Tools and Techniques for Evaluating Reliability Trade-offs for Nano-Architectures

Debayan Bhaduri

Thesis submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Sandeep K. Shukla, Chair
Dong S. Ha, Member
Binoy Ravindran, Member

April 30, 2004
Blacksburg, Virginia

Keywords: Nanotechnology, Gibbs distribution, TMR, CTMR, PRISM, Gaussian, reliability, entropy, probabilistic model checking, interconnect noise, modeling, defect-tolerant architecture, granularity

Tools and Techniques for Evaluating Reliability Trade-offs for Nano-Architectures

Debayan Bhaduri

Abstract

It is expected that nano-scale devices and interconnections will introduce unprecedented level of defects in the substrates, and architectural designs need to accommodate the uncertainty inherent at such scales. This consideration motivates the search for new architectural paradigms based on redundancy based defect-tolerant designs. However, redundancy is not always a solution to the reliability problem, and often too much or too little redundancy may cause degradation in reliability. The key challenge is in determining the granularity at which defect tolerance is designed, and the level of redundancy to achieve a specific level of reliability. Analytical probabilistic models to evaluate such reliability-redundancy trade-offs are error prone and cumbersome, and do not scale well for complex networks of gates. In this thesis we develop different tools and techniques that can evaluate the reliability measures of combinational circuits, and can be used to analyze reliability-redundancy trade-offs for different defect-tolerant architectural configurations. In particular, we have developed two tools, one of which is based on probabilistic model checking and is named NANOPRISM, and another MATLAB based tool called NANOLAB. We also illustrate the effectiveness of our reliability analysis tools by pointing out certain anomalies which are counter-intuitive but can be easily discovered by these tools, thereby providing better insight into defect-tolerant design decisions. We believe that these tools will help furthering research and pedagogical interests in this area, expedite the reliability analysis process and enhance the accuracy of establishing reliability-redundancy trade-off points.

Acknowledgements

I would like to thank Dr. Sandeep K. Shukla for his guidance, motivation and support throughout this work. I also thank Dr. Dong S. Ha and Dr. Binoy Ravindran for serving as committee members for this thesis.

We acknowledge the support of NSF grant CCR-0340740 which provided for the funding for the work reported in this thesis.

I extend my love and affection to **Didi**, without her I would not have come this far.

I would like to thank my childhood buddy **Tamal Bhattacharya** for being there for me through thick and thin. Growing up together with him has been a wonderful experience.

From the FERMAT Lab, I would like to thank the following:

Suhaib Syed: For "tolerating me at home and at work, and being my best friend at Tech".

Hiren D. Patel: For "motivating me with his hard work and dedication".

Deepak Abraham MathaiKutty: For "keeping me amused by his quaint dress codes".

David Berner: For "being supportive about weekend parties".

Gaurav Singh: For "helping me be in touch with hindi".

I would also like to take this opportunity to thank **Habeeb Abdullah**, **Animesh Patcha**, **Madhup Chandra** and **Shekhar Sharad Agrawal** for being great pals at Tech. In spite of our busy schedule, we always found time to hang out together.

Dedication

I dedicate this thesis to my parents

Mrs. Bharati Bhaduri

and

Mr. Dwipendra Nath Bhaduri

Contents

List of Figures

List of Tables

Chapter 1

Introduction

New technologies for building nanometer-scale devices are expected to provide the means for constructing much denser logic and thinner wires. These technologies provide a mechanism for the construction of a useful Avogadro computer [34] that makes efficient use of extremely large number (Avogadro number is in the order of 10^{23}) of small devices computing in parallel. But the economic fabrication of complete circuits at the nanometer level remains challenging because of the difficulty of connecting nanodevices to one another. Also, experts predict that these devices will have high defect density due to their minuscule dimension, quantum physical effects, reduced noise margins, system energy levels reaching thermal limits of computation, manufacturing defects, aging and many other factors. In the near future we will not be able to manufacture large, defect-free integrated circuits. Thus, designing reliable system architectures that can work around these problems at run-time becomes important.

