FP7-ICT / FET-OPEN – 309129 / i-RISC

D5.2

Report on Fault Tolerant Synthesis through Error Correcting Codes Driven Graph Augmentation

Editor:	Emanuel Popovici
Deliverable nature:	Public
Due date:	January 31, 2015
Delivery date:	February 27, 2015
Version	2.0
Date of current version:	April 15, 2016
Total number of pages:	61
Reviewed by:	i-RISC members
Keywords:	Codeword Prediction Encoder, Fault tolerance, IC synthesis, Error Correction Control, VLSI, Soft error, reliability, single-event upset, single-event transient, fault-tolerance, Cut enumeration, Boolean Matching, Multi-Objective Optimization.

Abstract

This deliverable presents an overview of the work carried out in relation to Work Package 5 (WP5) during the second year of the project. The document reports all the activities aimed to develop error coding driven graph augmentation (Task 5.3), a first step towards fault-tolerant circuit design. A novel method to design fault tolerant circuitry with prime focus on improving the fault tolerance capability of combinatorial logic by means of error correction codes is discussed. Mathematical analysis on the Boole Shannon limit (Task 5.5) for linear circuits is also articulated. Different methodologies aimed at multi objective optimization (Task 5.4) are detailed, explaining the combinational circuit optimization techniques for different constraints like reliability, power & delay.

List of Authors

Participant	Author
UCC	Emanuel Popovici (e.popovici@ucc.ie) Satish Grandhi (sagrand@ue.ucc.ie) Christian Spagnol (Christian.spagnol@ue.ucc.ie) Jiaoyan Chen (chenj@ue.ucc.ie) David McCarthy (davidmc@ue.ucc.ie)
CEA	Valentin Savin(valentin.savin@cea.fr)
TUD	Sorin Cotofana(s.d.cotofana@tudelft.nl)
UPT	Alexandru Amaricai (alexandru.amaricai@cs.upt.ro)

Table of Contents

List of Dissemination Activities

Accepted Papers

[P1] S. Grandhi, C. Spagnol, and E. Popovici, "Reliability analysis of logic circuits using probabilistic techniques," *10th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp.1-4, June 30-July 3, 2014.

[P2] J. Chen, C. Spagnol, S. Grandhi, E. Popovici, S. Cotofana, and A. Amaricai, "Linear Compositional Delay Model for the Timing Analysis of Sub-Powered Combinational Circuits," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp.380-385, July 9-11, 2014.

[P3] S. Grandhi, S. Spagnol, J. Chen, E. Popovici, and S. Cotafona, "Reliability aware logic synthesis through rewriting," *27th IEEE International System-on-Chip Conference (SOCC)*, pp.274-279, Sept. 2-5 , 2014.

Submitted Papers

[P4] J. Chen, C. Spagnol, S. Grandhi, E. Popovici, and S. Cotofana, “Inverse Gaussian Distribution Based Timing Analysis of Sub-threshold CMOS Circuits“, *Microelectronics Journal*. {**Submitted**}

[P5] S. Grandhi, D. McCarthy, E. Popovici, and S. Cotofana, “Studying the impact of Technology Mapping on the Reliability of Combinational circuits”, *IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2015 {**Submitted**}

[P6] S. Grandhi, D. McCarthy, C. Spagnol, J. Chen, and E. Popovici, “A CAD Framework for Reliability Estimation and Optimization of Combinational Circuits”, *Journal of Low Power Electronics*. {**In preparation**}

List of Figures

List of Tables

Abbreviations

BCH	Bose Chaudhuri Hocquenghem
BER	Bit Error Rate
BSC	Binary Symmetric Channel
CAD	Computer Aided Design
CDF	Cumulative Distributive Function
CPE	Codeword Prediction Encoder
CT	Criticality Threshold
DES	Data Encryption Standard
DRM	Dynamic Reliability Management
DWAA	Dynamic Weighted Average Algorithm
ECC	Error Correcting Codes
EDA	Electronic Design Automation
FEC	Forward Error Correction
FER	Frame Error Rate
IC	Integrated Circuit
IP	Intellectual property
LDPC	Low Density Parity Codes
LDGM	Low Density Generator Matrix
MCNC	Microelectronics Centre of North Carolina
NBTI	Negative Bias Temperature Instability
PI	Primary Input
PO	Primary Output
RMR	R-fold Modular Redundancy
SER	Soft Error Rate
TMR	Triple Modular Redundancy
VLSI	Very Large Scale Integration
WP	Work Package

1. Executive Summary

In the second year of the i-RISC project (M13-M24), Work Package 5 (WP5) addressed three important research problems: Fault tolerant techniques employing the traditional Error Coding Driven Graph Augmentation (Task 5.3), multi objective gate level optimization with the aim of improving circuit reliability (Task 5.4) and developing mathematical framework for computing the Boole-Shannon limit for noisy circuits (Task 5.5).

Work Package (WP) 5 has been tasked with two major contributions towards the i-RISC goals:

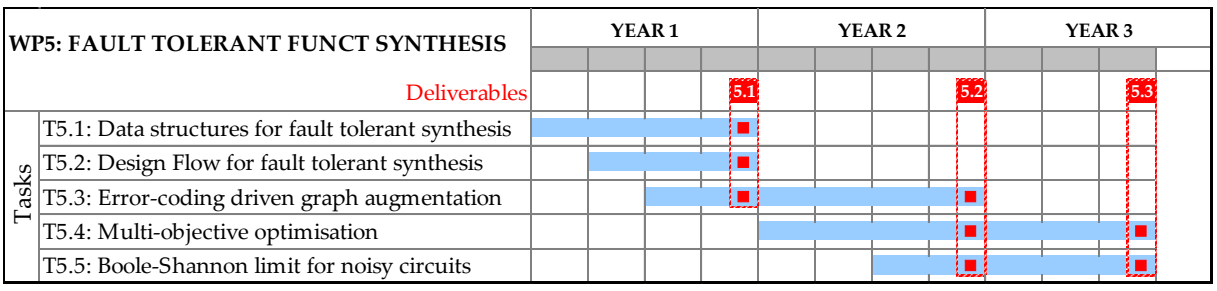
- The **first WP5 goal** is the achievement of systematic synthesis and optimization of reliable circuits, culminating with a multi-objective circuit design optimization, with respect to its area, delay, power, and all driven by reliability. The introduction of reliability as circuit figure of merit leads to a 4-dimensional (area, delay, power, reliability) solution space, and has a tremendous influence on the complexity of the synthesis process.
- **The second WP5 goal** is a fundamental study on the effectiveness of integrating error-correcting codes into the structural (Boolean network) implementation of the circuit logical functionality. More precisely, this aims at researching on potential links between the logic representation of a digital circuit and error correcting codes in order to generate fault tolerant implementation of the logical functionality of the circuit.

To give a quick recap of the first year research study, the following issues have already been addressed:

- We analyzed a number of data structures and singled out the And-Inverter-Graph (AIG) as the most appropriate for our goals due to its compactness, versatility to incorporate many parameters of concern (switching activity, delay, reliability, etc.), and scalability to any circuit size.
- A primitive design flow has been proposed combining custom tools, academic tools with more established/industry accepted tools that would allow evaluation and validation of our designs.
- Gate level reliability aware logic optimization techniques have been presented.
- We reported an initial framework for the synthesis of chips made from unreliable components, through the concept of error correcting codes driven graph augmentation.

A Gantt chart of WP5 tasks and their time distribution, which indicates the tasks addressed and initiated during the period M13-M24, is presented below:

- ✓ Task 5.3 : Fault tolerant techniques employing error coding driven graph augmentation
- ✓ Task 5.4 : Reliability heavy multi objective logic optimization techniques
- ✓ Task 5.5 : Mathematical analysis to define Boole-Shannon limit for noisy circuits



The main technical contributions presented in this deliverable are summarized as follows:

- **A CAD framework completely automating the flow of systematic synthesis of fault tolerant combinational circuits (Task 5.3):** A novel methodology to design fault tolerant circuitry by employing the error correction codes mechanism that is generic and is applicable to any combinatorial logic. The study presented here focuses mainly on encoding aspects, adapted from the field of forward error correction, for protecting fault prone Boolean functions. All the scripts developed have been completely integrated unto the traditional VLSI EDA design flow as reported in the i-RISC deliverable D6.1.
- **Performance analysis and plotting results using Codeword Prediction Encoder (CPE) simulator (Task 5.3):** To evaluate the performance of the CPE methodology, we have developed simulator in 'C'. It converts any input file unto a proprietary internal format and process the circuit for different characteristics. Different types of LDPC decoders are employed to determine the best possible configuration that provides the least possible error on the output node. We introduce the novel concept of "**Critical Nodes**" which are basically the high octane nodes that have to be safe guarded under all circumstances. This work also presents results from the application of the methodology on various benchmark circuits as well as IP cores and investigation of various error correction codes.
- **Pre-Dominantly Reliability driven logic optimization along with power and timing optimization techniques (Task 5.4):** We propose a highly accurate reliability estimation methodology developed in line with the traditional simulated based power estimation techniques. Mersenne twister is used to generate highly random set of input patterns. Further, we also introduce a cut-enumeration based reliability driven optimization technique. And-Inverter graph based rewriting algorithm using 4-input cuts are employed to introduce logically equivalent modifications unto the input circuit representation. Initial simulation results report up to 10% improvement in circuit reliability values.
- **Mathematical approach to define the Boole-Shannon limit for noisy circuits (Task 5.5):** Theoretical analysis describing the goodness of the Codeword Prediction Encoder (CPE) and its performance limits are presented. We consider all the four parameters: Area, Timing, Power and Reliability to evaluate the improvement achieved by employing the newly proposed scheme. Currently, we limit our analysis of computing the extra cost incurred due to the CPE approach to linear circuits.

The deliverable is organized as follows: Chapter 2 is dedicated to the fault tolerant techniques through graph augmentation by means of Codeword Prediction Encoder (CPE). All the simulation results and the performance analysis through CPE simulator are also reported. Chapter 3 describes the mathematical analysis of the CPE approach its performance limits. Chapter 4 introduces a new highly accurate reliability estimation technique based on simulation based methodologies. The Multi-objective logic optimization technique is also detailed out here. The last chapter is dedicated to concluding remarks and future work.

2. Error Coding Driven Graph Augmentation (Task 5.3)

Abstract: The continuous scaling of the transistor length has resulted in significant increase in the soft error rate in combinational circuits. As a result, fault tolerant techniques that improve circuit reliability are the need of the hour. Traditional fault tolerant techniques analyze the circuit reliability issue from a static point of view neglecting the dynamic errors. This work proposes a novel methodology to design fault tolerant circuitry by employing the error correction codes mechanism. The method is generic and is applicable to any combinatorial logic. The study presented here focuses mainly on how to generate redundancy for protecting fault prone Boolean functions. The idea is adapted from the field of forward error correction for telecommunications and in particular this section will focus on both encoding aspects and error correction capabilities. We further present results from the application of the method in various IP cores and investigation of various error correction codes.

2.1. Introduction

This work introduces a novel reliability driven fault tolerant methodology for combinational circuits by augmenting extra logic to correct some of the errors. Fault tolerant techniques for improving reliability of digital circuitry have been of interest from long time. Von Neumann **[Neumann56]** first introduced a classification of the error type and proposed solutions based on multiplexing techniques as early as 1956. One of the most fundamental and effective fault-tolerant mechanisms introduced in that work was the well-known R-fold modular redundancy (RMR), 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 (TMR) has been widely used in the design of systems where reliability is considered a key issue. However, despite the wide 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. 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 particular 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.

2.2. Low Density Parity Check (LDPC): Overview

Low-density parity check (LDPC) codes are a class of linear block codes and were invented by Gallager **[Gallager 63]**. Since the proposed class of codes required that the parity-check matrix representing the code has a small number of ones to be efficiently decoded, the codes have been called low density parity check codes. Like all linear block codes they can be described via matrices. The second possibility is a graphical representation. The matrix defined in Figure 2-1 is a parity check matrix with dimension $m \times n$ for a (8, 4) code. Two important conventions used in describing LDPC codes are: w_r for the number of 1´s in each row and w_c for the columns. For a matrix to be called low-density, the two conditions $w_c << n$ and $w_r << m$ is a must.

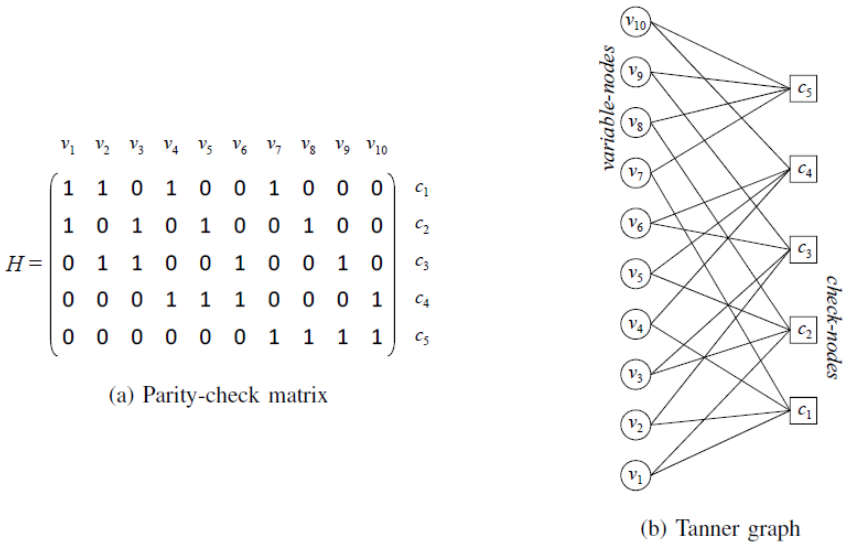


Figure 2-1: Example of parity-check matrix and corresponding bipartite Tanner graph

Tanner introduced an effective graphical representation for LDPC codes as shown in Figure 2-1. Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes in a Tanner graph are called variable-nodes (v-nodes) and check-nodes (c-nodes). The representation of the code by means of the Tanner graph gives an underlying structure that facilitates the decoding process. The message y received from the channel is the codeword x sent corrupted by some error e that has been added by the channel. Since the channel model used is the 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. We further denote the received syndrome:

$$z = Hy = Hx + He \tag{2.1}$$

From the property of linear block codes every codeword satisfies $Hx = 0$, hence the syndrome is $z = He$. The decoding problem consists of finding the most probable vector e that explains the observation of such a syndrome. For LDPC codes, decoding can be implemented by message passing algorithms that exchange messages between v-nodes and c-nodes, as shown in Figure 2-2. For more details, the reader may refer to **[i-RISC/D3.1]**.

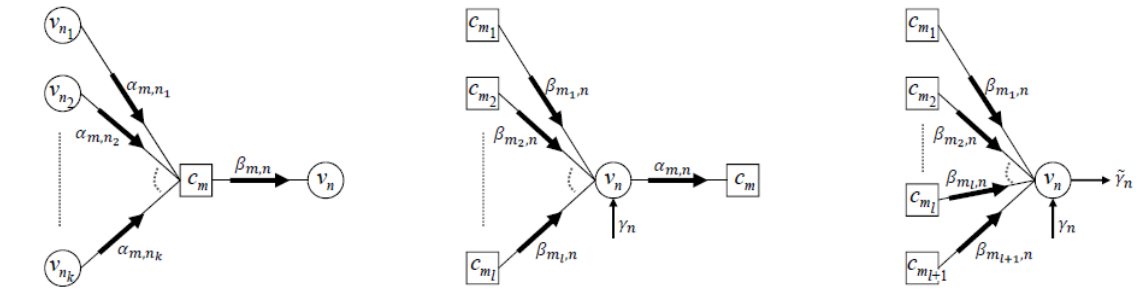


Figure 2-2 : Computation of extrinsic messages and of the a posteriori information

2.3. Codeword Prediction Encoder (CPE) for Fault Prone Boolean Functions

The approach presented in **[i-RISC/D5.1]** attempts to reduce the error probability on the output of a combinatorial circuit by choosing an optimal realization of the Boolean function under investigation. 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 particular Boolean Function. The focus of this approach is not on changing the combinational logic but on augmenting extra combinational logic to enable the retrieval of the correct output even if errors have occurred. The main challenges include:

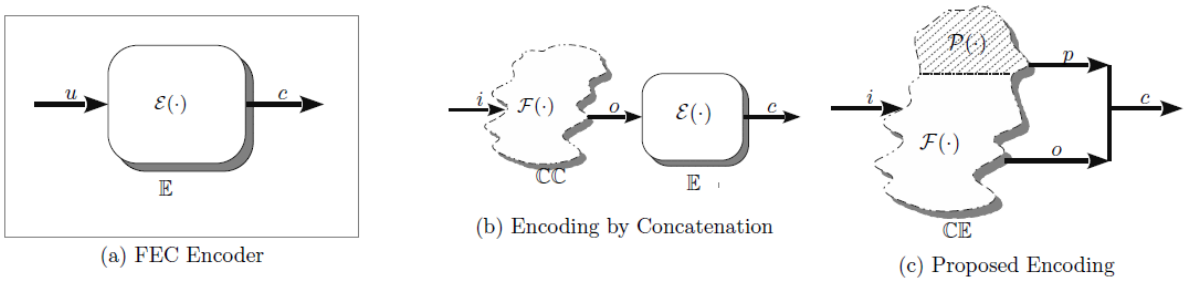
- How to implement such a system capable of taking advantage of the added redundancy?
- How to do it in an optimal way (minimum amount of redundancy) add minimum amount of redundancy to correct the given number of errors?

