nodes are two-input AND gates. SIS works on one copy of a logic network, defined by the current boundaries of its logic nodes, while ABC works on an AIG. A cut computed for an AND node in the AIG can be seen as a logic node. Since there are many cuts per logic node, the AIG can be seen as an implicit representation of many logic networks. When AIG rewriting is performed in ABC, a minimal description is found among all decompositions of all structural cuts in the AIG, while global logic sharing is captured using a structural hashing table. Thus, ABC is more likely to find a smaller representation concerning AIG nodes than SIS.

4.3 Rule based Rewriting

A set of local transformation rules for logic optimization from a reliability perspective. The proposed transformation rules (i) maintain the logical equivalence of the new circuit with the original one and (ii) provide a set of standard rules that when applied in a guided fashion would result in improved circuit reliability. The impact of the gate error probability on equivalent logic configurations to determine the best realization is then studied. The transformation rules are built upon the application of Boolean algebra logical equivalence laws such as swapping and reduction of variables. Evaluation of the logic transformation rule set on a test circuit shows a reliability improvement in the order of 10%. Given a combinatorial circuit implementing the Boolean function f lets call S_I the first AIG network before rule application and S_F the final AIG network after rule application. The reliability-aware synthesis aims to find a sequence of transformations leading to an AIG network $S_F = S_{opt}$ that minimise the cost function Err(f). It is evident that logic synthesis based on local transformation is an NP-hard problem [113]. A heuristic approach is employed to find an acceptable solution, i.e., an AIG providing reliability higher than a certain threshold. The strategy chosen is to apply local optimization based on a transformation rule set. The challenge is in deciding whether to transform a sub-graph to the new state S0 based on the impact of the transformation on the PO reliability.

A fast and accurate algorithm is presented in the previous section to compute the impact of input and gate error probabilities on the reliability of the circuit output. In this section, logically equivalent configurations of a given circuit are compared concerning error probabil-

ities operating under similar conditions. Though reliability driven logic optimisation is in its infancy when compared to power and delay driven optimization, the method presented here is still based on the popular and successful concept of local transformations [106], [107]. A set of AIG based local transformation rules that can be applied on any combinational circuit are defined. The impact of the gate error probability on equivalent logic configurations is studied to determine the best realization. The transformation rules are built upon the application of Boolean algebra logical equivalence laws such as swapping and reduction of variables. In this way, each transformation rule alters the structure of the AIG network but maintains the logical functionality intact.

Given a combinatorial circuit implementing the Boolean function f, S_I the first AIG network before rule application and S_F the final AIG network after rule application. The reliability-aware synthesis aims to find a sequence of transformations leading to an AIG network S_F=S_{opt} that minimise the cost function Err(f). It is well known that logic synthesis based on local transformation is an NP-hard problem. Thus, the algorithm relies on a heuristic approach to find an acceptable solution, i.e., an AIG providing a sign of reliability higher than a certain threshold. The strategy chosen is to apply local optimisation based on the transformation ruleset. The constraint is in deciding whether to transform a sub-graph to the new state S_0 based on the impact of the transformation on the PO reliability.

4.3.1 Local Transformation Rules

In this section, the set of transformation rules utilized in the quest for the reliability optimised implementation of Boolean functions are presented.

–Rule1: The first transformation as shown in Fig. 4.4 is modeled based on the law of distributivity. Consider re-convergent fanout node n3 with two fan-in nodes n1 and n2. If both n1 and n2 have no other fanout except n3, then I1 and I3 can be swapped with determinate negations. The new configuration shall have higher reliability as well as node count decreases by one.

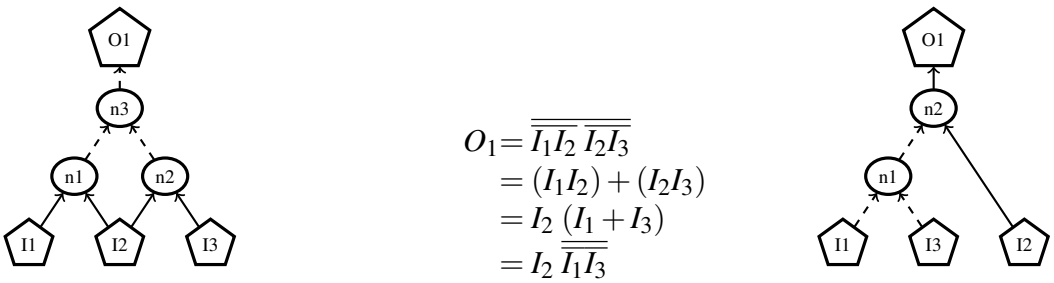


Fig. 4.4 Logic Transformation Rule1

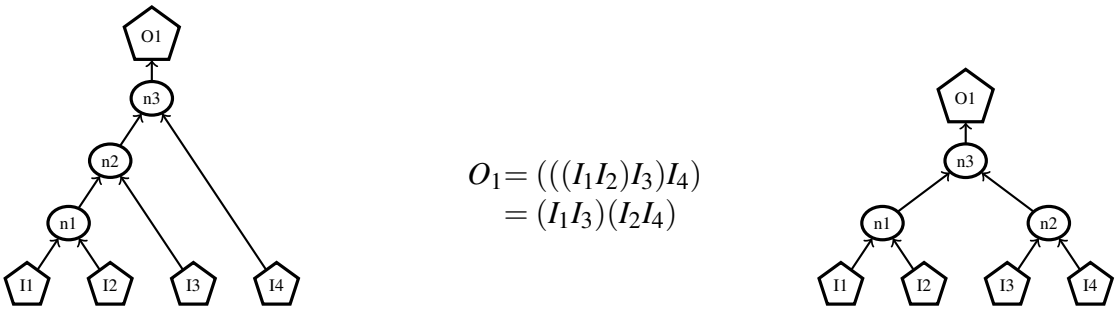


Fig. 4.5 Logic Transformation Rule2

–Rule2: This transformation as shown in Fig. 4.5 is based on the law of associativity. The underlying assumption is that reducing the length of the longest path will improve the reliability of the circuit.

–Rule3: This rule as shown in Fig. 4.6 is also based on the law of associativity. It suggests equally distributed inverters on both the legs of the graph will improve the reliability of the circuit. This rule specifically targets the configuration for a two-input XOR gate. Its application can be seen predominantly in circuits like priority encoders, CORDIC processors, etc.

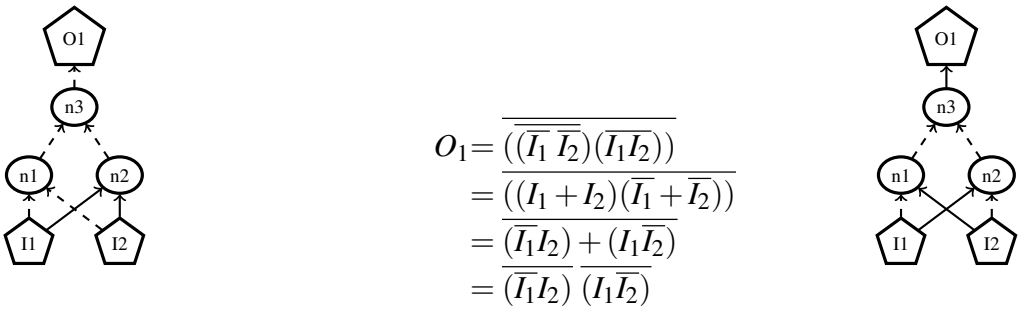


Fig. 4.6 Logic Transformation Rule3

–Rule4: This rule is derived based on principles of associativity and insertion and defines the best representation for 3-bit majority voter. Fig. 4.7 depicts the three possible configurations of the majority voter. This rule in principle applies rule1 to reduce node count and then applies rule3 to place the inverters equally on both the legs of the output. (Eq. 4.1) proves the logical equivalence of all the three configurations.

$$
\begin{aligned}
O_1 &= \overline{\overline{(I_1I_2)}\ \overline{(I_2I_3)}\ \overline{(I_3I_1)}} \\
&= I_1I_2 + I_2I_3 + I_3I_1 \\
&= I_3I_1 + I_1I_2 + I_2I_3 + I_2I_3 \\
&= I_3I_1 + I_2I_3I_3 + I_1I_2 + I_2I_2I_3 \\
&= I_3(I_1 + I_2I_3) + I_2(I_1 + I_2I_3) \\
&= (I_3 + I_2)(I_1 + I_2I_3) \\
&= (\overline{\overline{I_3}\ \overline{I_2}})(\overline{\overline{I_1}(\overline{I_2I_3})})
\end{aligned}
\tag{4.1}
$$

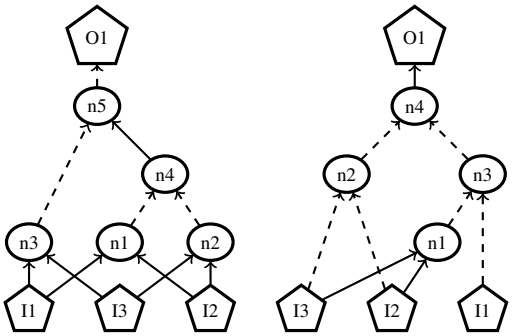
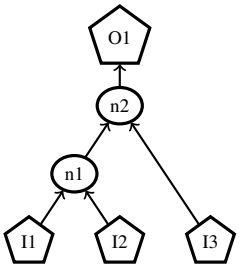


Fig. 4.7 Logic Transformation Rule4

–Rule5: The fifth transformation rule is based on the commutative law. The rule states that the signals with the lowest static probability of '1' in an AIG tree should be closer to the output, or closer to the root node of a subgraph. Intuitively, this rule is maximizing the masking effect of the AND gate to minimize the impact of any error coming from the left side of the graph. The reliability improvement is strongly dependent on the static probability of the input pins.



$$
\begin{aligned}
O_1 &= ((I_1I_2)I_3) \\
&= ((I_1I_3)I_2)
\end{aligned}
$$

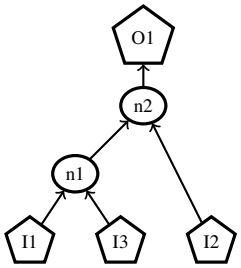


Fig. 4.8 Logic Transformation Rule5

4.3.2 Exhaustive Analysis of Rules

Each of the rewriting rules presented earlier is a transformation between two logical equivalent circuits. A mathematical equation that describes the primary output reliability as a function of the input error probabilities input static probabilities and the gate error probability can be associated with each of this logical equivalent circuit by recursively applying Eq. 3.4 and Eq. 3.5 for each node of the circuits. The problem now turns into choosing one among the original and the modified circuit.

$$\begin{aligned} R_{Org} &= f_1\{PE_I, SP_I, GE\} \\ R_{Mod} &= f_2\{PE_I, SP_I, GE\} \end{aligned} \tag{4.2}$$

where,

- R_{Org} - Reliability of the original circuit
- R_{Mod} - Reliability of the modified circuit
- PE_I - Error Probability of the Primary Inputs
- SP_I - Static Probability of the Primary Inputs
- GE - Individual gate error

Unfortunately, even for a small circuit, the direct mathematical analysis is not feasible to give a complete mathematical characterization to determine the reliability performance of the circuit configurations from these equations as the number of variables in the equations used to compute the output error probability is too high. Consider a three input graph. Eq. 4.2 has seven variables: three input error probabilities, three static input probabilities, and the gate error probability. As an example of the complexity, apply Eq. 4.2 to Rule4. It results in an equation of the output error probability that is a polynomial in seven variable of degree 20. Hence, it is clear that direct mathematical analysis is not feasible to formally compare the reliability performance of the circuit configurations from these equations. As an approximate solution, a framework has been developed in Matlab to investigate the behavior of the equivalent circuits. The most commonly used signal patterns Gaussian, Inverse Gaussian, monotonically increasing and monotonically decreasing distributions across the input nodes for error probability and static probability are applied and then evaluate Eq. 4.2 varying the gate error probability. These four are randomly picked patterns as they are the most commonly seen patterns. An exhaustive analysis of these rules has been performed for different input static probability values and gate errors.

Scenario		Pattern	
Error Probability	Static Probability	Error Probability	Static Probability
Gaussian	Constant	0.01..0.05..0.01	0.5
Rev_Gaussian	Constant	0.05..0.01..0.05	0.5
Monotonically Increasing	Constant	0.01..0.03..0.05	0.5
Monotonically Decreasing	Constant	0.05..0.03..0.01	0.5
Constant	Gaussian	0.5	0.01..0.05..0.01
Constant	Rev_Gaussian	0.5	0.05..0.01..0.05
Constant	Monotonically Increasing	0.5	0.01..0.03..0.05
Constant	Monotonically Decreasing	0.5	0.05..0.03..0.01

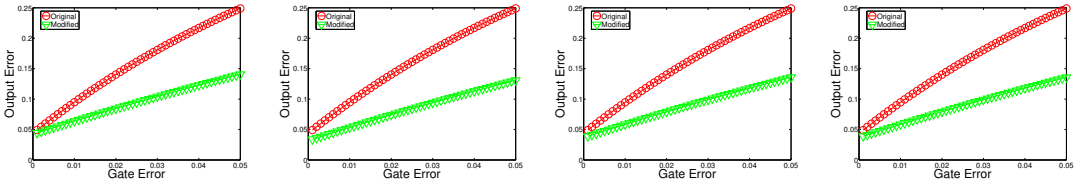
Table 4.1 Scenarios used for rules analysis

These four patterns are then applied to the input pins error probabilities as well as static probability independently assuming the other parameter to be constant. In Tab. 4.1, the eight different scenarios analyzed are listed. The set of simulation results covering all the patterns listed in Tab. 4.1 are plotted in Fig. 4.9 - Fig. 4.12. From the Rule1 plots, it is clear that considerable reliability improvement is achieved by reducing the node count. While valid for this specific rule, the statement cannot be generalized for any circuit. During the analysis, it has been observed that insertion of extra nodes can result in reliability improvement.

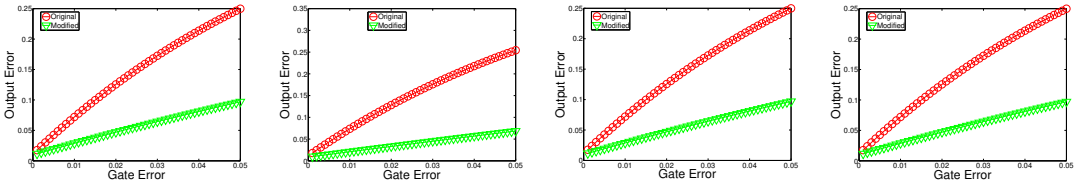
Rule3 is pre-dominantly applicable on most of the primary communication blocks. It can have a higher impact factor as xor is the most commonly used configurations in circuits like cordic, parity encoder, etc. Rule4 scales up the reliability numbers by almost 10% in all the cases. Reducing the number of re-convergent nodes helps in computing the reliability and static probability numbers in much more accurate fashion. The simulation results presented show how for Rule1 to Rule4 the proposed transformation is beneficial for all input scenario. While performing a different set of simulations, it is observed that insertion of extra nodes can result in reliability improvement. The question of how to insert such additional nodes systematically is explained in the cut based synthesis methodology later on.

4.3.3 The CAD algorithm

To validate the proposed approach and rules, a local optimization search algorithm is presented to quantify the reliability improvement. The aim is to find a sequence of transformations leading to an AIG network that minimise the cost function. The algorithm relies on a

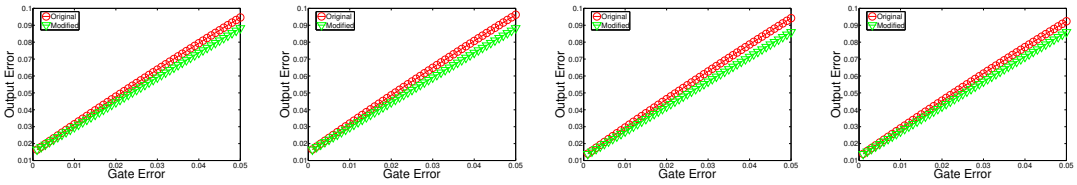


(a) PE_Gaussian_SP_Static (b) PE_Rev_Gaussian_SP_Static (c) PE_Rising_SP_Static (d) PE_Falling_SP_Static

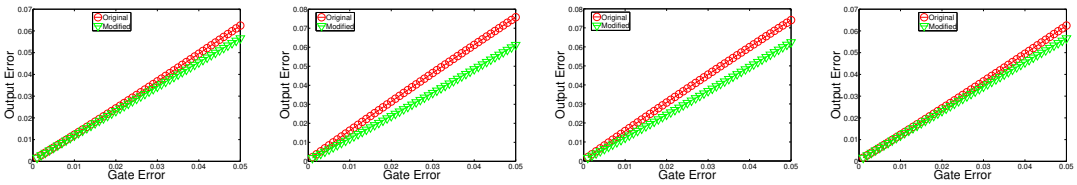


(e) PE_Static_SP_Gaussian (f) PE_Static_SP_Rev_Gaussian (g) PE_Static_SP_Rising (h) PE_Static_SP_Falling

Fig. 4.9 All simulation results for Rule1

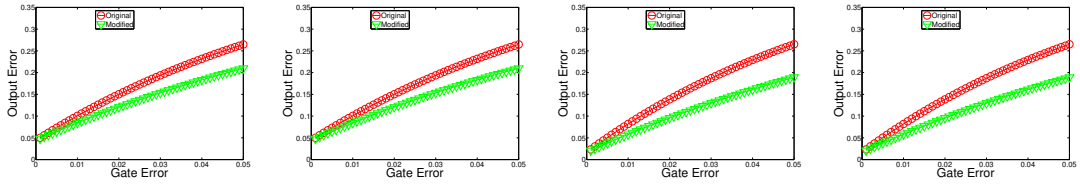


(a) PE_Gaussian_SP_Static (b) PE_Rev_Gaussian_SP_Static (c) PE_Rising_SP_Static (d) PE_Falling_SP_Static



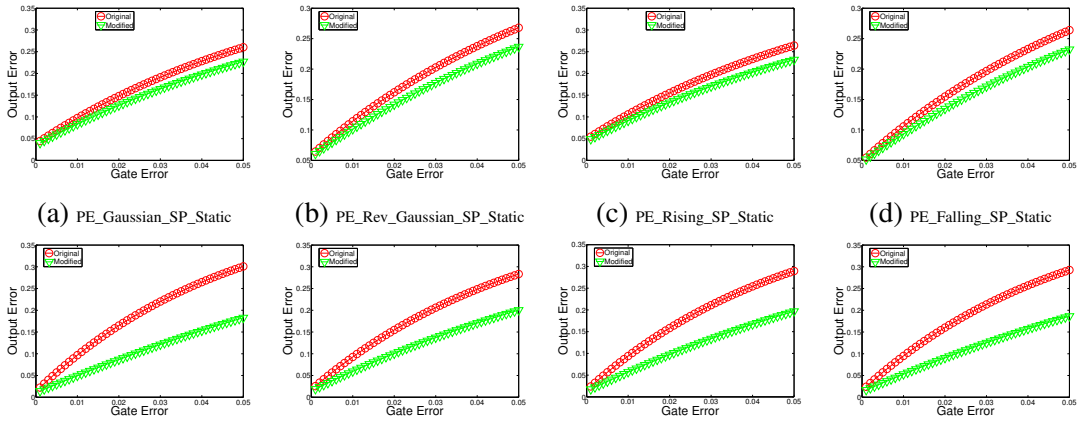
(e) PE_Static_SP_Gaussian (f) PE_Static_SP_Rev_Gaussian (g) PE_Static_SP_Rising (h) PE_Static_SP_Falling

Fig. 4.10 All simulation results for Rule2



(a) PE_Rising_SP_Static (b) PE_Falling_SP_Static (c) PE_Static_SP_Rising (d) PE_Static_SP_Falling

Fig. 4.11 All simulation results for Rule3



(a) PE_Gaussian_SP_Static (b) PE_Rev_Gaussian_SP_Static (c) PE_Rising_SP_Static (d) PE_Falling_SP_Static

(e) PE_Static_SP_Gaussian (f) PE_Static_SP_Rev_Gaussian (g) PE_Static_SP_Rising (h) PE_Static_SP_Falling

Fig. 4.12 All simulation results for Rule4

Algorithm 3 Local transformation based reliability aware optimization

Require: N, total number of nodes in the AIG network
Require: R_N, total number of transformation rules

```
for i = 1 to N do do
    for j = 1 to RN do do
        if Rule R_j is applicable on Node_i then
            Implement the transformation and calculate new reliability W_ij
        end if
        Switch Back to normal configuration
    end for
    select R_J s.t. W_J = min(W_ij)
    Implement R_J
end for
```

heuristic approach to find an acceptable solution, i.e., an AIG providing reliability higher than the original circuit. The strategy chosen is to apply local optimization based on the transformation ruleset. Given a combinational circuit implementing the Boolean function f and its AIG network, the graph is wholly traversed to see if any of the rules are applicable on the given node. For every possible transformation, the new reliability of the circuit is

computed. The configuration that yields the highest improvement in circuit reliability is chosen, and the new topology is generated. This process continues on every node on the graph until the primary outputs are reached where no more transformations are applicable.

Fig. 4.13 describes the complete flow of the tool. The output error probability of two netlists is computed; (i)the default circuit (the original MCNC netlist) without any optimization and (ii)the circuit optimised by the best 'ABC' synthesis algorithm. Since the synthesis algorithm on the ABC tool focuses on area and delay optimisation, it may deteriorate the reliability of the circuit. After selecting the better circuit configuration between the two, the internal tool developed is employed to improve the circuit reliability further. The synthesis tool traverses through every single node in the circuit performing Boolean matching to see if any matching rewriting rules can be applied. If there is more than one applicable rule, the one which provides the highest reliability improvement is selected. This process continues until no more rules can be applied to this node that can improve the circuit reliability. A similar set of operations are performed on all the nodes in the circuit. This methodology will continue to benefit with further expansion of the local transformation ruleset.

The rules presented have been tested in various conditions, and the simulation results show reliability improvement in every scenario. However, due to the large variable count, it is impossible to derive a rigorous proof and to test for all input scenario. Hence, one cannot generalize the improvement in all circumstances. As a result, a local optimization search algorithm is used to confirm the reliability improvement before its application on the circuit. Alg. 3 details the process adopted for performing local transformations. Starting from the initial circuit configuration, the graph is traversed throughout to see if any of the rules are applicable on the given node. For every possible transformation, the new reliability of the circuit is computed. The configuration that yields the highest improvement in circuit reliability is fixed, and the new topology is generated. This process is continued on every node on the graph until the primary outputs are reached where no more transformations are applicable.

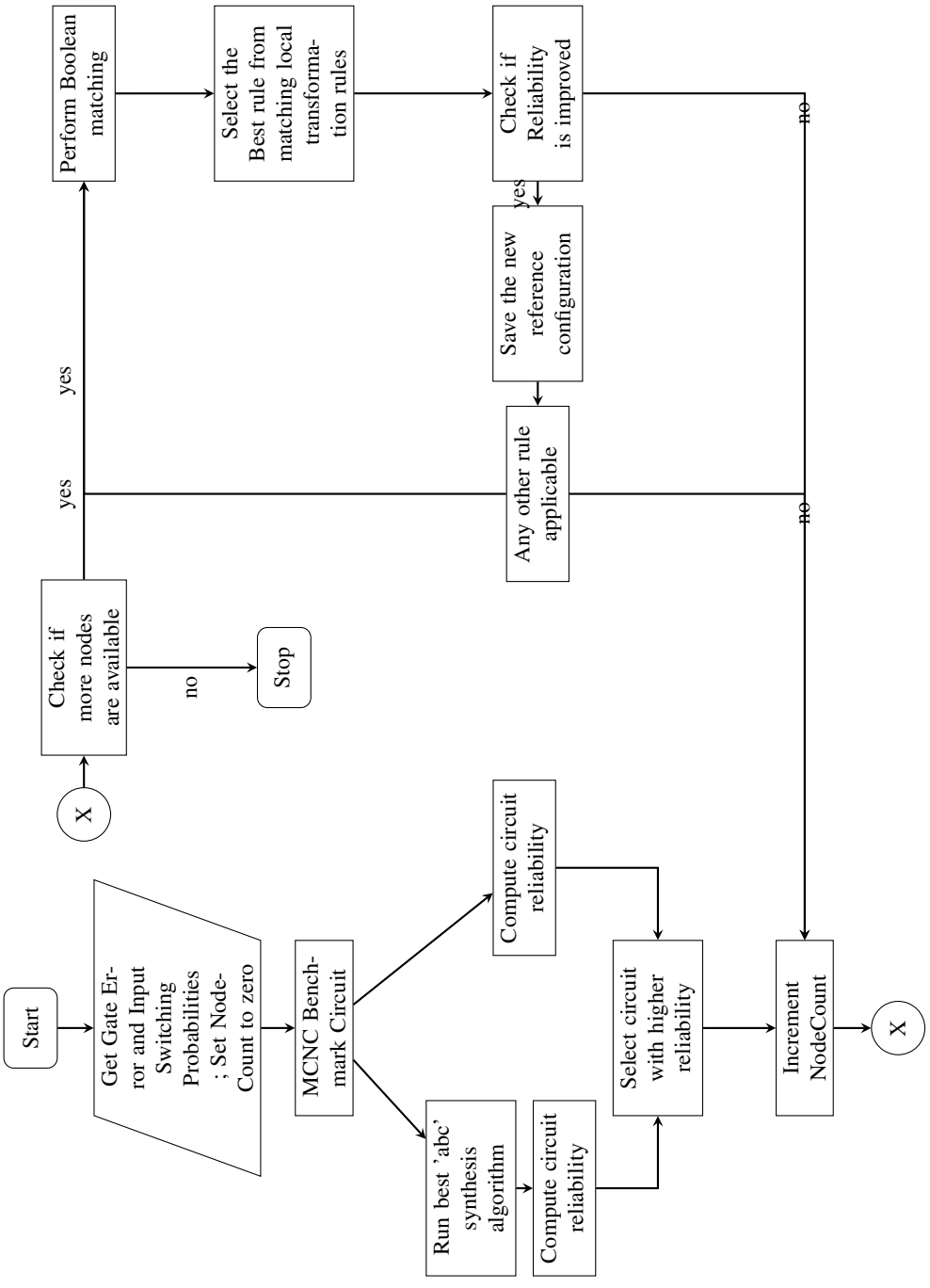


Fig. 4.13 Reliability Aware Logic Synthesis Flow

4.4 Experimental Results

In this section, the potential practical implications of the proposed reliability-aware synthesis tool using local transformations are evaluated. In this view, Alg. 3 was implemented and integrated into ABC synthesis tool as a command $'rwrel'$. The total savings achieved concerning circuit reliability is studied by performing a set of experiments using MCNC benchmark circuits [114] with more than 3000 node AIGs after structural hashing. A unit gate error probability of 0.01 is assumed in all the simulations. The networks were loaded in Berkeley Logic Interchange Format (BLIF) format and strashed to AIGs. Functional equivalence was extensively verified during development using the ABC equivalence check command $'cec'$. In this section, a case study highlighting the important differences in the way the Boolean matcher finds alternative matching local transformation rule when reliability is the constraint is first presented. Then, the results obtained for various MCNC benchmark circuits and 8051 functional units are discussed in detail.

4.4.1 Case study

As a case study, the proposed reliability-aware synthesis algorithm is applied on the AIG depicted in Fig. 4.14(a). As the circuit is simple, only two local transformations are applicable. Rule2 is applicable on node n1. The new error probability of the circuit after applying this rule is presented in Fig. 4.15. It is clear that applying Rule2 on node n1 results in higher reliability. No other rule is applicable on node n1. So, this would be the new reference topology of the circuit. Also, Rule2 can be applied on node n2 of Fig. 4.14(a). After this transformation, the reliability of the circuit reduces, and hence this transformation is not applicable. Further, rule2 can also be applied on node n3 of Fig. 4.14(b). Simulation results presented in Fig. 4.15 show that such transformation also results in improvement in the reliability of the circuit. No further rules can be applied on any of the nodes, and this would remain as the most optimised version of the reference circuit. Fig. 4.15 shows even for such a small circuit, the application of the reliability-aware synthesis algorithm can achieve a marked improvement of 25% over the initial configuration and about 10% over the netlist synthesized by the ABC tool. Another point of interest is the fact that in spite of higher node count in the configuration shown in Fig. 4.14(b), the reliability of the circuit is higher when compared to Fig. 4.14(c). This leads the research to further investigate on developing novel techniques that could result in improving reliability by inserting extra nodes.