General computer architectures till date have been based on principles that differentiate between memory and processing and rely on communication over buses [10]. Nano-

electronics may change these basic principles. Processing will be cheap and copious, interconnection expensive and prevalent. This will tend to move computer architectures in the direction of locally connected, redundant and reconfigurable hardware meshes that merge processing and memory. At the same time, due to fundamental limitations at the nanoscale, micro-architects will be presented with new design challenges. For instance, the methodology of using global interconnections and assuming error-free computing may no longer be possible. Due to the small feature size, there will be a large number of nanodevices at a designer's disposal. This will lead to redundancy based defect-tolerant architectures, and thus some conventional techniques such as Triple Modular Redundancy (TMR), Cascaded Triple Modular Redundancy (CTMR) and multistage iterations of these may be implemented to obtain high reliability. However, too much redundancy does not necessarily lead to higher reliability, since the defects affect the redundant parts as well. As a result, in-depth analysis is required to find redundancy levels for specific reliability measures of different architectural configurations.

1.1 Motivation

[62, 8, 9] show that redundancy based defect-tolerant architectures (resource redundancy) for different Boolean networks have unique reliability-redundancy trade-off points. But, there are certain key challenges involved in determining such trade-off points [61, 6]. These are as follows:

- The arbitrary augmentation of unreliable devices could result in the decrease of the reliability of an architecture.
- For each specific architecture and a given failure distribution of devices, once an

optimal redundancy level is reached, any increase or decrease in the number of devices may lead to less reliable computation.

- Redundancy may be applied at different levels of granularity, such as gate level, logic block level, functional unit level etc.
- Determining the correct granularity level for a specific Boolean network is crucial in such trade-off analysis.

Theoretical models in [47, 48, 60] have been proposed in the past, and results from information theory [65, 36] have been used to analyze redundancy-reliability trade-offs for different defect-tolerant architectures. However, since such *analytical* methodologies involve complex combinatorial arguments and conditional probability computations, they often lead to erroneous results. [62] proposes a probabilistic model checking based automation approach, that found such a flaw in the analytical approach of [47]. This motivates the need for automating reliability analysis of systems. We have developed automation methodologies to evaluate reliability measures of different redundant architectural configurations for arbitrary Boolean networks. Before we explain our tools and techniques for evaluating such trade-offs, let us discuss some interesting analytical reliability computation models and hybrid defect-tolerant architectures.

1.2 Analytical Approaches for Reliability Analysis

In this section we survey some theoretical models that have been used to analyze reliability-redundancy trade-off points of some classical defect-tolerant architectures. Several techniques have been suggested in the past to mitigate the effects of both faults and defects in designs [85, 50, 48]. One of the techniques discussed in fault

tolerant literature is the multiplexing based defect-tolerance technique initiated by von Neumann [85], and is based on massive duplication of imperfect devices and randomized error-prone interconnects.

In 1952, von Neumann introduced a redundancy technique called NAND multiplexing [85] for constructing reliable computation from unreliable devices. He applied this technique to majority gates as well and illustrated the generality of this multiplexing based defect-tolerant architectural configuration. Von Neumann also showed that such an architectural configuration can perform computations with high reliability measures, if the failure probabilities of the gates are sufficiently small and failures are independent. Pippenger [65] showed that von Neumann's construction works only when the probability of failure per gate is strictly less than $1/2$, and that computation in the presence of noise (which can be seen as the presence of transient fault) requires more layers of redundancy. Dobrushin et al. [31] also showed that a logarithmic redundancy is necessary for some Boolean function computations, and is sufficient for all Boolean functions. In [60], NAND multiplexing was compared to other techniques of fault-tolerance and theoretical calculations showed that the redundancy level must be quite high to obtain acceptable levels of reliability.

The multiplexing based defect-tolerance scheme replaces a single logic device by a multiplexed unit with N copies of every input and output of the device. The N devices in the multiplexing unit process the copies of the inputs in parallel to give N outputs. Each element of the output set will be identical and equal to the original output of the logic device, if all the copies of the inputs and devices are reliable. However, if the inputs and devices are in error, the outputs will not be identical. To tackle such an error-prone scenario, some critical level $\Delta \in (0, 0.5)$ is defined. The output of the multiplexing unit is considered stimulated (taking logical value `true`) if at least $(1-\Delta)\cdot N$ of the outputs are stimulated, and non-stimulated (taking logical value `false`) if no more