Important work to cross the field of circuit design with the knowledge of error correction theory has been done by Taylor **[Taylor-68a, Taylor-68b]** that used LDPC codes and faulty memories and logic, to build fault tolerant architectures for reliable systems. Taylor's approach has been the base for numerous other works **[Kuznetsov73, adjicostis05, Vasic07]**. The approach presented here can be seen as an expansion of the Check Symbols Generation **[Touba97] [Mohanram03]** and the Parity Prediction Function **[Sogomonjan93] [Manich96] [Ko01]**, where circuitry is added to a combinatorial network to generate extra bit to ensure parity. We formalize these approaches and extend them to take full advantage of the power of error correction codes to enable correction of the faults not just detection.

2.3.1. Proposed Scheme

Forward error correction methods generate a codeword $c \in \boldsymbol{C}$, with $\boldsymbol{C}$ being the particular code being used (e.g. an LDPC code), by encoding an information message u. We call the encoder $\mathbb{E}$. Assuming without loss of generality that the encoder is systematic the codeword c is composed by u and several extra symbols called the parity p.

We look for a way to apply such procedure on an unreliable combinatorial circuit $\mathbb{CC}$ with inputs i and outputs o. The simplest solution would be to encode the outputs o. However this approach suffers from the fact that if a fault incurs in $\mathbb{CC}$, then input of the encoder is $\tilde{o}$ (the error corrupted output) that gives $\tilde{c}$ after encoding. $\tilde{c}$ may or may not be a codeword of $\boldsymbol{C}$, depending of the presence of faults in the encoder logic, however no decoder will be able to retrieve i from it. Another disadvantage of the scheme is that from hardware prospective the concatenation of $\mathbb{CC}$ and $\mathbb{E}$ increases the critical path and hence limits the throughput. We propose to combine the encoding process in the combinatorial circuit to ensure that the outputs o of the new combinatorial logic $\mathbb{CE}$ is a codeword as depicted in Figure 2-3. In this new scenario, $\mathbb{CE}$ computes not only the outputs o but also a set of parity p that guarantee$[o|p] \in \boldsymbol{C}$, (systematic encoding is used). Different from the previous approach, in this case the parity p is computed directly from the inputs i.



(a) FEC Encoder

(b) Encoding by Concatenation

(c) Proposed Encoding

Figure 2-3: Methodology of Codeword Prediction Encoder

To understand how the new combinatorial logic is obtained, lets define as $\mathcal{E}(\cdot)$ the function computed by $\mathbb{E}$ and let's assume that we are working with binary codes, then $\mathcal{E}: GF(2)^k \rightarrow GF(2)^n$, where k is the dimension and n is the length of the code $\boldsymbol{C}$. The number of outputs of the combinatorial circuit $\mathbb{CC}$ is also assumed to be equal to k. The functionality of $\mathbb{CC}$ can be described as a function $\mathcal{F}(\cdot)$ with $\mathcal{F}: GF(2)^l \rightarrow GF(2)^k$ where l is the number of binary inputs and k is the number of binary outputs. The function $\mathcal{P}(\cdot)$ that maps the inputs directly to the parity is $\mathcal{P}: GF(2)^l \rightarrow GF(2)^m$ and is the composition $\mathcal{E}(\mathcal{F}(\cdot))$. This can be seen simply by considering how its result must be equivalent of computing $\mathcal{F}(\cdot)$ and then $\mathcal{E}(\cdot)$ 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 $\mathcal{P}(\cdot)$ **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 $\boldsymbol{G}$, the composition $\mathcal{E}(\mathcal{F}(\cdot))$ is then equivalent to:

$$\mathcal{P}(j) = \sum_{b=0}^{k} \mathcal{F}_{b(j)} \boldsymbol{G}(b,j) \quad j \in \{0, \dots, m\} \tag{2.2}$$

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

2.4. CPE Simulator & CAD Automation

To implement the CPE methodology and evaluate its performance, we have automated the complete methodology. Quite a few scripts were developed to initially convert the traditional verilog netlists into proprietary format that is easy to parse by the CPE simulation engine. The whole automation comprises of '2' steps, namely:

- Pre-Processing
- Testing Using CPE simulator

In order to understand the whole process, it is imperative to explain the internal proprietary format that we have employed. We shall also introduce the important term "Criticality Threshold".

2.4.1. Netlist Format

Any circuit in VLSI chip design is represented as collection of gates written a file generally called as the netlist. Since CPE simulator is a tool completely developed in C-language, it cannot understand the terminology of gates. Hence, we have developed an internal proprietary format to represent the verilog netlists. The Verilog list may correspond either to the original circuit F or to the parity circuit P. Each gate in the circuit is converted into a node within the internal format.

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 *indegree* 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 defined as the number of its successors
  - output nodes must have outdegree equal to zero

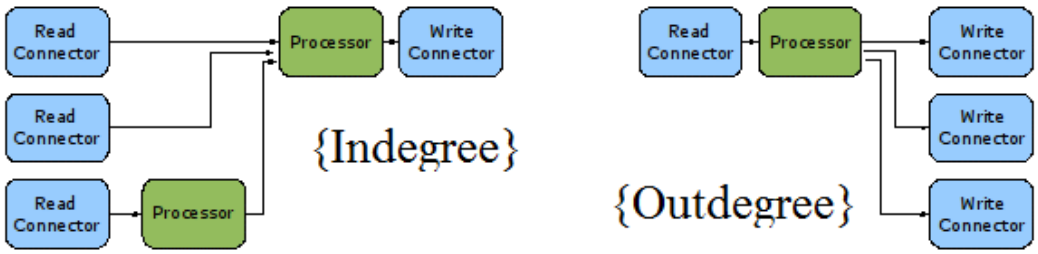


Figure 2-4: Indegree and Outdegree graphical representation.

The following convention is used to represent all possible gates:

0 = NOT, 1 = AND, 2 = OR, 3 = XOR, 4 = NAND, 5 = NOR, 6 = XNOR

Nodes of type 0 must be of indegree = 1. Nodes of type 2-6 must be of indegree >= 2.

The following table depicts a snapshot of small netlist representation:

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

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

2.4.2. Fault Injection

The internal format of circuit representation allows simulating the netlist in C. Furthermore, we can inject errors by flipping the output of each gate with a pre defined error probability on each gate. 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 similar to the one described in deliverable D5.1 **[i-RISC/D5.1]**. It is called as "*Gate output probabilistic mutant*" – it alters the gate output with a given probability.

2.4.3. Criticality Threshold

Some particular gates of the circuit may be *critical*, in the sense that injecting only one error at the output of such a gate may result in a very large number of errors at the output of the circuit. This happens when the output of the gate is propagated to a large number of outputs of the circuit. Such error events are characterized by a very high number of errors on the output of the circuit and cause decoding failures with very high probability. In order to identify these gates at the design time, we use the graphical description of the netlist introduced above, and proceed as described below.

The criticality degree of a node X, denoted by $cdeg(X)$, is defined as the number of output nodes to which X is connected by at least one path. Thus, injecting an error in node X, may produce at most $cdeg(X)$ errors on the output. In our simulations, we fix a *criticality threshold* (CT): Nodes X with $cdeg(X) > CT$ are considered to be "protected" (e.g. by increasing area), so as to make then reliable (error-free). Hence, errors are injected only in nodes X with $cdeg(X) \leq CT$. For example,

- For example, fixing CT = 5, means that errors are injected only in those nodes that are connected to at most 5 output nodes (cdeg(n) ≤ 5)
- A particular case is CT = -1, which means that all nodes are error-prone (no "protected" nodes)

2.4.4. The CAD flow

There are a number of scripts that were developed which perform pre-processing of all the input files before they can be run through the CPE simulator. Figure 2-5 shows the top level representation of the complete tool.

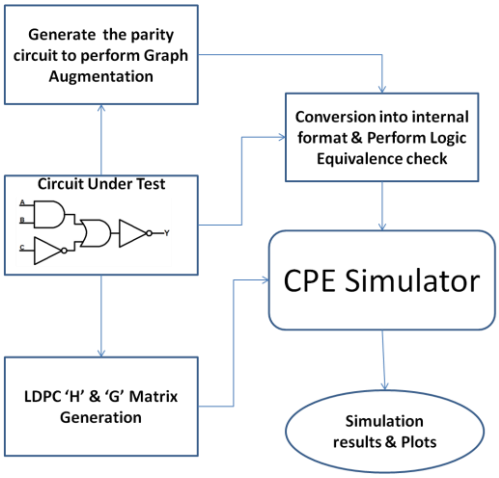


Figure 2-5 : The graph augmentation tool flow

- Based on the codeword length & the LDPC rate, we generate the Generator matrices. Similarly, the reference combinational logic, either the MCNC benchmark circuits or the standalone IP's are also converted into the internal format which is easily understood by the CPE simulator.

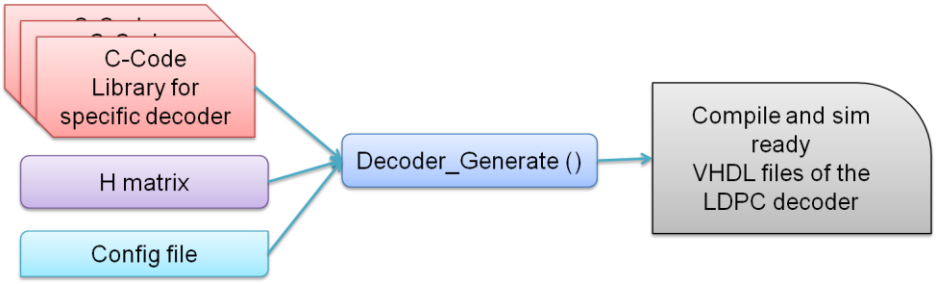


Figure 2-6: LDPC Decoder Architecture within CPE Framework

- A number of consistency checks are performed in order to make sure 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 ($l \geq 2$) are those whose predecessors are either input nodes or have been processed during stages 1, …, $l-1$.
    - If a processing order cannot be find consistency check error.
- The CPE simulator is the framework developed to automate the process of simulating the standard MCNC benchmark circuits. It accepts all the input files,
  - The combinational circuit netlist called 'F'
  - The parity circuit netlist called 'P'
  - The LDPC generator matrix

  and propagates the error through the netlist to simulate the final the BER & FER values. Figure 2-7 shows the top level architecture of the CPE tool :
  - An SRC module which generates the values of the inputs
  - The output of the SRC goes into a module 'F' and a second module 'P' – these modules are simulating the corresponding netlists, including the error injection at the gate level.
  - The outputs of F and P modules go into the decoder module.
  - Finally, output from the decoder goes into the BER/FER estimation module, where it is compared to the original input generated by the SRC module.

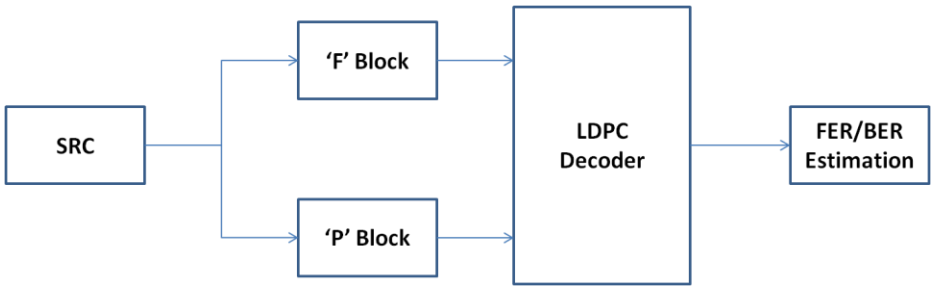


Figure 2-7: CPE tool top level representation

2.5. Experimental Results

A number of benchmark circuits have been simulated using the cad setup for various different constraints. It should be mentioned here that all the simulation results presented in this section assumes that the decoder is run on reliable hardware (error free hardware). This is referred to as asymmetric setting where in the circuit and the corresponding augmented logic is error prone and we assume that the decoder is fault free. Going forward, we would like to perform analysis for the symmetric case as well.

2.5.1. Critical Nodes

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). We have injected only one error in 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 Figure 2-8, 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 10 errors on the output which is acceptable. However, for some gates, the error injection can generate up to 191 errors on the output. Hence, we conclude that decoder failures are due to a very large number of errors on the output of F (or P) even if the gate error probability is quite low.

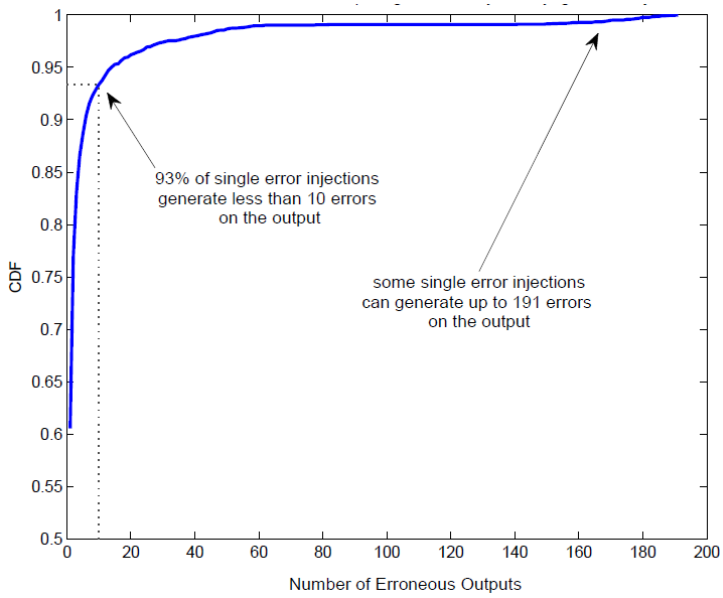


Figure 2-8: CDF of the number of erroneous outputs generated by one single error injection

We have further moved on and computed the criticality degrees of all nodes in F and P. As depicted in Figure 2-9, only 1.67% of the nodes of F have a criticality degree > 8 (which is good). However, for P the situation is completely different. There are many nodes with very high criticality degree (38.8% of nodes have criticality degree > 50). We cannot protect all these nodes, because that would not make sense. There seems to be some kind of trade-off popping up here: 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 (less nodes, but more critical); we are in the process of finding the optimal trade-off between number of nodes and criticality.

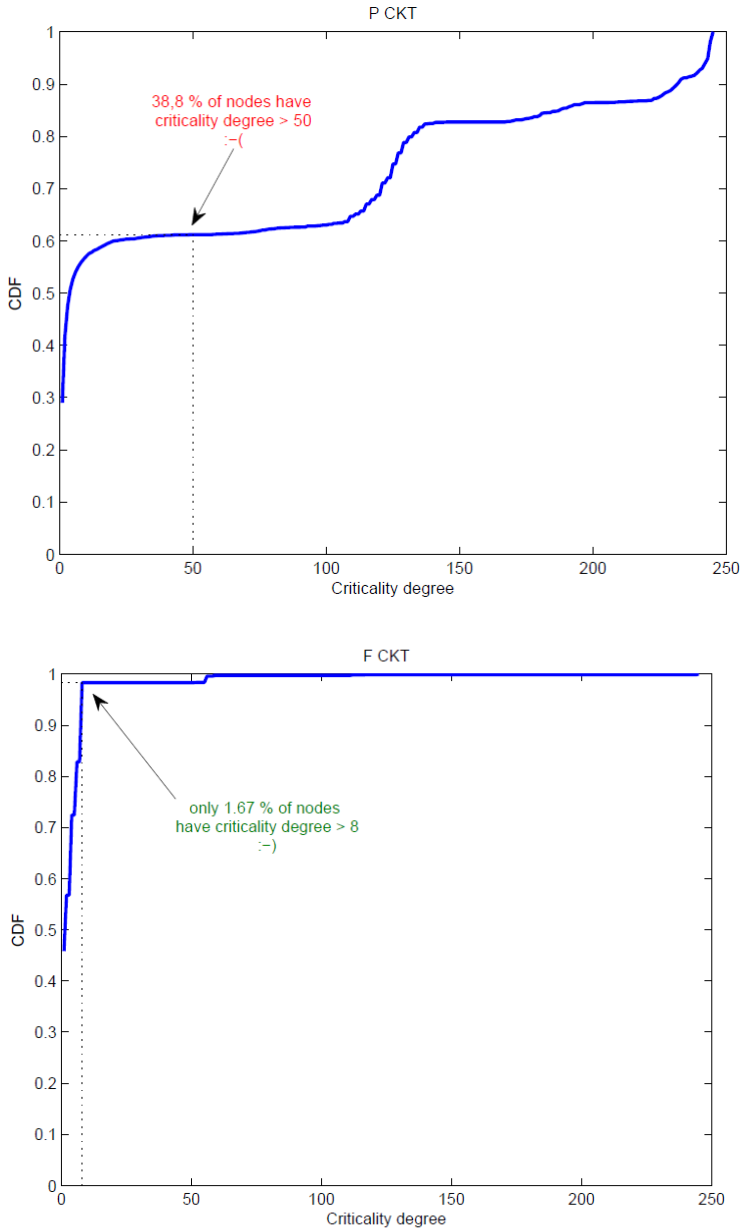


Figure 2-9 : CDF plot of Criticality degree of 'F' & 'P' Circuits.

As an important of point, the number of critical nodes is currently a bigger problem within the nonlinear circuits compared to linear circuits. As depicted in Figure 2-10, 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. We concatenate the functional unit with the generator matrix which is not the best possible approach. We are looking at different ways to overcome this issue.