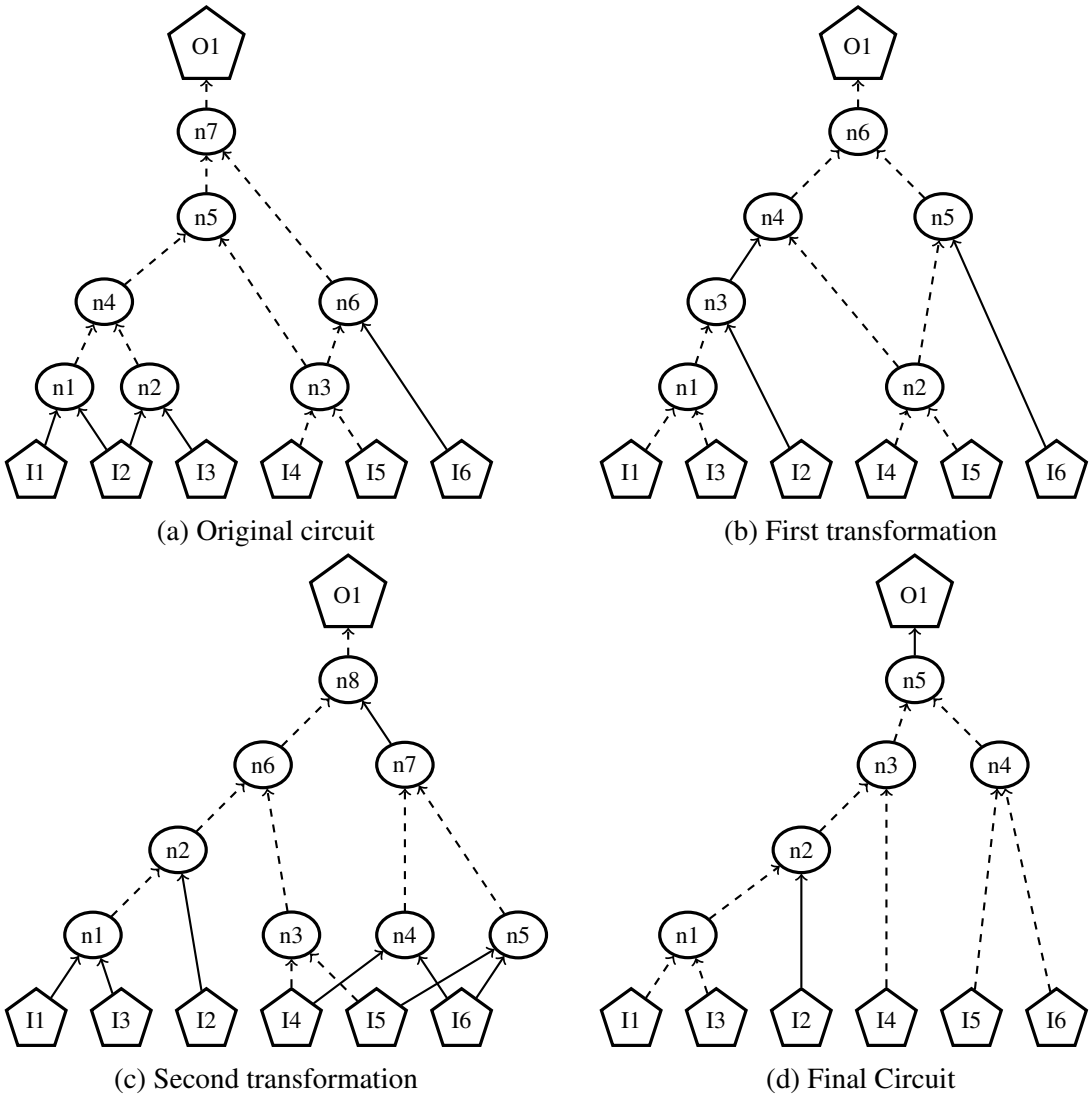


Fig. 4.14 Application of logic transformation rule-set on the original Circuit

4.4.2 Evaluation of MCNC Benchmark Circuits

To prove that the whole methodology is scalable, the tool is tested on a set of MCNC benchmark circuits. Simulation results comparing the circuit reliability of default and the optimised configuration obtained from the tool are reported in Tab. 4.2. In the table, columns 1 and 2 give the name and number of gates in the benchmark circuit. The reliability of both the original configuration and the optimised one are computed and tabulated in the third column. The fourth column reports the percentage improvement achieved through the synthesis algorithm. Column 5 lists the total number of output nodes, and those reliability improvements are greater than 0.5%. The reliability improvement is computed using Eq. 4.4.

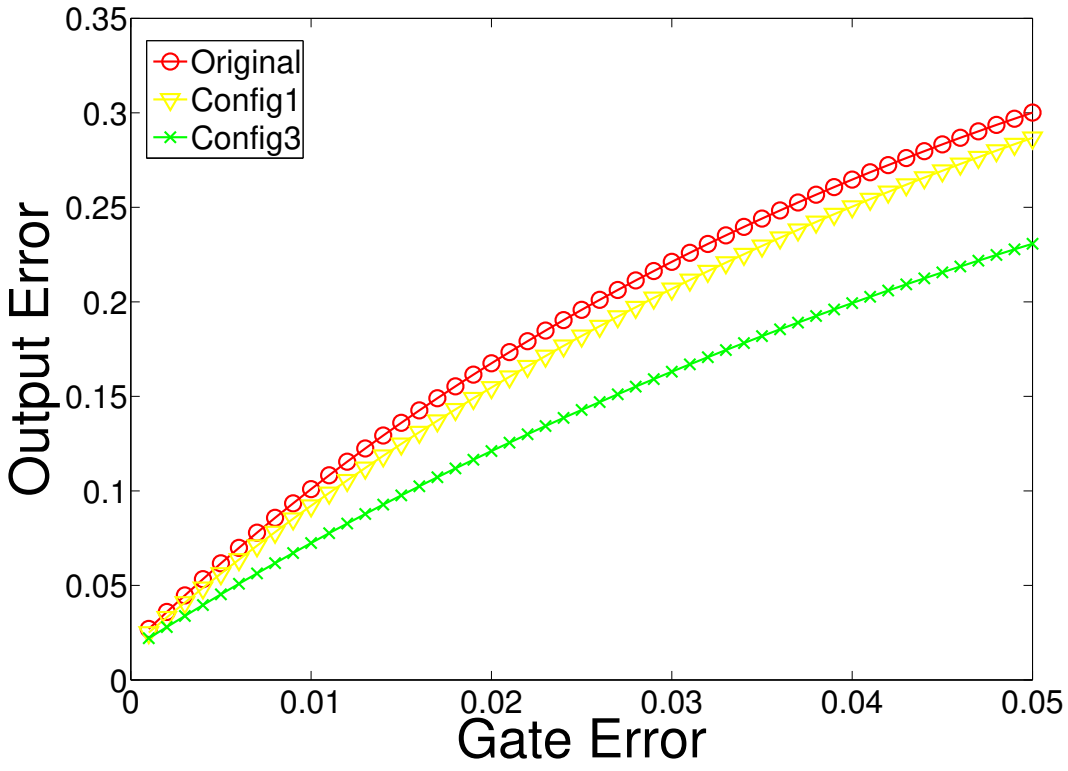


Fig. 4.15 Comparison of results

$$R_{Metric} = \sum_{i=1}^{n} \frac{R_{Org} - R_{Mod}}{R_{Org}} \tag{4.3}$$

where 'n' is the number of nodes. The maximum improvement in reliability is 7.5% for the benchmark circuit, *x2*. Complex circuits with higher output node count include many paths with gate count less than 10 and generally there are no local transformation rules that are found suitable for application. Hence, *vda* and *pair* circuits with higher gate count report lower improvement in reliability.

4.5 Cut Based AIG Rewriting

This section presents an AIG 4-input cut based rewriting algorithm which targets the optimization of circuit reliability rather than area, delay or power. Rewriting is a conventional logic optimization approach which relies on local transformations. Most commercially available logic synthesis tools include a rewriting engine that may operate multiple times on the same netlist during the optimization process. For traditional constraints like area/or delay,

Benchmark	GateCount	Output Error Probability		$R_{Metric}\%$	Output Node Details	
		Original	Optimised		Total	$R_{Metric} \geq 0.5\%$
b9	99	0.16023	0.15036	6.15808	20	7
cm162a	33	0.22993	0.21427	6.81026	5	4
cm85a	35	0.20816	0.19640	5.65277	3	2
cu	45	0.13332	0.12700	4.73912	5	5
dalu	1371	0.32429	0.31516	2.81394	16	15
frg1	125	0.17372	0.17089	1.62925	3	1
pair	1500	0.20542	0.20429	0.54835	131	28
unreg	112	0.09779	0.09365	4.23406	16	16
vda	924	0.15885	0.15724	1.01155	39	18
x2	60	0.16726	0.15468	7.51923	7	6

Table 4.2 MCNC Benchmark Circuits Performance Evaluation (gate error: $\varepsilon = 0.05$.)

the goal is to reduce the number of nodes and the circuit, respectively. But, when circuit reliability is the optimization constraint, the entire optimization algorithm, and its output change drastically.

In cut based rewriting, the first step is to pre-compute the best circuits for a subset of NPN classes of 4-cuts from all possible such functions. Forest (list of all useful cuts) construction is done once and saved to a file, and when rewriting a circuit, the forest is loaded from a file rather than being computed each time. The primary forest constructed in this work consists of 2004 nodes implementing 106 NPN function classes. Comparatively, ABC's original area based rewriting uses a forest of 1785 nodes implementing 139 function classes. This is because, in the ABC's default rewriting algorithm targeting area optimization, unit XOR nodes are also used in the forest, but these are excluded from the reliability-aware rewriting algorithm since XOR node can result in significant output error.

The second step is cut-enumeration and Boolean matching. Cuts that are rooted at a particular node are identified, and functionally equivalent alternative cuts from the precomputed forest are evaluated as replacements for that node. For each cut, the nodes that are computing this function and no others can be removed. New nodes that are not duplicates of existing nodes must be added. The cut implementation is then scored by the number of nodes it removes less the amount it must add. Each node in the forest exists as a function itself and also as a fan-in for further nodes. Auxiliary tables and linked lists are maintained by grouping each node in the forest into NPN function classes. The conditions for a node being tried to be added to the forest is that it is either less than two levels deep or implements a more complex NPN function class deemed practical and that it is better than each existing function executing that class in its output error probability.

The decomposition module within ABC encompasses all the coding details that handle network modifications during rewriting. A decomposition includes the new nodes that must be added or reused (found in the network by strashing). Nodes in the original cut are marked for removal if they are not needed elsewhere. In area-based rewriting, a decomposition must be formed for every implementation of every cut as building the decomposition is how area change is calculated. All these decompositions are constructed still when rewriting for reliability to retain the same, known correct, code structure, even though the decomposition could be delayed and one calculated per cut since the reliability information is available in the forest. For each alternative implementation, a decomposition (possible alteration) of the network based on substituting that implementation is formed.

4.5.1 The CAD Algorithm

The main algorithm implemented in this work is called "Rewrite for Reliability" or $'rwrel'$ for its command in ABC. This algorithm is based on the standard cut-rewriting algorithm from ABC, adapted so that the goal is reliability instead of area. Modified versions of the data structures were created to include node error probability values. The critical functions of the rewriting process were then rewritten to use reliability as a goal. All utility functions from the rewriting module were duplicated, just changing them to work with the expanded structures instead of the originals. The cut enumeration and decomposition modules are self-contained and are used directly. Eq. 4.4 quantifies the 'percent decrease in error' goal function that is used in guiding the optimization algorithm to select the better cut.

$$R_{Metric} = \frac{P_{\varepsilon}(Old_Cut) - P_{\varepsilon}(New_Cut)}{P_{\varepsilon}(Old_Cut)} \tag{4.4}$$

A pseudo-code of this algorithm is presented in Alg. 4. The aim is to find a sequence of transformations leading to an AIG network that minimize the cost function. A heuristic approach is employed to find an acceptable solution, i.e., an AIG providing higher reliability than the original circuit. The strategy chosen is to figure out all the functionality equivalent cuts for a given node and determine the best among them. Given a combinational circuit implementing the Boolean function f and its AIG network, the algorithm, starting from PI's, traverses through the graph to see if any of the transformations are applicable on the given node. For every possible transformation, the new circuit reliability value is computed. The configuration that yields the highest circuit reliability improvement is chosen, and the new topology is generated. This process is continued on every graph node until the primary outputs are reached where no more transformations are applicable.

Algorithm 4 Cut based Reliability Optimization

```
1: INPUT : P_ntk, NODE_count, PI_sp , G_err
2: OUTPUT: PO_err
3: for all nodes I= 1 to NODE_count do
4:     for Node N in AIG do
5:         Get cuts based at N
6:         for 4-input cut C based at N do
7:             Get truth-table F of N concerning C
8:             for Possible graph S of function F do
9:                 Make decomposition D corresponding to cut C
10:                Remove original nodes from D
11:                Add nodes of S to D
12:                if Level > MaxLevel then
13:                    Go to Next S
14:                end if
15:                Compute savings as nodes that can be dropped from network with D
16:                Compute cost of adding new nodes.
17:                Error = Error(S)
18:                if Error < BestError then
19:                    BestS = S
20:                    BestD = D
21:                end if
22:            end for
23:        end for
24:        Apply decomposition BestD to the AIG
25:    end for
26: end for
```

Two versions of the algorithm were developed in this work. In both of them, the rewrite for a given cut is selected for the least error, based on the reliability information available in the forest. Choosing which cut to use for node rewriting is the difference between the two algorithm versions. The first version selects the least error for each cut and then decides the least error cut for the node. The second version instead selects the most improved cut concerning percentage improvement in reliability, for each node. Initially, the first version of the algorithm was implemented, but while the obtained results were good, better performance was expected. One possibility considered was that the absolute error goal function favored situations in which the optimization process ended up in a local minimum. Choosing the cut with the best reliability naturally favors already optimised cuts where little progress can be made, where it might be more favorable to select a slightly unreliable cut and improve it to have better reliability. Thus, the second algorithm version makes use of reliability

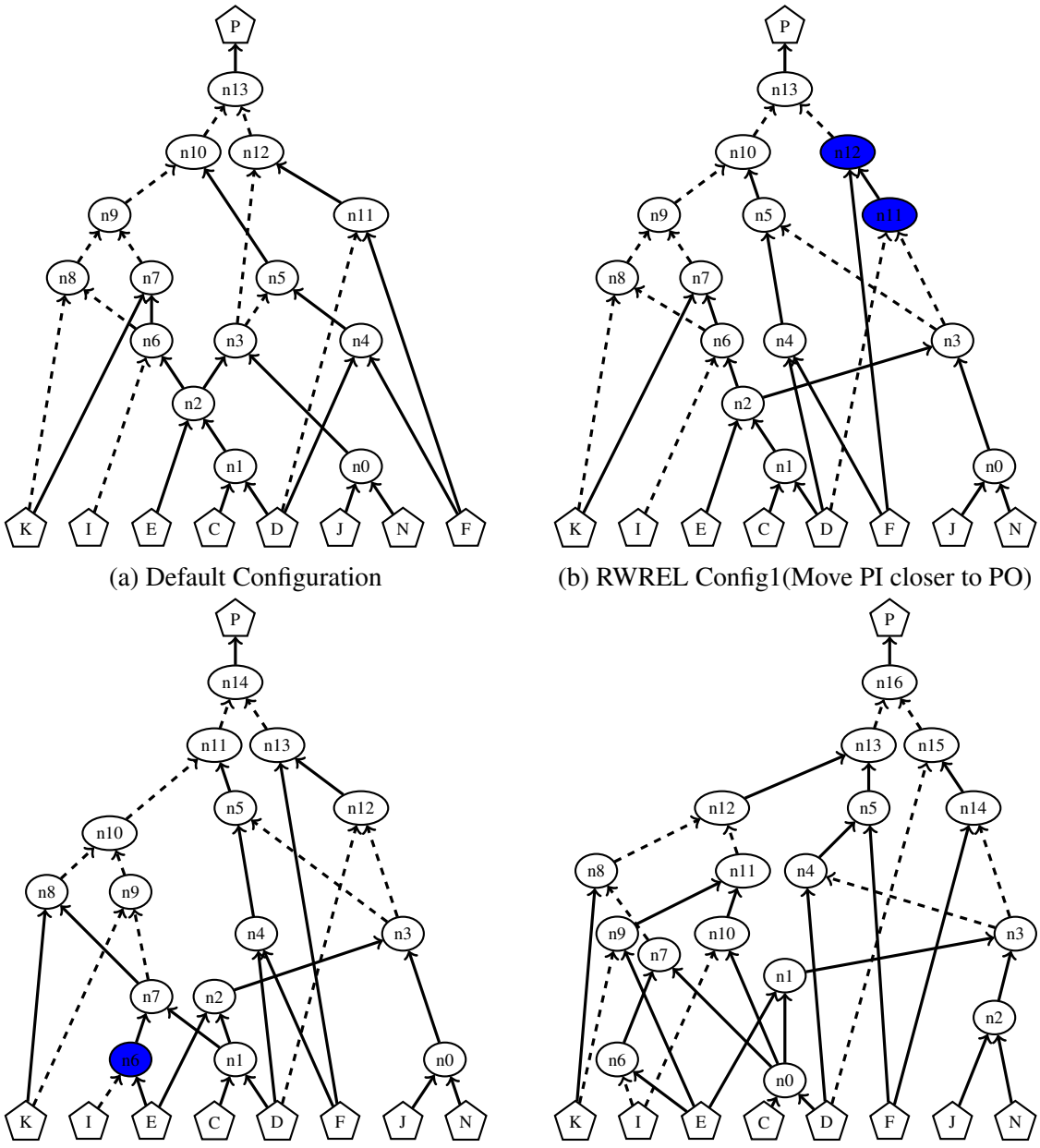
improvement as a metric instead of the absolute error. This metric change resulted in about three times better improvement and given this, all the results presented in the next section are based on the second algorithm.

4.6 Experimental Results

In this section, the potential practical implications of the proposed reliability-aware synthesis tool are evaluated. In this view, Alg. 4 was implemented and integrated in ABC synthesis tool as a command $'rwrel'$. Another command $'printrel'$ is developed to display the node reliability statistics in the network. The total savings achieved concerning circuit reliability is studied by performing a set of experiments using MCNC benchmark circuits [114] with more than 3000 node AIGs after structural hashing. A unit gate error probability of 10^{-2} is assumed in all the simulations. The networks were loaded in Berkeley Logic Interchange Format (BLIF) format and strashed to AIGs. Functional equivalence was extensively verified during development using the ABC equivalence check command $'cec'$. We use internal technology mapping algorithms to map the AIG network to a gate level Verilog netlist using the TSMC 65 GP CMOS standard cell library. Synopsys Design Compiler is employed for power, timing and area estimation on the two netlists. Also, the two netlists are verified for equivalence via Synopsys Formality. After the algorithm correctness and performance is well established, its effect on the implementation of 8051 micro-controller basic blocks like an adder, multiplier, and divider was also tested. In this section, a case study is initially presented highlighting the crucial differences in the way the Boolean matcher finds alternative cuts when reliability is the constraint. Then, the results obtained for various MCNC benchmark circuits and 8051 functional units are discussed.

4.6.1 CM162a – A Case Study

The proposed reliability-aware synthesis algorithm is applied on the MCNC benchmark circuit $'cm162a'$, which implements a 14-input 5-output logic function. The current case study considers the cone is comprising of all the gates driving the output pin 'P'. Fig. 4.16(a) represents the default AIG configuration. The total number of nodes in the default AIG representation of the circuit is 14. With a unit gate error probability of 10^{-2}, the output node 'P' has an error probability $Err_{org} = 0.0717$. Three specific functionally equivalent logic configurations generated by the tool are now explained in detail. The idea is to emphasize on how the Boolean matcher works when the constraint is set to output error optimization instead of area.



(a) Default Configuration

(b) RWREL Config1(Move PI closer to PO)

(c) RWREL Config2(Insert Redundant Node

(d) RWREL Config3 (Optimised)

Fig. 4.16 MCNC Benchmark CM162A – A Case Study

Fig. 4.16(b) depicts the first sub-case: Moving the PI's closer to the PO's wherever possible. It is clear that the PI node 'F' has been pushed closer to the PO. This helped in reducing the overall output error on the output node to $Err_{rwrel1} = 0.068$. Fig. 4.16(c) depicts the second sub-case: insert redundant nodes. Extra nodes if added pragmatically can result in significant error masking effect that reduces the overall error on the output nodes

considerably. A redundant node $'n6'$ has been appended onto the logic, thus total node count is increased to 15 and reduces the overall output error to $Err_{rwrel1} = 0.0676$.

After multiple iterations, the final version of the optimised circuit configuration in Fig. 4.16(d) is obtained. The final output error value of the optimised circuit is $Err_{opt} = 0.0633$ and the total node count is 17. Two redundant nodes $'n9'$ and $'n10'$ are added into the configuration. Also, node $'n4'$ in Fig. 4.16(c) is modified such that the PI node 'F' has been pushed closer to the PO in similar lines with the first sub-case. Some of the important differences observed in the way Boolean matching is performed by the reliability-aware rewriting algorithm when compared to the area driven approach in [28] are listed below:

– Cuts with Primary Inputs closer to the Primary Outputs are preferred. Primary Inputs have lower error probability when compared to the internal nodes. Thus, moving a highly reliable node closer to the output can result in canceling out some of the errors.

– Cones with more considerable circuit depth can result in higher critical path delay and are not preferable in case of delay optimization. But, they can result in lowering the output error values and are often employed by all the algorithm.

– Redundant Node insertion is commonly performed. This increases the total area but has higher masking effect and thereby reducing the output node error probability.

– One other significant difference that has been observed is the matching of path lengths. Most of the time the Boolean matcher selected cuts in such a way that the resulting circuit is more balanced so that the level of both the left and right paths are similar.

Benchmark	No. of Outputs	Node Count			Circuit Depth		Averaged Absolute Error		Output Error Improvement (%)		
		NC_{org}	NC_{opt}	Increase (%)	CD_{org}	CD_{opt}	Err_{org}	Err_{opt}	Avg	Max	Min
apex7	36	221	252	14.03	14	14	0.005895	0.005240	10.44	51.33	1.74
cm162a	5	36	44	22.22	9	8	0.005874	0.005128	13.88	15.66	9.74
comp	3	107	119	11.21	18	19	0.009510	0.007232	18.31	25.31	5.98
dalu	16	1371	1602	16.85	35	35	0.019144	0.016449	13.98	19.45	7.98
I6	67	692	523	-24.42	5	4	0.008381	0.007291	12.83	15.19	4.23
I7	67	903	702	-22.26	6	5	0.009235	0.008303	10.21	75.5	3.16
I8	81	3310	2187	-33.93	21	20	0.018626	0.015219	18.83	32.04	6.8
k2	45	1998	2152	7.71	23	19	0.031802	0.028530	12.94	25.65	4.52
unreg	16	112	111	-0.89	5	5	0.006506	0.005382	18.43	19.03	17.78
vda	39	924	1042	12.77	16	18	0.030577	0.027491	12.81	65.6	5.31
too_large	3	824	773	-6.19	30	27	0.053492	0.046468	13.87	17.43	11.33
8051_add	19	175	222	26.86	28	29	0.014149	0.013330	16.99	41.56	5.86
8051_mul	17	630	765	21.43	48	47	0.030946	0.029475	10.58	32.65	1.58
8051_div	17	1010	1181	16.93	198	181	0.023831	0.022888	12.31	31.75	2.18

Table 4.3 RWREL Performance Evaluation on different Benchmark Circuits (Gate error ε = 0.001)

Benchmark	Area			Delay			Dynamic Power			Leakage Power		
	A_{org}	A_{opt}	Increase(%)	T_{org}	T_{opt}	Increase(%)	P_{org}	P_{opt}	Increase(%)	P_{org}	P_{opt}	Increase(%)
apex7	705.24	779.04	10.46	1.10	1.16	5.45	46.25	53.32	15.28	3.12	3.47	11.28
cm162a	121.68	137.88	13.31	0.69	0.56	-18.84	7.91	8.97	13.47	0.54	0.61	13.29
comp	360.36	405.72	12.59	1.36	1.38	1.47	34.22	36.11	5.54	1.61	1.79	11.46
dalu	4232.52	4876.2	15.21	3.41	3.15	-7.62	310.43	403.97	30.13	18.66	21.93	17.52
I6	1861.92	1677.6	-9.9	1.37	0.89	-35.04	133.68	137.11	2.57	8.28	7.62	-7.97
I7	2386.8	2144.16	-10.17	1.48	1.38	-6.76	174.82	175.92	0.63	10.66	9.65	-9.47
I8	8147.52	5780.16	-29.06	4.59	3.11	-32.24	720.15	522.14	-27.5	37.7	26.66	-29.29
k2	4675.32	5343.84	14.3	1.96	1.87	-4.59	119.51	177.1	48.19	23.53	26.07	10.79
unreg	367.2	372.6	1.47	0.48	0.52	8.33	29.36	32.16	9.53	1.64	1.71	4.34
vda	2158.92	2638.08	22.19	1.08	1.31	21.3	88.61	129.34	45.97	11.18	13.09	17.08
too_large	2176.2	2087.64	-4.07	2.32	2.20	-5.17	118.6	121.57	2.5	9.45	9.16	-3.08
8051_add	595.08	723.6	21.6	2.49	2.60	4.42	74.19	84.25	13.55	2.67	3.24	21.34
8051_mul	2038.32	2432.16	19.32	3.93	3.82	-2.8	304.85	363.33	19.18	9.17	10.87	18.51
8051_div	3295.8	3781.44	14.74	16.96	15.43	-9.02	555.68	612.15	10.16	14.95	17.1	14.38

Table 4.4 Area, Delay and Power Analysis – A comparative Study

4.6.2 Evaluation of Benchmark Circuits

To prove that the proposed methodology is scalable, the tool is tested on a set of MCNC benchmark circuits. Simulation results comparing the average circuit output errors of the original and the optimised configuration obtained from the tool are reported in Tab. 4.3. The name of each circuit is given in the first column while column two provides the output pin count of each circuit. The following three columns list the node count of the original circuit, the optimised circuit, and the optimization induced node count change in percentage. Columns 6 (7) provides the circuit depth of the default (optimised) circuits. Columns 8 (9) lists the average output error value for the default (optimised) circuits. Columns 10 through 12 lists the average, maximum, and the minimum relative output error improvement. Tab. 4.4 reports the area, timing and power analysis after applying RWREL on MCNC benchmark circuits. In the table, P_{org}, T_{org} and A_{org} are the power, timing and area results from the original MCNC netlist. P_{opt}, T_{opt} and A_{opt} are the power, timing and area results from the netlist obtained via RWREL optimization.