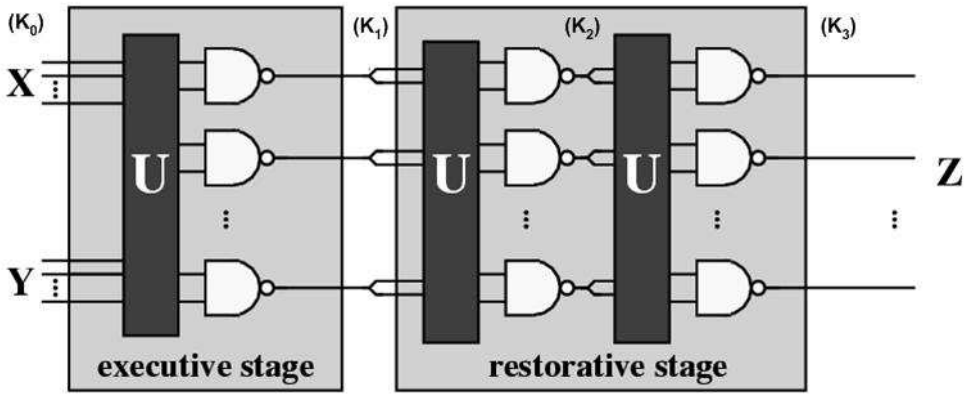


Figure 1.1: A NAND multiplexing unit from [62]

than $\Delta \cdot N$ outputs are stimulated. If the number of stimulated outputs is in the interval $(\Delta \cdot N, (1-\Delta) \cdot N)$, then the output is undecided, and hence a malfunction will occur. The basic design of a von Neumann multiplexing technique consists of two stages: the *executive stage* which performs the basic function of the processing unit to be replaced, and the *restorative stage* which reduces the degradation in the executive stage caused by erroneous inputs and faulty devices. As shown in Figure 1.1, NAND multiplexing is an architectural configuration wherein a single NAND gate is the processing unit of a multiplexing based defect-tolerant system. The unit U performs random permutation of the input signals i.e. each signal from the first input bundle is randomly paired with a signal from the second input bundle to form the input pair of one of the duplicated NAND gates.

Han and Jonker [47] extend this technique to a rather low degree of redundancy, and elucidate the stochastic markov [63] nature of the system. They also investigate a system architecture based on NAND multiplexing for SETS, where transient errors are introduced due to random background charges. Stochastic analysis of the chains of

stages (Figure 1.1) is performed to evaluate optimal redundancy factors for systems, and bounds on the failure probability per gate that can be tolerated for specific reliability and redundancy levels.

For the NAND multiplexing system shown in Figure 1.1, let X be the set of lines in the first input bundle that are stimulated. Consequently, $(N - X)$ lines are not stimulated (logic low). Y and Z are also corresponding sets for the second input and the output bundles. [47] also assumes a constant probability of gate failure ε, and that the faulty gates invert their outputs (von Neumann fault). If the sets X, Y and Z have $(\bar{x}, \bar{y}, \bar{z})$ elements respectively, then these are the relative excitation levels of the two input bundles and of the output bundle, respectively. The stochastic distribution of $\bar{z}$ has to be determined in terms of the given input distributions ($\bar{x}$ and $\bar{y}$).

In [85], von Neumann concluded that, for extremely large N, the stochastic variable $\bar{z}$ is approximately normally distributed. He also determined an upper bound of 0.0107 for the gate failure probability that can be tolerated. In other words, according to von Neumann, if the failure probability per gate is greater than this threshold, then the probability of the NAND multiplexing system failing will be larger than a fixed, positive lower bound, no matter how large a bundle size is used. Recently, it was shown that, if each NAND gate fails independently, the tolerable threshold probability of each gate will be 0.08856 [36].

For a single NAND gate in the multiplexing scheme (Figure 1.1), assume $\bar{x}N$ and $\bar{y}N$ are input lines stimulated in each input bundle respectively. If the error probabilities for the two input bundles of the multiplexing configuration are independent, the probability of each gate's output being logic high is $\bar{z} = 1 - \bar{x}\bar{y}$ (assuming error-free NAND operation). If the gate failure probability is ε, the probability of the output being stimulated is:

$$\bar{z} = (1 - \bar{x}\bar{y}) + \varepsilon(2\bar{x}\bar{y} - 1) \tag{1.1}$$

Equation 1.1 is valid only for the von Neumann fault model (erroneous gate inverts the correct output). The NAND multiplexing unit constitutes a Bernoulli sequence because the gates are assumed to function independently. Therefore, the probability distribution of the stimulated outputs can be expressed as a binomial distribution. The probability of exactly k outputs being stimulated is:

$$P(k) = \binom{N}{k} \bar{z}^k (1 - \bar{z})^{N-k} \tag{1.2}$$

When N is sufficiently large and $\bar{z}$ is extremely small, the probability distribution of exactly k outputs being stimulated from the N output lines of the NAND multiplexing stage can be approximated to a Poisson distribution. If both inputs of the NAND gates have high probability of being logic high, the stimulated outputs are then considered erroneous. The reliability of the system can then be computed as the probability of the number of stimulated outputs being below a threshold ($P(k \leq x)$). Since the number of stimulated outputs is a stochastic variable that is binomially distributed, the central limit (De Moivre-Laplace) theorem applies when N is sufficiently large and $0 < \bar{z} < 1$. In this case, Equation 1.2 can be rewritten as:

$$P(k \leq x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sqrt{N\bar{z}(1-\bar{z})}} e^{-1/2(t - N\bar{z}/\sqrt{N\bar{z}(1-\bar{z})})^2} \tag{1.3}$$