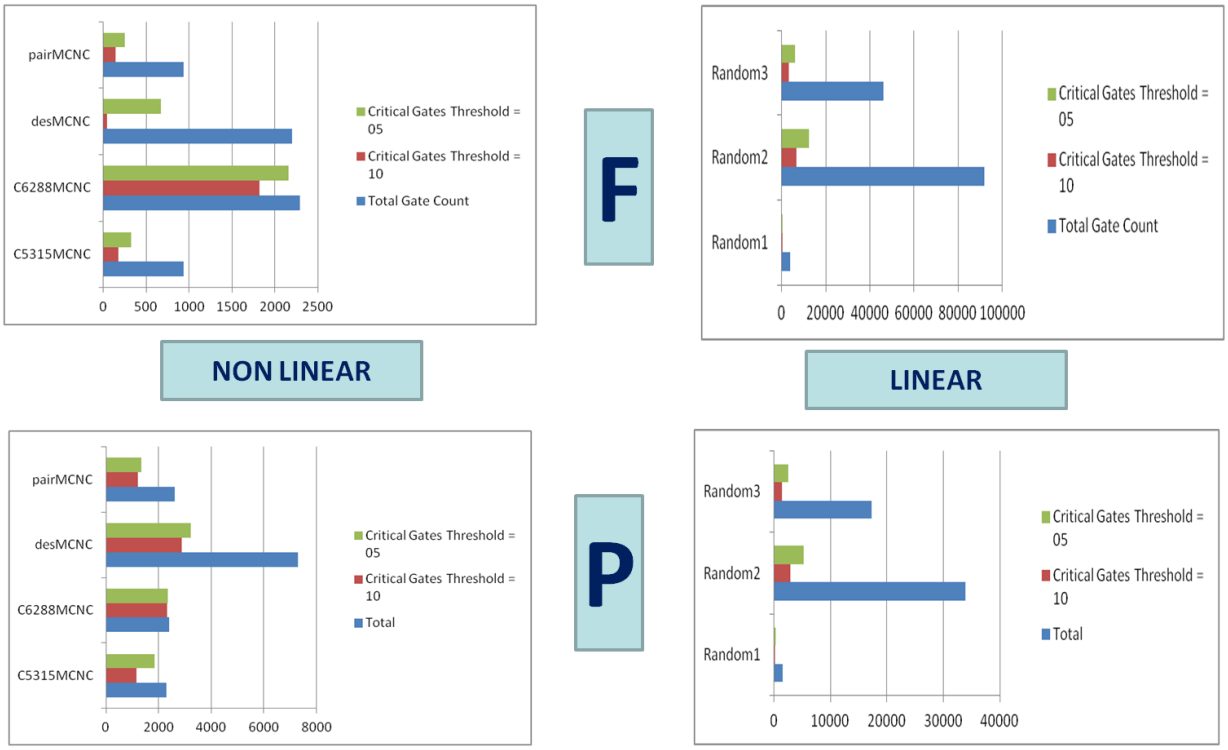


Figure 2-10: Critical node count for different Linear and Non-Linear circuits.

2.5.2. Area Overhead

Any error correction technique comes with the extra overhead of the parity circuit that needs to augmented to the existing logic circuit. As shown in Figure 2-11, for linear circuits, the size of the parity circuits is much smaller. This is because we multiply two matrices to generate the new circuit. But, in the case of non linear circuits, the overhead is slightly on the higher side. Currently, we are in the middle of coming up with better methodologies to reduce the size of the parity circuit.

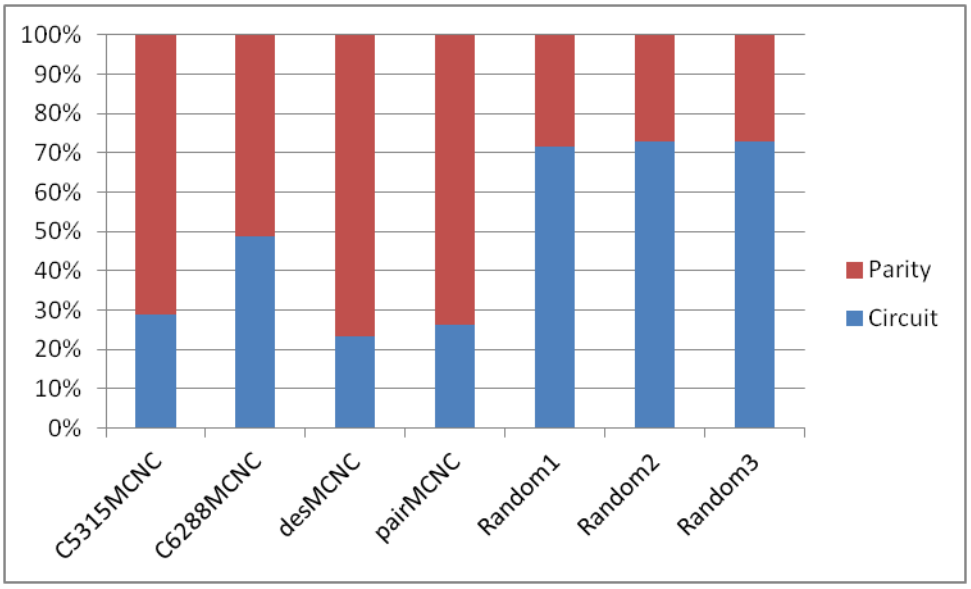


Figure 2-11: Area overhead due to parity augmentation.

2.5.3. NMR Vs CPE

Figure 2-12 plot portrays the bit error probability of the CPE and TMR methodologies for the MCNC benchmark circuit 'DES'. In the simulator, we have declared all the gates whose failure generates more than 10 errors on the output of F or P 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 5 orders of magnitude!! Note that in order to achieve a bit error probability of 4e-9 using N-modular redundancy, one should take N = 11!

We are currently looking at solutions to turn the critical gates into always reliable. One possibility would be to use modular redundancy for these gates which means that each gate is repeated N times (say 3 times) + a majority logic gate. Is this realistic? The other alternative is to make sure we define a different voltage island for all these critical gates so that they are powered up by higher voltage.

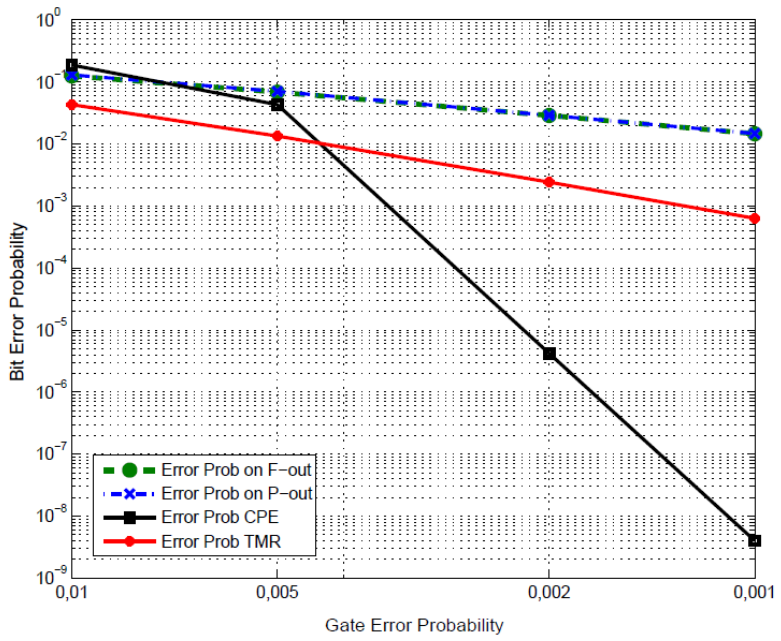


Figure 2-12: Performance of BER Vs CPE

2.5.4. Hardware Impact Investigation

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 **[Taylor68-a]**. These results have been already documented in **[iRISC – D5.1]** and only the final results are reported for the sake of completion. 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 in terms of 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 Figure 2-13. 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.

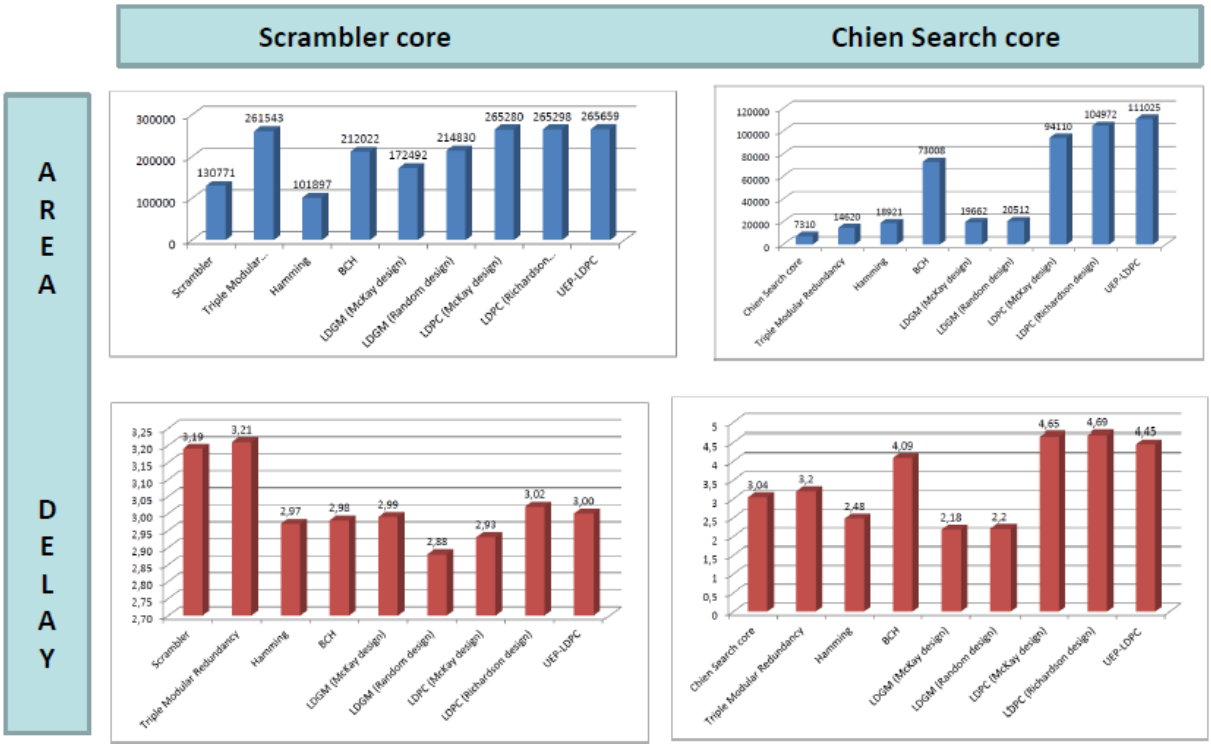


Figure 2-13: Area & timing Analysis on IP cores using CPE methodology.

2.5.5. Case Study

We have selected MCNC benchmark circuit "DES" as our test case and performed various simulations to study our methodology. The numbers of critical nodes for different criticality threshold (CT) levels are tabulated in Table 2-1. 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 that the total number of nodes of F (3249). Thus, it doesn't really make sense to use CPE under such circumstances, because a more simple and efficient solution would be to protect all the nodes in F.

Table 2-1: Protected Nodes Vs Criticality Threshold

Criticality Threshold	Number of protected nodes F CKT	Number of protected nodes P CKT
CT = -1	0 (0%)	0 (0%)
CT = 2	1299 (39,98%)	10553 (25,64%)
CT = 5	841 (25,88%)	6891 (16,75%)
CT = 8	75 (2,31%)	5911 (14,36%)
CT = 10	75 (2,31%)	5524 (13,42%)

Figure 2-14 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 due to the fact that the decoder converges to a wrong codeword. But with CT= 2, 5, 8, 10; the decoder never converged to a wrong codeword. Figure 2-15 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.

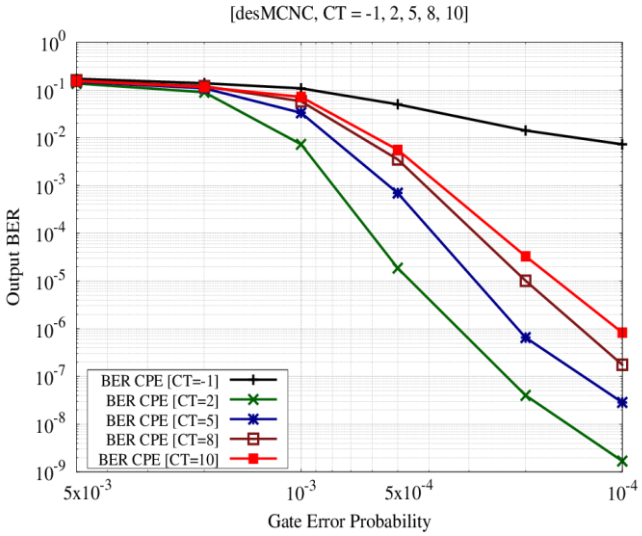


Figure 2-14: Output BER for various Criticality thresholds

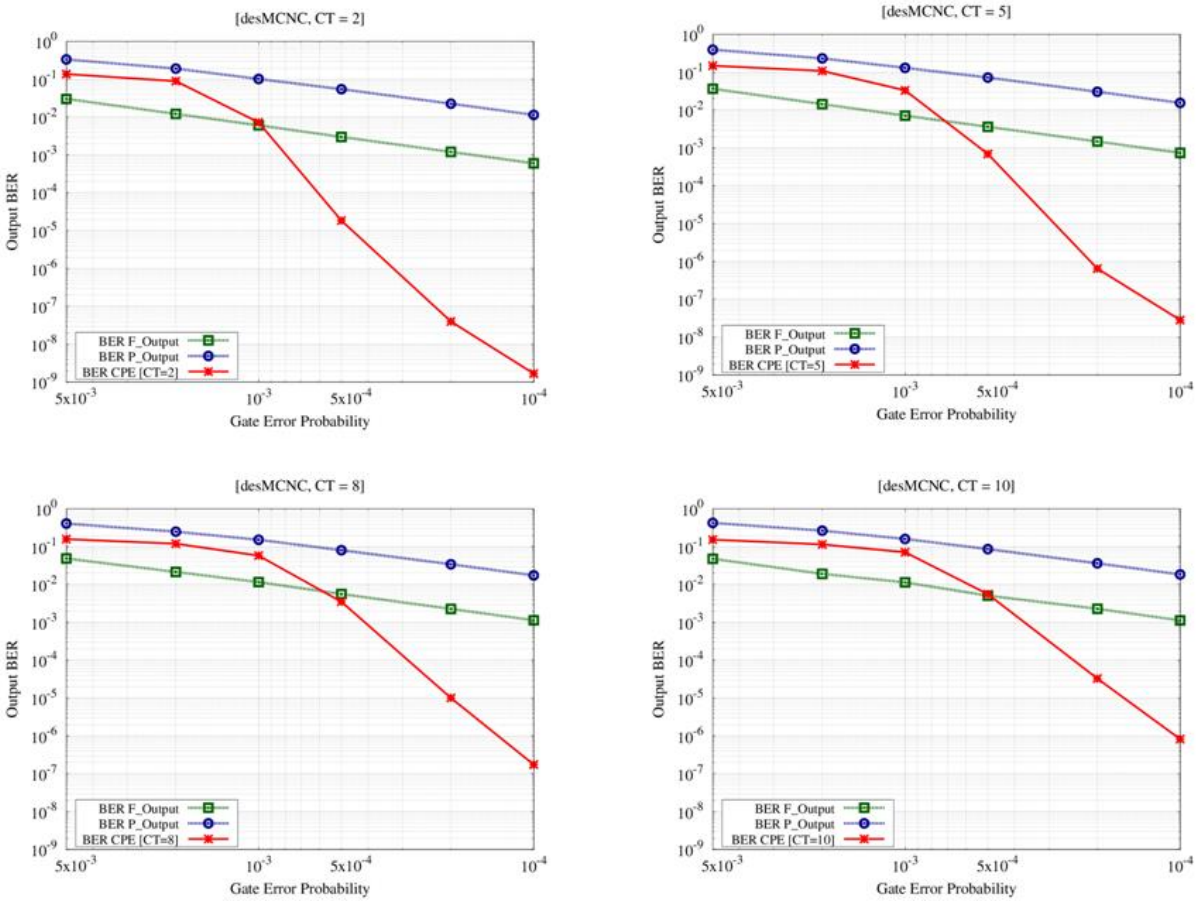


Figure 2-15: Detailed plots for output error on F, P and decoder output nodes.

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 results in an area significantly larger which is not really desirable. Size of P must be considerably reduced, in order for CPE to provide an effective solution.

2.5.6. Impact of LDPC code sizes on Area

Throughout this section, by *size* of a circuit we mean the number of the elementary logic gates that have to be interconnected to form the circuit function. Also this discussion is limited to linear circuits at this moment and would be extended to non linear circuits in the future. The size of the fault-tolerant CPE implementation is the sum of the combinational circuit 'F' , the parity circuit 'P' and the decoder 'D'. It is given by:

$$\text{Size}(\textbf{CPE}) = \text{Size}(F) + \text{Size}(P) + \text{Size}(D) \tag{2.3}$$

Although LDPC decoding algorithms consist of iterative message passing procedures, in practical implementations only the circuitry corresponding to a decoding iteration has to be instantiated in hardware. In this case, the size of the decoding circuit is known to increase linearly with the size of the decoding input. Thus, assuming that the rate k/n of the code is constant, we can write Size(D) = $\mathcal{O}(n)$ = $\mathcal{O}(k)$. If F is a random matrix, the size of the combinatorial circuit implementing F increases linearly with $l \times k$. Therefore, assuming that the ratio l/k of matrix dimensions remains constant, we can write Size($\mathbb{CC}$) = $\mathcal{O}(l \times k)$ = $\mathcal{O}(k^2)$. Similarly, we have Size($\mathbb{S}$) = $\mathcal{O}(k^2)$. Moreover, since $\mathbb{CC}$ and $\mathbb{S}$ are of comparable sizes, which asymptotically dominate the size of the decoding circuit D, we can (asymptotically) approximate the size of the fault-tolerant CPE implementation by:

$$\text{Size}(\textbf{CPE}) \approx 2 \times \text{Size}(\mathbb{CC}) \tag{2.4}$$

A higher rate code can result in significant reduction in the size of the 'P' circuit and hence the overall area of the implementation as shown in Figure 2-16.