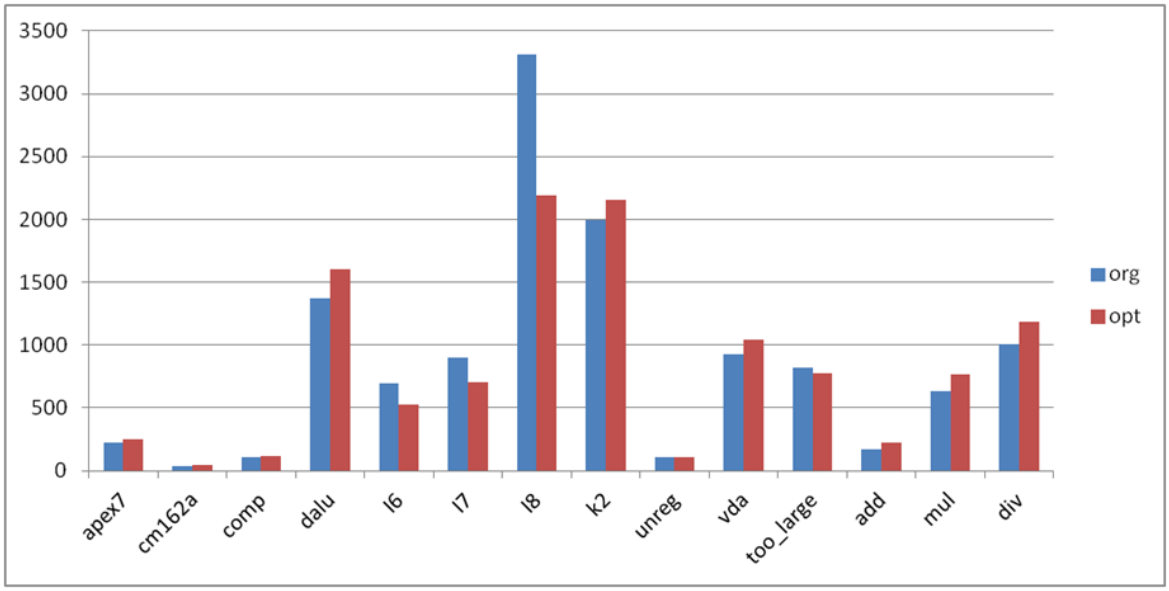


Fig. 4.17 Comparison of results : Node count reduction

From the tables, it is clear that significant SER reduction can be achieved very low area overhead of 6.57% that results in 13.52% higher power consumption by employing the optimization algorithms. An average improvement of 14% and a peak improvement of up to 75.5% was observed. $'I7'$ benchmark is a simple logic circuit consisting of 199 PI's and 67 PO's and a total node count of 903. Output pin $V266(6)$ provides the peak error improvement of 75.5%. Another important fact worth mentioning is that benchmark circuits $'I6'$, $'I7'$, $'I8'$ and $'too_large'$ report reduction in node count. This proves that the algorithm is intelligent

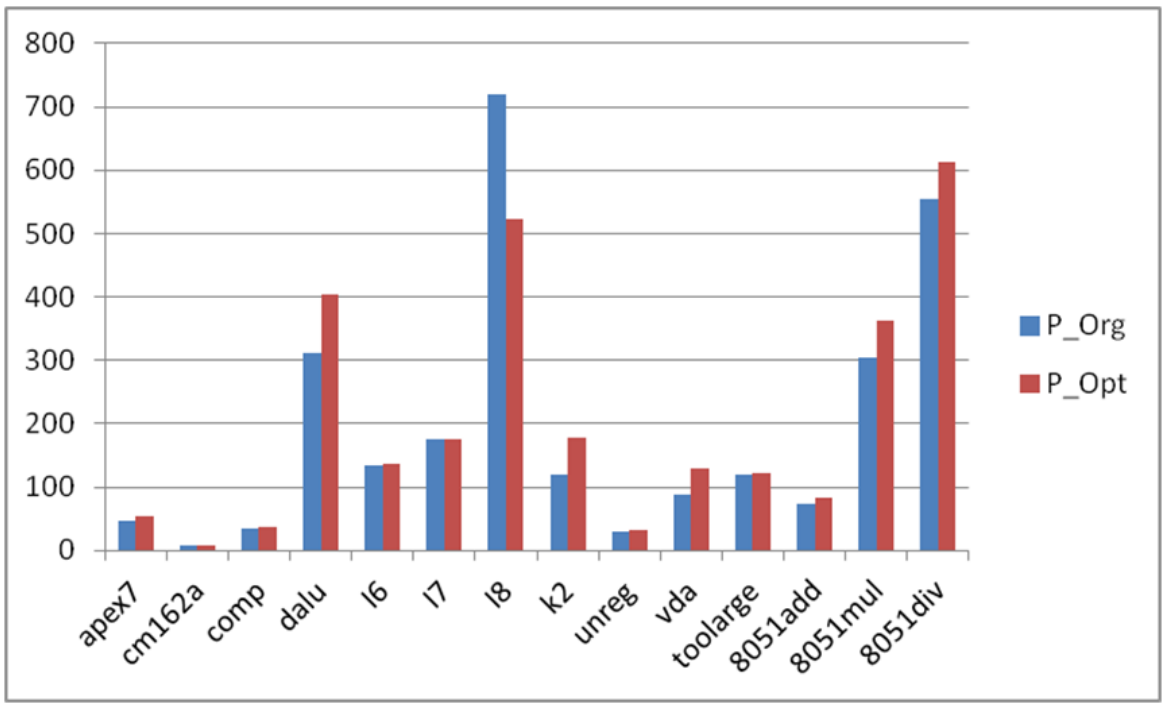


Fig. 4.18 Comparison of results: Power reduction

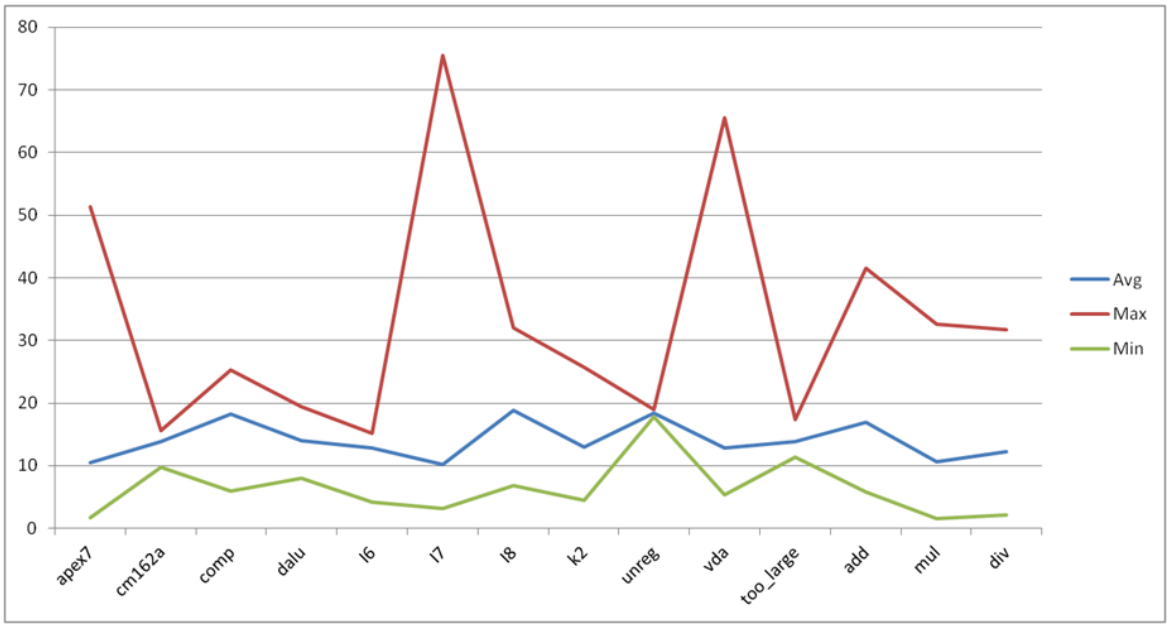


Fig. 4.19 Comparison of results : Reliability improvement

to decide when to add redundancy and when to remove unnecessary nodes that can result in better output reliability. Also benchmark circuit $'too_large'$ reports a high output error. This is because it has 824 nodes and just three output pins. It implies that if a significant number

of gates are placed within the cone to emulate a particular function, higher the probability of error on the output. Also, the circuit depth of the logic resembles no direct relation to the output error as circuit depth has reduced in some instances while in others it has been increased. The tabulated data has been plotted to better project the results. Fig. 4.17 depicts the node count between the original and optimised netlist. Fig. 4.18 depicts the power change between the original and optimised netlist. Fig. 4.19 depicts the reliability improvement between the original and optimised netlist.

4.7 Conclusions

Two different methodologies have been developed to perform reliability driven combinational circuit synthesis. First, a simple synthesis algorithm was presented that optimises the circuit reliability based on a small set of local transformation rules. Application of these rules on the standard MCNC benchmark circuits with node count from 30 to 1500 resulted in improving overall circuit reliability by up to 7.5%. Another methodology which is more generic was developed extending the local transformation rule set to encompass more possible scenarios. A reliability driven 4-cut enumeration and Boolean matching technique that improves circuit reliability has been proposed. The technique of rewriting for reliability was developed by extending an existing cut based rewriting tool to make use of local transforms targeting a reliability metric improvement instead of area. A synthesis algorithm that optimises the circuit output nodes error probability was also presented. The application of the proposed reliability-aware synthesis algorithm on various MCNC benchmark circuits with a node count from 50 to 3000 resulted in up to 75.5% output error probability reduction. On average, about 14% SER reduction was obtained at the expense of less than 4.5% area overhead.

In this chapter, multiple optimisation algorithms are analysed with the aim of improving reliability through netlist transformation, without altering the number of inputs or outputs. In the next chapter, error control coding techniques are employed to improve circuit reliability by redundancy addition into the circuit hence altering the number of inputs/outputs.

Chapter 5

CPE : Codeword Prediction Encoder

5.1 Introduction

In the previous chapter, logic synthesis techniques aimed at improving reliability through netlist transformation, without altering the number of inputs or outputs, were explored. In this chapter, error control coding based redundancy addition techniques are explored to improve reliability. In traditional models of communication or storage systems with error correction coding, it is assumed that the operations of an error correction encoder and decoder are deterministic and that the randomness exists only in the transmission or storage channel [115]. However, with the advent of nano-electronics, the reliability of the forthcoming circuits and computation devices is becoming questionable. Due to massive increases in density integration, lower supply voltages, and variations in the technological process, MOS, and emerging nano-electronic devices will be inherently unreliable [9], [10]. Besides, a significant challenge to current CMOS design is to lower the energy consumption by several factors of magnitude, with the apparent goal of energy preservation. Diminishing the energy consumption can be addressed by aggressive supply voltage scaling, with the drawback that bringing the signal level closer to the noise level reduces noise immunity and leads to unreliable computing [6].

Consequently, in the future systems of communication and storage, errors may not only come from the transmission channels, but also from the faulty hardware. It is then becoming crucial to design and analyze error correcting decoders able to provide reliable error correction even if they are made of unreliable components. Thus, the study of novel techniques for reliable data transmission using unreliable hardware is an increasing priority. Low-density parity check (LDPC) codes are known to provide excellent error correction performance that closely approaches the Shannon capacity of noisy transmission channels [116], [117]. As a result, they have been adopted in many current and next-generation wireless protocols such

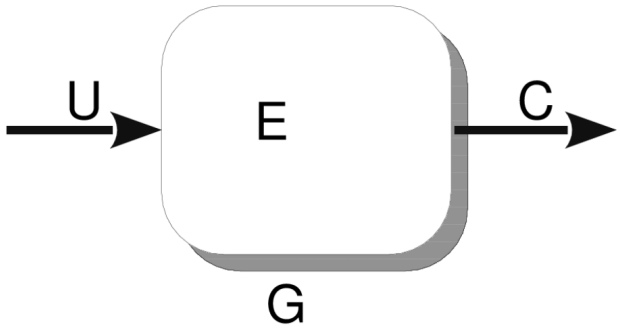
as IEEE 802.16E (WiMAX), IEEE 802.11 (Wi-Fi), and DVB-S2/T2/C2 standards, besides many other applications [118]. A new fault-tolerant technique called CPE using ECC based methods/architectures to improve circuit reliability is now presented.

5.1.1 Main Contributions and Outline

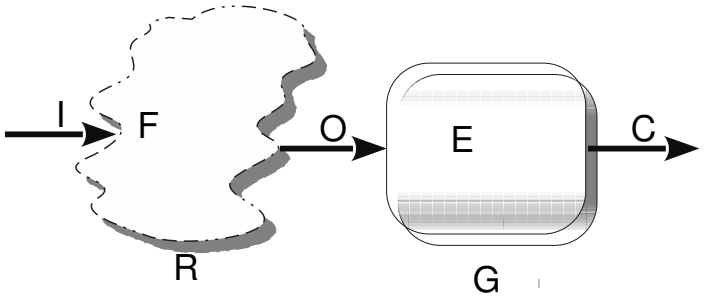
Work on fault tolerance and reliability can either be focused on protecting combinational circuits, memories or dynamic systems where combinational logic and small memory (state) co-exist. This work deals only with the problem of protecting combinatorial logic, and in particular, the focus is on the encoding process. The interest on the encoding process is born from the fact that they are often assumed to be fault free (even when the decoder process is assumed unreliable) [119]. This work introduces novel reliability driven fault tolerant methodology known as Codeword Prediction Encoder (CPE) for reliable LDPC encoding by augmenting extra logic to correct the errors introduced during the encoding process [3] [4]. The approach presented here is an expansion of the Check Symbols Generation [120] and the Parity Prediction Function [121], [122], where circuitry is added to a combinatorial network to generate extra bit to ensure parity. These approaches are formalized and extended to take full advantage of the power of error correction codes to enable correction of the faults and not just detection.

The CPE approach, as shown in Fig. 5.1 focuses on using ECC based methods/architectures to improve the reliability of combinatorial circuits. A significant difference when compared to standard EDA optimization techniques is the fact that there is no attempt to modify or alter the logic internally but the aim is to augment the circuit in such a way that it adds "redundancy" that can then be exploited to recover the correct output. A parallel can be drawn with the communication systems where the redundant symbols are added to the message and transmitted. This redundant symbol allows the recovery of the message even in the presence of transmission error. The CPE simulator provides a unified platform which comprises the novel encoder and fault tolerant LDPC decoders. Encoders using regular LDPC codes with different column weights(d_v) for the parity check matrix, namely $d_v = 3$ and $d_v = 4$, and different coding rates (r), namely $r = 1/2$ and $r = 3/4$ were employed. Also, different state-of-the-art reliability enhanced LDPC decoding mechanisms like Self-Corrected Min-Sum (SCMS), and Gallager B with Extended Alphabet(Gal-B) was used. Simulations results prove that it is possible to retrieve the original information by employing particular configurations of these encoders and decoders. In general, output BER is reduced by up to 10,000 times by adopting CPE mechanism as compared to transmitting data directly.

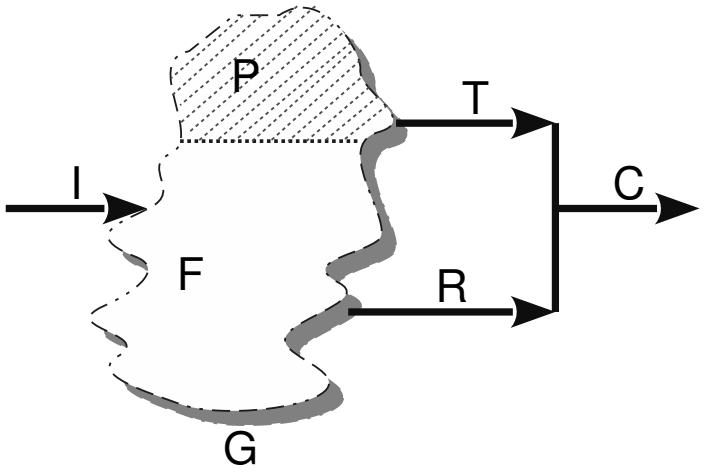
For the mathematical analysis presented, the system where the CPE methodology is applied can be partitioned based on two characteristics:



(a) Encoder



(b) Encoding by Concatenation



(c) Proposed Encoding

Fig. 5.1 Methodology of Codeword Prediction Encoder

- if the circuit to protect is linear or non-linear. Here, a linear circuit is a circuit consisting only of XOR gates. Such a system can be described as a linear system of equations similar to a linear code.
- if the system is symmetric (encoding and decoding process are faulty) or asymmetric (perfect decoding process)

The analysis presented here focuses on the case of linear circuits, for both the symmetric and asymmetric scenarios. Non-linear circuits present several difficulties due to the lack of mathematical methods to work with such circuits. The symmetric-linear case is of great interest from the analysis point of view because in the symmetric scenario the only suitable coding scheme is LDPC codes, thanks to their ability to decode the received message even in the presence of faulty hardware. LDPC codes are of great interest from the ECC augmented hardware point of view due to their low complexity and the complete understanding of their asymptotic performance and theoretical limits. The first factor turns them into most favored scheme to be used in hardware where area and the throughput are the dominating factors. The second factor allows for an asymptotic analysis of reachable performances to be carried out.

5.2 LDPC Codes and Error Models

Consider the data transmission scheme depicted in Fig. 5.2. In this scheme, a binary information sequence u of length k must be transmitted through a noisy channel. To recover u correctly at the output of the channel, the information sequence can be protected by adding some redundancy in the transmitted data. As u must be successfully recovered at the output of the channel, the information sequence must be preserved by adding some redundancy in the transmitted data. The data protection can be done with an LDPC code that encodes the information sequence u into a codeword x of length $n > k$. At the output of the channel, the received sequence y is passed through an LDPC decoder that aims at reconstructing u.

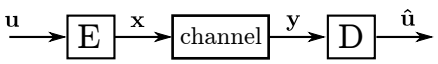
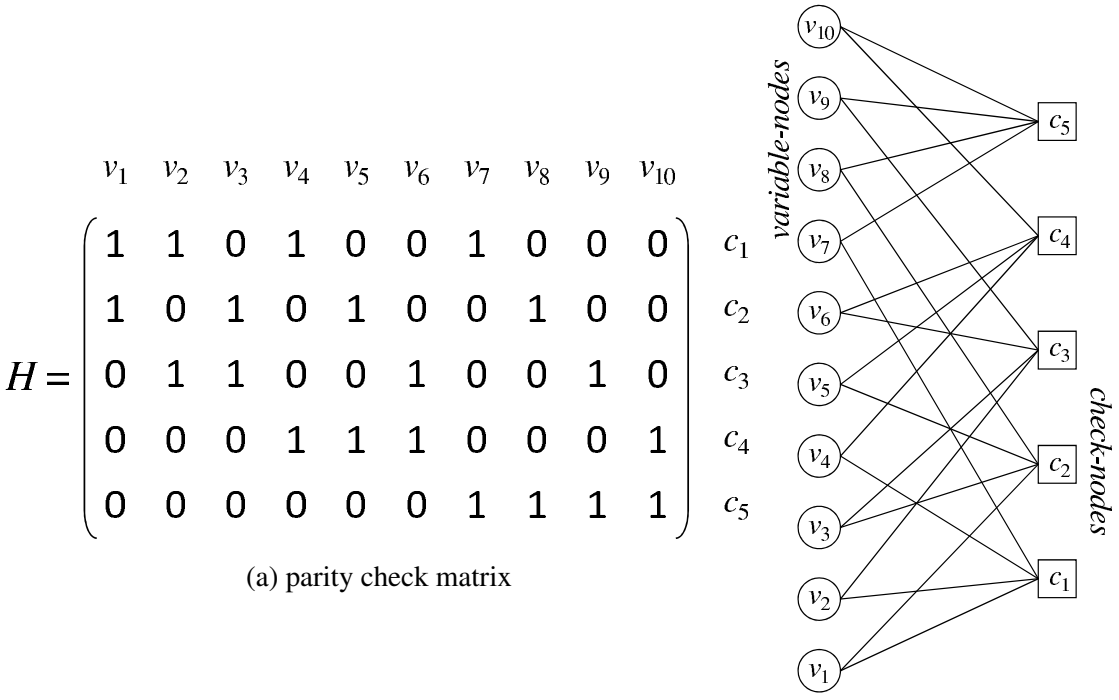


Fig. 5.2 Data transmission scheme

In this section, LDPC codes, as well as the encoding and decoding operations, are first introduced. Then, the gate error model employed in modeling the unreliable gates that constitute the encoder and the decoder is explained.



(a) parity check matrix

(b) Tanner graph

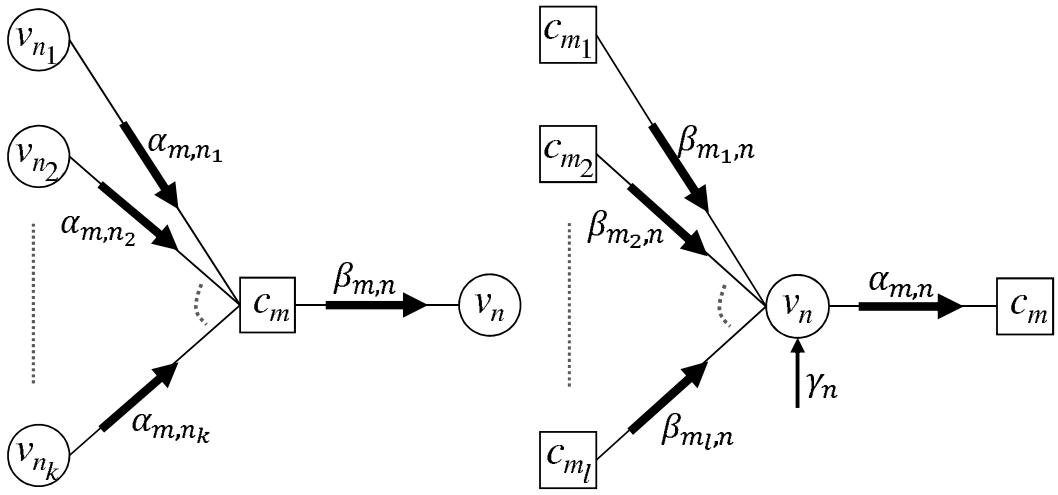
Fig. 5.3 LDPC Codes

5.2.1 LDPC codes

Low-density parity check (LDPC) codes are a class of linear block codes invented by Gallager [116]. A LDPC code is defined by its binary parity check matrix H of size $m \times n$, see Fig. 5.3 (a). A binary vector x of length n is a codeword of the LDPC code if it satisfies Eq. 5.1.

$$Hx^T = 0, \tag{5.1}$$

where T is the transpose operator. For LDPC codes, the parity check matrix H is sparse, *i.e.*, it contains only a few non-zero components. In the following, d_v and d_c denote the number of 1's in each row and in each column of H, respectively. Tanner also introduced a graphical representation of LDPC codes as shown in Fig. 5.3 (b). Tanner graphs are bipartite, which means that the nodes of the graph are separated into two distinct sets. The first set contains n variable nodes and the second set includes m check nodes. Edges are only connecting nodes of two different types, and there is an edge between variable node v and check node c if and only if there is a one at the corresponding position in the parity check matrix. In the end, the matrix representation of the LDPC code is used for encoding, while the graph representation is used for decoding, as described in the next section.



(a) Check-to-variable messages

(b) Variable-to-check messages

Fig. 5.4 LDPC message computation

5.2.2 LDPC Encoding and Decoding

Once the LDPC parity check matrix H is fixed, it remains to construct the corresponding encoder that transforms any information sequence u into a codeword x that satisfies Eq. (5.1). From H, one can construct a generator matrix G of size $k \times n$, where $k = n - m$, that verifies $HG^{\mathrm{T}} = 0$. The encoding operation can then be realized from the generator matrix as

$$x = uG \tag{5.2}$$

Several solutions have been proposed to construct a generator matrix G from the parity check matrix H [123]. Most of the usual solutions consider systematic encoding, for which the codeword $x = [u, p]$ contains both the information sequence u and m parity bits given by p. In this case, the left hand side of the generator matrix G is the identity matrix of size $k \times k$.

After encoding, the codeword x is transmitted on the channel, which outputs y. If the channel is a Binary Symmetric Channel (BSC), the received word can be written as $y = x + e$, where e is the binary error pattern and the sum above is computed modulo 2. From condition (5.1), one can compute the syndrome $z = Hx^T + Hy^T = He^T$. The decoding problem consists of finding the most probable vector e that explains the observation of the syndrome z. For LDPC codes, decoding can be implemented by message passing algorithms that exchange messages between variable-nodes and check-nodes, as shown in Fig. 5.4. Several LDPC decoders (Gallager B, Min-Sum, Belief Propagation, etc.) have been proposed, which consist of different processing rules and simplifications of the message-passing algorithm. These decoders have different complexity and different decoding performance.

LDPC codes were initially introduced under the assumption of reliable hardware. To analyze the performance of LDPC codes under unreliable hardware, an error model to mimic unpredictable behavior in the logic gates that are used in the encoder and the decoder is presented.

5.2.3 Gate Error Model

The unreliable gates are modeled in-line with the Von Neumann model [41]. A Von Neumann erroneous gate is modeled as an ideal logic gate cascaded with an error injecting XOR gate. The error-injecting XOR gate determines the stochastic error behavior by toggling the gate output with a pre-defined probability. As an example, Fig. 5.5 graphically represents an unreliable AND gate.

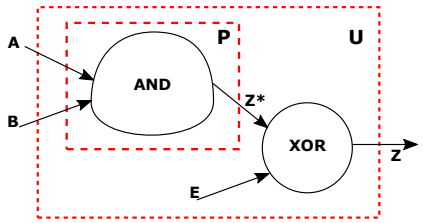


Fig. 5.5 Gate Error Model.

The noise variable E takes value 1 with a given probability p_g that represents the gate error probability. Under this model, it was shown that LDPC decoders built from unreliable gates are naturally robust to errors, with no need for additional circuit protection [124], [82]. Unfortunately, it was also shown that most of the standard LDPC encoders fail entirely when they are built from unreliable gates [83]. Now, the novel robust LDPC encoding solution that consists of computing extra parity bits is described.

5.3 Codeword Prediction Encoder (CPE)

Consider the systematic encoding operation described by Eq. (5.2). Fig. 5.6 represents the encoding error probability P_e with respect to the gate error probability p_g for various LDPC codes, with $k = 1000$, $d_v = 3$ and $r = 1/4, 2/5, 1/2, 5/8$, respectively. The encoding error probability does not depend on the coding rate. Fig. 5.6 also shows that the encoding error probability is dramatically increased concerning the gate error probability p_g. For example, a gate error probability $p_g = 10^{-4}$ will give an encoding error probability $P_e = 10^{-2}$, which represents an increase by a factor of 100. Even small gate error probability values give high encoding error probabilities. As a result, the high encoding error probability is going

to combine with the channel noise, and at the end, the decoder will not be able to recover the original information sequence u from y. Hence there is a need to drastically reduce the encoding error probability before transmission on the channel.

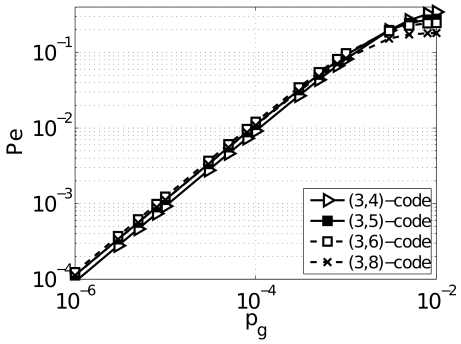


Fig. 5.6 Encoding error probability P_e with respect to gate error probability p_g. In the legend, the $(3,x)$-code represents the code with $d_v = 3$ and $d_c = x$.

As described in Fig. 5.7, a first solution consists of passing the noisy codeword $\tilde{x}$ through an LDPC decoder before channel transmission to eliminate the encoding errors. This solution will be practical only if the gate error probability leads to an encoding error probability lower than the correction capability of the LDPC code. As the encoding error probability can be high, the correction capability of the LDPC code alone is not sufficient. The proposed Codeword Prediction Encoder (CPE) approach depicted in Fig 5.8 which consists of two methodologies: non-systematic and systematic.

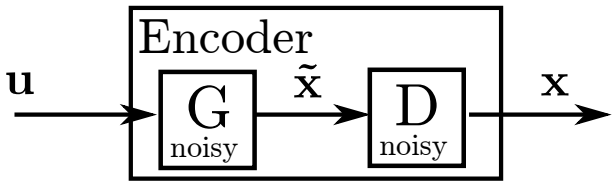
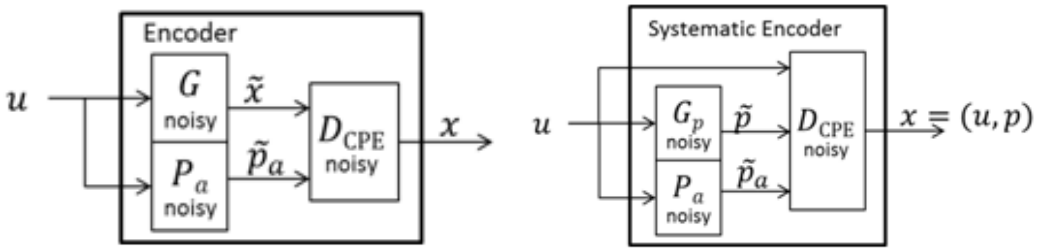


Fig. 5.7 First encoding solution

In case of non-systematic encoding as described in Fig 5.8(a), in addition to the parity bits p contained in $\tilde{x}$, m_a extra parity bits $\tilde{p_a}$ are also computed from u. The vector $\tilde{x_a} = [\tilde{x}, \tilde{p_a}]$ is called the augmented codeword. Before channel transmission, $\tilde{x_a}$ is passed through a different LDPC decoder, denoted by D_{CPE} to eliminate the encoding errors. The extra parity bits $\tilde{p_a}$ serve only to help eliminate the encoding errors, and, after decoding, only $\tilde{x}$ is transmitted through the channel. Thus, both the LDPC code that produces $\tilde{x}$ and the one that generates $\tilde{x_a}$ have to lead to good decoding performance.