Thus, [47] shows analytically that, for the executive stage of NAND multiplexing, the probability of the number of stimulated outputs is theoretically a binomial distribution for small values of N. The authors also show that the probability could be approximated

to a Gaussian distribution with mean $N\bar{z}$ and standard deviation $\sqrt{N\bar{z}(1-\bar{z})}$, when N is sufficiently large and $0 < \bar{z} < 1$. The authors then go on to demonstrate how additional restorative stages improve fault-tolerance. The discussion above illustrates that the number of stimulated outputs of each NAND multiplexing stage is a stochastic variable and its state space is $A = [0, 1, 2, ...N-1, N]$. This stochastic variable is denoted as $\bar{\xi}_n$, where n is the index of the multiplexing stage. Thus, the evolution of $\bar{\xi}_n$ in the multiplexing configuration is a stochastic process, and with fixed bundle size and gate failure probability, the distribution of $\bar{\xi}_n \ \forall n$ (stages) depends on the number of stimulated inputs of the nth multiplexing unit (outputs of the $n-1$ th unit). This can be expressed as follows:

$$P(\bar{\xi}_n \in A \mid \bar{\xi}_1 = k_1, \bar{\xi}_2 = k_2, ...\bar{\xi}_{n-1} = k_{n-1}) \\ = P(\bar{\xi}_n \in A \mid \bar{\xi}_{n-1} = k_{n-1}) \tag{1.4}$$

Equation 1.4 is the condition for a stochastic process to be a Markov process. As indicated in Figure 1.1, $k_1,\ k_2, ...k_{n-1}$ are the number of stimulated outputs of the stages represented by the indices respectively. The evolution of $\bar{\xi}_n$ in the NAND multiplexing system is therefore a Markov process, or a discrete Markov chain. The transition probability of a stochastic Markov chain indicates the conditional probability from one specified state to another. As illustrated in [47], the transition probability matrix Ψ for each $\bar{\xi}_n$ is identical and independent of the multiplexing stage (n). Thus, it can be inferred that $\bar{\xi}_n$ evolves as a homogeneous Markov chain. Therefore, an initial probability distribution and a transition probability matrix are sufficient to get all output distributions. For a NAND multiplexing system with n individual stages, the output distribution of the configuration is:

$$P_n = P_0 \Psi^n \tag{1.5}$$

It is also pointed out in [47] that when the number of multiplexing stages (n) is large, Ψ^n approaches a constant matrix π, and each row of π is identical. This indicates that as n becomes extremely large, not only the transition probabilities in a NAND multiplexing system will get stable, but also the output distribution will become stable and independent of the number of multiplexing stages. Experiments indicate that this defect-tolerant technique requires a rather large amount of redundant components to give acceptable reliability levels. This makes NAND multiplexing inefficient for the protection against permanent faults, normally compensated by reconfiguration techniques. However, this architectural configuration may be a solution for ultra large integration of highly unreliable nanometer-scale devices affected by dominant transient errors.

[48] reviews the NAND multiplexing and reconfiguration fault tolerant techniques, and presents a defect and fault tolerant architecture in which multiplexing (low degree of redundancy) is combined with a massively reconfigurable architecture. Reconfigurable architectures are discussed in details in Chapter 2. The system performance of this architecture is evaluated by studying its reliability, and this shows that the suggested architecture can tolerate a device error rate of up to 10^{-2}. The architectural configuration is seen to be efficiently robust against both permanent and transient faults for an ultra-large integration of highly unreliable nanodevices. For the NAND multiplexing configuration, Equation 1.2 can be used to compute the reliability of the system if the faulty devices are independent and uniformly distributed. This scenario may be reasonable when the dominant faults are transient in nature. However, this binomial distribution model is not sufficient to describe the manufacturing defects and permanent faults. The device components are not statistically independent but rather

correlated since defects tend to cluster on a chip [51]. Thus, Equation 1.2 is not appropriate for computing reliability measures of the multiplexing system. [48] indicates that such manufacturing defects can be modeled with a continuous probability distribution function $f(r)$ where r is the component reliability. Thus, the new reliability evaluation formula is:

$$R(k) = \int_0^1 \binom{N}{k} \bar{z}^k (1-\bar{z})^{N-k} f(r) dr \tag{1.6}$$

The success of the approach depends on finding appropriate parameters for Equation 1.6, and [48] follows Stappers beta distribution model [46]. Han and Jonker also discuss about the analytical methodology to compute reliability of reconfigurable architectures. The basic logic circuit blocks in the processor arrays on a reconfigurable chip are referred to as processing elements (PEs), and these are sometimes associated with local memories. In very large chips, reliability can be enhanced by adding spare PEs to the architectural configuration. Instead of trying to achieve complete fault tolerance, most techniques aim at optimizing probability of survival, defined as the percentage of defects that can be eliminated by the reconfiguration approach. Reconfiguration approaches are categorized as local or global [37]. Global approaches usually involve far more complex reconfiguration algorithms than local solutions. [48] assumes that all PEs (also called modules) are identical, so that any spare module can be substituted for a defective one, provided sufficient interconnection paths exist in the cluster. If in an array there are r spares out of n identical modules, then at least $n - r$ must be error-free for proper operation. The reliability of the array is given by $R_n = \sum_{m=n-r}^{n} R_{mn}$, where R_{mn} is the probability of exactly m out of n modules being fault free. Assuming the modules have the same reliability measure R_0, and are statistically independent, the probability distribution of the number of defect free modules m can be modeled as a binomial distribution:

$$R_{mn} = \binom{n}{m} R_0^m (1 - R_0)^{n-m} \tag{1.7}$$

Again, these defective modules in an array are not uniformly distributed but rather correlated, and Stappers model is used to improve the reliability calculation of correlated modules [46]. The authors compute the reliability measures of a hierarchical approach that uses NAND multiplexing at the lowest level and then uses redundancy (spares) for reconfiguration at three additional implementation levels. The authors show that for an individual gate failure probability of 10^{-2} and with 10^{12} devices on the chip, this architectural approach achieves a chip-level reliability of 99.8% with a redundancy factor of less than 10.

Nikolic et al. [60] also use theoretical models to compare the relative reliability measures of R-fold modular redundancy (details in Chapter 2), NAND multiplexing and reconfiguration for different device failure rates. It is shown that for a chip with 10^{12} devices, and with a reliability level that specifies that the chip must work with 90% probability, RMR is the least effective followed by NAND multiplexing and reconfiguration providing the best computational reliability. It is also indicated in [60] that for individual device failure rates of 0.01 to 0.1, the redundancy factors required to provide high reliability of computation (90 % overall working probability) are very large (10^3 to 10^5) for such defect-tolerant architectural configurations.

1.3 Main Contributions of this Thesis

Figure 1.2 shows the design flow of a digital system. It indicates the different stages of the system design starting from the specification (higher level of abstraction) to the netlist generation (detailed implementation/lower abstraction level). There are other