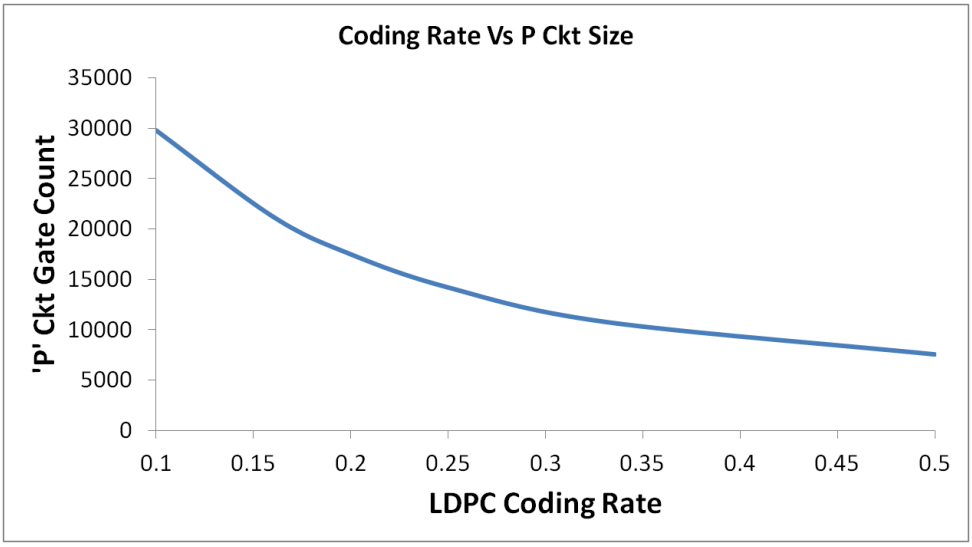


Figure 2-16: Area Vs Coding Rate Plot

But, on the positive note, higher rate codes can result is less number of critical gates as shown in Figure 2-17. Hence, it is a tradeoff between area vs. performance. A higher rate code occupies less area but it has less capability of correcting errors. Similarly, a low rate code increases the size of the parity circuit drastically but improves the performance of the circuit from a reliability perspective.

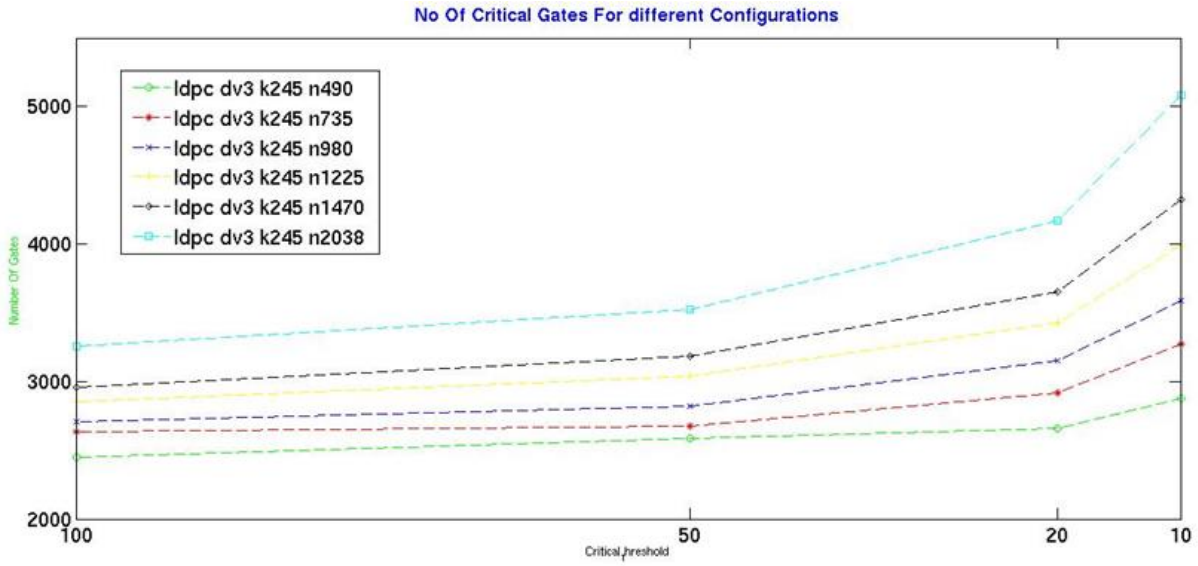


Figure 2-17: Critical Gate Count Vs Coding Rate

2.5.7. Validation on MCNC Benchmark Circuits

We have tested the CPE methodology on a selected of non linear MCNC benchmark circuits and a set of randomly generated linear circuits. Table 2-2 to Table 2-7 describe the simulation results for the following six scenarios:

- Gate error set of 1e-03, 1e-04, 1e-05
- Critical Gate threshold set of '5' & '10'

In Table 2-2 to Table 2-7, column 1 describes the circuit under consideration. Column 2 and 3 provide the gate count of both the 'F' {Circuit under test} and the 'P' {parity logic} circuit. Column 4 and 5 provide the critical gate count of both the 'F' {Circuit under test} and the 'P' {parity logic} circuit. Column 6, 7 and 8 provide the error probability on the output node of the 'F' {Circuit under test}, the 'P' {parity logic} circuit and the decoder.

Table 2-2: Gate Error = 1.00e-03; Critical Gate Threshold = 10 {Case1}

Benchmark	Gate Count		Critical Gates		Error Probability		
	F	P	F	P	out-F	out-P	CPE
C5315MCNC	936	2307	176	1151	6.23e-03	2.26e-02	5.947e-03
C6288MCNC	2290	2408	1822	2326	1.97e-02	9.33e-03	1.284e-02
desMCNC	2204	7292	41	2877	8.05e-03	4.00e-02	2.921e-04
pairMCNC	932	2617	139	1198	3.53e-03	2.55e-02	4.917e-04
Random1	4007	1578	306	132	8.66e-03	1.00e-02	3.493e-02
Random2	91905	33926	6732	2839	1.58e-01	1.77e-01	1.614e-01
Random3	46244	17289	3330	1401	8.32e-02	9.94e-02	9.442e-02

Table 2-3 : Gate Error = 1.00e-03; Critical Gate Threshold = 5 {Case2}

Benchmark	Gate Count		Critical Gates		Error Probability		
	F	P	F	P	out-F	out-P	CPE
C5315MCNC	936	2307	321	1845	3.83e-04	2.53e-03	5.875e-05
C6288MCNC	2290	2408	2162	2341	5.45e-03	4.76e-03	1.900e-03
desMCNC	2204	7292	675	3219	5.39e-03	3.04e-02	4.491e-06
pairMCNC	932	2617	247	1333	2.92e-03	1.81e-02	1.048e-05
Random1	4007	1578	530	225	6.87e-03	8.17e-03	1.230e-02
Random2	91905	33926	12398	5275	1.26e-01	1.40e-01	1.311e-01
Random3	46244	17289	5996	2557	6.56e-02	7.64e-02	8.494e-02

Table 2-4: Gate Error = 1.00e-04; Critical Gate Threshold = 10 {Case3}

Benchmark	Gate Count		Critical Gates		Error Probability		
	F	P	F	P	out-F	out-P	CPE
C5315MCNC	936	2307	176	1151	2.73e-04	2.26e-02	5.947e-03
C6288MCNC	2290	2408	1822	2326	2.27e-03	8.21e-04	1.012e-03
desMCNC	2204	7292	41	2877	8.27e-04	4.19e-03	3.265e-08
pairMCNC	932	2617	139	1198	3.62e-04	2.60e-03	1.628e-06
Random1	4007	1578	306	132	8.74e-04	1.04e-03	8.692e-05
Random2	91905	33926	6732	2839	1.85e-02	1.94e-02	1.729e-02
Random3	46244	17289	3330	1401	9.19e-03	1.09e-02	1.500e-03

Table 2-5: Gate Error = 1.00e-04; Critical Gate Threshold = 5{Case4}

Benchmark	Gate Count		Critical Gates		Error Probability		
	F	P	F	P	out-F	out-P	CPE
C5315MCNC	936	2307	321	1845			
C6288MCNC	2290	2408	2162	2341	5.70e-04	4.74e-04	1.272e-04
desMCNC	2204	7292	675	3219	----	----	0.00
pairMCNC	932	2617	247	1333	---	----	0.00
Random1	4007	1578	530	225	7.01e-04	8.24e-04	2.432e-06
Random2	91905	33926	12398	5275	1.45e-02	1.66e-02	6.786e-03
Random3	46244	17289	5996	2557	7.11e-03	8.30e-03	1.084e-04

Table 2-6: Gate Error = 1.00e-05; Critical Gate Threshold = 10 {Case5}

Benchmark	Gate Count		Critical Gates		Error Probability		
	F	P	F	P	out-F	out-P	CPE
C5315MCNC	936	2307	176	1151			
C6288MCNC	2290	2408	1822	2326	2.37e-04	9.52e-05	9.899e-05
desMCNC	2204	7292	41	2877	----	----	0.00
pairMCNC	932	2617	139	1198	---	----	0.00
Random1	4007	1578	306	132	8.74e-05	1.05e-04	1.058e-06
Random2	91905	33926	6732	2839	1.88e-03	2.23e-03	1.859e-06
Random3	46244	17289	3330	1401	9.20e-04	1.11e-03	1.538e-07

Table 2-7: Gate Error = 1.00e-05; Critical Gate Threshold = 5 {Case6}

Benchmark	Gate Count		Critical Gates		Error Probability		
	F	P	F	P	out-F	out-P	CPE
C5315MCNC	936	2307	321	1845			
C6288MCNC	2290	2408	2162	2341	5.46e-05	4.65e-05	1.397e-05
desMCNC	2204	7292	675	3219	---	----	0.00
pairMCNC	932	2617	247	1333	---	---	0.00
Random1	4007	1578	530	225	---	---	0.00
Random2	91905	33926	12398	5275	1.44e-03	1.68e-03	0.00
Random3	46244	17289	5996	2557	---	---	0.00

From Table 2-2 to Table 2-7, it is clear that the CPE approach provides complete error free decoding ability for any error less than 1e-05. Even at 1e-04, the performance is very good. Further analysis is going on to develop techniques which can improve the performance of the methodology even at higher gate error rates.

2.6. Conclusion

We have proposed a novel methodology to implement error correction codes driven graph augmentation techniques to improve the fault tolerance of the circuits that is generic and is applicable to any combinatorial logic. CPE methodology of implementation and performance evaluation has been completely automated. Initial simulation results shows that performance of CPE is much better as compared to that of the NMR approach. We need a replication of up to '11' times of the combinational circuit to achieve similar kind of performance as the CPE. Apart from this, some other important set of conclusions to make are as follow {for any LDPC code rate & for any critical gate threshold}:

- With lower code rate, we achieve significant amount of performance improvement where in 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.

3. Boole Shannon Limit of noisy combinational logic (Task 5.5)

The CPE approach has already been presented in Section 2.3 and evaluation/simulation of its performances when applied to real circuits has been presented in Section 2.5. In this section theoretical analysis of its goodness and its performance limits are presented. This is in line with task 5.5 which is aimed at computing the Boole Shannon limit of noisy combinational logic performance.

3.1. Problem setting

The CPE approach focuses on using ECC based methods/architectures to improve reliability of combinatorial circuits. A significant difference with standard EDA approach is the fact that there is no attempt to internally modify or alter the logic 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 presence of transmission error.

For the analysis presented the system where the CPE is applied can be partitioned based on two characteristics, first if the circuit to protect is linear or no non-linear, second if the system is symmetric (encoding[1] 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. The situation where non-linear circuits are presents several difficulties due to the lack of mathematical methods to work with such circuits. Analysis of the CPE limits in such scenario will require fundamental discoveries and will be focus of future long term work.

The symmetric-linear case is of great interest from analysis point of view because in the symmetric scenario the only usable coding scheme is LDPC codes, thanks to their ability of decoding the received message even in presence of faulty hardware, as reported in Deliverable D3.1 **[i-RISC/D3.1]**. 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 asymptotical 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.

3.2. Cost analysis

In VLSI circuit design, there are three parameters that are considered when comparing equivalent circuits: Area, Timing and Power. Whit the advent of unreliable gates a fourth parameter is introduced namely reliability. To evaluate the goodness of any new proposed scheme, it is necessary to consider all these four constraints and the inter-relation between them. In this line of thought, we introduce a general notation to evaluate proposed schemes. We will use the term cost C() to mean any of the four goals, or any linear combination of the four.

For the moment to simplify analysis, we consider the cost of a XOR gate and assume that the cost of any circuit can be represented as a multiple of this cost. This would suite the analysis for the case of linear circuits while an extension for non-linear cases would require consideration of the cost

[1] Here, "encoding" means the computation of the original circuit (F) and of the parity circuit (P). Encoding is always assumed to be error prone.

of universal gates and how these costs can be combined to estimate the cost of the complete circuit. These topics will form the core of future investigation on how to expand the proposed analysis for non-linear circuits.

3.3. CPE Cost Analysis

In this section, we analyze the cost incurred due to the CPE approach in the case of linear circuits. To do so specific corner case scenario are considered. These are scenarios where we are interested on 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 if I do not care throughput reduction?
- Is it possible to reduce power at the cost of area?

The analysis gives an idea of what are the limits in the best case scenario. These are equivalent of situation when one of the parameter is free or much less important. The first analysis focuses on the cost of the CPE in term of area or power.

3.3.1. Notation and Conventions:

We associate a cost with each probabilistic XOR-gate $\bigoplus_{\varepsilon}$, which is denoted by C_{ε}:

- C_{ε} is a positive value that denotes the cost function in order to implement a XOR-gate with error probability ε. Throughput this section, cost should be understood as either area of power.

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 $\varepsilon = 0$, C_0 represents the cost of the ideal (error-free) XOR-gate.

We will further denote the cost associated with the implementation of circuit F with probabilistic XOR-gate $\bigoplus_{\varepsilon}$ as $C(F_{\varepsilon})$. In particular, $C(F_0)$ denotes the cost of the ideal circuit. By a slight abuse of notation, we denote by F both the implementing circuit and the corresponding linear function. 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). We shall assume 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 $\bigoplus_{\varepsilon}$, then $C(F_{\varepsilon}) = \theta C_{\varepsilon}$.

For asymptotic consideration, increasing the size of F to infinity means that both l and k go to infinity, while keeping the ratio l/k constant (shape of F).

Finally, we need one more assumption concerning the LDPC decoder, denote by D. First, we note that D is not a linear function, hence it 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, we may write:

- $C(D_{\varepsilon}) = n\mu\delta C_{\varepsilon}$ ($\varepsilon \geq 0$), 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 follows 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 we are interested in the area (cost = area), μ is equal to number of decoding iterations instantiated in hardware.
  - Usually $\mu = 1$, as only one iteration is instantiated in hardware.
  - In order to increase throughput, several decoding iteration 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 we are interested in power consumption (cost = power), then $\mu \cong \log(n)$.

3.4. Cost analysis for Area/Power against error free circuit

The focus is now to find the conditions that guarantee the CPE approach to consume less area (or power). Let's start to define the notation used in case of CPE with LDPC codes. 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 $H = [H_i \mid H_p]$, with H_i of size $m \times k$ and H_p of size $m \times m$, G can be computed as $G = H_i^T \cdot \left(H_p^T\right)^{-1}$. Note that G is of size $k \times m$ and it is generally not a sparse matrix.
- Let $P = F \cdot G$ 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 $o_F = i \cdot F$ and $o_P = i \cdot P$ 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 $H \cdot [o_F \mid o_P]^T = 0$.
- 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:
  - $C(F_\varepsilon) = \lambda_F lkC_\varepsilon$
  - $C(P_\varepsilon) = \lambda_P lmC_\varepsilon$

Under these assumptions, we can state the conditions that ensure the CPE to be less costly then 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:

$$C(F_\varepsilon) + C(P_\varepsilon) + C(D_0) < C(F_0) \tag{3.1}$$

After some manipulations, and denoting $r = \frac{k}{n}$ the coding rate, one gets:

$$C_\varepsilon < \frac{\lambda_F r - (\mu/l)\delta}{\lambda_F r + \lambda_P(1-r)}\, C_0 \tag{3.2}$$

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

$$C(F_\varepsilon) + C(P_\varepsilon) + C(D_\varepsilon) < C(F_0) \tag{3.3}$$

After some manipulations, and denoting $r = \frac{k}{n}$ the coding rate, one gets:

$$C_\varepsilon < \frac{\lambda_F r}{\lambda_F r + \lambda_P(1-r) + (\mu/l)\delta}\, C_0 \tag{3.4}$$

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

$$C_\varepsilon < \frac{\lambda_F r}{\lambda_F r + \lambda_P(1-r)}\, C_0 \tag{3.5}$$

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 $\lambda_F = \lambda_P = \frac{1}{2}$, which gives:

$$C_\varepsilon < r \cdot C_0 \tag{3.6}$$

The main result of interest is the condition found for the asymptotical case. It had drawn a clear connection between the cost of a single gate and the rate of the code needed to achieve a desired performance. There is here a strong parallel with the Shannon limit for communication systems, in the sense that it set the limit of the achievable regions once the code rate is set. It also set a simple limit that help deciding if the reduction in cost of new technology is sufficient high to justify switching to such technology.

However, the above analysis doesn't consider whether or not there exists a code of rate r capable to correct (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 doesn't ensure that it is also reliable (i.e. allows recovering the correct output). This issue is addressed in the next section.