The CPE approach for systematic encoding is illustrated in Fig 5.8(b). G_p represents the G sub-matrix of size $n-k \times k$, corresponding to parity bits. Thus, x= [u, p], and the parity



(a) Non-systematic Encoding (b) Systematic Encoding

Fig. 5.8 The CPE approach

bits p can be computed by $p = u \cdot G_p$. In this case only the parity bits p can be affected by gate errors, while the data bits u are assumed to be error free. The circuit composed either by G and P_a (non-systematic case) or by G_p and P_a (systematic case) is denoted as G_{CPE}.

The D_{CPE} decoder used at the encoding side makes use of both the original parity bits $\tilde{p}$ and of the extra parity bits $\tilde{p_a}$ to eliminate the encoding errors. If the LDPC codes are constructed carefully, this strategy will result in increased correction capabilities for the augmented codeword $\tilde{x_a}$ compared to the initial codeword $\tilde{x}$. For good code construction, an important condition is that the extra parity bits $\tilde{p_a}$ are independent of the original parity bits p, which means that $\tilde{p}$ and $\tilde{p_a}$ are computed from different combinations of bits from u. To guarantee the independence while ensuring good decoding performance for both $\tilde{x_a}$ and $\tilde{x}$, split-extended construction introduced in [125] is employed. Precisely, the additional parity bits $\tilde{p_a}$ are computed in the same way as the 'extended bits' in [125], and the D_{CPE} decoder utilizes the split-extended parity-check matrix defined therein. In the end, the performance of the proposed CPE approach will depend on the choice of the code (rate, degrees, etc.) and of the considered LDPC decoder. In the following section, the CPE mathematical analysis is presented.

5.4 CPE Mathematical Analysis

The CPE methodology has already been presented in the previous section. In this section, theoretical analysis of its performance limits is described.

5.4.1 CPE Cost Analysis

To evaluate the performance of any new proposed scheme, it is necessary to consider all the four constraints(area, timing, power, and reliability) and the inter-relation between them. In this line of thought, a general notation is introduced to evaluate proposed schemes. The term

cost function, C(), is employed to reflect the performance of the proposed scheme, either for any one of the four goals or any linear combination of the four. The cost of an XOR gate is considered as the reference, and the cost of any circuit can be represented as a multiple of this cost.

This would suit the analysis for the case of linear circuits while an extension for non-linear cases would require consideration of the cost of universal gates and how these costs can be combined to estimate the cost of the complete circuit. A specific corner case scenario has to be considered to compute the cost incurred due to the CPE approach. These are scenarios which would be of interest in evaluating the cost of the CPE approach on one of the four goals without considering the effect on the others. For example

- Is it possible to reduce area ignoring throughput reduction?
- Is it possible to reduce power at the cost of the area?

The analysis gives an idea of what the limits are in the best case scenario. These are equivalent to the situation when one of the parameters is less critical compared to others. The first analysis focuses on the cost of the CPE in term of area or power.

5.4.2 Notation and Conventions

A cost is associated with each probabilistic XOR-gate $\oplus_\varepsilon$, which is denoted by C_ε that indicates the cost function to implement an XOR-gate with error probability ε. Clearly, the cost must be a decreasing function of ε, that is $C_{\varepsilon_1} > C_{\varepsilon_2}$ for $\varepsilon_1 < \varepsilon_2$, or put differently, more reliable gates are also more expensive. For ε=0, C_0 represents the cost of the ideal (error-free) XOR-gate. Further, the cost associated with the implementation of circuit F with probabilistic XOR-gate $\oplus_\varepsilon$ is represented by $C(F_\varepsilon)$. In particular, $C(F_0)$ denotes the cost of the ideal circuit. Both the implementing circuit and the corresponding linear function are represented using F. Since F is linear, it can also be represented by a binary matrix of size $l \times k$ (l is the number of binary inputs, and k is the number of binary outputs). It is assumed that the number of XOR-gates used to implement F is proportional with the number of non-zero (1's) entries of the corresponding binary matrix. If F_ε is synthesized from θ XOR-gates $\oplus_\varepsilon$, then C(F_ε) = θC_ε.

For asymptotic consideration, increasing the size of F to ∞ means that both l and k go to ∞ while keeping the ratio l/k constant (shape of F). The LDPC decoder, denote by D, is not a linear function and cannot be implemented with XOR-gates only. However, the complexity of D depends linearly on the code-length and the number of decoding iterations. Therefore,

by expressing the cost of each gate composing D as a multiple of the cost of the XOR-gate, it may be expressed as $C(D_\varepsilon) = n\mu\delta C_\varepsilon$ $(\varepsilon \geq 0)$, where

- n is the code-length
- μ expresses the contribution of the number of decoding iterations to the total cost of the decoder. More details are as listed below:
- δ is a constant value depending on the implemented decoder

It is important to mention that the optimal number of decoding iterations increases logarithmically with the code-length. Here, optimal means that increasing the number of decoding iterations above this value should not provide any further coding gain. However, if the parameter of interest is the area, μ is equal to the number of decoding iterations instantiated in hardware. Usually, $\mu = 1$, as only one iteration is instantiated in hardware. To increase throughput, several decoding iterations may be instantiated in hardware (the decoder is said to be unrolled). In this case, one may assume that $1 \leq \mu \leq log(n)$. If the constraint of interest is power consumption (cost = power), then $\mu \cong log(n)$.

5.4.3 Cost analysis for Area/Power

The focus is now to find the conditions that guarantee the CPE approach to consuming less area (or power). Assuming,

- F to be a non-sparse matrix of size $l \times k$ - the original linear circuit
- H to be the sparse parity-check matrix of the LDPC code, with dimension $m \times n$, where m=n-k
- G to be the parity part of a generator matrix of the LDPC code. Hence, writing H= $[H_i|H_p]$, with H_i of size $m \times k$ and H_p of size $m \times m$, G can be computed as G= H_i^T . $H(\binom{T}{p})^{-1}$. Note that G is of size $k \times m$ and it is generally not a sparse matrix.
- Let P=F.G be the matrix of size $l \times m$, corresponding to the parity circuit within the CPE approach. Indeed, if the vector $i \in 0,1^l$ denotes the binary inputs of F, and o_F =$i.F$ and o_P = $i.P$ denote the binary outputs of F and P, respectively, then $[o_F \mid o_P]$ is a codeword of the LDPC code defined by H, since one has $H.[o_F|o_P]^T$=0.
- We further denote by λ_F and λ_P the fraction of non-zero (1's) entries of F and P, respectively. According to the cost linearity assumption, we may write: $C(F_\varepsilon) = \lambda_F lkC_\varepsilon$ and $C(P_\varepsilon) = \lambda_P lkC_\varepsilon$

Under these assumptions, it can be stated that the conditions to ensure the CPE to be less costly than the error-free alternative:

- In the asymmetric case, the cost of the CPE approach is less than the cost of the fault-free circuit if and only if:

$$C(F_\varepsilon)+C(P_\varepsilon)+C(D_0)<C(F_0) \tag{5.3}$$

  After some manipulations, and denoting r=k/n the coding rate, one gets:

$$C_\varepsilon < \frac{\lambda_F r-(\mu/l)\delta}{\lambda_F r+\lambda_P(1-r)}C_o \tag{5.4}$$

- In the symmetric case, the cost of the CPE approach is less than the cost of the fault-free circuit if and only if:

$$C(F_\varepsilon)+C(P_\varepsilon)+C(D_\varepsilon)<C(F_0) \tag{5.5}$$

  After some manipulations, and denoting r=k/n the coding rate, one gets:

$$C_\varepsilon < \frac{\lambda_F r}{\lambda_F r+\lambda_P(1-r)+(\mu/l)\delta}C_o \tag{5.6}$$

- In the asymptotic case, i.e., assuming that l tends to ∞ while keeping the shape of F (l/k) and the coding rate (k/n) constant, in both symmetric and asymmetric cases, we get:

$$C_\varepsilon < \frac{\lambda_F r}{\lambda_F r+\lambda_P(1-r)}C_o \tag{5.7}$$

  Since μ increases at most logarithmically with n, and thus with l. Moreover, assuming F to be a random matrix, the expected value of both λ_F and λ_P is λ_F=λ_P=1/2, which gives:

$$C_\varepsilon < r.C_o \tag{5.8}$$

The main result of interest is the condition found for the asymptotic case. From the analysis, a clear connection between the cost of a single gate and the rate of the code needed to achieve the desired performance is established. This is in line with the Shannon limit for communication systems, in the sense that the boundary of the achievable regions is fixed, once the code rate is adjusted. It is also established that a single threshold can decide if the reduction in the cost of new technology is sufficiently high to justify switching to such technology. However, the above analysis does not consider whether or not there exists a code

of rate 'r' capable of correcting (with high probability) the errors occurring at the output of F and P. In other words, the above inequality ensures that the CPE approach is cost-effective, but does not guarantee that it is also reliable (i.e., allows recovering the correct output). This issue is addressed in the next section.

5.4.4 Error Correction Capacity

Taking into account the error correction capacity of the proposed CPE approach, the above cost analysis is further elaborated. $\alpha_\varepsilon = \frac{c_{epsilon}}{c_o} \in [0,1]$ is defined as the cost reduction factor, when replacing an ideal (fault-free) technology with an error-prone one. Recall that ε is the error-probability of the faulty XOR-gate, and let σ_F and σ_P denote the error probability at the output of F and P, respectively(clearly, both σ_F and σ_P depend on ε). To simplify the analysis, the following is assumed throughout this section.

$$\sigma_F = \sigma_P = \sigma_\varepsilon \tag{5.9}$$

Hence, within the CPE approach, σ_ε represents the error probability at the decoder input. According to Eq. (4.8), the CPE approach is cost-effective if and only if:

$$r > \alpha_\varepsilon \tag{5.10}$$

Let $\sigma_\varepsilon \in [0,\frac{1}{2}]$ be maximum fraction of errors that can be corrected by a code of rate r>α_ε. According to Shannon's theorem [126], we have:

$$\alpha_\varepsilon = 1 - h(\sigma_\varepsilon) \leftrightarrow \sigma_\varepsilon = h^{-1}(1-\alpha_\varepsilon) \tag{5.11}$$

where $h(x) = -xlog_2(x) - 1 - xlog_2(1-x)$ is the binary entropy function.

Therefore, if $\sigma_\varepsilon < \sigma_\varepsilon$, then the CPE approach is not only cost-effective but also reliable, correcting the errors occurring at the output of F and P. Since F and P are linear circuits with sizes $l \times k$ and $l \times m$, any binary output of F and P can be computed by a chain of XOR-gates of length at most l. If every XOR-gate is in error with probability ε; the output error probability can be upper-bounded by (note that this upper-bound is not expected to be tight):

$$\sigma_\varepsilon < \sigma_\varepsilon^{'} = \frac{1-(1-2\varepsilon)^l}{2} \tag{5.12}$$

In particular, for $\sigma_{\varepsilon}^{'} < \sigma_{\varepsilon}^{'}$, the CPE approach is both cost effective and reliable. We also have:

$$\sigma_{\varepsilon}^{'} < \sigma_{\varepsilon}^{'} \leftrightarrow 1 - h(\frac{1-(1-2\varepsilon)^{l}}{2}) \tag{5.13}$$

For large l values, the above inequality establishes an upper-bound for the cost reduction factor, which ensures that the CPE approach to be both cost-effective and reliable. For small ε values, the CPE approach is both cost-effective and reliable for any cost-reduction factor $\alpha_{\varepsilon} \leq 1$. The intuition behind is quite apparent: for small ε, the σ_{ε} value - Error probability at the output of F - is also low, and the CPE approach may use a code with a rate close to 1 (note that coding rate = 1 corresponds to the original circuit only). As the ε value increases, the cost-reduction factor $\alpha\varepsilon$ is bounded below 1 and decreases until it gets close to 0 (i.e., the cost of the fault-prone technology must be negligible concerning the cost of the fault-free technology). The "waterfall" region (where $\alpha\varepsilon$ decreases from 1 to 0) corresponds to an increase in the value of ε by approximatively 3 to 4 orders of magnitude.

5.4.5 CPE and Modular Redundancy

The Modular Redundancy (MP) is a well-known methodology to improve reliability by mean of replication of the circuit. This can be seen as the equivalent to a repetition code where the same symbol is transmitted several times, and then a majority voting is applied. The scheme is straightforward and well understood, and it is hence interesting to compare the CPE approach to it. To compare the two architectures, it is important to consider that both schemes, CPE and MR, there are trade-offs between area and throughput. Ω-MR schemes can be implemented by instantiating Ω copies of the same circuit F, or by instantiating only ω copies and reusing each copy Ω/ω times. Similarly, an LDPC decoder can be implemented by unrolling some of all the iterations into a long pipeline or by reusing the same hardware to compute all the iterations. Some common notations used are listed below:

- A_F, A_P, A_{DEC} the area of the F, P and D blocks
  - A_{DEC}= μA_I with A_I area of single iteration
  - μ is the number of unrolled iterations
- A_{MR}^{ω} the area of the MR when ω instances of F are used
  - A_{MR}^{ω} =ωA_F
  - The corresponding latency is given by Ω/ω clock cycles
- A_{CPE}^{μ} the area of the CPE when LDPC decoder is unrolled

- $A^{\mu}_{CPE} = A_F + A_P + \mu A_I$
- The corresponding latency is given by I/μ clock cycles, where I is the total number of decoding iterations of the LDPC decoder.

Area/throughput comparison

The CPE and Ω-MR are compared regarding area and throughput without consideration of the total performance. Then, $A^{\mu}_{CPE} \leq A^{\omega}_{MR}$ if:

If better throughput is not necessary

$$\frac{A_I}{A_F} r \leq \frac{r_\omega - 1}{r_\mu} \tag{5.14}$$

If better throughput is desired

$$\frac{A_I}{A_F} r \leq \frac{\omega}{I} - \frac{1}{\mu r} \tag{5.15}$$

The following analysis can prove this:

$$A^{\mu}_{CPE} \leq A^{\omega}_{MR} \Leftrightarrow A_F + A_P + \mu A_I \leq \omega A_F \tag{5.16}$$

Considering that the area of the P circuit can be evaluated as:

$$A_P = (\frac{1}{r} - 1) A_F \tag{5.17}$$

The condition can be re-written as:

$$\mu A_I \leq (\omega - \frac{1}{r}) A_F \Leftrightarrow \frac{A_I}{A_F} \leq \frac{r\omega - 1}{r\mu} \tag{5.18}$$

Considering the throughput condition, if better throughput is desired then:

$$\frac{\Omega}{\omega} \geq \frac{I}{\mu} \Rightarrow \omega \leq \Omega \frac{\mu}{I} \tag{5.19}$$

Using the results for the previous case, we get:

$$\frac{A_I}{A_F} \leq \frac{\Omega}{I} - \frac{1}{\mu_r} \tag{5.20}$$

The analysis helps in computing the limits in the ratio between the area of iteration of the LDPC decoder and the area of the function being protected that guarantee the cost of using

the CPE scheme is less than the cost of using Ω-MR. Intuitively, it is understandable that for the small circuit it may be better to repeat the circuit rather than using an LDPC decoder.

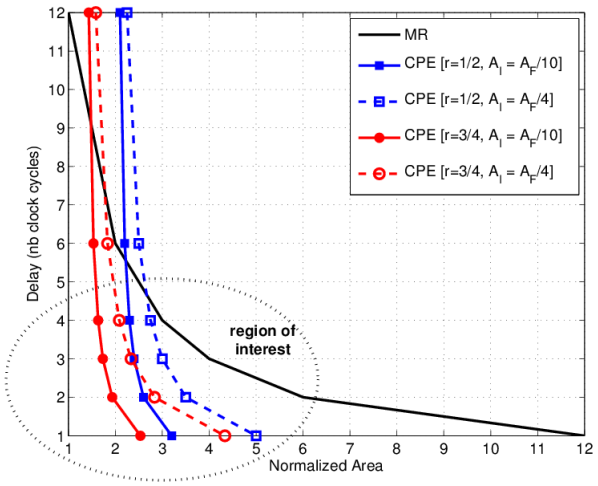


Fig. 5.9 Delay vs. normalized area for MR and CPE schemes with various unrolling factors.

Case Study: Consider an Ω-MR scheme with Ω=12 and unrolling factor (i.e., number of copies of F instantiated in hardware) $\omega \in 1, 2, 3, 4, 6, 12$. The delay $(\frac{\Omega}{\omega})$ vs. normalized area $(\frac{A_{MR}}{A_F} = \omega A_F)$ when varying the ω value is plotted in Fig. 5.9 (solid black curve, no markers). A CPE approach with I=12 decoding iterations and unrolling factor (i.e., number of decoding iterations instantiated in hardware) $\mu \in 1, 2, 3, 4, 6, 12$ is employed. Further, the rate of the LDPC code is either r=1/2 or r=3/4, and that the area of one decoding iteration is either $A_I = \frac{1}{4}A_F$ or $A_I = \frac{1}{10}A_F$. Considering two different values for A_I may reflect:

- The use of different decoders, one more powerful but having an increased area, and a second one less powerful but having a smaller area,
- An increase in the size of the circuit F under consideration, since A_I becomes negligible with respect to A_F when the size of F tends to ∞.

The delay $(\frac{I}{\mu})$ vs. normalized area $(\frac{A_{CPE}}{A_F} = \frac{1}{r} + \mu\frac{A_I}{A_F}$ when varying the μ value for the CPE approach is also plotted in Fig. 5.9 (solid/dashed, red/blue curves, corresponding to different r and $\frac{A_I}{A_F}$ values). It can be seen that the CPE approach offers more flexibility, giving the possibility to obtain a better compromise between area and delay. However, it is clear that focusing only on the area and delay is of limited value since there are no considerations of the fact that the CPE approach may also have better error protection performance.

Area/performance comparison

Next step in the analysis is to consider the performance, in term of error correction capability. It is assumed that both schemes are implemented so that they achieve the same throughput, a fully parallel Ω-MR and a fully unrolled LDPC decoder is considered. Moreover few assumptions are taken to simplify the analysis:

- Output Error Probability $\sigma_F = \sigma_P = \sigma$
- Complexity of F scales quadratically with k ($A_F = \gamma k^2$)

An asymptotic relationship between area and performances for the Ω-MR scheme is computed. The error probability of the modular redundancy scheme, denoted by σMR, is the probability of having the majority of the replica in error; the binomial combination can represent this. When the limit for Ω tends to ∞:

$$\sigma_{MR} \approx \sigma^{\frac{\Omega}{2}} \tag{5.21}$$

Put differently, in order to reach an output error probability of σMR, it is necessary to repeat the F function $\Omega \approx 2\frac{log(\sigma_{MR})}{log(\sigma)}$ times. Hence, $\frac{A_{MR}}{A_F} = \Omega \approx 2\frac{log(\sigma_{MR})}{log(\sigma)}$, which show that the area of the MR approach tends to ∞ as the target error probability σ_{MR} tends to zero.

An asymptotic relationship between area and performances for CPE approach is derived. Let r_σ denote the capacity of a BSC channel with error probability σ (i.e., maximum coding rate which can asymptotically correct any fraction of errors $\leq \sigma$). It is known that:

$$r_\sigma = 1 - h(\sigma) \tag{5.22}$$

where $h(\sigma) = -\sigma log(\sigma) - (1-\sigma)log(1-\sigma)$ is the binary entropy function. Assumption here is that a family of LDPC codes operating close to the channel capacity are available. This would mean that LDPC codes are of rate $\cong r_\sigma$ and any output error probability (arbitrarily close to zero) can be achieved when the code-length tends to ∞. The area of the CPE scheme using an LDPC code with rate, $\cong r_\sigma$ and fully unrolled decoder with I decoding iterations is given by:

$$A_{CPE} \cong \frac{1}{r_\sigma} A_F + I A_I \tag{5.23}$$

Considering now that the number of iterations required scales with the logarithm of the dimension of the LDPC code I ≈ log(k) and that the area of each iteration of the LDPC decoder is linearly dependent to the dimension of the LDPC code

$$A_I = \delta n = \delta \frac{1}{r_\sigma} k \tag{5.24}$$

We obtain:

$$A_{CPE} \cong \frac{1}{r_{sigma}} A_F + \delta k log(k) \tag{5.25}$$

Normalizing by $A_F (A_F = \gamma K^2)$, we get:

$$\frac{A_{CPE}}{A_F} = \frac{1}{r_{sigma}} + \frac{\delta}{\gamma} \frac{log(K)}{K} \tag{5.26}$$

Unlike the Ω-MR approach, the CPE approach can achieve an arbitrary small output error probability, with bounded area penalty. As k tends to ∞, we get $\frac{A_{CPE}}{A_F} = \frac{1}{r_\sigma}$, which also confirms the findings from the previous section.

5.5 CPE Simulator and CAD Automation

To implement the CPE methodology and evaluate its performance, the CPE simulator has been developed. The CPE simulator automates the process of simulating the standard data transmission over noisy channels with circuits build using faulty error prone gates. To understand the whole process, it is imperative to explain the internal proprietary format and various other core features that are part of the CPE methodology.

5.5.1 CPE Core Architecture

Fig. 5.10 shows the top level architecture of the CPE tool. The input patterns that are fed into the system are generated using the "SRC" module as depicted in the Fig. 5.10. The output of the SRC feeds into a module 'F' and a second module 'P.' These modules simulate the corresponding netlists, including the error injection at the gate level. The outputs of F and P modules is fed into the decoder module. Finally, the output from the decoder is fed into the BER/FER estimation module, where it is compared to the original input generated by the SRC module.

5.5.2 Criticality Threshold

A gate is critical when the output of the gate propagates to a large number of outputs of the circuit. Such gates result in an unusually high number of errors on the output of the circuit thereby cause decoding failures with very high probability. To identify the critical gates at the

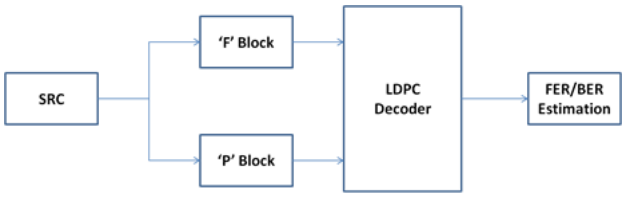


Fig. 5.10 The CPE Tool Architecture

design time, the graphical description of the netlist introduced above is used. The criticality degree of a node X, denoted by $\text{cdeg}(X)$, is defined as the number of output nodes to which at least one path connects x. Thus, injecting an error in node X may produce at most $\text{cdeg}(X)$ errors on the output. In all the simulations, the criticality threshold (CT) is fixed: nodes X with $\text{cdeg}(X) > \text{CT}$ are considered to be "protected" (e.g., by increasing area), to make then reliable. Hence, errors are injected only in nodes X with $\text{cdeg}(X) < \text{CT}$. For example, fixing CT = 5 infers that errors are injected only in those nodes that are connected to less than 5 output nodes ($\text{cdeg}(X) < 5$). A particular case is $\text{CT} = -1$, which means that all nodes are error-prone.

Critical gates have to be safeguarded from external aggressions that would toggle the output value resulting in errors. Multiple solutions to turn the critical gates into always reliable were analysed. One possibility would be to use modular redundancy for these gates which means that each gate is repeated N times (say 3 times) and a majority logic gate decides the output value. The other alternative is to define a different voltage island for all these critical gates so that they are powered up by the higher voltage.

5.5.3 Pre-Processing

The CPE simulator accepts all the input files; namely, the two encoders composing the G_{CPE} encoder, and the LDPC generator matrix. Many scripts were developed to perform the pre-processing of all the input files before they can be run through the CPE simulator. In particular, the traditional Verilog netlists that describe the encoding circuits have to be converted into the internal proprietary format that is easy to parse by the CPE simulation engine. As a key point to understand the whole process, the internal netlist proprietary format is first described.

Table 5.1 CPE Netlist Representation

```
14 14 775,
4 2 4 3 2
4 711 712 713 714,
2 252 253,
.....
.....
0 185
```

5.5.4 Netlist Format

Any gate level synthesized representation of the digital circuit is represented as a collection of gates commonly referred to a netlist. Since CPE simulator is a tool completely developed in C++, it cannot understand the terminology of gates. Hence, an internal proprietary format is introduced to represent the Verilog netlists. Each gate in the circuit is converted into a node within the internal format. The following convention is used to represent all possible gates: 0 = NOT, 1 = AND, 2 = OR, 3 = XOR, 4 = NAND, 5 = NOR, 6 = XNOR. A node X is called a predecessor of Y if the output of X is an input of Y (in graph terminology, there is a directed edge from X to Y). In this case, Y is said to be a successor of X.The in-degree of a node is defined as the number of its predecessors input nodes must have indegree equal to zero The outdegree of a node is represented as the number of its successors; output nodes must have outdegree equal to zero. In simple circuit design terminology, in degree and outdegree correspond to fan-in and fan-out of the gates. Nodes of type 0 must be of indegree = 1. Nodes of type 2-6 must be of indegree ≥ 2. Tab. 5.1 depicts a snapshot of CPE netlist representation format:

- Line1: $N_{inputs}N_{outputs}N_{internal}$
- Line2: Ordered list of indegrees: must contain $N_{internal}+N_{outputs}$ values, corresponding to the indegrees of internal and output nodes (in order)
- Line3: List of predecessors of the node N_{inputs}
- .....
- Line#: List of predecessors of the node $N_{inputs}+N_{outputs}-1$
- Line#: List of predecessors of the node $N_{inputs}+N_{outputs}$
- .....

- Line#: List of predecessors of the node $N_{inputs} + N_{outputs} + N_{internal} - 1$

Many consistency checks are performed to ensure that the netlists adhere to the syntax.

- Check that input nodes (numbered from $0 to N_{inputs} - 1$) have indegree zero.
- Check that output nodes (numbered from $N_{inputs} to N_{inputs} + N_{outputs} - 1$) have outdegree zero.
- Determine the processing order of internal and output nodes, i.e. which nodes must be processed in the first stage, which nodes must be processed in the second stage, etc.
  - Nodes processed in the first stage are those whose all predecessors are input nodes.
  - Nodes processed in stage l (l=2) are those whose predecessors are either input nodes or have been processed during stages 1, ..., l-1.
  - If a processing order cannot find consistency check error.

5.5.5 CPE Simulator

The CPE simulator is the framework developed to automate the process of simulating the standard MCNC benchmark circuits as well as linear circuits. Fig. 5.11 describes the complete CAD flow of the tool. It accepts all the input files: the combinational circuit netlist called 'F', the parity circuit netlist called 'P', and the LDPC generator matrix. The error is propagated through the netlist to simulate the final the Bit Error Rate(BER) and Frame Error Rate(FER) values. For simplicity purpose, the same value of probability is used for all the gates, irrespective of their type or their position within the graph. The methodology employed to insert faults is called "Gate output probabilistic mutant"- it alters the gate output with a given probability. Some gates of the circuit may be critical, in the sense that injecting only one error at the output of such gates may result in a substantial number of errors at the output of the circuit.

Based on the codeword length and the LDPC rate, the Generator matrices are generated which are then converted into the internal format which is easily understood by the CPE simulator. In the CPE simulator, a source module generates the pseudo-random input vectors. Errors are injected by flipping the output of each gate with a predefined error probability. The output of the source module feeds into the two encoder modules which propagate the faults through the circuits. The output of the encoders along with original data bits is used by the D_{CPE} decoder module to generate the codeword that can be transmitted over the

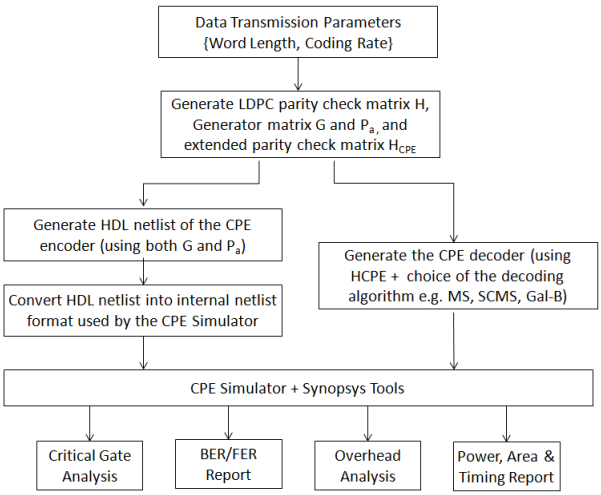


Fig. 5.11 The CPE CAD flow

channel. Finally, an output from the decoder drives the BER/FER estimation module where it is compared to the original input generated by the SRC module.