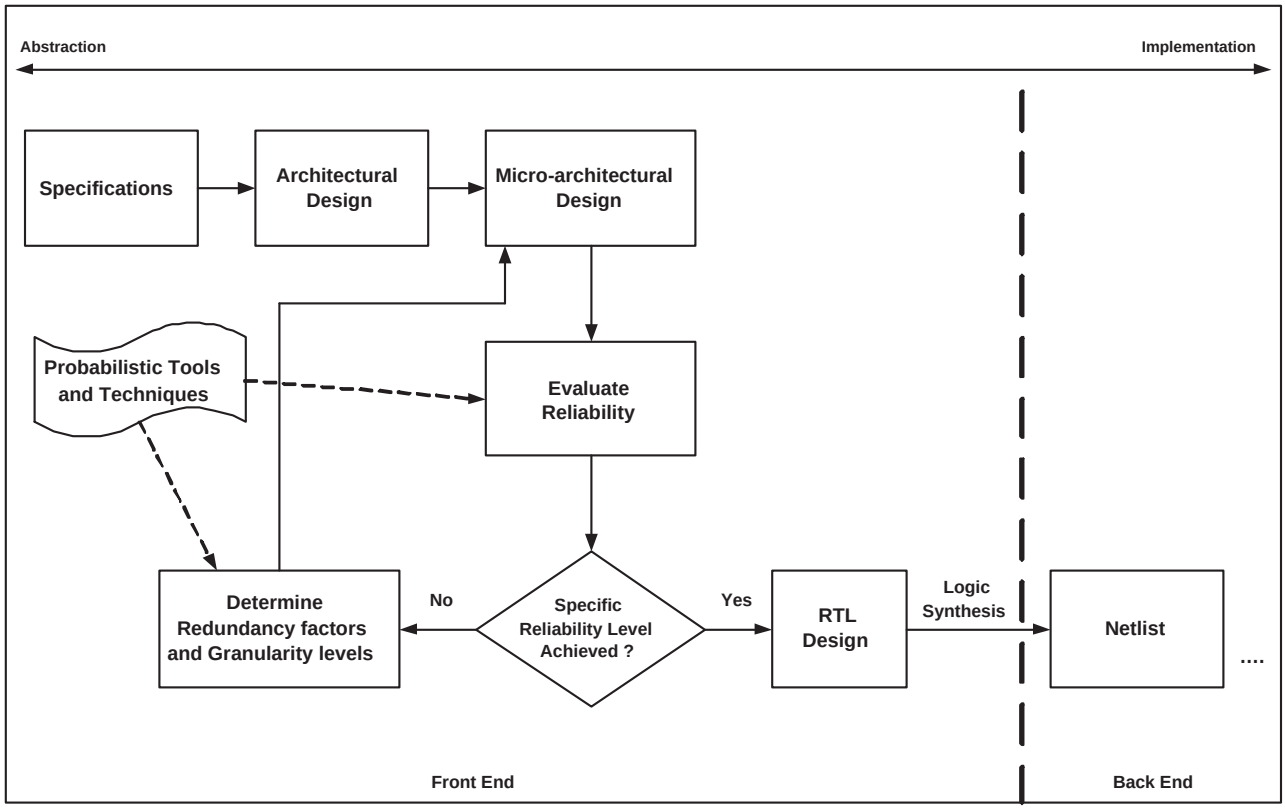


Figure 1.2: The design flow of an electronic or information-processing system

back-end specific processes performed after logic synthesis from the RTL design, such as logic optimization, physical design, layout etc, that finally lead to the fabricated system. These have been omitted in Figure 1.2 for simplicity. The front-end design consists of translation of the system specifications to an architectural design, which is refined to a micro-architectural design. This is a detailed architectural description of the system. Such a design methodology for digital and information processing systems has to guarantee acceptable reliability levels. Therefore, quick and easy techniques are required to measure the reliability of such micro-architectural designs. If the desired reliability levels cannot be achieved with the architectural configuration, the design has to be made more robust. This may involve augmenting more redundancy and at

different granularity levels (such as gate level, logic block level).

As discussed earlier, specific Boolean networks have different reliability-redundancy trade-off points and incorporating arbitrary number of redundant devices in the architecture may even degrade the reliability of computation [62]. Also, for complex Boolean networks, most of the analytical methodologies discussed earlier do not scale well, and are error-prone and cumbersome. Such limitations necessitate the automation of such methodologies. Figure 1.2 shows the *probabilistic tools and techniques* reported in this thesis that we believe will expedite and ease the evaluation of reliability measures and redundancy/granularity levels for specific micro-architectural descriptions. Thus, the main contributions of this thesis are as follows:

- Development of automation methodologies for the analysis of reliability-redundancy trade-offs of different redundancy (resource) based defect-tolerant architectural configurations.
- Development of tools that perform quick and accurate computation of best and worst case reliability characteristics by setting upper and lower bounds on device and interconnect failure probabilities. The inputs to these tools are:
    * Any arbitrary combinational circuit.
    * Any hardware redundancy based defect-tolerant technique.
    * Redundancy factor and granularity level.
    * Probability distribution of signal energy levels at the prime inputs of logic networks.
    * Error distributions of the component gates.
    * Signal noise at inputs and interconnects.

Given such inputs, these tools compute reliability measures of logic networks in terms of different metrics. The outputs of these tools may

* Probability distribution of signal energy levels at the prime outputs of the logic networks.
* Entropy at the prime outputs and interconnects of circuits.
* Probability of atmost 10% errors at the outputs.
* Expected percentage of errors at the outputs.

1.3.1 Brief Introduction to Our Tools

In this subsection, we describe our reliability evaluation tools, and how these will expedite the design cycle of a electronic or information processing system. These tools are targeted for systems that are built out of nano-scale devices in particular. We describe a MATLAB based tool code named **NANOLAB** since it is based on MATLAB [5, 7, 13], and a probabilistic model checking based tool named **NANOPRISM** [6, 8]. One difference between our automation techniques and the standard analytical approaches (discussed in the previous section) is as follows: we evaluate the reliability of specific cases as opposed to considering the general framework, and hence are not necessarily restricted by analytical bounds.