3.4.1. Error Correction Capacity

In this section we further elaborate on the above cost analysis, by taking into account the error correction capacity of the proposed CPE approach. We define $\alpha_\varepsilon = \frac{C_\varepsilon}{C_0} \in [0,1]$ to be 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, we will assume throughout this section that

- $\sigma_F = \sigma_P \stackrel{\text{def}}{=} \sigma_\varepsilon$

Hence, within the CPE approach, σ_ε represents the error probability at the decoder input. According to Equation (3.6), the CPE approach is cost-effective if and only if:

$$r > \alpha_\varepsilon \tag{3.7}$$

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

$$\alpha_\varepsilon = 1 - h(\bar{\sigma}_\varepsilon) \Leftrightarrow \bar{\sigma}_\varepsilon = h^{-1}(1 - \alpha_\varepsilon) \tag{3.8}$$

where $h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function.

Therefore, if $\sigma_\varepsilon < \bar{\sigma}_\varepsilon$ the CPE approach is not only cost effective but also *reliable*, in the sense that it is able to correct 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):

$$\sigma_\varepsilon < \sigma'_\varepsilon = \frac{1 - (1-2\varepsilon)^l}{2} \tag{3.9}$$

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

$$\sigma'_\varepsilon < \bar{\sigma}_\varepsilon \Leftrightarrow \alpha_\varepsilon < 1 - h\left(\frac{1 - (1-2\varepsilon)^l}{2}\right) \tag{3.10}$$

For large l values[2], 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. This upper-bound is plotted in Figure 3-1, for $l = 10^6$ and $l = 10^9$.

It can be seen that 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 obvious: for small ε, the σ_ε value – error probability at the output of F – is also small, and the CPE approach may use a code with rate close to 1 (note that coding rate = 1 corresponds to the original circuit only).

As the ε value increases, the cost-reduction factor α_ε is bounded below 1 and decreases until it gets close to 0 (*i.e.* the cost of the fault-prone technology must be negligible with respect to the cost of the fault-free technology). The "waterfall" region (where α_ε decreases from 1 to 0) corresponds to an increase in the value of ε by approximatively 3 to 4 orders of magnitude.

[2] Large enough such that the term $(\mu/l)\delta$ in Equations (3.2) and (3.4) can be considered negligible.

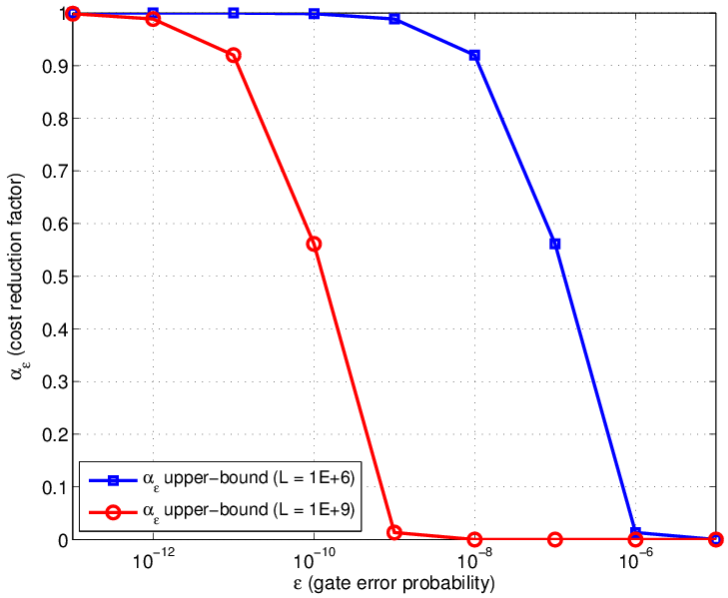


Figure 3-1: Cost-reduction upper-bound, ensuring both cost-effectiveness and reliability

3.5. CPE and Modular Redundancy comparison

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 extremely simple and well understood and it is hence interesting to compare the CPE approach against it. To compare the two architectures, it is important to consider that both schemes, CPE and MR, there are tradeoffs 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 for compute all the iterations. Let's now define some notation to consider this aspect:

- A_F, A_P, A_{DEC} the area of the F, P and D blocks
  - $A_{DEC} = \mu A_I$ 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
  - $A_{MR}^{\omega} = \omega A_F$
  - The corresponding latency is given by Ω/ω clock cycles
- A_{CPE}^{μ} the area of the CPE when LDPC decoder is unrolled
  - $A_{CPE}^{\mu} = A_F + A_P + \mu A_I$
  - The corresponding latency is given by I/μ clock cycles, where I is the total number of decoding iterations of the LDPC decoder.

Using the same notation and conventions as in Section 3.3.1, we can now state the conditions to ensure that the area of the CPE is smaller than the area of the Ω-MR.

3.5.1. Area/ throughput comparison

In this section, we first compare CPE and Ω-MR in term of area and throughput without consideration on the total performance. Then,

- $A_{CPE}^{\mu} \leq A_{MR}^{\omega}$ if:
  - If better throughput is not necessary :

$$\frac{A_I}{A_F} \leq \frac{r\omega - 1}{r\mu} \tag{3.11}$$

  - If better throughput is desired

$$\frac{A_I}{A_F} \leq \frac{\Omega}{\mathrm{I}} - \frac{1}{\mu r} \tag{3.12}$$

This can be simply proved by the following analysis:

$$A_{CPE}^{\mu} \leq A_{MR}^{\omega} \Leftrightarrow A_F + A_P + \mu A_I \leq \omega A_F \tag{3.13}$$

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

$$A_P = \left(\frac{1}{r} - 1\right) A_F \tag{3.14}$$

We can rewrite the condition (3.13) as:

$$\mu A_I \leq \left(\omega - \frac{1}{r}\right) A_F \Leftrightarrow \frac{A_I}{A_F} \leq \frac{r\omega - 1}{r\mu} \tag{3.15}$$

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

$$\frac{\Omega}{\omega} \geq \frac{\mathrm{I}}{\mu} \Rightarrow \omega \leq \Omega \frac{\mu}{\mathrm{I}} \tag{3.16}$$

Using the condition (3.14) for the previous case, we get:

$$\frac{A_I}{A_F} \leq \frac{\Omega}{\mathrm{I}} - \frac{1}{\mu r} \tag{3.17}$$

The two condition presented show 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 small circuit it may be better to repeat the circuit rather than using a LDPC decoder. Analysis of these zones and partitions need to be further investigated to understand their implications and conceptual value.

Case Study: We consider an Ω-MR schemes with $\Omega = 12$ and unrolling factor (*i.e.*, number of copies of F instantiated in hardware) $\omega \in \{1, 2, 3, 4, 6, 12\}$. The delay $\left(\frac{\Omega}{\omega}\right)$ vs. normalized area $\left(\frac{A_{MR}}{A_F} = \omega A_F\right)$ when varying the ω value is plotted in Figure 3-2 (solid black curve, no markers).

We further consider a CPE approach with $\mathrm{I} = 12$ decoding iterations and unrolling factor (*i.e.*, number of decoding iterations instantiated in hardware) $\mu \in \{1, 2, 3, 4, 6, 12\}$. We consider that 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:

(1) the use of different decoders, one more powerful but having an increased area, and a second one less powerful but having a smaller area,
(2) an increase on the size of the circuit F under consideration, since A_I becomes negligible with respect to A_F when the size of F goes to infinity.

The delay $\left(\frac{1}{\mu}\right)$ vs. normalized area $\left(\frac{A_{CPE}}{A_F} = \frac{1}{r} + \mu\frac{A_I}{A_F}\right)$ when varying the μ value for the CPE approach is also plotted in Figure 3-2 (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. Error correction performance of the two schemes is discussed in the next section.

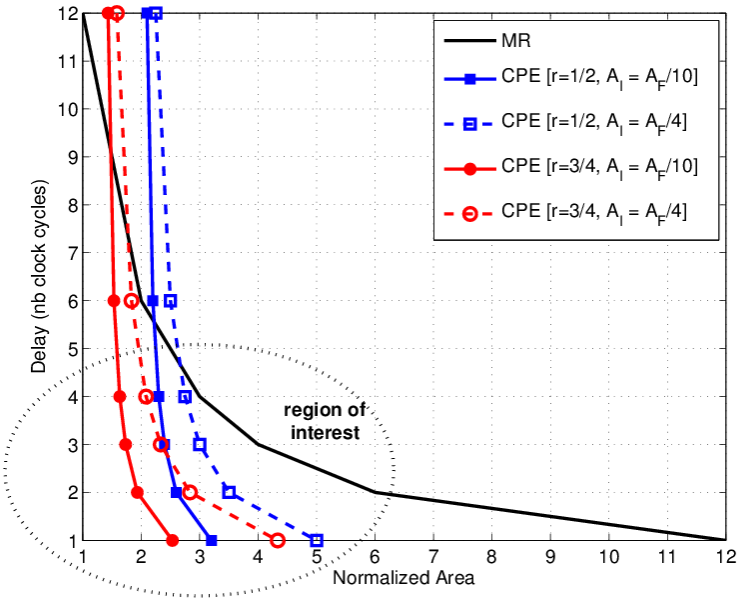


Figure 3-2: Delay vs. normalized area for MR and CPE schemes with various unrolling factors

3.5.2. Area/performance comparison

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

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

First let us find an asymptotical relationship between area and performances for the Ω-MR scheme. The error probability of the modular redundancy scheme, denoted by σ_{MR}, is the probability

of having the majority of the replica in error; this can be represented of binomial combination. When the limit for Ω that goes to infinity we have:

$$\sigma_{MR} \approx \sigma^{\Omega/2} \tag{3.18}$$

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

We now derive an asymptotical relationship between area and performances for CPE approach. 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:

$$r_\sigma = 1 - h(\sigma) \tag{3.19}$$

where $h(\sigma) = -\sigma\log(\sigma) - (1-\sigma)\log(1-\sigma)$ is the binary entropy function. Let us assume that we have a family of LDPC codes operating close to the channel capacity. This means 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 goes to infinity. 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:

$$A_{CPE} \cong \frac{1}{r_\sigma} A_F + \mathrm{I}A_I \tag{3.20}$$

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

$$A_I = \delta n = \delta\frac{1}{r_\sigma}k \tag{3.21}$$

We obtain:

$$A_{CPE} \cong \frac{1}{r_\sigma} A_F + \delta k\log(k) \tag{3.22}$$

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

$$\frac{A_{CPE}}{A_F} = \frac{1}{r_\sigma} + \frac{\delta}{\gamma}\frac{\log(K)}{K} \tag{3.23}$$

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

3.6. Conclusion

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 achievable trade off of the i-RISC approach to error prone circuitry. Two research directions are foreseen for continue this work. The first it is to understand the implication of the corner scenario cost analysis presented and expand it into a unify 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 to 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 limits in telecommunication are presented/valid only for restricted channels. It is foreseen that formulation of these limit for non-linear circuit may require years of fundamental research on many mathematical tools before being possible.

4. Multi Objective Optimization (Task 5.4)

Abstract: In this section, a novel multi-objective optimization tool is introduced to address and improve the reliability along with the traditional parameters like power, delay and area of combinational circuits. 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. The key idea is to employ local transformation rules that enhance the circuit reliability without altering the functionality of the circuit. This rewriting capability, along with the reliability evaluation method for assessing the SER of a circuit, enable the integration of the SER in a unified search algorithm that iteratively evolves the design in order to satisfy a given set of objectives. This technique was previously extensively used for optimizing the area, power and the delay and the current results show that it can reduce error probability as well. Experimental results based on simulations performed on MCNC benchmark circuits indicate that our reliability-driven synthesis methodology improves circuit reliability by up to 10%.

Publications: Two Journals under preparation; Three Conference papers accepted [P1-P3];

4.1. Introduction

The low reliability of advanced CMOS devices has become a critical issue that has to be considered in the digital IC design flow. To overcome the reliability related concerns, a variety of techniques have already been proposed. Negative Bias Temperature Instability (NBTI) and its mitigation techniques have received significant focus **[Vrudhula06]**. Dynamic Reliability Management (DRM) techniques, which try to hide the inherent pessimistic reliability while maintaining the system performance and lifetime expectation within the desired range were previously proposed **[Rivers04]**. Another novel idea is the concept of inexact computing **[Palem12]** where circuits are designed with erroneous gates. Reliability driven logic synthesis is one area that is gaining lot of importance in the last few years. In [**Smita07**], Soft Error Reliability (SER) is improved through localized circuit restructuring taking advantage of don't care based re-synthesis and local rewriting. In [**Diana13**], a technique to improve the circuit robustness to soft errors based on redundancy addition and removal (RAR) by eliminating gates with large contribution to the overall SER is proposed. Efficient algorithms for synthesizing approximate circuits for concurrent error masking of logical and timing errors were employed in [**Mohanram13**]. ATPG-based rewiring method to generate functionally-equivalent yet structurally-different implementations to reduce the SER was developed in [**Almukhaizim06**]. All these approaches employ redundant node addition techniques to improve circuit reliability. Our approach slightly differs as we use subset of NPN-equivalent (Negation-Permutation-Negation equivalent) logic configurations to improve circuit reliability. The biggest advantage is that we do not add any extra overhead by increasing node count. ABC **[Brayton10]**, a logic synthesis and verification tool which performs scalable logic optimisation based on AND-Inverter Graphs (AIGs) **[Mishchenko06]** has been used to accommodate all our algorithms. This section investigates gate level methodologies to improve circuit reliability by employing logic synthesis techniques. As part of our ongoing research, we are developing a reliability aware logic synthesis tool.

The **first step** in building a reliability aware synthesis tool is to develop an efficient algorithm that computes circuit reliability. Reliability analysis of logic circuits deals with computing the impact that the gate level have on the circuit Primary Outputs(PO). Traditional approach to reliability

analysis begins with elementary SPICE simulations to estimate the circuit error probability. Several analytical approaches for computing reliability have been previously reported [**Choudhury09, Han11**]. In this line of thought, we propose two different methodologies. One is based on probabilistic based methods and second is based on simulation based methods. As we represent the circuit in the AIG format, a novel algorithm based on probability principles is proposed, with the prime focus being AND & Inverter gates. The algorithm uses the dynamic weighted average algorithm (DWAA) [**Ercolani89**] approach to account for the impact of reconvergent fanout on the overall results. Besides, we also present a second methodology that is based on the well known simulation based techniques. Both these techniques have trade off's wrt each other and one supersedes the other in specific applications. Though the probabilistic based method is very fast, it can be slightly less accurate. This 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 in this stage which enables us do with slightly less accuracy. The simulation based methodology is very accurate but can be time consuming specially for large circuits. This is used in final steps which require accurate reliability numbers before signoff.

The **second phase** in the synthesis tool development is the optimization of combinational circuit for a given set of constraints. Logic optimization and synthesis is the process of taking in a higher level representation of a circuit and translating it into hardware. This generally involves three steps. The first is compiling the high level representation into an intermediate representation. The second is optimising the intermediate representation. The third is mapping the circuit onto the final output representations. The term "synthesis", besides describing the whole process, can refer to either the compilation or optimization steps individually. In this work the interest is in the optimisation step, and in particular optimising for the reliability constraint. Through optimising for reliability, it is hoped to indirectly allow power savings. Operating a circuit at reduced voltages, at near- or sub-threshold operation has a potential for great power savings. However this can cause the gates to be unreliable.

Logic optimization techniques are traditionally classified under two broad categories, local rule-based transformations (or rewriting) and technology independent/dependent algorithms **[Mishchenko 06-b]**. Rewriting is based on employing a set of local transformation rules on a small sub-section of the graph in order to improve area, power or timing. A rule transforms a pattern for a local sub-expression, or a sub-circuit, into another equivalent one. Since rules need to be described, and hence the type available of operations/gates must be known, the rule-based approach usually requires that the description of the logic is confined to a limited number of operation/gate types such as AND, OR, XOR, NOT etc. Algorithmic based approaches work on the observation that there exists certain set of operations, which are inherently good irrespective of technology. These methods use global transformations such as decomposition or factorization, and therefore they are much more powerful compared to the rule-based methods. However, general Boolean methods, including don't care optimization, do not scale well for large functions. Algebraic methods are fast and robust, but they are not complete and thus often give lower quality results. For this reasons, industrial logic synthesis systems normally use algebraic restructuring methods in a combination with rule-based methods.

The present work is based on the popular and successful concept of local transformations **[Darringer81] [Brayton87].** In this work, we explore the possibility of reducing output error probability by employing local transformation techniques a.k.a. rewriting. Reliability driven logic

optimization is in its infancy when compared to power and delay driven optimization. There are two methodologies that we propose in this work. The first is rule-based resynthesis, and the second is based on cut enumeration. Rule-based resynthesis approaches rely on searching the graph for specific substructures and replacing these with alternatives. The search and replacement substructures are hand derived and hard-coded. A slightly more general approach to this is using Boolean algebra transforms that can apply to various sub-circuits with a common feature. Hard-coded graph synthesis has been used for power in [**Mehrotra13**] and for reliability in [**Grandhi2014-b**]. The second approach to synthesis is cut rewriting [**Mishchenko06-b**].

4.2. Reliability Computation

One of the keys for developing an efficient optimization and synthesis tool is the availability of accurate reliability information as well as efficient/fast algorithms for computing the reliability of logic functions representing partial solutions during the optimization process. A pure reliability analysis based on HSPICE Monte Carlo simulations is not feasible due to a prohibitive computation time and excessive resource requirements. In this section, we address the problem of computing circuit reliability investigating two different approaches and presenting the tools that has been developed for each approach. The first method, simulation based approach, is based on simulating the combinatorial circuit under test and injecting error with specific characteristics to gather statistical information on how these error propagate through the circuit . The second method, probabilistic based approach, builds a model of the combinatorial circuit and calculates the reliability of the circuit based on mathematical analysis of the model. This has already been reported in the previous deliverable report [**iRISC-D5.1**].

4.2.1. Simulation Based Methodology

Simulation-based reliability estimation technique can be employed at various levels of abstraction, which include: switch-level, gate level, and block-level. In this paper, we only consider gate level simulation 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 not considered. 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, in order 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. Gate-level simulation is a well studied problem and much effort has been placed on improving its speed [**Burch93, Todorovich02, Betz99**]. It is highly accurate compared to probabilistic methodologies. But, despite many innovations, it comes with the overhead of extremely long run times for large system-level designs. Moreover, simulation requires input vectors, which are often not available when designing a new system. Keeping this criterion in mind, we use this methodology only for final estimation limiting ourselves to use the probabilistic methodology most of the time during the intermediate computations.