5.6 Experimental Results

Many encoders have been simulated using the CPE simulator employing different decoding algorithms. Two scenarios are under investigation

- Both encoder and decoder are assumed to be error-prone.
- Only the encoder block is faulty.

The first scenario is the most generic and considers both encoder and decoder to be fault-prone. Given such situation, only LDPC codes are usable for the coding schemes, the only known ECC for which fault-tolerant decoder exists. But, the perfect decoder case that presents an evident asymmetry between the encoder and the decoder is also considered. This allows one to use coding schemes other than Low-Density Parity-Check Codes as ECC codes are not available for a faulty decoder. From the encoder point of view, four different configurations are employed. Thus, regular LDPC codes are considered with different column weights for the parity check matrix, namely $d_v = 3$ and $d_v = 4$, and different coding rates, namely $r = 1/2$ and $r = 3/4$. From decoder perspective, three state-of-the-art reliability enhanced LDPC decoders are employed within the CPE CAD flow: Min-Sum (MS), Self-Corrected Min-Sum (SCMS), and Gallager B with Extended Alphabet(Gal-B). Next, simulation results of the CPE scheme for different scenarios are presented. The focus is mainly on the symmetric scenario; namely, both the encoder and the decoder are faulty.

5.6.1 Critical Nodes

Critical nodes are the ones which generate the maximum number of errors on the output nodes of the encoder. As defined in the previous section, criticality degree of a gate is defined as the number of erroneous outputs generated when the gate is in error (assuming that all the other gates are error-free). Tab. 5.2 lists the count of total and critical nodes within the four encoding schemes. The encoding scheme parameters are given in the first column while column two provides the count of total nodes within the G_{CPE} encoder. The following three columns list the count of critical nodes when critical threshold CT is set to 10, 20, and 50 respectively.

Table 5.2 Critical Gate count for different encoding schemes

Encoder	G_{CPE} Node Count	CT=10	CT=20	CT=50
dv3-r12	44399	3373	1844	833
dv3-r34	28182	2288	1240	537
dv4-r12	45175	3424	1851	824
dv4-r34	27167	2112	1183	488

As expected, lower the value of the critical threshold, higher the number of critical nodes within the encoder. As a trade-off, a lower critical threshold is also expected to lead to lower encoding error probability. To illustrate this, an encoder with $r = 3/4$ and $d_v = 4$ and D_{CPE} was set to Min-Sum model [127] is employed. As depicted in Fig. 5.12, the output BER value reduces the critical threshold values. It infers that more the number of nodes safeguarded from possible soft errors, higher the possibility of retrieving the original information.

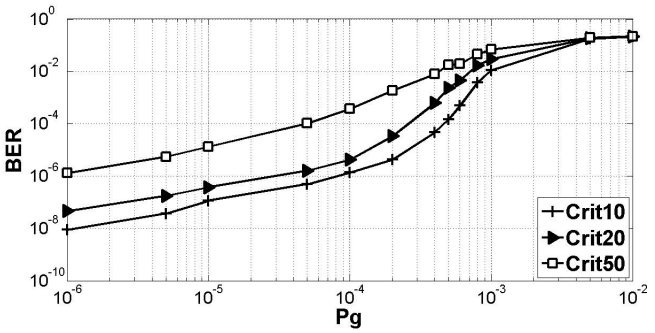


Fig. 5.12 Critical Threshold impact on Output BER

5.6.2 Impact of Decoder Configuration

The reliability enhanced LDPC decoders are benchmarked based on their performance as well as their ability to deal with the circuit fault-induced probabilistic behavior effectively. For faulty decoders, it is assumed that the output of every variable and check node function computation is flipped with a probability $p = 10^{-3}$. For non-binary message alphabets, flipping the output value means that a value different from the correct one is selected uniformly at random from the alphabet. Fig. 5.13 highlights the output BER values for different faulty decoders and the default non-CPE approach when the critical threshold is set to 20. The encoder employed in this particular case has the following parameters $r = 3/4$ and $d_v = 3$. Self-Correcting Min-Sum(SCMS) and Min-Sum(MS) [127] decoders provide the best performance by reducing the error rates go up to 10K times better than the default encoder. Gal-B provides up to 100 times improvement concerning error correction.

It also illustrates the performance of CPE compared to the default encoding mechanism. For example, for CT $= 10$ and for a gate error probability P_g = $1e^{-4}$, the BER of CPE is $5.83e^{-8}$, while the BER of the encoder without protection is $5.54e^{-3}$. This represents a significant improvement, by more than five orders of magnitude. Furthermore, by injecting errors only on non-critical gates, the performance of CPE fared much better than the default encoding mechanism.

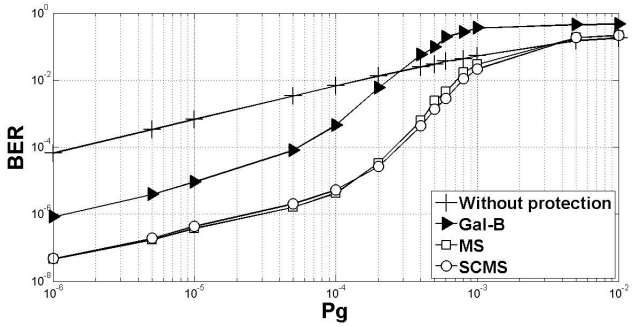


Fig. 5.13 Decoder Configuration impact on Output BER

For the encoder with $d_v = 4$ and $r = 1/2$, by setting the CT value to 10, it is possible to provide fault free information when the gate error rate is less than 10^{-4}. As depicted in Fig. 5.14, the CPE mechanism provides error-free output for gate errors smaller than P_g = $6e^{-4}$. MS decoding scheme is employed and adopted an encoder with $r = 1/2$ and $d_v = 4$ for achieving this kind of performance. For the sake of completion, Fig. 5.15 illustrates the similar scenario is assuming a perfect decoder. In such case, the CPE performance improves marginally as compared to employing faulty decoder.

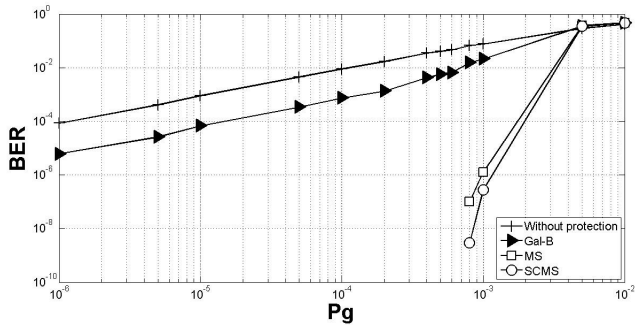


Fig. 5.14 CPE error free scenario employing faulty decoder

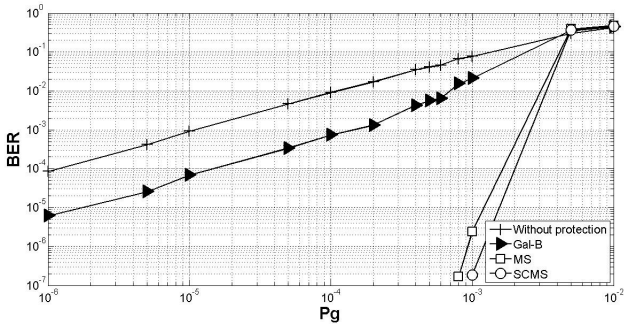


Fig. 5.15 CPE error free scenario employing perfect decoder

5.7 CPE for Fault Prone Boolean Functions

The proposed CPE approach has been extended to nonlinear circuits in this section. The idea is to improve fault tolerance using methods derived from Error Control Coding (ECC) theory to protect the combinational logic that implements a particular Boolean Function. The focus of this approach is still not on modifying the combinational logic but on augmenting extra combinational logic to enable the retrieval of the correct output even if errors have occurred just like in the case of linear encoding approach.

5.7.1 Encoding Mechanism

To understand how the new combinatorial logic is derived, A new function computed by E $\varepsilon(.)$ is defined. In case of working with binary codes, then ε:GF(2)^k $\rightarrow$GF(2)^n, where k is the dimension and n is the length of the code C. The number of outputs of the combinatorial circuit *CC* is also assumed to be equal to k. The functionality of *CC* can be described as a function F(.) with F: GF(2)^l $\rightarrow$ GF(2)^k where I is the number of binary inputs, and k is the number of binary outputs. The function P(.) that maps the inputs directly to the parity is P:

$GF(2)^l \rightarrow GF(2)^m$ and is the composition E(F.). This can be seen merely by considering how its result must be equivalent to computing F(.) and then E(.) on the results.

It is evident that even if the operation is equivalent to serially concatenating the two blocks, it computes the parity on an independent path than the original combinatorial logic, and hence it does not suffer from the fault propagation scenario discussed above. In a sense, the function P(.) predicts the parity of the outputs o from the inputs i as if a standards encoder were implemented with o as inputs. For all linear FEC codes, the encoding process can be expressed as a matrix multiplication of the input message and the generator matrix G, the composition E(F.) is equivalent to:

$$P_j = \sum_{b=1}^{k} F_{b(j)} G(b, j) \, j \in 0, ..., m \tag{5.27}$$

The combinatorial logic that computes the parity is a linear combination of the various functions that calculate the output bits, as such the resulting logic may be costly, both with regards to the area consumption and regarding the critical path.

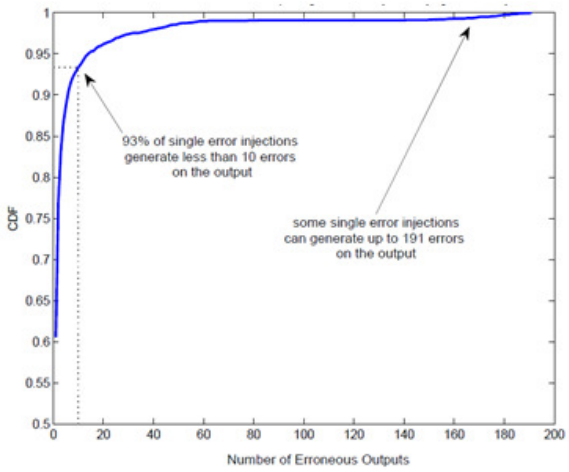
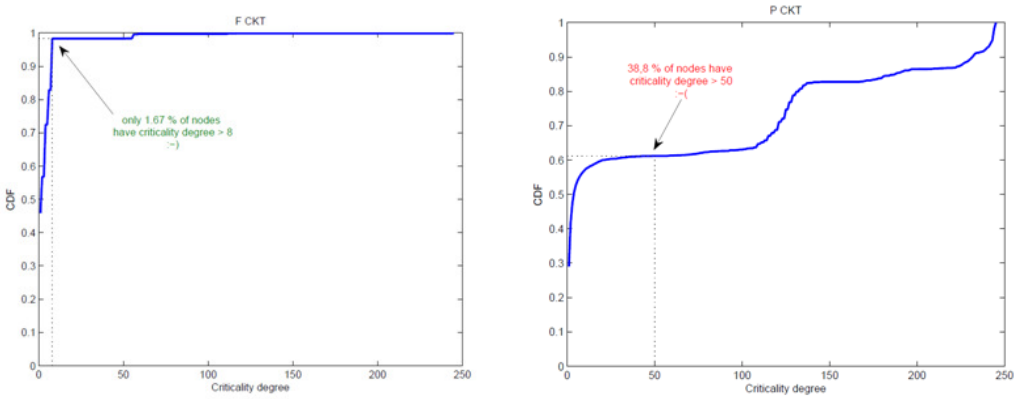


Fig. 5.16 CDF of the number of erroneous outputs generated by one single error injection

5.8 Experimental Results

Many benchmark circuits have been simulated using the CPE CAD setup for different constraints. It should be mentioned here that all the simulation results presented in this section assume that the decoder is run on reliable hardware (error-free hardware). This is referred to as asymmetric setting wherein the circuit and the corresponding augmented logic is error-prone, and the decoder is assumed to be fault free.



(a) F circuit Criticality

(b) P circuit Criticality

Fig. 5.17 CDF plot of criticality degree

5.8.1 Critical Nodes

Only one error is injected into the circuit (at the output of one single gate) and counted the number of errors generated on the output of the circuit. For a randomly generated logic circuit, this behavior is illustrated in Fig. 5.16, which plots the CDF data of the number of errors on F-output generated by one single error injection. It turns out that for about 93% of gates, injecting an error generates less than ten errors on the output which is acceptable. However, for some gates, the error injection can generate up to 191 errors in the output. Hence, it can be concluded that decoder failures are due to a considerable number of errors on the output of F (or P) even if the gate error probability is quite low.

Further, the criticality degrees of all nodes in F and P are computed. As depicted in Fig. 5.17(a), only 1.67% of the nodes of F have a criticality degree > 8. However, for P the situation is entirely different as shown in Fig. 5.17(b). There are many nodes with very high criticality degree (38.8% of nodes have criticality degree > 50). All these nodes cannot be protected due to high cost. The trade-off here is increasing the number of nodes of P may allow decreasing their criticality level (more nodes, but less critical). On the contrary, decreasing the number of nodes in P could increase their criticality level (fewer nodes, but more critical).

As an important point, the number of critical nodes is currently more significant problem within the nonlinear circuits compared to linear circuits. As depicted in Fig. 5.18, the number of critical nodes is pretty huge in the non-linear circuits. The parity circuit often dominates the total count of critical nodes. This is because of the current methodology used to generate the augmented logic circuit (F) circuit. Concatenate the functional unit with the generator matrix which is not the best possible approach in the case of non-linear circuits.

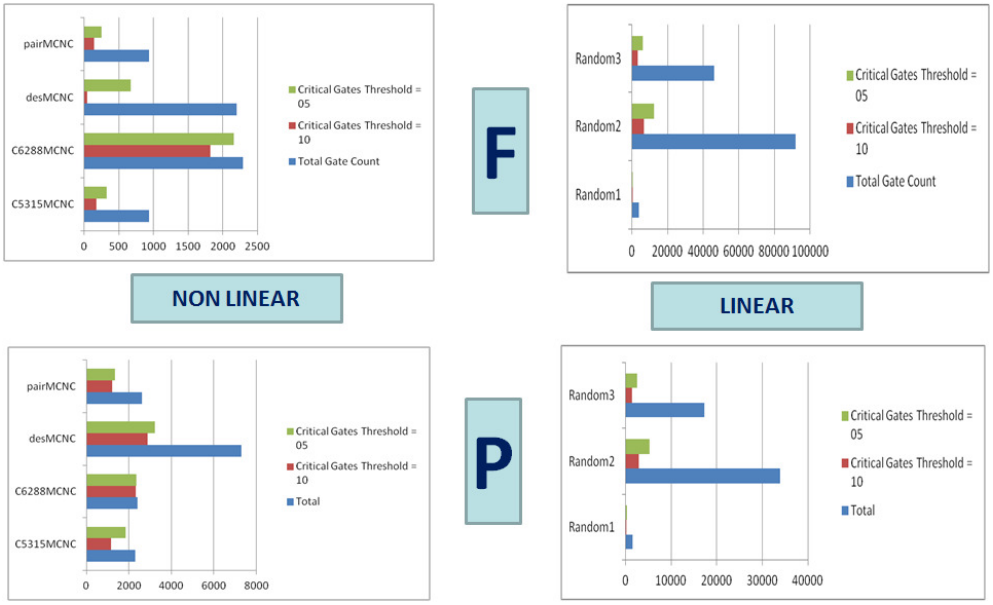


Fig. 5.18 Critical node count for different Linear and Non-Linear circuits.

5.8.2 Area Overhead

Any error correction technique comes with the extra overhead of the parity circuit that needs to augment to the existing logic circuit. As shown in Fig. 5.19, for linear circuits, the size of the parity circuits is much smaller. This is because two matrices are multiplied to generate the new circuit. But, in the case of non-linear circuits, the overhead is slightly on the higher side.

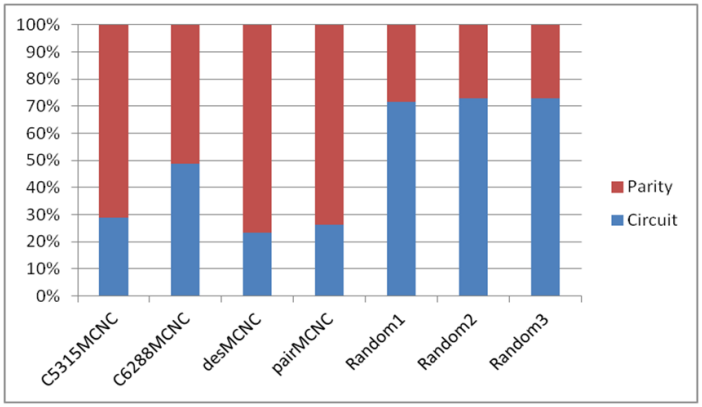


Fig. 5.19 Area overhead due to parity augmentation.

5.8.3 NMR Vs. CPE

Fig. 5.20 plot portrays the bit error probability of the CPE and TMR methodologies for the MCNC benchmark circuit 'DES'. In the simulator, all the gates whose failure generates more than ten errors on the output of F or P are declared as critical. By injecting errors only on non-critical gates, the performance of CPE fared much better than TMR. For a gate error probability of 0.001, the bit error probability of CPE = 4e-9, while the bit error probability of TMR = 6.3e-4. This represents a significant improvement, by more than five orders of magnitude. Note that to achieve a bit error probability of 4e-9 using N-modular redundancy, one should take N = 11.

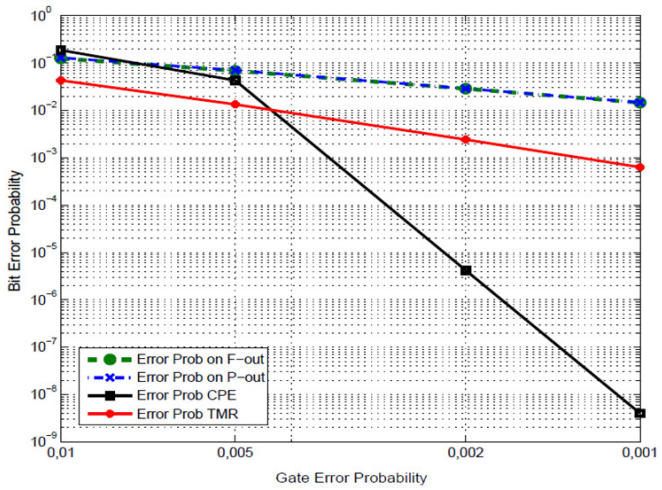


Fig. 5.20 Performance of NMR Vs CPE

5.8.4 Impact of LDPC code sizes on Area

This section presents several applications of this encoding technique to two IP cores. The use of different ECC codes is investigated. For reference, the results are compared with the original size/timing and with the Triple Modular Redundancy (TMR) scheme. The size of the fault-tolerant CPE implementation is the sum of the combinational circuit ' F', the parity circuit 'P' and the decoder 'D'.

Two IP cores commonly used in telecommunication systems, the Scrambler and the Chien Search block for Reed Solomon decoding were considered. Several codes are investigated to span a large variety of possibilities concerning code length, error correction capability, encoding/decoding complexity, etc. These include Hamming code, BCH codes, Low-Density Parity Check (LDPC) codes and the Low-Density Generator Matrix (LDGM) codes. Area and Delay results for the application of the proposed fault protection scheme for the Scrambler

and Chien Search cores are as shown in Fig. 5.21. It can be seen how the implementation of the CPE protection can have a significant impact in term of area and delay. It is also evident how the cost of the scheme is dependent on the combinatorial logic to be secured.

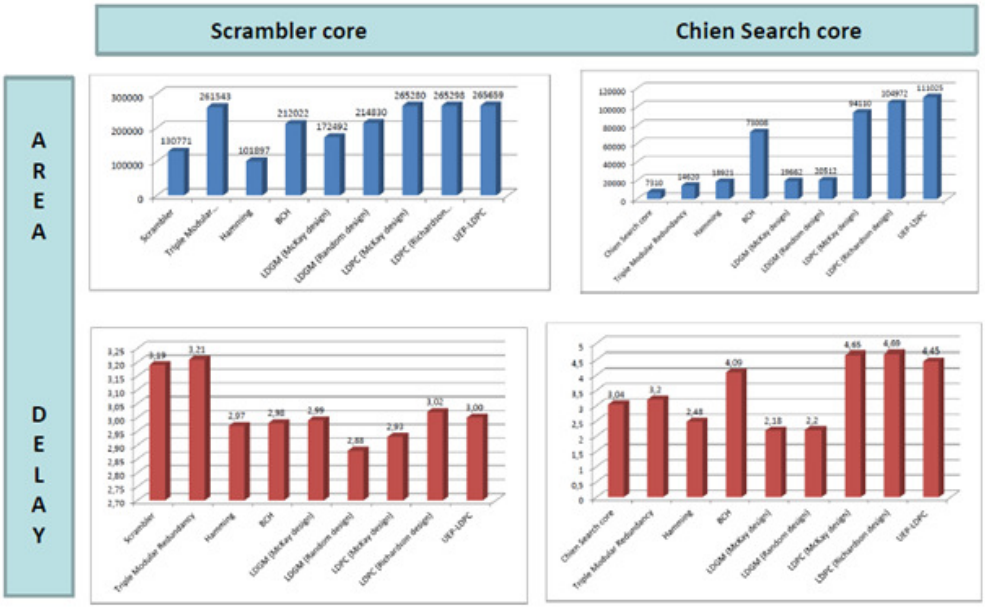


Fig. 5.21 Area and timing Analysis on IP cores using CPE methodology.

5.8.5 Case Study

MCNC benchmark circuit "DES" has been selected as the default test case to perform various simulations to study the methodology. The reliability of the CPE approach depends on the CT value. However, even if we consider a CT value as high as CT = 10, the number of nodes to be protected in P (5524) is still higher than the total number of nodes of F (3249). Thus, CPE under such circumstances is not the right solution, because a more efficient and straightforward solution would be to protect all the nodes in F.

Fig. 5.22 depicts the output error probability plot. When CT is set to -1, all nodes are possibly error-prone, most of the CPE output bit errors are because the decoder converges to a wrong codeword. But with CT= 2, 5, 8, 10; the decoder never converged to a wrong codeword. Fig. 5.23 gives a detailed analysis of the output error on all the three output nodes; the 'F' circuit, the 'P' circuit as well as the decoder.

Besides, one other limitation is the extremely gate count of the parity circuit (P). For the current test case, there are a total of '3249' nodes in the circuit and '41151' nodes in the corresponding parity circuit (P). This results in an increased error probability on output nodes of parity circuit as compared to the logic circuit (F). This also leads in an area significantly

larger which is not desirable. Size of P must be reduced considerably, for CPE to provide a practical solution.

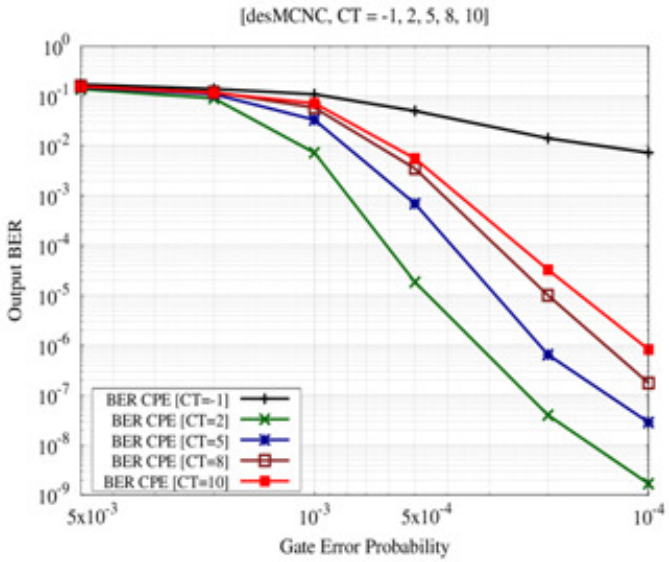


Fig. 5.22 Output BER for various Criticality thresholds

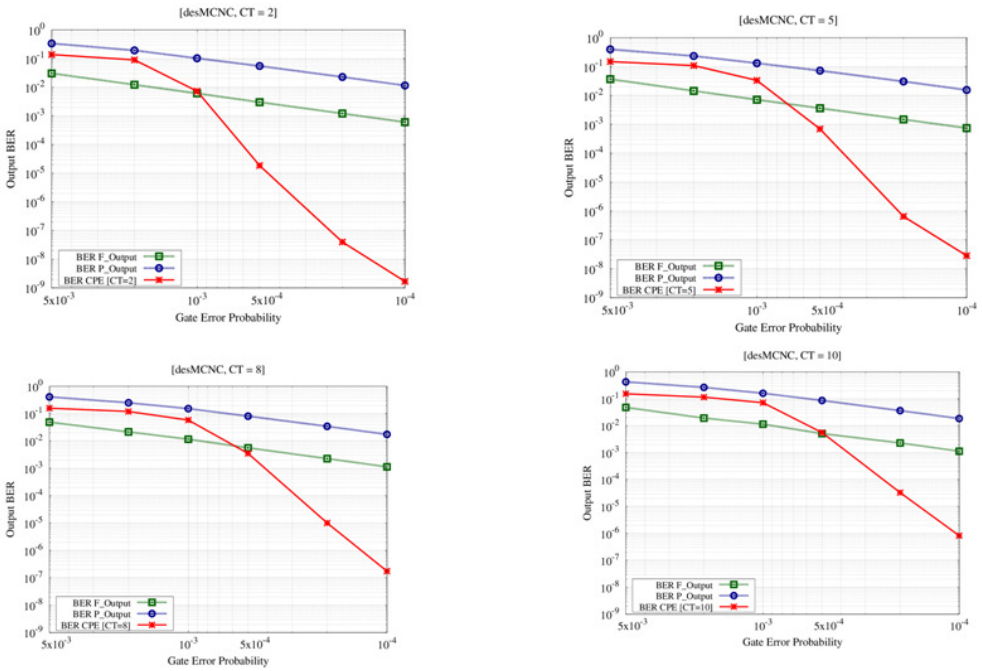


Fig. 5.23 Detailed plots for output error on F, P and decoder output nodes.

5.9 Conclusions

A novel fault tolerant methodology known as Codeword Prediction Encoder (CPE) for reliable data transmission using unreliable hardware is proposed. The principle idea is to implement error correction codes driven graph augmentation techniques applicable to any combinatorial logic individually to correct dynamic errors introduced during the encoding process. The mathematical analysis presented reflects only a preliminary investigation of the various possible avenues to reach mathematical formulation of what are the limitation and the potential trade-off of the CPE technique.

The CAD flow for CPE methodology is implemented, and performance evaluation has been entirely automated. Simulation results show that performance of CPE is much better as compared to transmitting data by employing traditional encoding methodology. A replication of up to '11' times of the combinational circuit to achieve similar kind of performance as the CPE. Performance evaluation using various fault tolerant LDPC decoders were discussed. By employing Min-sum decoding mechanisms and a strong encoder $r = 1/2$ and $d_v = 4$, it is possible to correct all errors given that gate errors smaller than $P_g = 6e^{-4}$. In general, CPE performance improvement of upto 10K is observed when compared to the normal encoding mechanism. Apart from this, some other important sets of conclusions are for any LDPC code rate and for any critical gate threshold:

- With lower code rate, we achieve a significant amount of performance improvement wherein the CPE can correct all errors for gate error probability of 1e-2.
- Lower the code rate, higher the number of gates in the parity circuit 'P'.
- Number of critical gates is not rising drastically with code rate.

In this chapter, A new technique called CPE based on error control coding techniques are presented which improve the circuit reliability by redundancy addition into the circuit hence altering the number of inputs/outputs. One of the issues encountered with CPE was the difficulty of generating the error probability model for the outputs of the circuit and also the encoder. This is primarily due to the fact that the output functions are not completely independent. The coding scheme has to take into consideration these models for more effective decoding. In the next chapter we propose some new models which are based on delay degradation analysis.