1 Conventional digital signals are bimodal, meaning that the logic low or high are defined as discrete voltage/current levels. Due to device miniaturization at the nanoscale, the notion of being a binary zero or one will change. A probabilistic design methodology based on Markov Random Fields (MRF) [57] proposed in [2] introduces a new information encoding and computation scheme where signals

are interpreted to be logic low or high over a continuous energy distribution. The inputs and outputs of the gates in a combinational block are realized as nodes of a Markov network and the logic function for each gate is interpreted by a Gibbs distribution [71] based transformation. This computational scheme also exploits the fact that maximizing the probability of energy levels at each node of the network is equivalent to minimizing the entropy of Gibbs energy distribution that depends on neighboring nodes. The probability of a logic variable can be calculated by marginalizing over the set of possible states of the neighboring logic variables and propagated to the next node in a Boolean network by using Belief Propagation [52]. **NANOLAB** automates this probabilistic design methodology by computing energy distribution and entropy at the primary/intermediate outputs and interconnects of Boolean networks, and by implementing Belief Propagation. The logic compatibility functions (similar to truth table) [2] for the different component gates of the Boolean network and the energy distribution at the primary inputs are specified to the tool. We have also augmented the capability to model uniform and Gaussian noise at the primary inputs and interconnects of combinational blocks and analyze such systems in terms of entropy at the outputs. Such modeling features in NANOLAB will expedite and enhance the analysis of reliability measures for different defect tolerant architectures.

2 **NANOPRISM** is a probabilistic model checking based tool that applies probabilistic model checking techniques to calculate the likelihood of occurrence of transient defects in the devices and interconnections of nano architectures. NANOPRISM is based on the conventional Boolean model of computation and can automatically evaluate reliability at different redundancy and granularity levels, and most importantly show the trade-offs and saturation points. By saturation point we mean the granularity based redundancy vs. reliability reaches a plateau

meaning that there cannot be any more improvements in reliability by increasing redundancy or granularity levels. It consists of libraries built on PRISM [54, 73] (probabilistic model checker) for different redundancy based defect-tolerant architectural configurations. These libraries also support modeling of redundancy at different levels of granularity, such as gate level, logic block level, logic function level, unit level etc. Arbitrary Boolean networks are given as inputs to these libraries and reliability measures of these circuits are evaluated. This methodology also allows accurate measurements of variations in reliability on slight changes in the behavior of the system's components, for example the change in reliability as the probability of gate failure varies.

1.3.2 Features of these Tools

Here we enlist some major advantages of our tools:

- Our tools allow fast analysis of reliability-redundancy trade-offs for alternative defect tolerant architectures.
- The idea of a new model of computation in [2] for uncertainty based computation is extended in the direction of reliability evaluation and automated by NANOLAB. Previously [2, 21] only show the viability of a MRF based computation scheme and how it can be mapped to nano devices such as Y CNTs [82]. These do not extend the model of computation in the direction of reliability-redundancy trade-off analysis.
- NANOLAB can also compute reliability of systems in the presence of signal noise at interconnects and inputs. Noise is modeled as uniform or Gaussian distribution or combinations of both. This models practical situations the circuits

are subjected to during normal operation. Analysis of reliability measures for redundant architectural configurations of these logic circuits when exposed to such real world noise distributions makes our methodology more effective.

- NANOPRISM is a tool that automates the evaluation of redundancy vs. reliability vs. granularity trade-offs. In [62] a probabilistic model checking based CAD framework is used to evaluate reliability measures of a specific redundancy technique namely NAND multiplexing, and only redundancy vs. reliability trade-offs are computed. NANOPRISM uses such a framework too, but we have developed a library of generic redundancy based defect tolerant architectures where different Boolean networks can be plugged in and analyzed. NANOPRISM analyzes optimal redundancy and granularity levels for specific logic networks and expedites the process of finding the correct trade-offs.

- We are able to illustrate some anomalous counter intuitive trade-off points that would not be possible to observe without significant and extensive theoretical analysis, and the automation makes it easier to analyze these critical parameters.

- From our literature search, we found that the results on reliability measures for different defect tolerant architectures were mostly analytical [77, 47, 60, 38] and none considered *granularity* and *entropy* (discussed in details in Chapter 2) as parameters. However, for complex network of gates, such analytical results may be error-prone. As a result, we believe that scalable automation methodologies to quickly evaluate these figures of merits is crucial for practical use by engineers.