The tool generates random vectors and applies them on the primary inputs of the AIG based circuit representation. The tool can process multiple simulation runs simultaneously, analyzing how reliability varies with input vectors. We employ extra xor gates to model the error on each gate to

compute the numbers under ideal and faulty scenario. First of all, we define the following four important variables:

- pNtk : It is a pointer to the network (circuit) under consideration as shown in Figure 4-1
- pNode: It is a pointer to the node under consideration in the circuit as shown in Figure 4-1 (all the Primary inputs, Internal Nodes & 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.

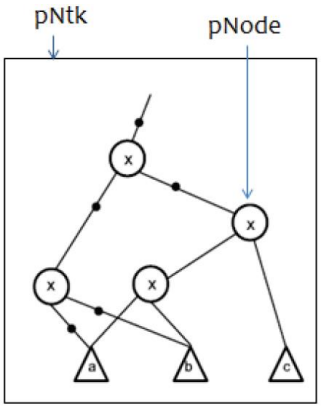


Figure 4-1 : AIG representation of a circuit

Next, count the number of inverters by traversing through every Internal node and check whether the left child or right child have inverters or not. If yes, then increase the inverter count. Repeat the process until all the nodes are traversed and correspondingly increase the AND gate counter every time a internal node is traversed. Follow the same procedure for counting the number of inverters for primary output and simultaneously create a dynamic array for storing the values of Internal nodes and Primary output using `Calloc' function. Once the counting and allocation has been done the next step is to generate a random pattern for primary inputs as well as for nodes and their respective left and right Child (these random pattern will be required when modelling the gates to be faulty).

4.2.2. Mersenne twister & Random Number Generation

For generating random pattern, we can either use rand() or random function() available in C library but the disadvantage with them is that they have a period of length 2^{32} - 1 and since we will be running quite millions of simulations so they are high chances of patterns to get repeated and moreover the patterns generated are correlated to each other. To overcome the problem of the period length, we used a pseudo random number generator called Mersenne twister. The name has derived from the concept of Mersenne prime. In mathematics, a Mersenne prime is a prime number of the form $M_n = 2^{n-1}$. This is to say that it is a prime number which is one less than a power of two. The four Mersenne primes are 3, 7, 31 and 127. It has a very long period of 2^{19937} - 1. It is k-distributed to 32-bit accuracy for every 1 < k < 623. It passes numerous tests for statistical randomness, including the diehard tests. We have used two commands for manipulating the default generator.

- mt_seed(void): Choose a seed from random input
- mt_lrand(void): Generates 32-bit random value.

Once the random patterns are generated the next task is to apply them to the primary inputs of the circuits and calculate the output value at each and every internal node by considering the circuit to be ideal. The computation is carried out as follows: traverse through the node in a sequential order and for every node calculate the right and left child using the built in function:

- pLeft_Child = Abc_ObjFanin0(pNode_UC);
- pRight_Child = Abc_ObjFanin1(pNode_UC);

After the calculation of the left and right child ,next check whether there is a inverter at the childs or not by using the function : pNode_UC -> fCompl0 ; 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 indicates the presence of a inverter at the left child. pNode_UC -> fCompl1: if this value is equal to zero then it indicates there is no inverter at the right child and if the value is 1 then it indicates the presence of a inverter at the right child.

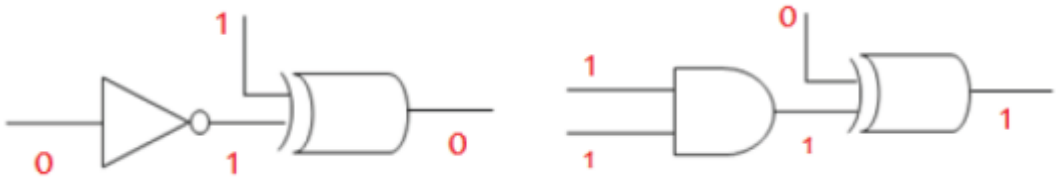


Figure 4-2: Error Injection on AND & Inverter gates.

Till now, all the AND gates as well as inverters has been considered to be fault free but to calculate the reliability we need to model the circuit as faulty .this can be achieved by the use of XOR gates . A XOR gate is applied just after each gate with one input from the gate and other from a randomly generated vector as shown in Figure 4-2. 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 we have both Ideal and faulty values at all the nodes, we can compute the error probability and in turn the reliability. 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. Repeat the procedure till all the node have been traversed and keep updating their respective error probability count after each iteration. After the completion of all the iteration, for each node divide its error probability count by the no. of iteration to calculate the error probability and repeat the same for the remaining nodes. Now, the reliability for a particular node can be computed using the formula:

$$R(t) = 1 - P(r), \tag{4.1}$$

where P(r) is error probability.

4.3. Reliability Driven Synthesis

We propose two different methodologies both based on the rewriting technique. The first is Rule-based resynthesis and the second one being cut rewriting. Rule based resynthesis approaches rely on searching the graph for specific substructures and replacing these with alternatives. The search and replacement substructures are hand derived and hard-coded. We introduce set of local transformation rules for logic optimization from a reliability perspective. These set of rules along 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 upto 7.5%. The second approach to synthesis is 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 as a certain node are identified node, 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 alternatives are selected. This methodology is much more superior in terms of its performance compared to the initial methodology proposed.

4.3.1. Rule Based Methodology

The basic idea behind our proposal is to implement reliability aware transformations. We introduce 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. We then study the impact of the gate error probability on equivalent logic configurations 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. We have evaluated our logic transformation rule set on a test circuit and results show a reliability improvement in the order of 8%.

4.3.1.1. Local Transformation Rules

In this section, we present the set of transformation rules utilized in our quest for the reliability optimized implementation of Boolean functions. Most of the details behind the local transformation rules, its application and others have been reported in previous years report [**iRISC-D5.1**]. Only the top level representation of each of the rules is presented in Figure 4-3 for quick reference.

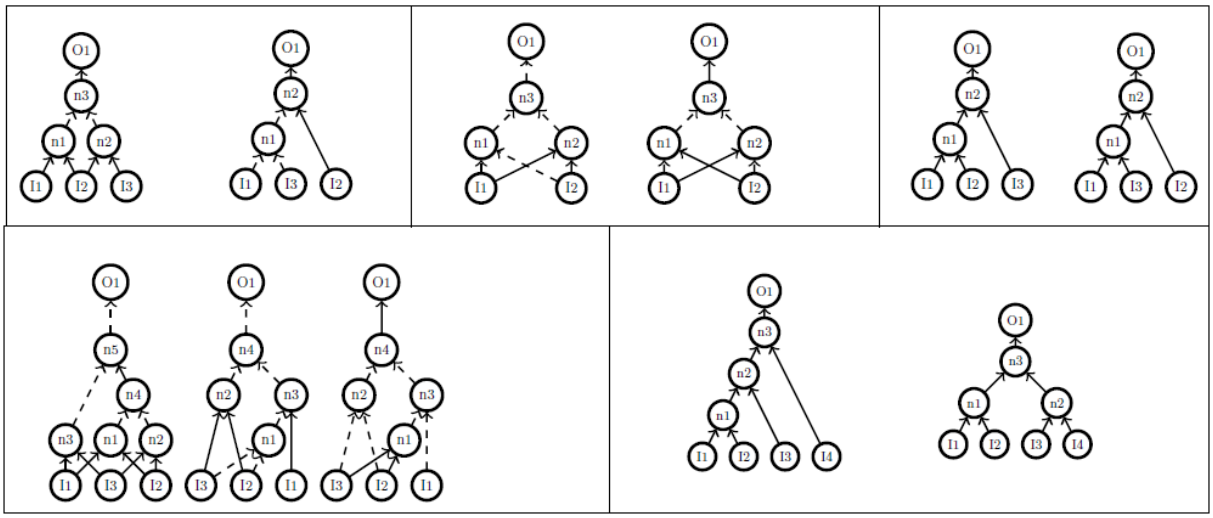


Figure 4-3: Ad-Hoc Rules used in Logic Optimization.

4.3.1.2. Exhaustive Analysis of Rules

Direct mathematical analysis is not feasible to compute the improvement achieved as the number of variables in the equations used to compute the output error probability are too high. Instead, we set the input node static probability to 0.5 and the input error probability to 0.01 and perform simulation using the tool previously described. All the rules have also been investigated for other generic patterns such as Gaussian. Simulation results for different input patterns confirm that rules [1-4] improve the reliability of the circuit and hence suggest that they are applicable in any general scenario. In contrast, rule5 is applicable only under certain circumstances as detailed out later in the section.

Table 4-1: Different Scenarios for Rule Analysis.

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

Each of the logical transformation presented in the main paper can be seen a transformation between two logical equivalent circuit. Mathematical equations that describe the primary output reliability/error probability in function of the input error probabilities input static probabilities and the gate error probability of the primary output can be associated with each of this logical equivalent circuit.

$$\begin{aligned} Ro_{Org} &= f1\{PE_I , SP_I , G_E\} \\ Ro_{Mod} &= f2\{PE_I , SP_I , G_E\} \end{aligned} \tag{4.2}$$

Unfortunately, even for small circuit, it is not possible to give a complete mathematical characterization to determine the reliability performance of the circuit configurations from these equations. Consider a three input graph. Eq. (4.2) has seven variables: three input error probabilities, three input static probabilities and the gate error probability. It is hence impossible to find a symbolical solution to determine if and when one function is lower than the other. As an example of the complexity, apply Eq. (4.2) to Rule4 a closed form equation of the output error probability can be obtained, however, the resulting polynomial has degree 20. Hence, it is clearly understood that there exists no simple method to completely define the performance of the two configurations. As an approximate solution we investigate the behaviour of the equivalent circuits in different scenarios. We have identified four different patterns that are commonly observed in digital circuits, namely; Gaussian, reversed Gaussian, monotonically increasing and monotonically decreasing.

We apply these four patterns on the input pins error probabilities as well as static probability independently assuming the other parameter to be constant. Table 4-1 lists the eight different scenarios considered for exhaustive testing of the local transformation rules. The set of simulation

results covering all the patterns listed in Table 4-1 are plotted in Figure 4-4 to Figure 4-8. From the Rule1 plots, we see 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. In our research, we have observed that intelligent insertion of extra nodes can result in reliability improvement. The question of how to insert such extra nodes is still an open problem and subject of future investigation. Rule3 is pre-dominantly applicable on most of the basic 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. The simulation results presented show how for Rule1 to Rule4 the proposed transformation is beneficial for all input scenario.

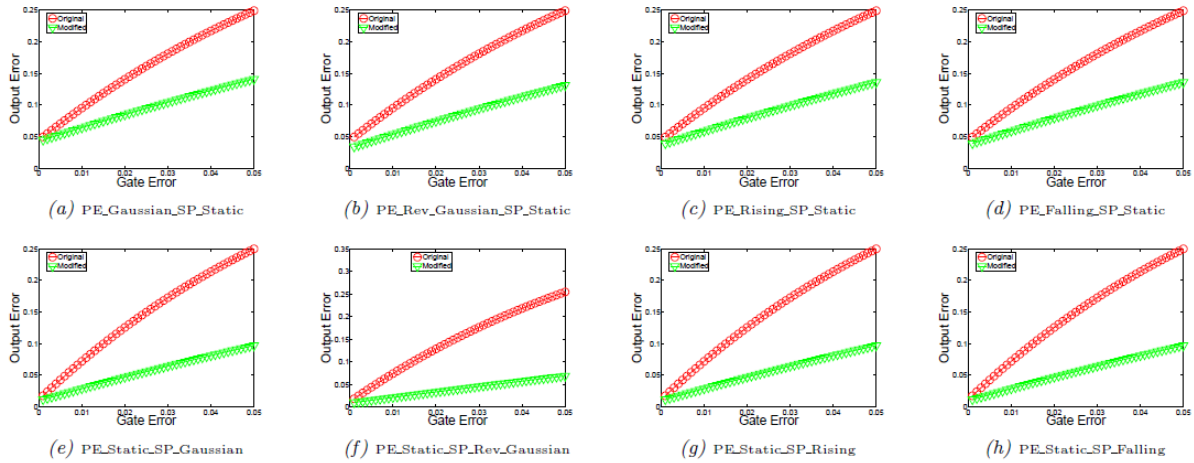


Figure 4-4 : Simulation results for rule1

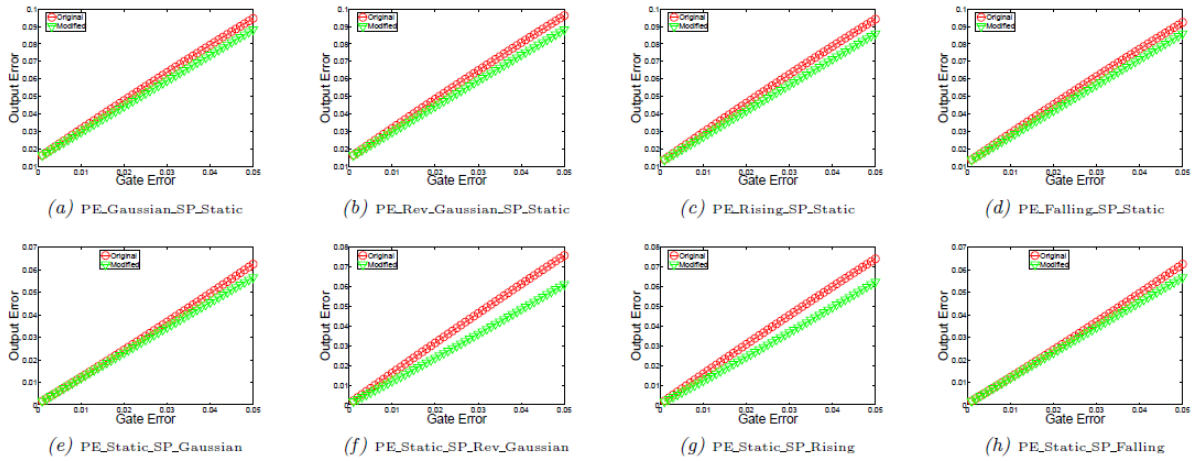
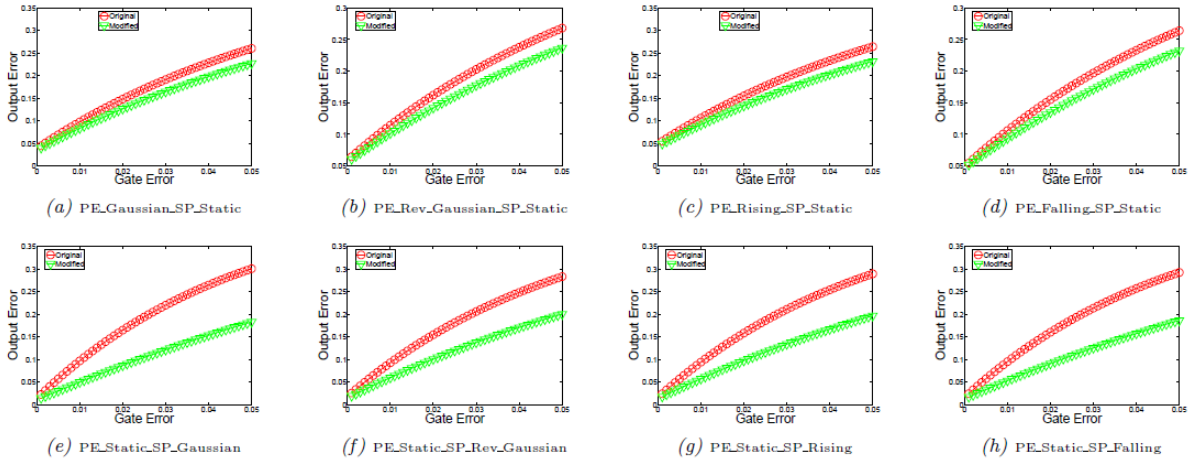


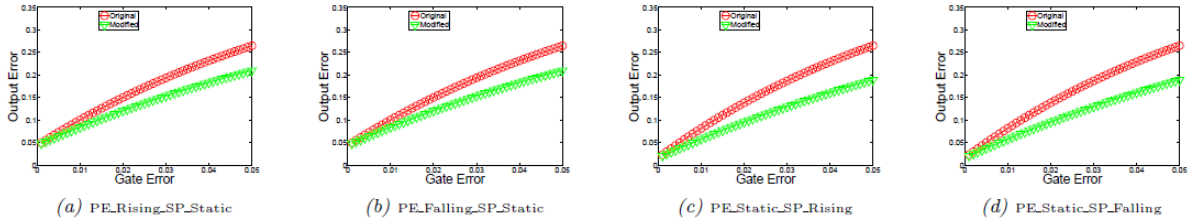
Figure 4-5 : Simulation results for rule2



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

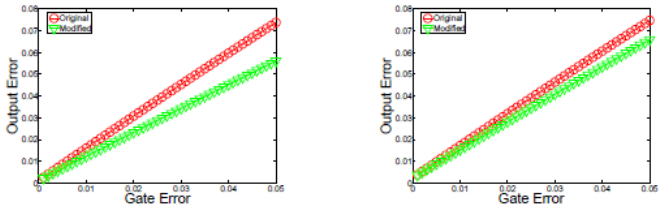
(e) PE_Static_SP_Gaussian (f) PE_Static_SP_Rev_Gaussian (g) PE_Static_SP_Rising (h) PE_Static_SP_Falling

Figure 4-6: Simulation results for rule4



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

Figure 4-7 : Simulation results for rule3



(a) Pe_{IX}=0.01,SP_{I1}=0.2, SP_{I2}=0.1,SP_{I3}=0.5 (b) Pe_{IX}=0.01,SP_{I1}=0.15, SP_{I2}=0.3,SP_{I3}=0.5

Figure 4-8: Simulation results for rule5

4.3.1.3. Simulation Results

The rules presented have been tested in various conditions and the simulation results show reliability improvement in every scenario. However, due to the high variable count, it is impossible to derive a rigorous proof and/or 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. 1 details the process adopted for performing local transformations. Starting from the initial circuit configuration, we traverse through the graph 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 we reach the primary outputs where no more transformations are applicable.