Chapter 6

Delay Degradation Analysis

6.1 Introduction

In the early days of IC design(the early 1970s), the full custom methodology was adopted wherein the layout of each transistor, and the interconnections between them were laid out manually. However, with the development of sophisticated Electronic Design Automation (EDA) tools, the design of the state-of-the-art complex ICs was made possible by adopting semi-custom VLSI design flow. In this flow, a set of predefined basic gates have already been designed and stored in the library as a part of the current design. With semi-custom flow, there may be a slight degradation in performance compared to an equivalent full custom VLSI chip. But the minimization of the design cycle time due to re-usability of components from the library makes it outperform its full custom counterpart. Offlate, full custom designs are limited to performance-critical applications and in special cases when a new technology library is adopted for designing a circuit.

The most critical aspect of semi-custom ASIC design flow is Static Timing Analysis(STA). STA is the process of computing the expected timing of a digital circuit without requiring a full circuit simulation. With the introduction of sub-threshold logic [128], the CMOS circuits supply voltage value has been pushed to an extremely low level, which is below the MOSFET threshold voltage. This dramatically reduces the power consumption but aggravates the process variations thereby compromising circuit functional performance [129]. Unpredictable gate delay values induced by local variations [130] along with other variations like CMOS channel length, threshold voltage, and oxide thickness are some of the critical factors that complicate timing analysis and estimation [131]. Hence, accurate delay analysis and estimation are turning out to be even more challenging, and the capabilities of conventional delay models and timing analysis approaches are proving to be inadequate.

6.1.1 Main Contributions and Outline

In this work, delay approximation methodology based on probability density functions(PDF) based on Inverse Gaussian function is proposed [132] [23]. This approach is significantly faster than the state-of-the-art since only the essential cells must be fully simulated to obtain the key model parameters and the delay model for complex circuits. Unlike other techniques or tools, which demand large look-up tables or complicated calculations [85], [86], the proposed approach is remarkably more straightforward. For the simulation set-up, process variations and voltage variations are investigated during this key-parameters estimation step. The room temperature is chosen for all the simulations, and temperature variation is not taken into consideration due to its less impact compared with the two parameters on delay degradation [128].

The primitive version of the IGD based delay model was first presented in [23] applicable to combinational elements in digital CMOS circuits to compute critical parameters of the model and was later extended to sequential circuits in [132]. The main idea behind the proposal is first to calculate the primary gate critical parameters utilizing MC simulations and then linearly extrapolate (propagate) them through the logic network at the circuit level. In the refined approach, the effect of fan-out value and input transition time on the gate delay is also taken into consideration. The proposed IGD model is endorsed by physical phenomena and provides considerable delay estimation accuracy and efficiency. Furthermore, the model is highly accurate even for different power supply voltage values ranging from nominal V_{dd} to sub-threshold V_{dd}. To demonstrate the practicability of the IGD based statistical approach, the estimated delay with the current model and MC SPICE simulations for several combinational and sequential logic blocks were compared. 32nm technology models were employed across all the simulations. Experimental results indicate that the proposed method outperforms GD fitting and provides a very close match with MC simulations, i.e., less than 1.2% and 1.9% error for the two circuits, 8-bit Ripple Carry Adder(RCA), and 8-bit De-Multiplexer(DEMUX) and Multiplexer(MUX) respectively.

6.2 Library Cells Simulation Methodology

In semi-custom VLSI design flow, any design is implemented using a set of predefined basic gates as the fundamental building blocks. This creates a need for accurate abstraction of each building block (gate or cell) so that other tools that use these blocks can predict the behavior of the design. Those aspects of the library cell behavior are abstracted that appear as constraints to the designer. Traditionally, all the cells in the library database are characterized by metrics like timing, power, and noise. Modeling and predicting propagation

delay or switching speed of CMOS logic gates is one of the major obstacles in VLSI design cycle. Microscopic simulation provides accurate results for propagation delays. However, simulations take a significantly long time to calculate. The Liberty Format [133], an open source ASCII library format was first introduced in 1987 for logic synthesis, and it has been refined continuously since. In 1988 a linear CMOS timing model was introduced, followed in 1992 by the introduction of the Non Linear Delay Model(NLDM) [134] [135]. The new Composite Current Source(CCS) model [133] allows for accurate analysis of sub-micron designs.

6.2.1 Library Characterization

Library characterization is the process of exhaustively analyzing an entity at a low level of abstraction to extract all relevant and meaningful information about it, and then to faithfully represent that information in a model at a higher level of abstraction. Cell characterization is the foundation on which the entire high-level RTL-to-GDSII flow has been built. This data is used in STA to compute timing delays on the critical timing paths that determine the maximum clock frequency at which the chip can safely operate thereby guaranteeing the correct chip functionality post fabrication [136]. Design Closure is a part of the chip design by which an Integrated Circuit(IC) design is modified from its initial description to meet a growing list of design constraints and objectives(timing, power, noise, test, and reliability). Many effects are showing up at 65nm and below that can no longer be ignored or simply handled by margining. A great deal of work has been performed on perfecting the timing models for static timing analyses [137] [138], [139], [140], [141] [142], [143]. Composite Current Source(CCS) [144], [145] modeling extends Liberty to include current waveform data which allows for more accurate analysis and unification of library data. CCS model formats, as part of the Liberty standard, are an open and unified model for timing, noise, and power. CCS addresses the analysis of these effects in today smaller technologies, effects such as high impedance nets, double switching and miller effects.

6.2.2 Timing Verification

Timing verification is a process of validating that a design meets its specifications by operating at a specific clock frequency without errors caused by a signal arriving too soon or too late. Transistor-level static timing technology has been available for well over a decade. Dynamic timing analysis can achieve more accurate estimates of path delays than static timing analysis because a set of input patterns is simulated to exercise all the paths in the circuit, as described in [89]. However, the computational complexity of generating such input patterns can be

very high, and the simulation is time-consuming for large circuits. Unlike the dynamic simulation approach, STA tools remove the need for simulating the entire block under all possible scenarios. Instead, they use fast but accurate approaches to estimate the delay of sub-circuits within the block and use graph analysis techniques to quickly seek out the slowest and fastest paths in the block. The result is that an STA tool can typically find all timing violations in a block in a fraction of the time it would take a dynamic circuit simulator. It is essential to reaching timing closure for digital designs by identifying paths that are limiting chip performance.

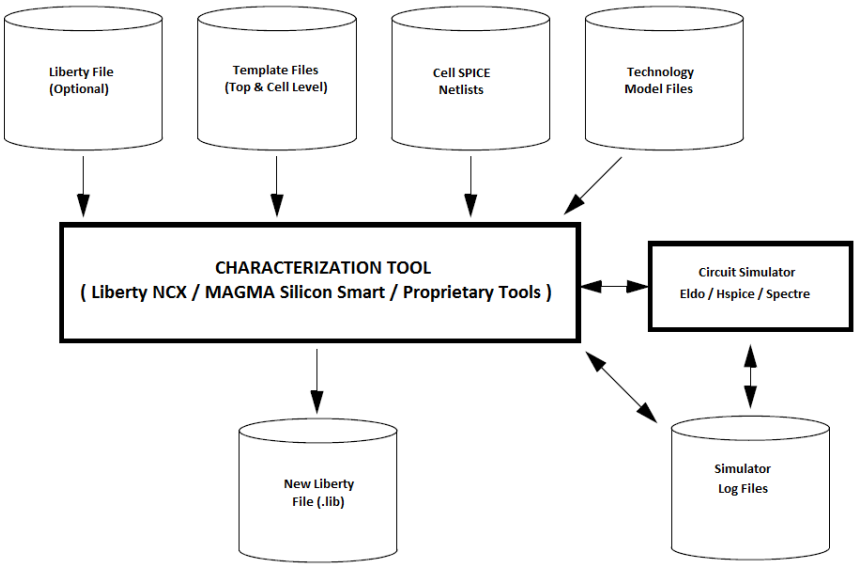


Fig. 6.1 Library Characterization Flow

6.2.3 Simulation Methodology

The basic cells must be fully simulated to obtain the key model parameters and capture the delay model for complex circuits. The underlying simulation methodology adopted is aligned with the industry standard library characterization flow as depicted in Fig. 6.1. The only difference is the fact that SPICE simulator is directly used instead of the licensed library characterization tool. Also, custom scripts are employed to gather useful data from the report files dumped by SPICE.

6.3 Linear Compositional Delay Model

In probability theory, Inverse Gaussian Distribution(IGD) also known as Wald distribution is a two-parameter family of continuous probability distributions [146]. The distribution

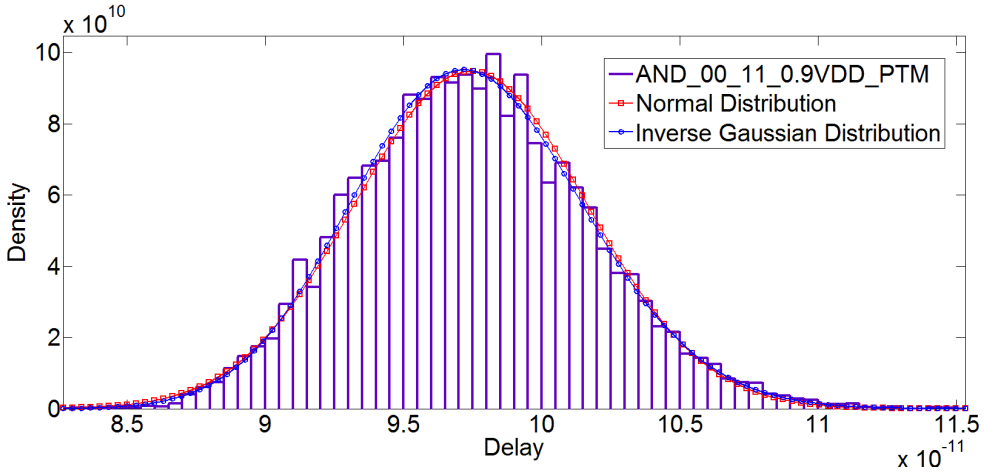
support is [0, ∞] and it can be symmetric or asymmetric around μ. Its probability density function IGD (μ, λ) expressed in Eq. 6.1 where μ is the mean and λ the shape parameter, can overcome both the shortfalls. As λ tends to ∞, the inverse Gaussian distribution becomes more like the Gaussian distribution. A comprehensive delay approximation methodology based on Inverse Gaussian Distribution (IGD) is proposed and compared with other related work namely, the Gaussian Distribution (GD) delay model [87]. To demonstrate the accuracy and flexibility of the IGD based approach, both conventional V_{dd} and near/sub-threshold V_{dd} values in conjunction with several types of process variations are considered.

$$f(x,\lambda,\mu) = \Big[\frac{\lambda}{2\pi x^3}\Big]^{\frac{1}{2}} exp\Big(\frac{-\lambda(x-\mu)^2)}{2\mu^2 x}\Big); x > 0 \tag{6.1}$$

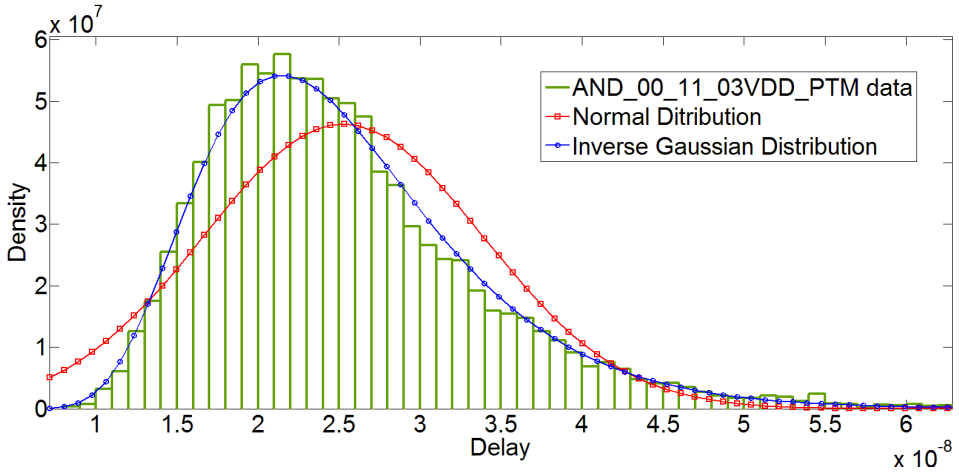
There is an intuitive reason why IGD fits better than GD with CMOS delay propagation data under various PVT variations. Brownian motion can well model the random movements of the carrier particles in electronic circuits in a steady state also called as Wiener process [147]. For particles under Brownian motion, GD captures the motion distribution of all particles at a given moment in time, while IGD reflects the particle motion when drift is applied. IGD provides the number of particles in random motion with a positive drift that reaches a fixed level in a given period. In electronic devices, the drift is the voltage difference between device terminals producing an electric field thus inducing carrier movements. It is observed that the IGD shape can change significantly depending on the two parameters and is not restricted like the GD approximation which must always be symmetric. Based on this argument, it can be concluded that the inverse Gaussian distribution is potentially better suited than the normal distribution to represent delay distributions in electronic circuits.

6.3.1 Flexibility (universality) of IGD model

SPICE simulations were performed on 2-input AND gate with threshold voltage variation (employing Gaussian distribution) being the most dominant element of all process variations. 32nm Predictive Transistor Models (PTM) under a nominal supply voltage of 0.9V was employed for the Monte Carlo simulations. To compare the IGD model with the GD model, similar experiment, i.e., a 2-input AND gate with inputs switching from 00 to 11, has been reproduced. The threshold voltage (V_{th}) variation is generated following the GD, where the mean value is the nominal V_{th}, V_{thn}=0.322V for nFETs and V_{thp}=-0.302V for pFETs. The standard deviation is set to 50mV, which is sufficient to reflect the threshold voltage variation in state of the art circuits. Both GD and IGD are used to fit the propagation delay data profile. The resulting delay histogram and the corresponding GD and IGD fittings are presented in Fig. 6.2(a) which indicates that both fit well the delay data. However, the similarity in



(a) Normal V_{dd}



(b) Sub-Threshold V_{dd}

Fig. 6.2 IGD and GD fittings for 2 Input AND gate

fitting capability no longer holds true, when the gate operates in the near-threshold regime. A similar experiment is repeated for the same V_{th} distribution with a V_{dd} of 0.3V and Fig. 6.2(b) presents the delay histogram and the GD and IGD fittings. It is clear that IGD almost perfectly fits the delay histogram which has a non-symmetrical shape with a steep slope towards zero and a long tail towards ∞. On the other hand, unlike in Fig. 6.2(a), GD does not provide a proper fitting in this case. It is also worth mentioning that the GD fitting curve does not start from zero, which is unrealistic because no circuit can operate without propagation delay. The experiment highlights the shortfall of using GD for fitting delay data.

To further demonstrate the IGD fitting accuracy, a chain of 5 AND gates in which all AND gates, except for the first one, are fed by the output of the previous gate is considered to demonstrate the IGD fitting accuracy further. The chain has been simulated with its primary inputs switching from 11 to 00 but employing similar process variations as in the previous experiment. A standard deviation of 50mV in the power supply voltage is assumed at 0.9V V_{dd}, which reflects real circuit power supply voltage fluctuations. In Fig. 6.3, the histogram and their corresponding fittings for the 3rd and 5th AND gates are presented (the other stages are omitted for clarity).

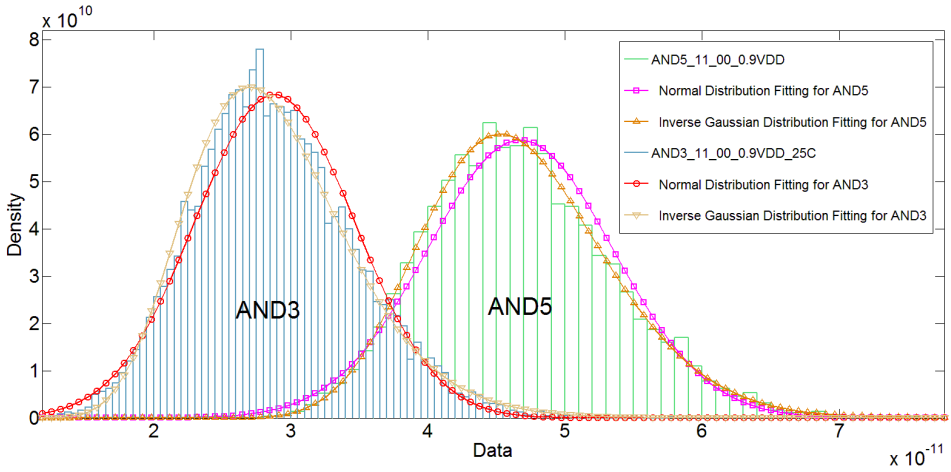


Fig. 6.3 Seven AND gate chain

One specific case of interest is to consider voltage variation alone. The current nanometer technologies present up to 20% of voltage variation, thereby changing the expected circuit performance drastically. Fig. 6.4 depicts the simulations results and mapping of both IGD and GD delay models wrt Monte Carlo simulation results. It proves that IGD is better compared to GD to model the delay degradation taking the impact of voltage variations alone. Thus, it is evident that IGD fits better the experimental data than the normal GD. Based on the

simulations, it can be concluded that IGD can accurately capture gate and simple circuit propagation delays for different process and voltage variations.

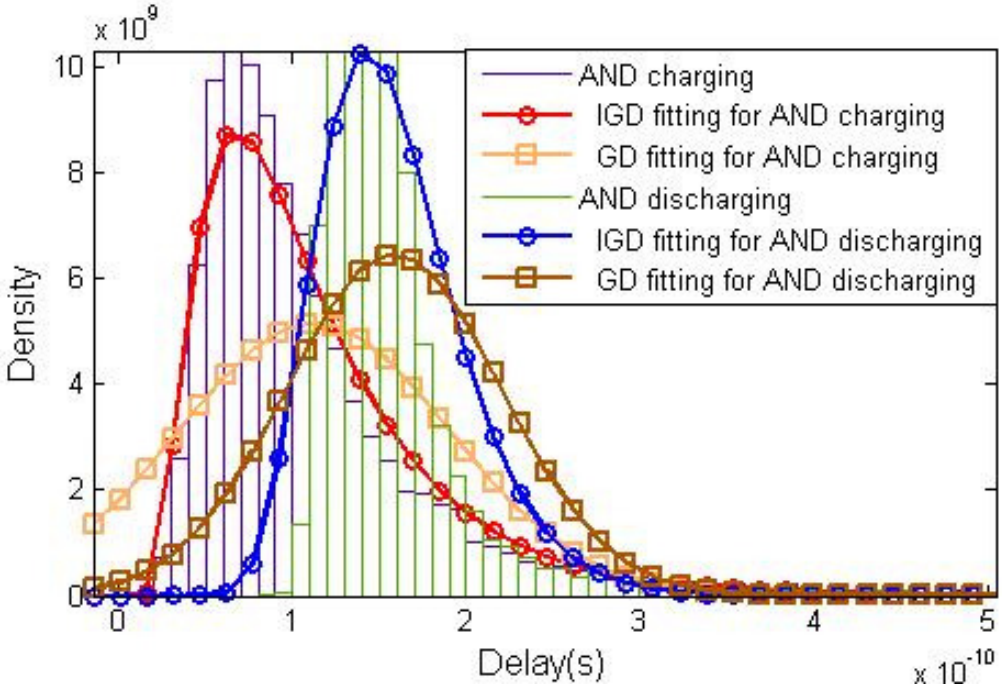


Fig. 6.4 IGD vs GD fitting for 2-input AND gate with only supply variation.

6.4 IGD Based Delay Model for Combinational and Sequential circuits

In this section, the IGD model is extended for both combinational and sequential circuits. The critical parameters are obtained that are utilized to estimate larger circuit probability distributions.

6.4.1 Typical timing path in synchronous CMOS circuits

In synchronous CMOS circuits, D-Latches (DL) and D Flip-Flops (DFFs) are employed for data synchronization. This is then passed onto the combinational logic blocks which are regarded as timing segments in static timing analysis. The cumulative sum of each delay segment caused by all these blocks determines the timing analysis typically known as Register to Register (R2R) delay as depicted in Eq.6.2.

$$D_{R2R} = D_{C2Q} + D_{LOGIC} + D_{SETUP}, \tag{6.2}$$

where,

- D_{R2R} is the total delay of a timing path;
- D_{C2Q} is the delay of D flip-flop from the clock rising/falling edge to the output;
- D_{LOGIC} is the propagation delay through the combinational logic;
- D_{SETUP} is the setup time of the output registers.

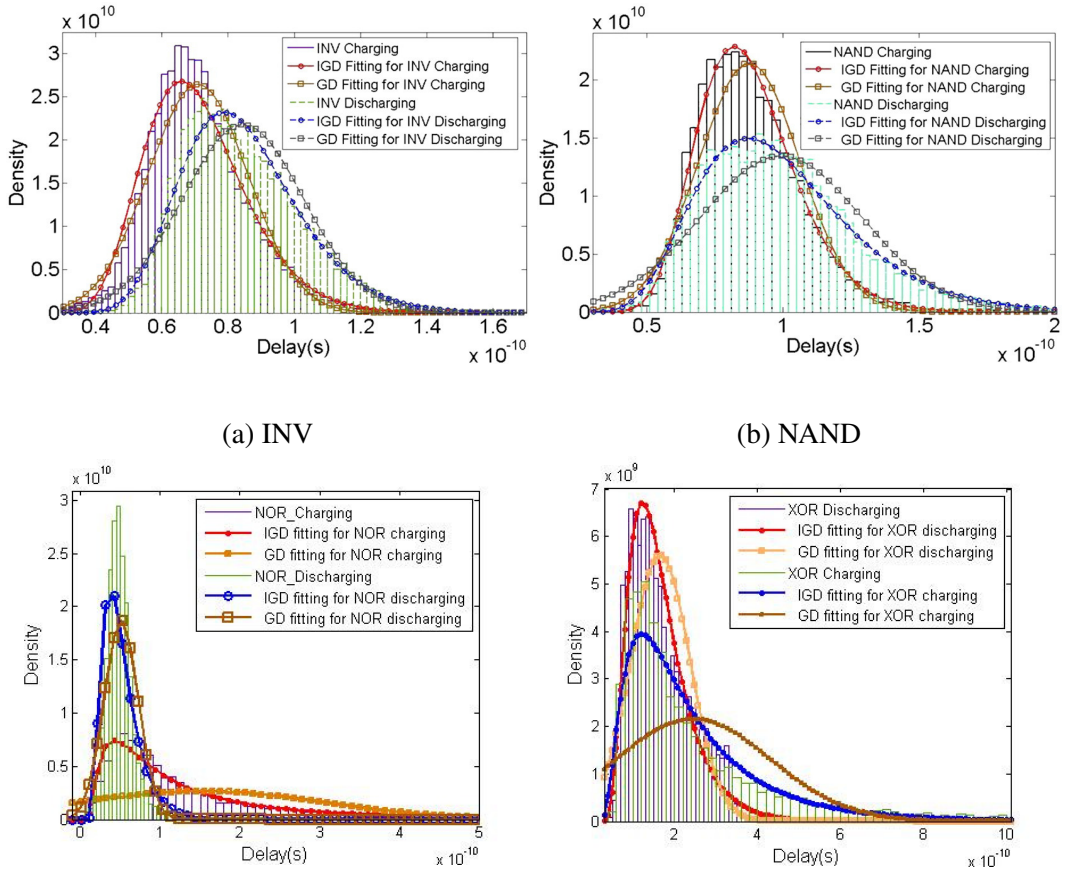
Unlike the conventional corner analysis that comprises of exact delay value for each component, the current approach proposes to estimate the IGD key parameters μ and λ for each of the terms in Eq.6.2. Major emphasis is placed on the first two terms of the equation and leaves D_{SETUP} for future consideration given that it is significantly smaller. The methodology of μ and λ computation for the longest path using a linear combination of the parameters neglecting D_{SETUP} was well explained in [23]. Eq.6.2 can therefore be translated into:

$$\begin{aligned} \mu_{R2R} &= \mu_{C2Q} + \mu_{LOGIC} \\ \lambda_{R2R} &= \lambda_{C2Q} + \lambda_{LOGIC} \end{aligned} \tag{6.3}$$

6.4.2 Combinational Gates - INV, NAND, NOR & XOR

The basic gates are the building blocks in any digital CMOS circuits. As the simplest gate in the CMOS family, INV is widely used in signal regulation and for enhancing signal strength. NAND and NOR gates are universal gates, and it is well known that all Boolean circuits can be synthesized with either of these universal gates alone. XOR is the most commonly used gate in all error correcting circuits. The IGD and GD fittings of all these gates based on ten thousand MC simulations and Fan-Out(FO) value set to one are depicted in Fig. 6.5. Both charging and discharging events at the output node are considered. The list of parameter variations employed within the simulation setup is:

- V_{dd} - mean value 0.3V and deviation 30mV;
- V_{th} - mean value 0.322V for nFETs and -0.302V for pFETs, and standard deviation 50mV;



(a) INV (b) NAND

(c) NOR (d) XOR

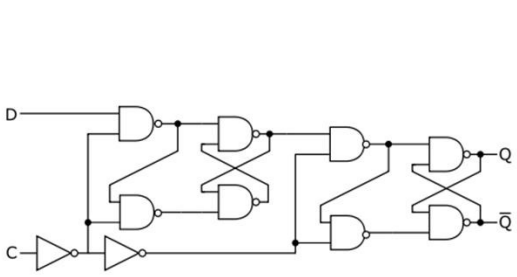
Fig. 6.5 IGD and GD fittings for basic gates charging and discharging events

- T_{ox} - 10% deviation for both nMOS and pMOS transistors.

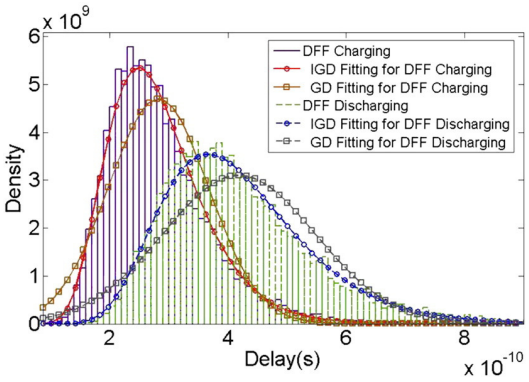
From the plots, it is clear that IGD correlates well with that of the MC simulation results as compared to GD. The critical parameters, μ and λ, for INV and NAND which serve as fundamental values for the proposed delay model.

6.4.3 Sequential Circuits- Master Slave DFF

Sequential elements such as DL and DFF are employed in logic circuits for data synchronization and are extremely important from timing analysis perspective. Unlike combinational circuits, which do not include any feedback loops, cross-coupled circuits are utilized for data retention in these elements. Therefore, it is important to verify if the IGD model fits well also for sequential elements as well. A DFF is composed of two adjacent DLs, known as Master and Slave, controlled by complementary clock signals. A DFF implementation built



(a) DFF Schematic



(b) PDF based Fitting

Fig. 6.6 IGD and GD fittings for DFF charging and discharging events

with eight NAND gates and two INVs is depicted in Fig. 6.6(a). The DFF IGD and GD fitted PDFs along with MC simulation data for FO=1 is presented in Fig. 6.6(b) where both the discharging ($1 \rightarrow 0$) and charging ($0 \rightarrow 1$) events at the DFF input (D) are considered.

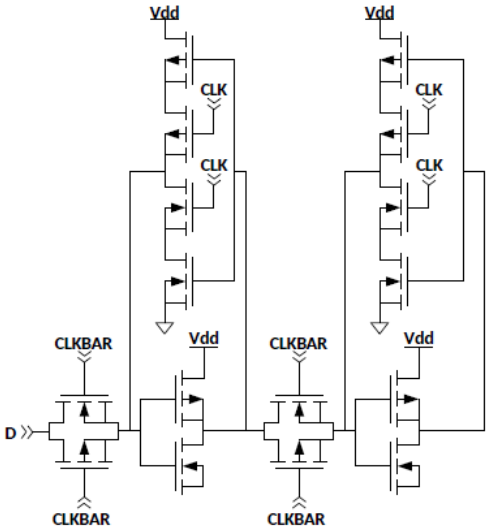
Table 6.1 μ and λ for NAND, INV and DFF

Gatetype	Charging		Discharging	
	μ	λ	μ	λ
INV	4.8	9.3	5.8	9.6
NAND	6.2	11.3	7.9	7.7
DFF	28.2	33.4	41.8	47.2

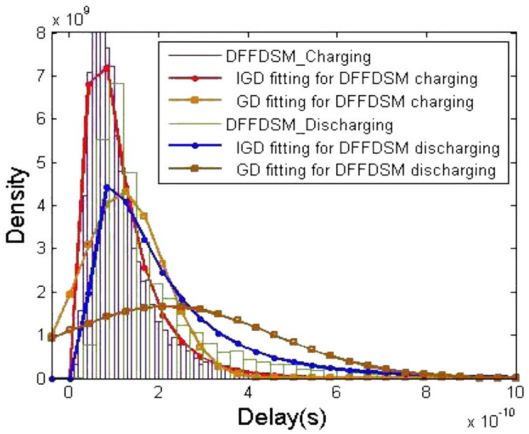
The key parameters, μ and λ under different conditions are also summarized in Tab. 6.1. It can be observed that the falling transition values are greater than the ones for the rising transition, which means that the discharging event takes more time than the charging event. Again one can easily observe that IGD correlates well with MC simulation data as compared to GD.

6.4.4 Sequential Circuits- Sub-Threshold DFF

For ultra-low-power applications, operating the transistors in their sub-threshold region is an effective way of reducing the power dissipation of a circuit. An optimal flip-flop has low power dissipation, imposes no delay and gives a valid output at all time. In a practical implementation, trade-offs between these parameters must be done. To confirm that the model works even with any specific flop designs, the analysis is extended unto sub-threshold specific flop design as depicted in Fig. 6.7(a). PowerPC 603 flip-flop, which was used in the



(a) DFF Schematic



(b) PDF based Fitting

Fig. 6.7 IGD and GD fittings for Sub-threshold based DFF architecture.

PowerPC 603 microprocessor data-path [148], is a combination of the Transmission-Gate Master-Slave (TGMS) and C2MOS flip-flops, using clocked inverters instead of feedback transmission gates. The DFF IGD and GD fitted PDFs along with MC simulation data for FO=1 are presented in Fig. 6.7(b) where both the discharging ($1 \rightarrow 0$) and charging ($0 \rightarrow 1$) events at the DFF input (D) are considered. It is observed that IGD correlates well with MC simulation data as compared to GD.

Thus, the practicability of fitting the behavior of both combinational and sequential gates using the IGD model is demonstrated. The shapes of the data and IGD fitting curves for both combinational and sequential elements are not symmetric, which once more provides evidence of the GD model inappropriateness. Besides, it is also demonstrated that discharging events for all gate types take longer than charging events. This provides a robust platform for delay estimation in a typical timing path. In the next section, the effect of Fan-Out(FO) and input transition time on the current methodology is discussed.

6.5 Fan-out Aware IGD Model

In the previous section, a fundamental equation to calculate the IGD key parameters for the critical path was introduced along with several IGD fittings and the corresponding essential parameters for sub-powered gates. Moreover, the linear compositionality of the IGD model

for combinational circuits have been demonstrated in [23]. To complete the delay model, fan-out a crucial component must be taken into account to depict more realistic scenarios. Fan-out, also regarded as the capacitive load at the output of a gate, can significantly affect output signal transition time and the propagation delay. In fact, there are two types of fan-out related phenomena that affect the gate delay:

- Fan-Out of the Current gate (FOC);
- Fan-Out of the Previous gate (FOP);

FOC has a direct impact on the gate delay. On the other hand, higher FOP would result in long signal transition time on the gate inputs which eventually results in higher propagation delay. The driving ability of sub-powered circuits is relatively weak and limits the maximum acceptable output load. In other words, a high fan-out is not suitable or requires careful designs in near/sub-threshold circuits. In this work, the maximum fan-out value is set to 4. When high fan-out is needed for sequential elements, i.e., DFFs, buffers insertion technique would be employed for load distribution instead of overloading the DFFs output node. This practice justifies narrowing the study on the FO value impact on combinatorial elements only. This section will address three key issues:

- Extend the methodology to capture FOC;
- Establish the relation between FOP and the IGD key parameters;
- Key IGD parameters computation for different FOC and FOP values.

Table 6.2 INV & NAND FOC key parameter values.

Gatetype	Charging				Discharging			
	e^-11		e^-10		e^-11		e^-10	
	P_μ	F_μ	P_λ	F_λ	P_μ	F_μ	P_λ	F_λ
INV	3.8	0.9	7.2	1.26	4.6	1.2	8.3	0.4
NAND	5.0	1.2	9.9	0.6	6.0	1.9	6.5	0.8

6.5.1 Fan-out effect estimation methodology

The logical effort, a straightforward technique to estimate delay in CMOS circuits, links the FOC and propagation delay [149]. The normalized delay D of a logic gate can be expressed

as the sum of two factors: the parasitic delay P and the stage effort F which depends on the gate load [149]:

$$D = NF + P \tag{6.4}$$

where N is the path branching effort, which indicates the fan-out number, which can be represented as the FOP in here. The key final output IGD parameters are calculated by applying the logical effort methods to them by using the following equations:

$$\begin{aligned} \mu_{FOC} &= FOC * F_{\mu} + P_{\mu} \\ \lambda_{FOC} &= FOC * F_{\lambda} + P_{\lambda} \end{aligned} \tag{6.5}$$

To derive F_{μ}, P_{μ}, F_{λ}, and P_{λ} for INV and NAND, two sets of data (μ and λ), i.e., FOC=1 and 2 for each gate, are collected and thereafter calculated by means of Eq.6.5. Once all values (F_{μ}, P_{μ} and F_{λ}, P_{λ}) are calculated, μ and λ with various FOC values can be evaluated. Those key coefficients are summarized in Tab. 6.2. It should be noted that the input transition time is 100ps.

Table 6.3 FOP effect on output transition time (all values in ps).

I/P Transition	T_{rise}	Increment	T_{fall}	Increment
FOP=1	85	-	99	-
FOP=2	124	39	153	54
FOP=3	165	41	204	51
FOP=4	206	41	256	52

6.5.2 Transition time effect estimation methodology

In the previous subsection, the FOC effect on the IGD model key parameters has been investigated and a methodology to calculate these values has been introduced. Now, the FOP effect on propagation delay in the form of transition time degradation will be discussed. It is understandable that high FO values cause long output transition time thereby increasing the propagation delays of the following gates. A look-up table is generated to capture the direct link between FOP and corresponding output transition time. INV gates with FOP=1, 2, 3, and 4 are simulated with the same variation set-up utilized previously. The corresponding output transition time is listed in Tab. 6.3, where it can be observed that with 100ps input transition time, the INV output transition time increases notably and the increment is quite steady following the FOP increase for both charging and discharging events. When it comes

to FOP=4, the output rise and fall time (input for the following gates) exceed 200ps, which can greatly increase the propagation delay of the driven gates. Based on the Tab. 6.3 data, it is of interest to investigate the corresponding change of the μ and λ values for different FOP when FOC=1 for both INV and NAND gate. The corresponding data are listed in Tab. 6.4, which provides the key parameter difference between two successive FOP values, namely T_μ and T_λ.

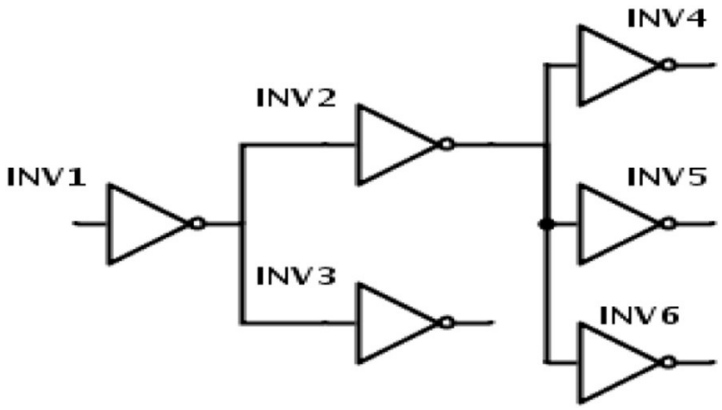


Fig. 6.8 A sample circuit with FOC= 3 and FOP= 2.

According to Tab. 6.4 and Tab. 6.5, the increment in μ and λ represented by constant T_μ and T_λ is steady. Therefore, the FOP effect on the IGD model can be simply calculated as follows:

$$\begin{aligned} \mu_{FOC} &= (FOP-1)T_\mu \\ \lambda_{FOC} &= (FOP-1)T_\lambda \end{aligned} \tag{6.6}$$

Table 6.4 FOP effect on key parameters for INV.

Gatetype	Charging				Discharging			
	e^-11		e^-10		e^-11		e^-10	
	P_μ	F_μ	P_λ	F_λ	P_μ	F_μ	P_λ	F_λ
FOP=1	3.8	0.8	7.2	0.9	4.6	-	8.3	-
FOP=2	4.6	0.8	8.1	0.9	5.5	0.9	12.5	4.2
FOP=3	5.4	0.8	9.2	1.1	7.2	0.9	16.9	4.4
FOP=4	6.2	0.8	10.2	1.0	8.0	0.9	21.2	4.3

6.5.3 The FOC and FOP effects

After investigating FOC and FOP effects respectively on the propagation delay as well as the IGD model key parameters, the straightforward combination of these two parts is given in the following equations.

$$\begin{aligned} \mu_{LOGIC} &= FOC * F_{\mu} + P_{\mu} + (FOP-1)T_{\mu} \\ \lambda_{LOGIC} &= FOC * F_{\lambda} + P_{\lambda} + (FOP-1)T_{\lambda} \end{aligned} \tag{6.7}$$

An INV based example where INV2 has FOC=3, and FOP=2 is illustrated in Fig. 6.8. The μ and λ calculation for the INV2 charging is carried out based on the values in Tab. 6.2 and Tab. 6.4. The μ and λ calculation is carried out by utilizing Eq. 6.7.

$\mu_{INV2} = 3*0.9e-11+3.8e-11+(2-1)*0.8e-11 = 7.3e-11$ $\lambda_{INV2} = 3*1.26e-10+7.2e-10+(2-1)*0.9e-10 = 11.88e-10$

Table 6.5 FOP effect on key parameters for NAND.

Gatetype	Charging				Discharging			
	$e^{-}11$		$e^{-}10$		$e^{-}11$		$e^{-}10$	
	P_{μ}	F_{μ}	P_{λ}	F_{λ}	P_{μ}	F_{μ}	P_{λ}	F_{λ}
FOP=1	5.0	0.8	9.9	2.4	6.0	0.9	6.5	-
FOP=2	5.8	0.8	12.3	2.4	6.9	0.9	7.9	1.4
FOP=3	6.6	0.8	14.8	2.5	7.8	0.9	9.3	1.4
FOP=4	7.4	0.8	17.4	2.6	8.6	0.9	10.5	1.2

6.5.4 Model Validation for Synchronous Circuits

To prove that the proposed IGD model and the method to propagate the key parameters is valid and applicable to generic circuits, SPICE simulation results are compared against the results obtained with the IGD approach for the following circuits:

- DFFs + 8-bit Ripple Carry Adder(RCA);
- DFFs + 8-bit Demultiplex(DEMUX) and Multiplexer(MUX).

The use of the model on similar circuits has already been presented, and Full Adder (FA) circuit has been thoroughly analyzed. To complete a synchronous timing path, the previously submitted data is being re-used along with the sub-threshold DFF values to complete the analysis. In this circuit, all involved gates have a fan-out of 1; thereby, the expansions of

the IGD model explained in the previous section are not required. Regarding the second circuit, different fan-out values are in place thus the method discussed in before is utilized. The Cumulative Distribution Functions (CDF), which is the integral of PDF is being used to more clearly quantify the difference between the proposed model results and MC data. CDF is used instead of PDF as it provides the average probability of switching event occurrence. PDF captures the likelihood of the happening of switching activity at that instant. From the circuit point of view, only switching event happening any time before the clock arrival are of interest. Consequently, the difference between the measured and the computed CDF is a better metric to evaluate the accuracy of the model.

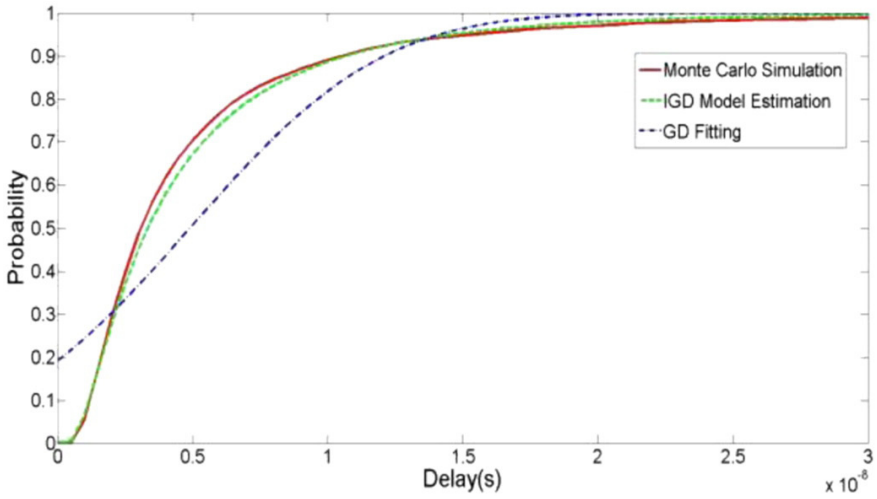


Fig. 6.9 DFFs +8-bit RCA CDFs.

DFFs + 8-bit Ripple Carry Adder

Given the FA parameters derived in and the already presented DFF parameters, the corresponding μ, and λ for an 8-bit RCA with DFFs is computed. From static timing analysis perspective, the worst analysis of the longest propagation delay is of utmost importance. The critical path within the FA is the one from Carry-In to Sum since its delay is larger when compared to Carry-In to Carry-Out within an 8-bit RCA, Hence, the longest delay occurs when the inputs A, B, Carry-In switch from all 0s to A=01111111, B=00000000, and Carry-In =1, resulting in an 10000000 output. In Fig. 6.9, the CDFs obtained by Monte Carlo simulation, the one obtained using the IGD model estimation, and GD based fitting are displayed for a delay range between 0ns to 30ns. The proposed IGD based delay prediction method closely approximates the MC simulation. An average mismatch of only 1.9% is recorded. On the other hand, the discrepancy between real simulation data and the GD

fitting is obvious. Once more, the GD fitted CDF curve starts from a non-zero value with a 0.2 probability, which is clearly unrealistic. It is important to clarify that the IGD curve is obtained by plotting the IGD CDF function whose parameters have been computed using the proposed propagation methodology and starting from the single block key parameters (no MC simulation necessary). On the contrary, the GD curve is an attempt to fit the MC data with a Gaussian curve. Even in this situation, where the GD fitting has the advantage of knowing the MC results, this method provides a much better approximation. CDF deviations between MC simulation and the IGD estimation for the 5ns to a 30ns range with a 5ns step are summarized in Tab. 6.6, case in which the highest deviation is 4.5% at 5ns, which is at an early stage of propagation, while all the others are below 1%.

Table 6.6 DFFS +8-BIT RCA CDF deviations.

Deviation(ns)	5	10	15	20	25	30	Average
IGD Estimation(%)	4.5	0.5	0.6	0.9	0.7	0.7	1.9

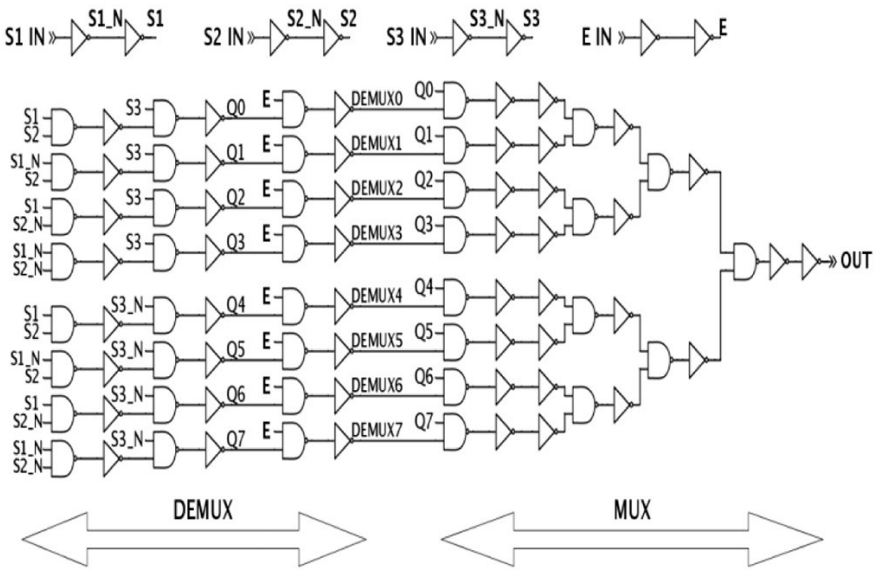


Fig. 6.10 Schematic of 8-bit DEMUX and MUX.

DFFs + 8-bit DEMUX and MUX

The corresponding key IGD parameters of the output of an 8-bit DEMUX and MUX with DFFs can be evaluated by using the data and methodology that was presented previously, i.e., fan-out number, and the entailing transition time. The schematic of the 8-bit DEMUX and MUX are depicted in Fig. 6.10 where only INVs and NANDs are being used. In Fig. 6.11,

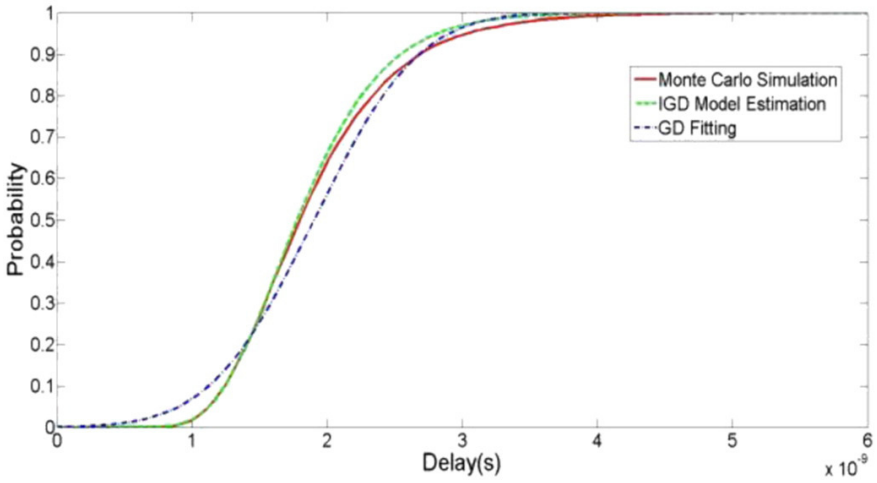


Fig. 6.11 DFFs +8-bit DEMUX and MUX CDFs.

the CDFs obtained utilizing Monte Carlo simulation, by IGD based estimation, and by GD fitting, are depicted for a delay range between 0ns to 6ns. The IGD estimation based curves closely follow the MC simulation with a slight deviation around 2ns and overall, it is better than the GD fitting. Tab. 6.7 lists the CDF deviations from MC simulation and the IGD estimation as well as the GD fitting for a delay range from 1ns to 6ns. The highest mismatch is 3.4% at 2ns, and the average overall error is 1.2% and 7.3% for the IGD estimation and the GD fitting, respectively. The fact that GD fitting performs better than IGD in the case of RCA circuit can be related to the smaller circuit size. It is worth mentioning that, due to the non-zero-crossing of the GD CDF, the GD deviation fitting will be too large if timing range starting from 0ns is chosen.

Table 6.7 DFFS +8-BIT DEMUX and MUX CDF deviations.

Deviation(ns)	1	2	3	4	5	6	Average
IGD Estimation(%)	2.6	3.4	2.4	0.6	0.1	0.1	1.2

6.6 Conclusions

A comprehensive fan-out aware Inverse Gaussian Distribution (IGD) based delay model was introduced. The IGD model was verified for both combinational and sequential gates, and fan-out effects were considered in two situations namely FOC and FOP. While both the GD and IGD models work well for normal voltage operation, IGD is much better suited as compared to GD under the sub-threshold region of operation. The model is not only highly accurate (close match to SPICE Monte Carlo simulation results), but more importantly, it exhibits

excellent flexibility against process and voltage supply variations. The key IGD model parameters evaluation is very straightforward, which is beneficial for the delay estimation of complex circuits. When compared to MC SPICE simulation data, obtained for the following circuits: (1) DFFs + 8-bit Ripple Carry Adder and (2) DFFs + 8-bit DEMUX and MUX, this method provides average mismatches of 1.9% and 1.2% respectively, while reducing run-time by up to 50 times. Moreover, when compared to the GD fitting results, the IGD based estimation was more accurate in both cases. The average deviation for the IGD estimation was 85% smaller than the one corresponding to GD fitting, e.g., 1.2% versus 7.3% for the second circuit.

In this chapter, IGD based delay degradation timing models were presented and the results are in close match to MC based simulation results. In the next chapter, the summary of the research work along with possible directions for future research are presented.

Chapter 7

Summary and Future work

Low power design is becoming increasingly important in today's technology as wireless communication becomes increasingly desirable. Although power dissipation is significant in portable systems, reliability is an equally important target for digital designers. With the advent of deep submicron technologies, transient faults are an increasing concern in semiconductor products. Most circuit-level mitigation techniques deal with power, area and timing constraints. But, accurate and early modeling, as well as error rate predictions, are the need of the hour. Hence the motive of the thesis is to explore the modeling of transient faults in logic circuits and its practical implementation to allow for the analysis of the impact of transient faults. The primary focus of the work is to design combinational logic circuits with reliability as the main driving force.

This research work first provided a basic model for transient fault propagation in combinational circuits, including modeling of essential masking factors. The discussion in Chapter 3 provided more details about these aspects of the proposed probabilistic gate error model. The analysis of delay degradation analysis and PDF propagation was included. After the modeling of transient faults, incorporating an optimization technique when transient faults are considered, to show that transient-fault reduction can be achieved by employing traditional rewriting techniques. Lastly, a novel methodology to design fault tolerant circuitry by using the error correction codes mechanism that was discussed.

7.1 Contributions

The thesis introduced new methodologies and new synthesis methods for reliability-aware low power digital design. In our approach, the reliability of digital circuits was estimated on AND-Inverter graphs. A series of graph reordering rules were applied on these circuits. These rules were based on common rules of Boolean Algebra and worked on the restructuring

of the AIG network keeping the functionality the same. This reordering and restructuring of the AIG nodes achieved a significant reduction in circuit error probability.

7.1.1 Data Structure

Data structures and algorithms largely determine the efficiency of the tool in implementing new applications. AIG deals with simple and flexible data structures, which promise improvements in quality and runtime of several applications which include both sequential as well as combinational logic. Moreover, the BDD is not the best suitable for the aim of the i-RISC project of computing reliability. The major problem in this regards is the fact that the BDD represents only the logic functionality of the combinatorial logic losing any direct link with the physical implementation. Technology mapping task is more difficult in BDD AIG is better suited to this task since each node on the graph represents an actual physical port. For these reasons, the AND-Inverter graph has been chosen as the underlying data structure. The data structure has then been enhanced to store information regarding the reliability of the circuit it represents. A number of academic EDA synthesis tools proposed in the literature were studied. These open-source tools provide a programming environment and a solid platform for research in logic synthesis, technology mapping, reliability estimation and optimization. Once the AIG graph is chosen as the most suitable data structure the ABC tool is a natural choice. The ABC tool has several advantages: a) it is open source b) It is actively maintained and c) it has several routing suitable for synthesis/mapping and verification. The ABC tool (including the sub-packages) is then incorporated in a design flow which involves also industry tools, namely from Synopsys.

7.1.2 Inverse Gaussian Distribution Based Timing Analysis

Successful timing analysis is the method of computing the maximum clock frequency at which the circuit can safely operate thereby guaranteeing the right chip functionality post-fabrication. With the continuous scaling of the transistors, unpredictable gate delay values induced by local variations along with other variations like CMOS channel length, threshold voltage, and oxide thickness are some of the critical factors that complicate timing analysis and estimation. A comprehensive fan-out aware Inverse Gaussian Distribution (IGD) based delay model was introduced. The main idea behind the proposal is first to gather the necessary gate key parameters employing MC simulations, Look Up Table(LUT) for all the gates in the library is maintained to capture these critical parameters. Then, by applying the principles of convolution, an algorithm is developed that propagates the PDF's onto the circuit output nodes by traversing the critical path. The IGD model was verified for both combinational and

sequential gates, and fan-out effects were considered in two situations namely FOC and FOP. While both the GD and IGD models work well for normal voltage operation, IGD is much better suited as compared to GD under the sub-threshold region of operation. The model is highly accurate even for different power supply voltage values ranging from nominal V_{dd} to sub-threshold V_{dd}, When compared to MC SPICE simulation data, obtained for the following circuits: (1) DFFs + 8-bit Ripple Carry Adder and (2) DFFs + 8-bit DEMUX and MUX, our methods provide average mismatches of 1.9% and 1.2% respectively, while requiring orders of magnitude less simulation time. Moreover, when compared to the GD fitting results, the IGD based estimation was more accurate in both cases. The average deviation for the IGD estimation was 85% smaller than the one corresponding to GD fitting, e.g., 1.2% versus 7.3% for the second circuit.

7.1.3 ROST-C: Reliability driven Optimization and Synthesis

The increasing demand for reliability analysis calls for automated tools that could analyze the circuit reliability in quick time without compromising on the accuracy. new gate error model called Conditional Probabilistic Error Propagation (CPEP) builds a probabilistic error model of the combinatorial circuit to study the impact of logical error on individual gates on the overall circuit. A complete analytical treatment was provided which provides a close formed equation to describe the gate error probability is also accounting for the impact of re-convergent fanout. Experimental results obtained with the proposed CPEP framework is within 2% average error and up to 1000 times faster when compared to Monte Carlo simulations. Subsequently, reliability driven 4-cut enumeration and Boolean matching technique that improves circuit reliability have been proposed. The method of rewriting for reliability was developed by extending an existing cut based rewriting tool to make use of local transforms targeting a reliability metric improvement instead of area. A synthesis algorithm that optimizes the circuit output nodes error probability was also presented. A primary tool incorporating the AIG and some local transformation rules based on Boolean algebra has been proposed for computing the reliability function, and it was demonstrated that through the selective application of the proposed rules, the reliability could be significantly improved. Application of this tool on the standard MCNC benchmarks resulted in an average improvement of 4.06% and a peak improvement of 7.52%. Another logic optimisation technique based on cut enumeration and Boolean matching is also presented. Version 1 gave an average increase of 5.11% on the MCNC benchmarks, with a peak of 27.75%. Version 2 gave an average improvement of 15.33% and peak improvement of 37.62%. For comparison, standard area-goal rewriting was also tested to see if the error would reduce on a 'fewer gates is less to go wrong' basis. An average improvement of 4.50% was obtained for standard

rewriting on the MCNC benchmarks which is way less than our new methodologies proposed. The proposed framework will be used to explore systematic multi-objective optimization methodology of fault-tolerant circuits.

7.1.4 Error Coding Driven Graph Augmentation

A novel methodology is proposed to implement error correction codes driven graph augmentation techniques to improve the fault tolerance of the circuits that is generic and applies to any combinatorial logic. CPE methodology of implementation and performance evaluation has been completely automated. Initial simulation results show that performance of CPE is much better as compared to that of the NMR approach. Replication of up to '11' times of the combinational circuit was needed to achieve similar kind of performance as the CPE. Apart from this, some other important sets of conclusions to make are as follow for any LDPC code rate and for any critical gate threshold:

- With lower code rate, a significant amount of performance improvement is achieved wherein the CPE can correct all errors for gate error probability of 1e-2.
- Lower the code rate, higher the number of gates in the parity circuit 'P'.
- Number of critical gates is not rising drastically with code rate.

7.1.5 Boole-Shannon Limit of noisy combinational logic

The analysis presented in this section reflect only a preliminary investigation on the various possible avenues to reach mathematical formulation of what are the limitation and the potential trade-off of the i-RISC approach to the error-prone circuitry. Two research directions are foreseen to continue this work. The first it is to understand the implication of the corner scenario cost analysis presented and expand it into a unified approach that could evaluate all costs (area/power/throughput/performance) in a multidimensional way. In this direction, it would also be necessary to expand the analysis to non-linear circuits. The second direction would improve the asymptotic analysis presented to arrive at a formulation of a Boole-Shannon eventually limit for ECC and error-prone circuitry. The research in this direction should first only consider linear circuits to obtain fundamental limits in the same way many conceptual boundaries in telecommunication are presented/valid just for restricted channels.

7.2 Future Work

While this work introduced some solutions for the existing problems, it also opened the door for new questions and methodologies for reliability-aware logic optimization in digital circuits.