To prove that the whole methodology is scalable, we have run our tool on various MCNC benchmark circuits. In Table 4-2, Columns 1 and 2 give the name and number of gates in the benchmark circuit. Column 3 captures the error probability of the original and the optimized circuit while Column 4 highlights the improvement achieved. The reliability improvement is computed using Eq. (4.3).

$$R_{Metric} = \frac{\left(\sum_{i=1}^{n} R_{org}\right) - \left(\sum_{i=1}^{n} R_{new}\right)}{\sum_{i=1}^{n} R_{org}} \tag{4.3}$$

where 'n' is the number of nodes. Columns 5 list the total number of output nodes and column 6 lists the count of nodes whose reliability improvement is greater than 0.5%. While for smaller circuits, there is not many paths which go unoptimized, for larger circuits, this can be quite high. We have performed a thorough analysis and figured out that the main reason is most of these paths have gate count less than '10'.

Table 4-2: MCNC benchmark circuits performance evaluation employing local transformation rules

Benchmark	GateCount	Output Error Probability		Reliability Improvement R_{Metric}%	Output Node Details	
		Original	Optimized		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

4.3.2. Cut Enumeration & Boolean matching approach

Rewriting is a common approach to logic optimization based on local transformations. Most commercially available logic synthesis tools include a rewriting engine that may be used multiple times on the same netlist during optimization. This section presents an And-Inverter graph based rewriting algorithm using 4-input cuts. The best circuits are pre-computed for a subset of NPN classes of 4-variable functions. Cut enumeration and Boolean matching are used to identify replacement candidates.

4.3.2.1. Primitive Terminology

4.3.2.1.1. Cuts

A *cut* of a node *n* is a set *C* of nodes such that any path from a PI to *n* must pass through at least one node in *C*. Node *n* itself forms a *trivial cut*. The nodes in *C* are called the *leaves* of cut *C*. A cut *C* is *K-feasible* if $|C| \leq K$; additionally, *C* is called a *K-input cut* if $|C| = K$. An example is shown in Figure 4-9. The node set {*d*, *f* ,*g*, *k*} defines a cut *C* of node *l*. *C* is a 4-input cut since it has four leaves, *d*, *f* , *g*, and *k*, and every path from a PI to *l*, if exists, must pass through at least one of them. Nodes *i*, *j*, and *h* are the internal nodes of the cut. However, set {*g*, *h*, *i*} is not a cut of *l*, because path *a*– *f* –*k*–*l* does not pass through any of the nodes in the set.

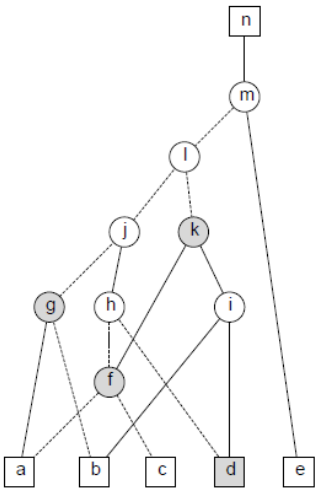


Figure 4-9: Simple CUT analogy

4.3.2.1.2. NPN equivalence

Two Boolean functions, *F* and *G*, are *NPN-equivalent* and belong to the same *NPN equivalence class*, if *F* can be transformed into *G* through negation of inputs (N), permutation of inputs (P), and negation of the output (N) [**Hurst1985**]. For example, Boolean functions $F = ab + c$ and $G = \bar{a}\bar{b} + \bar{a}c$ are NPN-equivalent, since $G = \bar{a}\left(\bar{b} + c\right) = \overline{a + (\bar{\bar{b}} + c)} = \overline{a + b\bar{c}}$ and *G* can be transformed from *F* through

- Negating *c*,
- Swapping the position of *a* and *c*, and
- Negating the output

Table 4-3: Maximum number of NPN-equivalent functions.

N	2	3	4	5	6	7
$\mathbf{2^{N+1}N!}$	16	96	768	7680	92160	1290240

4.3.2.1.3. Boolean matching

Boolean matching is a technique widely used in technology mapping [**DeMicheli1990**]. It compares a sub graph with library cells by functionality, with the consideration of functional equivalence. Unlike structural pattern matching, Boolean matching is normally done by calculating the canonical form representation of functions under permutation and negation of inputs/output. Thus, a Boolean matcher can also be used as a canonicalizer. The most obvious and the easiest solution is exhaustive search. For a function with *N* inputs, the maximum number of its NPN-equivalent functions is $2^{N+1}N!$, which is huge even for a small *N*, as shown in Table 4-3.

4.3.2.2. AIG rewriting

AIG rewriting technique presented in [**Bjesse2004**] is used as a way to compress circuits before formal verification. Rewriting is performed in two steps. In the first step, which happens only once when the program starts, all two-level AIG sub graphs are pre-computed and stored in a table by their Boolean functions. In the second step, the AIG is traversed in topological order. The two-level

AIG sub graphs of each node are found and the functionally equivalent pre-computed sub graphs are tried as the implementation of the node, while logic sharing with existing nodes is considered. The sub graph leading to least error on the output node is used as the replacement of the original sub graph.

An improved AIG rewriting technique for pre-mapping optimization is presented in [**Mishchenko06-a**]. It uses 4-input cuts instead of two-level sub graphs in rewriting, and preserves the number of logic levels so the area is reduced without increasing delay. Additionally, AIG balancing, which minimizes delay without increasing area, is used together with rewriting, to achieve better results. Iterating these two processes forms a new technology-independent optimization flow, which is implemented in the sequential logic synthesis and verification system, ABC [**ABC12**]. We use this technique but modify the parameter of optimization from area to reliability.

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 as a certain node are identified node, and alternative cuts that compute the same function are evaluated as replacements for that node. For the traditional goals of area and timing, the goals are to reduce the number of nodes and the depth of nodes, respectively. The rewriting implementation used in ABC attempts to reduce area in particular, and also not to worsen timing (leaving actual timing improvements to rule-based synthesis). A precomputed forest of possible alternatives is provided from which possible alternatives are selected.

To improve the area, ABC's rewriting algorithm takes a node, and gets a set of cuts of the node. It then tries alternative implementations for each of these 4- input cuts. For each cut, the nodes that are computing this function and no other can be removed, and new nodes that aren't duplicates of existing nodes must be added. The cut implementation is then scored by the number of nodes it removes, less the number it must add. For each cut, several function implementations are tried and the best implementation of any cut is selected and applied to the network. An option to the algorithm optionally permits zero-improvement cuts, to seek permutations of the network that might be more amenable to other improvement methods.

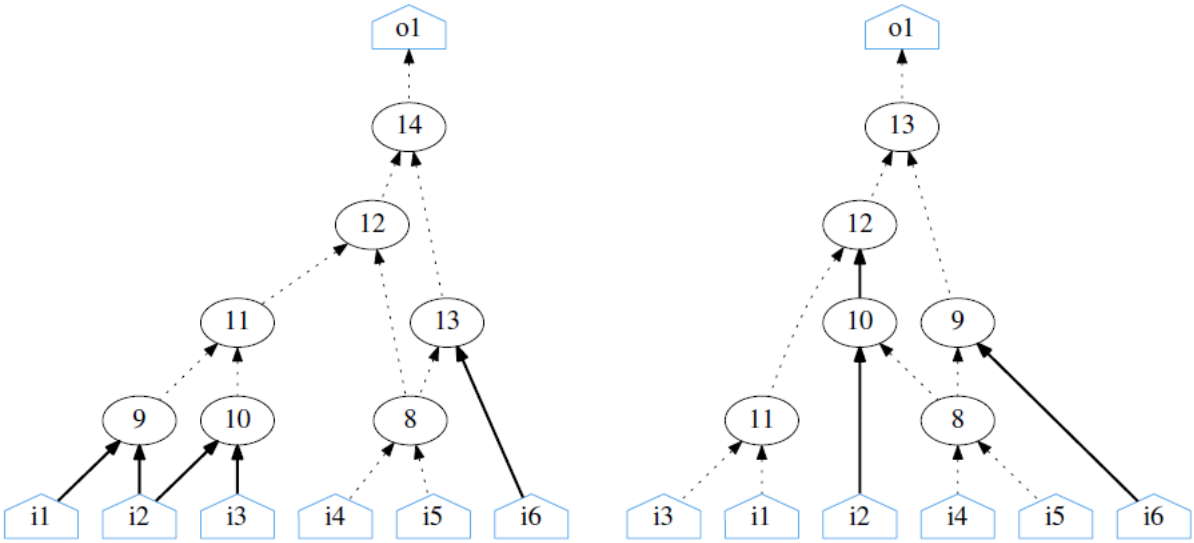


Figure 4-10 : Simple logic circuit before (a) and after (b) rewriting.

Figure 4-10 shows the effect of cut-based rewriting on the simple logic circuit. Though this was obtained with ABC's standard rewriting algorithm, the same result is produced by most of the algorithms for reliability in this work, due to the simplicity of the circuit.

4.3.2.3. Experimental Results

The main algorithm implemented in this work is called "Rewrite for Reliability" or rwrel for its command in ABC. A pseudo-code of this algorithm is shown in Alg. 1. This algorithm is based on the standard cut-rewriting algorithm from ABC, adapted so that the goal is reliability instead of area. Two versions of the algorithm were developed in this work. The first version selects the least error for each cut, and then selects the least error cut for the node. The second version instead selects the most improved cut for each node. A 'percent decrease in error' goal function is used, written as:

$$R_{Improvement} = \frac{P_{\varepsilon}(oldcut) - P_{\varepsilon}(newcut)}{P_{\varepsilon}(oldcut)} \tag{4.4}$$

The rewriting algorithm is applied to each AND-node in the AIG network in topological order. That is, the nodes connected directly to the inputs are rewritten first and the nodes nearest the outputs are rewritten last. This search order is easily enforced in ABC since it ensures that the nodes are numbered in topological order. It is ensured that only nodes that existed in the initial network are rewritten, since nodes created by the rewriting process will topologically be earlier in the network but their index numbers will be higher.

For each node, the cut enumerator is invoked to find a set of cuts fanning in to that node. These cuts are iterated over; taking only those with exactly 4 inputs (the cut enumerator will produce cuts with up to 4 inputs). For each cut, the truth table of that cut is evaluated, that is the truth table of the cut output in terms of the cut inputs. The function is converted into the canonical form of the NPN class, and possible implementations of this are looked up in a precomputed substitution node forest.

For each alternative implementation, a decomposition (possible alteration) of the network based on substituting that implementation is formed. The probability of error for the cut is read directly from the forest. For each cut, the rewrite that gives the least error for that cut is chosen. Selecting which cut to use to rewrite the node is the difference between the two version of the algorithm. In the first version, the cut giving least error is chosen. In the second version, the most improved cut is chosen.

Algorithm 1: Reliability Aware Optimization Employing Cut Enumeration

```
Require: N, total number of nodes in the AIG network
For node N in AIG do
Get cuts based at N
  For 4-input cut C based at N do
   Get truth-table F of N in terms of C
    For Possible graph S of function F do
      Make decomposition D corresponding to cut C
      Remove original nodes from D
      Add nodes of S to D
      IF Level > MaxLevel then
        Go to Next S
      end if
      Compute savings as nodes that can be dropped from network with D
      Compute cost of adding new nodes.
```

```
14.          Error = Error(S)
15.          IF Error < BestError then
16.              BestS = S
17.              BestD = D
18.          end if
19.       end for
20.    end for
21.    Apply decomposition BestD to the AIG
22. end for
```

This algorithm was implemented as an adaptation of the existing rewriting code in ABC. Modified versions of the data structures were created to include probability information. The key functions of the rewriting process were then rewritten to use reliability as a goal. Cut enumeration is carried out by a dedicated existing ABC module, and is described in more detail below.

The decomposition module of ABC is a module for representing possible modifications of the network. 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, decomposition must be formed for every implementation of every cut as building the decomposition is how area change is calculated. All these decompositions are still built when rewriting for reliability so as 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.

In both versions of the algorithm, the rewrite for a given cut is selected for least error, based on the reliability information in the forest. For the first version of the algorithm the cut that gives least error is chosen, based on the forest information. For the second version of the algorithm, the reliability of the initial form of the cut is evaluated as described above, and the improvement calculated using this. 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.

The presented algorithm is implemented using structurally hashed AIG as an internal circuit representation and integrated in ABC synthesis tool as a command '*rwrel*'. 'print_rel' command is developed to display the network reliability statistics.To evaluate its effectiveness, we performed a set of experiments using IWLS 2005 benchmarks with more than 5000 AIG nodes after structural hashing. A unit gate reliability of 10−4 is used throughout. The networks were loaded in from Berkeley Logic Interchange Format (BLIF) files and strashed to AIGs. Correctness was checked using the ABC *cec* equivalence check command, extensively during development as well as spot checks during evaluation. Having established the correctness and performance of the algorithms, its effect on the 8051 microcontroller circuits implemented using ACSL were tested.

The results for each test are shown in Table 4-4. In the table, column 1 lists the name of the benchmark circuit. Column 2&3 provides the number of nodes and output error probability of the default circuit. The gate count, reliability and the percentage improvement in output error probability of both the original configuration and the optimized one are computed and tabulated in columns 4-9. From the table, it is clear that improvement in output error probability employing the UCC synthesis algorithms provides much better improvement compared to the default algorithms provided by the standard tool.

Table 4-4: Logic Optimization results using cut enumeration technique.

Circuit	Default config		UCC_{REL}			$Resyn_{abc}$		
	Gates	Error	Gates	Error	Improvement (%)	Gates	Error	Improvement (%)
Adder	**175**	**0.01879**	**224**	**0.01377**	**26.68**	**146**	**0.01858**	**1.11**
multiplier	**631**	**0.06016**	**769**	**0.04751**	**21.02**	**487**	**0.05890**	**2.09**
Divider	**1011**	**0.40089**	**1183**	**0.24584**	**38.68**	**837**	**0.32419**	**19.13**
b9	**105**	**0.00655**	**97**	**0.00630**	**3.89**	**87**	**0.00624**	**4.79**
cm162a	**52**	**0.00210**	**53**	**0.00173**	**17.73**	**35**	**0.00240**	**-14.08**
cm85a	**40**	**0.00220**	**48**	**0.00180**	**18.06**	**36**	**0.00159**	**27.57**
Cu	**67**	**0.00289**	**59**	**0.00223**	**22.66**	**50**	**0.00254**	**11.86**
Dalu	**1735**	**0.01868**	**1637**	**0.01699**	**9.04**	**1214**	**0.02031**	**-8.73**
frg1	**659**	**0.00569**	**533**	**0.00471**	**17.34**	**491**	**0.00515**	**9.55**
Pair	**1736**	**0.06542**	**1854**	**0.05895**	**9.90**	**1381**	**0.06346**	**3.00**
Unreg	**112**	**0.00551**	**112**	**0.00532**	**3.39**	**112**	**0.00551**	**0.00**
Vda	**1020**	**0.15659**	**1125**	**0.09768**	**37.62**	**875**	**0.14759**	**5.74**
x2	**54**	**0.00259**	**57**	**0.00223**	**13.70**	**48**	**0.00245**	**5.29**

4.4. Power and Delay Driven Synthesis

The circuit is structurally represented as an AIG graph and the longest path delay from one of the inputs to one of the outputs of the network is calculated under a given delay model. A non-zero-delay model (corresponding to the AIGs) is assumed which takes into account the delay due to AND gates, inverters and the fanout. Average power dissipation in digital CMOS circuits can be expressed as the sum of three main components $P_{short-circuit}$, $P_{leakage}$ and $P_{switching}$. Pswitching is the switching power dissipation, also called the dynamic power is proportional to α the switching activity factor (also called transition probability), C_l the overall capacitance to be charged and discharged in a reference clock cycle, V_{dd} the supply voltage and f_{clk} the clock frequency. The difference in arrival times of signals at a gate input (the difference in the arrival times of the fanins at the AND node), leads to spurious or unwanted transitions, also called glitches. These spurious transitions play a major role in dynamic power dissipation. Based on the fanins' arrival times and the delay of the node, time instants at which a possible signal transition can occur have been calculated. Figure 4-11 depicts the complete flow for power & timing optimization of combinational circuits.