- Delay Degradation Analysis: IGD based PDF propagation technique has been proposed to perform delay degradation analysis in combinational circuits. An improved variability-aware PDF based delay model that accurately captures the delay behavior of all the basic gates in the library database is currently in progress. The proposed methodology is then employed to perform Static Timing Analysis (STA) on complex circuits. Future work includes investigation on applying this methodology to study timing errors and developing a synthesis tool for reliability.

- Reliability Analysis: The current research performs probabilistic error propagation to identify gates that are more likely to propagate an SEU and currently only deals with logical masking errors. Further, the model can be improved upon to cover the electrical and latching-window masking factors. Also, the impacts of glitches on overall output error also can be developed. Overall, the analysis can be extended to sequential circuits as well.

- Reliability Optimization: A reliability driven 4-cut enumeration and Boolean matching technique that improves circuit reliability has been proposed. Going forward, the local transformation rule set can be extended to encompass more possible scenarios. From a reliability perspective, the AIG data structure is more appropriate to represent combinational circuits. In particular, the fact that AIG is non-canonical (i.e., there exist more graphs representing the same logic function) can be exploited to improve reliability further. It is intended to extend the reliability evaluator to sequential circuits as well that would enable us to characterize the more recent and complex IWLS 2005 benchmarks. Intelligent addition of nodes within the structure can remarkably reduce the error on the output node. A formal mathematical procedure that can guide the optimization algorithms to insert these nodes automatically based on the structure of the circuit. Further, validation of the flow on real benchmarks synthesized to the final gate netlist employing technology mapping can be another interesting case study.

- Fault Tolerant Circuit Design: Conventional LDPC coding scheme determines the codeword size (or the size of parity added) to compensate for the noise in the channel. This work considers additional parity added to compensate extra noise due to unreliable hard-

ware. Two research directions are foreseen to continue this work. The first it is to understand the implication of the corner scenario cost analysis presented and expand it into a unified approach that could evaluate all costs (area/power/throughput/performance) in a multidimensional way. In this direction, it would also be necessary to expand the analysis to non-linear circuits. The second direction would improve the asymptotical analysis presented to eventually arrive at a formulation of a Boole-Shannon limit for ECC and error-prone circuitry. The research in this direction should first only consider linear circuits to obtain fundamental limits in the same way many conceptual boundaries in telecommunication are presented/valid just for restricted channels.

References

[1] Bo Yang, Satish Grandhi, Christian Spagnol, Emanuel Popovici, and Sorin Cotofana. An approach for digital circuit error/reliability propagation analysis based on conditional probability. In *Signals and Systems Conference (ISSC), 2016 27th Irish*, pages 1–6, 2016.

[2] Satish Grandhi, Bo Yang, Christian Spagnol, Samarth Gupta, and Emanuel Popovici. An EDA framework for reliability estimation and optimization of combinational circuits. *Journal of Low Power Electronics*, 12(3):242–258, 2016.

[3] Satish Grandhi, Elsa Dupraz, Christian Spagnol, Valentin Savin, and Emanuel Popovici. CPE: Codeword prediction encoder. In *Test Symposium (ETS), 2016 21th IEEE European*, pages 1–2, 2016.

[4] Elsa Dupraz, Valentin Savin, Satish Kumar Grandhi, Emanuel Popovici, and David Declercq. Practical LDPC encoders robust to hardware errors. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6, 2016.

[5] Alfredo Benso and Paolo Prinetto. *Fault injection techniques and tools for embedded systems reliability evaluation*, volume 23. Springer Science & Business Media, 2003.

[6] Sasan Iman and Massoud Pedram. POSE: Power optimization and synthesis environment. In *Logic Synthesis for Low Power VLSI Designs*, pages 199–224. 1998.

[7] Rashmi Mehrotra, Tom English, Michel Schellekens, Steve Hollands, and Emanuel Popovici. Timing-driven power optimisation and power-driven timing optimisation of combinational circuits. *Journal of Low Power Electronics*, 7(3):364–380, 2011.

[8] Paul E Dodd and Lloyd W Massengill. Basic mechanisms and modeling of single-event upset in digital microelectronics. *Nuclear Science, IEEE Transactions on*, 50(3):583–602, 2003.

[9] Shekhar Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[10] Cristian Constantinescu. Trends and challenges in VLSI circuit reliability. *IEEE micro*, 23(4):14–19, 2003.

[11] Hubert Kaeslin. *Digital integrated circuit design: from VLSI architectures to CMOS fabrication*. Cambridge University Press, 2008.

[12] Eric Vittoz. *Weak inversion for ultimate low-power logic*. CRC Press, 2005.

[13] Robert Baumann. Soft errors in advanced computer systems. *Design & Test of Computers, IEEE*, 22(3):258–266, 2005.

[14] Shekhar Borkar. Tackling variability and reliability challenges. *IEEE Design & Test of Computers*, 6(23):520, 2006.

[15] Shekhar Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, pages 746–749, 2007.

[16] Subhasish Mitra, Tanay Karnik, Norbert Seifert, and Ming Zhang. Logic soft errors in sub-65nm technologies design and CAD challenges. In *Proceedings of the 42nd annual Design Automation Conference*, pages 2–4, 2005.

[17] Giacinto P Saggese, Nicholas J Wang, Zbigniew T Kalbarczyk, Sanjay J Patel, and Ravishankar K Iyer. An experimental study of soft errors in microprocessors. *IEEE micro*, (6):30–39, 2005.

[18] Narayanan Vijaykrishnan. Soft errors: is the concern for soft-errors overblown? In *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, pages 2–pp, 2005.

[19] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th annual Design Automation Conference*, pages 338–342, 2003.

[20] MP Baze and SP Buchner. Attenuation of single event induced pulses in CMOS combinational logic. *Nuclear Science, IEEE Transactions on*, 44(6):2217–2223, 1997.

[21] Peter Liden, Peter Dahlgren, Rolf Johansson, and Johan Karlsson. On latching probability of particle induced transients in combinational networks. In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, pages 340–349, 1994.

[22] James F Ziegler, Huntington W Curtis, Hans P Muhlfeld, Charles J Montrose, B Chin, Michael Nicewicz, CA Russell, Wen Y Wang, Leo B Freeman, P Hosier, et al. Ibm experiments in soft fails in computer electronics (1978–1994). *IBM journal of research and development*, 40(1):3–18, 1996.

[23] Jiaoyan Chen, Christian Spagnol, Satish Grandhi, Emanuel Popovici, Sorin Cotofana, and Alexandru Amaricai. Linear compositional delay model for the timing analysis of sub-powered combinational circuits. In *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, pages 380–385, 2014.

[24] Mihir R Choudhury and Kartik Mohanram. Reliability analysis of logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(3):392–405, 2009.

[25] Svetlana N Yanushkevich, D Michael Miller, Vlad P Shmerko, and Radomir S Stankovic. *Decision diagram techniques for micro-and nanoelectronic design handbook*. CRC Press, 2005.

[26] Ellen M Sentovich Kanwar Jit Singh, Luciano Lavagno Cho Moon Rajeev Murgai, and Robert K Brayton Alberto Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. page 4, 1992.

[27] Dennis Wu and Jianwen Zhu. FBDD: A folded logic synthesis system. In *ASIC, 2005. ASICON 2005. 6th International Conference On*, pages 746–751, 2005.

[28] Robert Brayton and Alan Mishchenko. ABC: An academic industrial-strength verification tool. In *International Conference on Computer Aided Verification*, pages 24–40, 2010.

[29] Alan Mishchenko, Satrajit Chatterjee, and Robert Brayton. Dag-aware AIG rewriting a fresh look at combinational logic synthesis. In *Proceedings of the 43rd annual Design Automation Conference*, pages 532–535, 2006.

[30] Mihir R Choudhury and Kartik Mohanram. Reliability analysis of logic circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(3):392–405, 2009.

[31] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[32] A. Mishchenko and R. K. Brayton. Scalable logic synthesis using a simple circuit structure. In *Proc. IWLS*, pages 15–22, 2006.

[33] Sachin S Sapatnekar and Jordi Cortadella. Static timing analysis. In *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*, pages 155–176. CRC Press, 2016.

[34] Natasa Miskov-Zivanov. *Probabilistic modeling and optimization for circuit reliability*. PhD thesis, Carnegie Mellon University, 2008.

[35] Schmid A. Stanisavljevic M. and Leblebici Y. *Reliability of Nanoscale Circuits and Systems: Methodologies and Circuit Architectures*. Springer Science & Business Media, 2010.

[36] Ieee standard definitions for use in reporting electric generating unit reliability, availability, and productivity. *ANSI/IEEE Std 762-1987*, pages 1–24, May 1987.

[37] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, et al. *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[38] Jean-Claude Laprie. Dependable computing and fault-tolerance. *Digest of Papers FTCS-15*, pages 2–11, 1985.

[39] Jeffrey A Clark and Dhiraj K Pradhan. Fault injection: A method for validating computer-system dependability. *Computer*, 28(6):47–56, 1995.

[40] Natasa Miskov-Zivanov and Diana Marculescu. Circuit reliability analysis using symbolic techniques. *IEEE transactions on computer-aided design of integrated circuits and systems*, 25(12):2638–2649, 2006.

[41] John Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34:43–98, 1956.

[42] Richard Burch, Farid N Najm, Ping Yang, and Timothy N Trick. A Monte Carlo approach for power estimation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 1(1):63–71, 1993.

[43] Elias Todorovich, M Gilabert, Gustavo Sutter, Sergio López-Buedo, and E Boemo. A tool for activity estimation in FPGAs. In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, pages 340–349. 2002.

[44] Graham Horton. A new paradigm for the numerical simulation of stochastic Petri nets with general firing times. In *Proceedings of the European Simulation Symposium*, pages 129–136, 2002.

[45] Ketan N Patel, Igor L Markov, and John P Hayes. Evaluating circuit reliability under probabilistic gate-level fault models. In *Proceedings of the International Workshop on Logic and Synthesis*, pages 59–64, 2003.

[46] Erin Taylor, Jie Han, and José Fortes. Towards accurate and efficient reliability modeling of nanoelectronic circuits. In *Nanotechnology, 2006. IEEE-NANO 2006. Sixth IEEE Conference on*, volume 1, pages 395–398, 2006.

[47] Jie Han, Erin Taylor, Jianbo Gao, and Jose Fortes. Faults, error bounds and reliability of nanoelectronic circuits. In *Application-Specific Systems, Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on*, pages 247–253, 2005.

[48] Jie Han, Hao Chen, Erin Boykin, and José Fortes. Reliability evaluation of logic circuits using probabilistic gate models. *Microelectronics Reliability*, 51(2):468–476, 2011.

[49] SJ Seyyed Mahdavi and Karim Mohammadi. SCRAP: Sequential circuits reliability analysis program. *Microelectronics Reliability*, 49(8):924–933, 2009.

[50] Denis Teixeira Franco, Mai Correia Vasconcelos, Lirida Naviner, and Jean-François Naviner. Reliability analysis of logic circuits based on signal probability. In *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pages 670–673, 2008.

[51] Denis Teixeira Franco, Maí Correia Vasconcelos, Lirida Naviner, and Jean-François Naviner. Signal probability for reliability evaluation of logic circuits. *Microelectronics Reliability*, 48(8):1586–1591, 2008.

[52] Walid Ibrahim and Valeriu Beiu. Using Bayesian networks to accurately calculate the reliability of complementary metal oxide semiconductor gates. *Reliability, IEEE Transactions on*, 60(3):538–549, 2011.

[53] Thara Rejimon and Sanjukta Bhanja. Time and space efficient method for accurate computation of error detection probabilities in VLSI circuits. In *Computers and Digital Techniques, IEE Proceedings-*, volume 152, pages 679–685, 2005.

[54] R Bahar, Joseph Mundy, and Jie Chen. A probabilistic-based design methodology for nanoscale computation. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 480, 2003.

[55] Debayan Bhaduri and Sandeep Shukla. NANOLAB-a tool for evaluating reliability of defect-tolerant nanoarchitectures. *Nanotechnology, IEEE Transactions on*, 4(4):381–394, 2005.

[56] Smita Krishnaswamy, George F Viamontes, Igor L Markov, and John P Hayes. Probabilistic transfer matrices in symbolic reliability analysis of logic circuits. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 13(1):8, 2008.

[57] Afshin Abdollahi. Probabilistic decision diagrams for exact probabilistic analysis. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pages 266–272, 2007.

[58] Rahul Rithe, Jie Gu, Alice Wang, Satyendra Datla, Gordon Gammie, Dennis Buss, and Anantha Chandrakasan. Non-linear operating point statistical analysis for local variations in logic timing at low voltage. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 965–968, 2010.

[59] Jayanth Srinivasan, Sarita V Adve, Pradip Bose, and Jude A Rivers. The case for lifetime reliability-aware microprocessors. In *ACM SIGARCH Computer Architecture News*, volume 32, page 276, 2004.

[60] Yao Wang, Marius Enachescu, Sorin Dan Cotofana, and Liang Fang. Variation tolerant on-chip degradation sensors for dynamic reliability management systems. *Microelectronics Reliability*, 52(9-10):1787–1791, 2012.

[61] Krishna Palem and Avinash Lingamneni. What to do about the end of moore's law, probably! In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 924–929, 2012.

[62] Smita Krishnaswamy, Stephen M Plaza, Igor L Markov, and John P Hayes. Enhancing design robustness with reliability-aware resynthesis and logic simulation. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pages 149–154, 2007.

[63] Kai-Chiang Wu and Diana Marculescu. A low-cost, systematic methodology for soft error robustness of logic circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(2):367–379, 2013.

[64] Mihir R Choudhury and Kartik Mohanram. Low cost concurrent error masking using approximate logic circuits. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 32(8):1163–1176, 2013.

[65] Sobeeh Almukhaizim, Yiorgos Makris, Yu-Shen Yang, and Andreas Veneris. Seamless integration of SER in rewiring-based design space exploration. In *Test Conference, 2006. ITC'06. IEEE International*, pages 1–9, 2006.

[66] Nivard Aymerich and Antonio Rubio. Reliability and performance tunable architecture: The partially asynchronous r-fold modular redundancy (pA-RMR). *IEEE Transactions on Nanotechnology*, 13(3):617–622, 2014.

[67] K Nikolic, A Sadek, and M Forshaw. Architectures for reliable computing with unreliable nanodevices. In *Nanotechnology, 2001. IEEE-NANO 2001. Proceedings of the 2001 1st IEEE Conference on*, pages 254–259, 2001.

[68] Robert E Lyons and Wouter Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*, 6(2):200–209, 1962.

[69] F Lima Kastensmidt, Luca Sterpone, Luigi Carro, and M Sonza Reorda. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pages 1290–1295, 2005.

[70] Behrooz Parhami. New classes of unidirectional error-detecting codes. In *Computer Design: VLSI in Computers and Processors, 1991. ICCD'91. Proceedings, 1991 IEEE International Conference on*, pages 574–577, 1991.

[71] Tenkasi V Ramabadran. A coding scheme for m-out-of-n codes. *IEEE Transactions on Communications*, 38(8):1156–1163, 1990.

[72] F Özgüner. Design of totally self-checking embedded two-rail code checkers. *Electronics letters*, 27(4):382–384, 1991.

[73] Haruhiko Kaneko. Error control coding for semiconductor memory systems in the space radiation environment. In *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, pages 93–101, 2005.

[74] Zhongfeng Wang, Hiroshi Suzuki, and Keshab K Parhi. VLSI implementation issues of turbo decoder design for wireless applications. In *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on*, pages 503–512, 1999.

[75] Subhasish Mitra and Edward J McCluskey. Which concurrent error detection scheme to choose? In *Test Conference, 2000. Proceedings. International*, pages 985–994, 2000.

[76] Kartik Mohanram, Egor S Sogomonyan, M Gossel, and Nur A Touba. Synthesis of low-cost parity-based partially self-checking circuits. In *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pages 35–40, 2003.

[77] Michael G Taylor. Reliable information storage in memories designed from unreliable components. *Bell Labs Technical Journal*, 47(10):2299–2337, 1968.

[78] Michael G Taylor. Reliable computation in computing systems designed from unreliable components. *Bell Labs Technical Journal*, 47(10):2339–2366, 1968.

[79] Aleksandr Vasil'evich Kuznetsov. Information storage in a memory assembled from unreliable components. *Problemy Peredachi Informatsii*, 9(3):100–114, 1973.

[80] Christoforos N Hadjicostis and George C Verghese. Coding approaches to fault tolerance in linear dynamic systems. *Information Theory, IEEE Transactions on*, 51(1):210–228, 2005.

[81] Bane Vasic and Shashi Kiran Chilappagari. An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(11):2438–2446, 2007.

[82] Christiane Kameni Ngassa, Valentin Savin, Elsa Dupraz, and David Declercq. Density evolution and functional threshold for the noisy min-sum decoder. *IEEE Transactions on Communications*, 63(5):1497–1509, 2015.

[83] Elsa Dupraz and David Declercq. Evaluation of the robustness of LDPC encoders to hardware noise. In *Communications and Networking (BlackSeaCom), 2015 IEEE International Black Sea Conference on*, pages 87–91, 2015.

[84] Myeong-Eun Hwang. Supply-voltage scaling close to the fundamental limit under process variations in nanometer technologies. *Electron Devices, IEEE Transactions on*, 58(8):2808–2813, 2011.

[85] Michael Merrett, Plamen Asenov, Yangang Wang, Mark Zwolinski, Dave Reid, Campbell Millar, Scott Roy, Zhenyu Liu, Steve Furber, and Asen Asenov. Modelling circuit performance variations due to statistical variability: Monte Carlo static timing analysis. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–4, 2011.

[86] Lu Wan and Deming Chen. Analysis of circuit dynamic behavior with timed ternary decision diagram. In *Proceedings of the International Conference on Computer-Aided Design*, pages 516–523, 2010.

[87] Samy Zaynoun, Muhammed S Khairy, Ahmed M Eltawil, Fadi J Kurdahi, and Amin Khajeh. Fast error aware model for arithmetic and logic circuits. In *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, pages 322–328, 2012.

[88] David Blaauw, Kaviraj Chopra, Ashish Srivastava, and Lou Scheffer. Statistical timing analysis: From basic principles to state of the art. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(4):589–607, 2008.

[89] David M Binkley. Tradeoffs and optimization in analog CMOS design. In *Mixed Design of Integrated Circuits and Systems, 2007. MIXDES'07. 14th International Conference on*, pages 47–60, 2007.

[90] Sean Keller, David Money Harris, and Alain J Martin. A compact transregional model for digital CMOS circuits operating near threshold. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(10):2041–2053, 2014.

[91] Satish Grandhi, Christian Spagnol, and Emanuel Popovici. Reliability analysis of logic circuits using probabilistic techniques. In *Ph. D. Research in Microelectronics and Electronics (PRIME), 2014 10th Conference on*, pages 1–4, 2014.

[92] S Ercolani, M Favalli, M Damiani, P Olivo, and B Ricco. Estimate of signal probability in combinational logic networks. In *European Test Conference, 1989., Proceedings of the 1st*, pages 132–138, 1989.

[93] Michael Bushnell and Vishwani Agrawal. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, volume 17. Springer Science & Business Media, 2004.

[94] Richard Burch, Farid N Najm, Ping Yang, and Timothy N Trick. A Monte Carlo approach for power estimation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(1):63–71, 1993.

[95] Elias Todorovich, M Gilabert, Gustavo Sutter, Sergio López-Buedo, and E Boemo. A tool for activity estimation in FPGAs. In *International Conference on Field Programmable Logic and Applications*, pages 340–349, 2002.

[96] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for deep-submicron FPGAs*. Kluwer Academic Publishers, 1999.

[97] Richard Burch, Farid Najm, Ping Yang, and Dale Hocevar. Pattern-independent current estimation for reliability analysis of CMOS circuits. In *Annual ACM IEEE Design Automation Conference: Proceedings of the 25 th ACM/IEEE conference on Design automation*, volume 12, pages 294–299, 1988.

[98] Sharad C Seth and Vishwani D Agrawal. A new model for computation of probabilistic testability in combinational circuits. *INTEGRATION, the VLSI journal*, 7(1):49–75, 1989.

[99] A Biere. The AIGER and-inverter graph (AIG) format, version 20070427. *Available at fmv. jku. at/aiger*, 2007.

[100] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[101] Shiy Xu and E Edirisuriya. A new way of detecting reconvergent fanout branch pairs in logic circuits. In *Test Symposium, 2004. 13th Asian*, pages 354–357, 2004.

[102] A Mishchenko et al. ABC: A system for sequential synthesis and verification. *URL http://www. eecs. berkeley. edu/˜ alanmi/abc*, 2007.

[103] Giovanni De Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education, 1994.

[104] Satish Grandhi. Data structures and design flow for fault tolerant circuit synthesis. Technical report, University College Cork, Ireland, January,2014.

[105] Satish Grandhi. Report on fault tolerant synthesis through error correcting codes driven graph augmentation. Technical report, University College Cork, Ireland, January,2015.

[106] John A Darringer, William H Joyner, C Leonard Berman, and Louise Trevillyan. Logic synthesis through local transformations. *IBM Journal of Research and Development*, 25(4):272–280, 1981.

[107] Robert K Brayton, Richard Rudell, Alberto Sangiovanni-Vincentelli, and Albert R Wang. MIS: A multiple-level logic optimization system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(6):1062–1081, 1987.

[108] Satish Grandhi, Christian Spagnol, Jiaoyan Chen, Emanuel Popovici, and Sorin Cotafona. Reliability aware logic synthesis through rewriting. In *System-on-Chip Conference (SOCC), 2014 27th IEEE International*, pages 274–279, 2014.

[109] Satish Grandhi, David McCarthy, Christian Spagnol, Emanuel Popovici, and Sorin Cotofana. ROST-C: Reliability driven optimisation and synthesis techniques for combinational circuits. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 431–434, 2015.

[110] Nan Li and Elena Dubrova. AIG rewriting using 5-input cuts. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 429–430, 2011.

[111] Stanley L. Hurst, Jon C. Muzio, and D. Michael Miller. *Spectral Techniques in Digital Logic*. Academic Press, Inc., Orlando, FL, USA, 1985.

[112] Frédéric Mailhot and Giovanni De Micheli. Technology mapping using boolean matching and don't care sets. In *Design Automation Conference, 1990., EDAC. Proceedings of the European*, pages 212–216, 1990.

[113] Sabih H Gerez. *Algorithms for VLSI design automation*, volume 8. Wiley New York, 1999.

[114] Saeyang Yang. *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.

[115] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[116] Robert G Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[117] David JC MacKay and Radford M Neal. Near shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645–1646, 1996.

[118] Eldad Perahia. IEEE 802.11 n development: History, process, and technology. 46(7):48–55, 2008.

[119] Lav R Varshney. Performance of LDPC codes under faulty iterative decoding. *Information Theory, IEEE Transactions on*, 57(7):4427–4444, 2011.

[120] Mihir R Choudhury and Kartik Mohanram. Low cost concurrent error masking using approximate logic circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(8):1163–1176, 2013.

[121] Egor S. Sogomonjan and Michael Gössel. Design of self-parity combinational circuits for self-testing and on-line detection. In *Defect and Fault Tolerance in VLSI Systems, 1993., The IEEE International Workshop on*, pages 239–246, 1993.

[122] Salvador Manich, Michael Nicolaidis, and Joan Figueras. Enhancing realistic fault secureness in parity prediction array arithmetic operators by IDDQ monitoring. In *VLSI Test Symposium, 1996., Proceedings of 14th*, pages 124–129, 1996.

[123] Thomas J Richardson and Rüdiger L Urbanke. Efficient encoding of low-density parity-check codes. *IEEE transactions on information theory*, 47(2):638–656, 2001.

[124] Chu-Hsiang Huang, Yao Li, and L. Dolecek. Gallager B LDPC Decoder with Transient and Permanent Errors. *IEEE Transactions on Communications*, 62(1):15–28, 2014.

[125] Valentin Savin. Split-extended LDPC codes for coded cooperation. In *Information Theory and its Applications (ISITA), 2010 International Symposium on*, pages 151–156, 2010.

[126] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(1):379–423, 1948.

[127] Valentin Savin. Self-corrected min-sum decoding of LDPC codes. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 146–150. IEEE, 2008.

[128] Hendrawan Soeleman, Kaushik Roy, and Bipul C Paul. Robust subthreshold logic for ultra-low power operation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 9(1):90–99, 2001.

[129] Aseem Agarwal, David Blaauw, Vladimir Zolotov, Savithiri Sundareswaran, Min Zhao, Kaushik Gala, and Rajendran Panda. Path-based statistical timing analysis considering inter-and intra-die correlations. In *Proc. TAU*, pages 16–21, 2002.

[130] Xinghai Tang, Vivek K De, and James D Meindl. Intrinsic MOSFET parameter fluctuations due to random dopant placement. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 5(4):369–376, 1997.

[131] Balkaran S Gill, Chris Papachristou, and Francis G Wolff. Soft delay error analysis in logic circuits. In *Design, Automation and Test in Europe, 2006. DATE'06. Proceedings*, pages 1–6, 2006.

[132] Jiaoyan Chen, Sorin Cotofana, Satish Grandhi, Christian Spagnol, and Emanuel Popovici. Inverse gaussian distribution based timing analysis of sub-threshold CMOS circuits. *Microelectronics Reliability*, 55(12):2754–2761, 2015.

[133] https://www.opensourceliberty.org//.

[134] John F Croix and DF Wong. A fast and accurate technique to optimize characterization tables for logic synthesis. In *Proceedings of the 34th annual Design Automation Conference*, pages 337–340, 1997.

[135] Sandeep Miryala, Baljit Kaur, Bulusu Anand, and Sanjeev Manhas. Efficient nanoscale VLSI standard cell library characterization using a novel delay model. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1–6, 2011.

[136] Luciano Lavagno, Louis Scheffer, and Grant Martin. *EDA for IC implementation, circuit design, and process technology*. CRC press, 2006.

[137] Florentin Dartu, Noel Menezes, and Lawrence T Pileggi. Performance computation for precharacterized CMOS gates with RC loads. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(5):544–553, 1996.

[138] Jessica Qian, Satyamurthy Pullela, and Lawrence Pillage. Modeling the" effective capacitance" for the rc interconnect of CMOS gates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(12):1526–1535, 1994.

[139] Ravishankar Arunachalam, Florentin Dartu, and Lawrence T Pileggi. CMOS gate delay models for general RLC loading. In *Computer Design: VLSI in Computers and Processors, 1997. ICCD'97. Proceedings., 1997 IEEE International Conference on*, pages 224–229, 1997.

[140] Weiyu Chen, Sandeep K Gupta, and Melvin A Breuer. Analytic models for crosstalk delay and pulse analysis under non-ideal inputs. In *Test Conference, 1997. Proceedings., International*, pages 809–818, 1997.

[141] John F Croix and DF Wong. Blade and razor: cell and interconnect delay analysis using current-based models. In *Design Automation Conference, 2003. Proceedings*, pages 386–389, 2003.

[142] Florentin Dartu and Lawrence T Pileggi. Calculating worst-case gate delays due to dominant capacitance coupling. In *Proceedings of the 34th annual Design Automation Conference*, pages 46–51, 1997.

[143] Igor Keller, Ken Tseng, and Nisath Verghese. A robust cell-level crosstalk delay change analysis. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 147–154, 2004.

[144] Xin Wang, Ali Kasnavi, and Harold Levy. An efficient method for fast delay and SI calculation using current source models. In *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, pages 57–61, 2008.

[145] Ruijing Shen and Xiangqing He. A fast nonlinear timing analysis method for nanometer technologies. In *Communications, Circuits and Systems, 2007. ICCCAS 2007. International Conference on*, pages 1150–1153, 2007.

[146] Raj Chhikara. *The Inverse Gaussian Distribution: Theory: Methodology, and Applications*, volume 95. CRC Press, 1988.

[147] Peter Morters and Yuval Peres. *Brownian motion*, volume 30. Cambridge University Press, 2010.

[148] Gian Gerosa, Sonya Gary, Carl Dietz, Dac Pham, Kathy Hoover, Jose Alvarez, Hector Sanchez, Pete Ippolito, Tai Ngo, Suzanne Litch, et al. A 2.2 w, 80 mhz superscalar risc microprocessor. *Solid-State Circuits, IEEE Journal of*, 29(12):1440–1454, 1994.

[149] Ivan Edward Sutherland, Robert F Sproull, and David F Harris. *Logical effort: designing fast CMOS circuits*. Morgan Kaufmann, 1999.