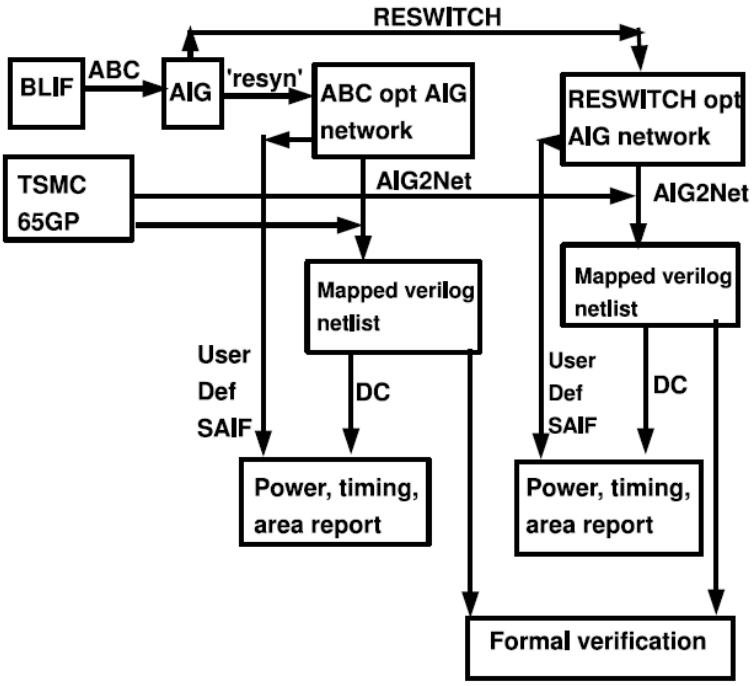


Figure 4-11 : The design flow for power, delay and area estimation

4.4.1. Implementation

Initially our method only targeted low power optimisation schemes. This implies decreasing the switching power (*SP*(*f*)) with no constraints on the longest path delay (*MAT*(*f*)) of the network. According to the conditions for the applicability of any rule, if all the four rules of power optimisation namely *Rp*1, *Rp*2, *Rp*3 and *Rp*4 are possible on a given node, then the rule which gives maximum reduction of switching power is chosen. Same applies when any three rules are possible on a given node. The rules applied in this method may decrease the switching power greedily but may increase the *MAT*(*f*) of the network largely. The algorithm applied in this implementation is a Greedy Algorithm [31] i.e. the rules are applied recursively at locally optimal AIG nodes until the minimal sum of switching power is achieved. The optimisation scheme proposed in this method is local and more emphasis will be on global and multi-objective optimisation in the further optimisation schemes. A tool called RESWITCH implemented in C as a sub-package in ABC which performs power optimisation. The tool OPT-PT is implemented on a set of MCNC Benchmark circuits. To complement the MCNC benchmark circuits, a set of large-scale random combinational logic circuits were prepared.

Table 4-5: Comparison of OPT-P with the best algorithm in ABC.

Circuit	Gates	P_{ABC}	T_{ABC}	A_{ABC}	RT_{ABC}	P_{PO}	T_{PO}	A_{PO}	RT_{PO}	△P	△T	△A
dalu	2000	281.69	25.48	2740572	0.50	235.42	24.59	2776628	0.69	-19.65%	-3.61%	1.29%
frg2	1355	365.24	11.42	1735806	0.45	273.79	11.75	1748241	0.56	-25.03%	2.80%	0.72%
vda	850	173.49	8.38	1360040	0.32	145.56	9.01	1427556	0.46	-19.18%	6.88%	4.72%
x3	1200	341.57	12.40	1509817	0.33	278.76	12.46	1551429	0.47	-18.38%	0.48%	2.68%
des	6250	1746.4	14.16	8916154	1.68	1422.2	14.59	9114893	2.85	- 22.79%	2.84%	2.18%
i8	1900	392.61	10.60	2632622	0.76	287.74	11.48	2644593	1.05	-26.71%	6.66%	0.45%
i9	1025	145.29	11.49	1368924	0.31	92.66	11.78	1387254	0.45	-36.22%	2.46%	1.32%
$\mathbf{R}_{o10,i8}$	1910	679.85	11.15	2631797	0.51	590.25	11.45	2728439	0.66	-15.18%	2.62%	3.54%
$\mathbf{R}_{o9,i7}$	920	378.42	9.47	1249644	0.21	322.18	9.84	1298947	0.31	-17.45%	3.7%	3.7%
$\mathbf{R}_{o12,i10}$	8530	2644.4	16.72	11653079	2.19	2215.2	16.86	12079592	3.02	-19.37%	0.8%	3.5%
$\mathbf{R}_{o13,i11}$	17860	5380.7	20.02	26461521	4.38	4639.8	20.36	26786046	8.38	-16%	1.66%	1.21%
$\mathbf{R}_{o10,i10}$	9380	2870.5	16.74	13169811	2.94	2497.9	16.96	13586011	4.47	-15%	1.29%	3%
$\mathbf{R}_{o14,i12}$	36270	10373	21.61	53068261	8.61	9040.9	21.58	53674448	15.95	-14.74%	0%	1.1%
$\mathbf{R}_{o11,i11}$	18850	5338.6	19.61	27286107	4.39	4169.9	19.19	27926652	8.19	-21.89%	-2.18%	2.29%
Average	10435	-	-	-	-	-	-	-	-	-20.24%	1.81%	2.38%

Table 4-6: comparison of OPT-T with the best algorithm in ABC.

Circuit	Gates	P_{ABC}	T_{ABC}	A_{ABC}	RT_{ABC}	P_{TO}	T_{TO}	A_{TO}	RT_{TO}	△P	△T	△A
count	300	101.78	14.79	324673	0.05	98.01	9.77	344453	0.09	-3.7%	-33.94 %	5.7%
frg2	1400	396.73	11.42	1735806	0.49	382.89	9.94	1751114	0.88	-4.5%	-14.88%	0%
x3	1200	330.9	12.40	1509817	0.33	320.254	11.44	1515892	0.52	-3.2%	-8.39%	0%
frg1	250	74.42	9.08	296687	0.07	74.77	8.10	300643	0.11	0%	-12.09%	1.3%
dalu	2100	268.40	25.48	2740572	0.46	288.50	23.22	2744669	0.82	6.9%	-10.73%	0%
toolarge	800	245.170	16.21	1104815	0.21	244.19	14.19	1110749	0.39	0%	-14.23%	0%
sct	100	29.91	4.89	138185	0.02	29.76	4.16	138185	0.025	0%	-17.54%	0%
term1	300	76.29	7.45	421872	0.14	80.09	6.82	423850	0.19	4.98%	-9.23%	0%
i2	500	175.12	5.99	564831	0.12	174.81	5.12	564831	0.19	0%	-16.99%	0%
k2	1600	224.879	10.74	2723015	0.61	251.49	9.17	2727112	1.1	9.5%	-17.12%	0%
DES	6000	1468.1	14.16	8916154	1.68	1425	12.96	8922090	3.12	-2.9%	-10.25%	0%
$\mathbf{R}_{o10,i10}$	10000	2553	16.72	13167870	2.29	2422.3	14.87	13174370	4.15	-5.3%	-12.44%	0%
$\mathbf{R}_{o11,i9}$	5000	1213.9	13.31	5579176	1.05	1200.6	11.37	5580589	1.95	0%	-17.06%	0%
Average	2273	-	-	-	-	-	-	-	-	3%	-15%	1%

4.5. Conclusion

In this section, we summarized the activities within the development of the multi-objective optimization & synthesis tool. On top of the probabilistic methodology described in the first year report, we have also developed a simulation based methodology which is more accurate. An initial 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%. We have also proposed another logic optimization technique based on cut enumeration and Boolean matching. Version 1 gave an average improvement 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 'less 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 in Task 5.4 during the 3rd year of the project. We also plan to complete the framework of a single synthesis tool which would optimize circuits based on all the input constraints; power, delay and reliability, in a single package.

5. Conclusion and Next Steps

In this report, we summarized the activities within WP5 for the second year of the i-RISC project.

From the task 5.4 perspective, a new simulation based methodology to study the impact of gate failure on complex combinational circuits is developed. We proposed two different circuit optimization methodologies based on local transformation rules have been presented. A synthesis algorithm based on these rewriting techniques was also presented that improves the circuit reliability. Application of the synthesis algorithm on the MCNC benchmark circuits with node count from 30 upto 1500 resulted in improving overall circuit reliability by up to 10%.

Going forward, we plan to extend the local transformation rule set to encompass more possible scenarios. From a reliability perspective, we believe that the AIG data structure is more appropriate to represent combinational circuits. In particular the fact that AIG are non canonical (i.e. there exist more graphs representing the same logic function) can be exploited to further improve reliability. We intend 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 have a huge impact as far as reducing the error on the output node is concerned. We are currently focussing on developing a solid mathematical based analogy that can guide our optimization algorithms to insert these nodes automatically based on the structure of the circuit.

We also started working on the Task 5.5 with a number of encouraging developments on computing the Boole-Shannon limit for noisy circuits. A preliminary investigation on the various possible avenues to reach mathematical formulation of what are the limitation and achievable trade off of the i-RISC approach to error prone circuitry is reported. Two research direction are foreseen for continue this work. The first it is to understand the implication of the corner scenario cost analysis presented and expand it into a unify 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 to 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 limits in telecommunication are presented/valid only for restricted channels. It is foreseen that formulation of these limit for non-linear circuit may require years of fundamental research on many mathematical tools before being possible.

References

[ABC12] ABC: A system for Sequential Synthesis and Verification, Berkeley Verification and Synthesis Research Center, 2012 (www.eecs.berkeley.edu/~alanmi/abc/abc.htm).

[Almukhaizim06] 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. IEEE, 2006.

[Bahar03] I. Bahar, J. L. Mundy, and J. Chen, "A probabilistic-based design methodology for nanoscale computation," in *International Conference on Computer Aided Design*, 2003, pp. 480–486.

[**Betz99**] V. Betz, J. Rose, A. Marquardt, Architecture and CAD for deep-submicron FPGAs, Kluwer Academic Publishers, 1999.

[Bhaduri05] D. Bhaduri and S. Shukla, "Nanolab: A tool for evaluating reliability of defect-tolerant nano architectures," in *IEEE Transactions on Nanotechnology, 4(4)*, 2005, pp. 381–394.

[**Bjesse04**] P. Bjesse and A. Boralv, "DAG-aware circuit compression for formal verification," in *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, 2004, pp. 42 – 49

[Borkar05] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,"Micro, IEEE, vol. 25, no. 6, pp. 10–16, 2005.

[Brayton10] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in Proceedings of the 22Nd International Conference on Computer Aided Verification, pp. 24–40, 2010.

[Brayton87] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "Mis: A multiple-level logic optimization system," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 6, no. 6, pp. 1062–1081, 1987.

[Brayton10] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool", *Proceedings of CAV'10, Springer, LNCS 6174*, 2010, pp. 24–40.

[Bryant86] R. Bryant, "Graph-based algorithms for Boolean function manipulation", *IEEE Transactions on Computers*, Vol. 35, No. 8, 1986, pp. 677–691.

[Burch93] R. Burch, F. Najm, P. Yang, T. Trick, A Monte Carlo approach to power estimation, in IEEE Trans. on VLSI Systems., Vol. 1, Issue 1, pp. 63-71, 1993.

[Chen14] Chen J; Spagnol, C.; Grandhi, S.; Popovici, E.; Cotofana, S.; Amaricai, A., "Linear Compositional Delay Model for the Timing Analysis of Sub-Powered Combinational Circuits," VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on , vol., no., pp.380,385, 9-11 July 2014.

[Choudhury09] M.R. Choudhury, and K. Mohanram, "Reliability analysis of logic circuits." in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28(3)*, 2009, pp. 392–405.

[Constantinescu03] C. Constantinescu, "Trends and challenges in vlsi circuit reliability,"Micro, IEEE, vol. 23, no. 4,pp. 14–19, 2003.

[Darringer81] J. Darringer, W. H. Joyner, C. Berman, and L. Trevillyan, "Logic synthesis through local transformations," IBM Journal of Research and Development, vol. 25, no. 4, pp. 272–280, 1981.

[DeMicheli90] F.Mailhot and G. DeMicheli, "Technology mapping using Boolean matching and don't care sets," in *Design Automation Conference, 1990. EDAC. Proceedings of the European*, Mar. 1990, pp. 212 –216.

[Diana13] 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.

[Ercolani89] S. Ercolani, M. Favalli, et. al., "Estimate of signal probability in combinational logic networks," in *Proceedings of the 1st European Test Conference*, 1989, pp. 132–138.

[Grandhi14-a] Grandhi, S.; Spagnol, C.; Popovici, E., "Reliability analysis of logic circuits using probabilistic techniques," Microelectronics and Electronics (PRIME), 2014 10th Conference on Ph.D. Research in , vol., no., pp.1,4, June 30 2014-July 3 2014.

[Grandhi14-b] Grandhi, S.; Spagnol, C.; Jiaoyan Chen; Popovici, E.; Cotafona, S., "Reliability aware logic synthesis through rewriting," System-on-Chip Conference (SOCC), 2014 27th IEEE International, vol., no., pp.274,279, 2-5 Sept. 2014.

[Gallager63] R. G. Gallager. Low-Density Parity-Check Codes . MIT Press, 1963.

[Kuznetsov73] A. Kuznetsov, "Information storage in a memory assembled from unreliable components," Prob. Inf. Transmission, vol. 9, pp. 254-264, 1973.

[Han11] J. Han, H. Chen, E. Boykin, and J. A. B. Fortes, "Reliability evaluation of logic circuits using probabilistic gate models," in *Microelectronics Reliability, 51*, 2011, pp. 468–476.

[Hadjicostis05] C. N. Hadjicostis and G. C. Verghese, "Coding approaches to fault tolerance in linear dynamic systems," IEEE Trans. Inf. Theory , vol. 51, no. 1, pp. 210-228, Jan. 2005

[i-RISC/D3.1] FP7-ICT/FET-OPEN/ i-RISC project, Deliverable 3.1, "Fault tolerant LDPC encoding and decoding", January 2014.

[i-RISC/D5.1] FP7-ICT/FET-OPEN/ i-RISC project, Deliverable 5.1, "Data Structures and Design Flow for Fault Tolerant Circuit Synthesis", January 2014.

[Kaeslin08] H. Kaeslin, Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication, 1st ed. New York, NY, USA: Cambridge University Press, 2008.

[Krishnaswamy05] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Ac- curate reliability evaluation and enhancement via probabilistic transfer matrices." in *Proceedings of the Design, Automation and Test in Europe*, 2005, pp. 282–287.

[Hurst85] S. Hurst, D.Miller, and J.Muzio, Spectral Techniques in Digital Logic. Academic Press, 1985.

[Mehrotra11] R. Mehrotra, T. English, M. Schellekens, S. Hollands, and E. Popovici, "Timing-driven power optimisation and power-driven timing optimisation of combinational circuits," Journal of Low Power Electronics, vol. 7, no. 3, pp. 364–380, 2011.

[Mehrotra13] Rashmi Mehrotra, "Systematic Delay-driven Power Optimisation and Power-driven Delay Optimisation of Combinational Circuits", PhD Thesis 2013, University College Cork, cora.ucc.ie.

[Mishchenko06-a] A. Mishchenko, S. Chatterjee and R. Brayton, "Dag-aware AIG rewriting a fresh look at combinational logic synthesis", *Proceedings of the 43rd annual conference on Design automation*, 2006, pp. 532–535.

[Mishchenko06-b] A. Mishchenko and R. K. Brayton, "Scalable logic synthesis using a simple circuit structure," in Proc. IWLS, pp. 15–22, 2006.

[Mischchenko13] A. Mishchenko, N. Een, R. Brayton, M. Case, P. Chauhan, and N. Sharma, "A semi-canonical form for sequential AIGs", Proc. DATE'13, pp. 797-802.

[Mohanram 09] M. Choudhury and K. Mohanram, “Reliability analysis of logic circuits,” Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 28, no. 3, pp. 392–405, 2009.

[Mohanram13] 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.

[Neumann56] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components", Automata Studies, vol. 34, pp.43 -98, 1956.

[Palem12] K. Palem and A. Lingamneni, “What to do about the end of moore’s law, probably!,” in Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, pp. 924–929, 2012.

[Patel03] K. Patel, I. Markov, and J. Hayes, “Evaluating circuit reliability under probabilistic gate-level fault models.” in *Proceedings of the International Workshop on Logic Synthesis*, 2003, pp. 59–64.

[Pedram96] S. Iman and M. Pedram, “Pose: power optimization and synthesis environment,” in Design Automation Conference Proceedings 1996, 33rd, pp. 21–26, 1996.

[Pippenger89] N. Pippenger, “Invariance of Complexity Measures for Networks with Unreliable Gates,” *J. Assoc. Comput. Mach*. 36, p. 531, 1989.

[Pippenger90] N. Pippenger, “Developments in 'The Synthesis of Reliable Organisms from Unreliable Gates,” *Proceedings of Symposia in Pure Mathematics*, pp. 311-324, 1990.

[Rejimon05] T. Rejimon and S. Bhanja, “Time and space efficient method for accurate computation of error detection probabilities in VLSI circuits,” in *IEE Proceedings on Computers and Digital Techniques, 152(5)*, 2005, pp. 679–685.

[Rejimon06] ——, “Probabilistic error model for unreliable nano-logic gates,” in *6th IEEE Conference on Nanotechnology, 1*, 2006, pp. 47–50.

[Smita07] 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. IEEE, 2007.

[Taylor06] E. Taylor, J. Han, and J. Fortes, “Towards accurate and efficient reliability modeling of nanoelectronic circuits,” in *6th IEEE Conference on Nanotechnology, 1*, 2006, pp. 395–398.

[Taylor-68a] M. Taylor, "Reliable information storage in memories designed from unreliable components," Bell Syst. Tech. J., vol. 47, pp. 2299-2337,1968.

[Taylor-68b] M. Taylor, "Reliable computation in computing systems designed from unreliable components," Bell Syst. Tech. J., vol. 47, pp. 2339-2266,Dec. 1968.

[Todorovich02] E. Todorovich, M. Gilabert, G. Sutter, S. Lopez-Buedo, and E. Boemo, A tool for activity estimation in FPGAs, in the Intl. Conf. on Field-Programmable Logic (FPL), pp. 340-349, 2002.

[Vasic07] Vasic, B.; Chilappagari, S.K., "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 , vol.54, no.11, pp.2438,2446, Nov. 2007.

[Vittoz14] E. A. Vittoz, Low-Power CMOS Circuits. CRC Press, Nov 2005, ch. Weak Inversion for Ultimate Low-Power Logic, pp. 1–18, 0.

[Vrudhula06] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the nbti effect for reliable design," in Custom Integrated Circuits Conference, 2006. CICC '06. IEEE, pp. 189–192, 2006.