1.4 Organization of this Thesis

Chapter 2 introduces certain basic concepts and terminologies that are used throughout this thesis. We discuss defect-tolerant computing, a probabilistic model of computation from [2] and how to use such a computational scheme to model noise. We also introduce granularity and measure of redundancy, and concepts in probabilistic model checking. Chapter 3 describes NANOLAB and our approach to analyze reliability of systems with the model of computation proposed by Bahar et al. A detailed example along with code snippets is used to elucidate this methodology. We also report reliability measures of different logic networks computed by this tool. Chapter 4 focuses on explaining how we use the NANOPRISM framework to model defect-tolerant architectures such as TMR, CTMR and their iterations for single gates and logic blocks respectively. Reliability-redundancy trade-off points for specific combinational circuits and interesting anomalies are also illustrated in this chapter. We also describe our von Neumann multiplexing based defect-tolerance library that is a part of our NANOPRISM tool in Chapter 5, and compare different multiplexing based architectural configurations. Finally, Chapter 6 concludes the thesis and summarizes plans to extend these automation methodologies in the future.

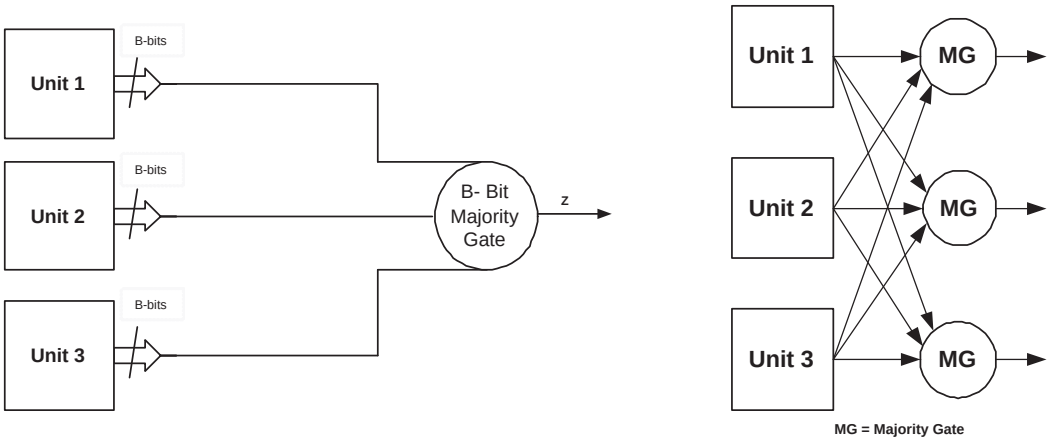
Chapter 2

Background

2.1 Defect Tolerant Architectures

Formally, a *defect-tolerant architecture* is one which uses techniques to mitigate the effects of defects in the devices that make up the architecture, and guarantees a given level of reliability. There are a number of different canonical defect-tolerant computing schemes most of which are based on redundancy. The techniques which we look at are highly generic and concerned with resource redundancy and can be easily extended to nano architectures. Some of these canonical techniques are *Triple modular Redundancy (TMR), Cascaded Triple Modular Redundancy (CTMR)* and multi-stage iterations of these [60]. We have also discussed *multiplexing based defect-tolerance* in Chapter 1.

Triple Modular Redundancy as shown in Figure 2.1(a) entails three similar units working in parallel, and comparison of their outputs with a majority voting logic [60]. The units could be gates, logic blocks, logic functions or functional units. The TMR provides a functionality similar to one of the three parallel units but provides a better probability



(a) Triple Modular Redundancy with a single voter

(b) Triple Modular Redundancy with three voters

Figure 2.1: Different TMR configurations

of working, and is fairly easy to apply to digital logic. The tradeoff is that instead of n devices, $3n$ devices and a majority gate are needed in this configuration, that results in increased circuit area and power, and decreased circuit speed. This architectural configuration can tolerate multiple failures in a single module. Faults or defects in two or more of the three modules will cause the system to fail. Also, the majority voting logic is a single failure point, and three voters can be used instead of just one to improve the reliability of the system further (shown in Figure 2.1(b)). The R-fold modular redundancy is a generalization of the TMR configuration where R units work in parallel instead of 3 and R $\in$ {3,5,7,9.....}. Note that R is always an odd number so as to prevent tie votes.

Cascaded Triple Modular Redundancy is similar to TMR, wherein the units working in parallel are TMR units combined with a majority gate. This configuration forms a CTMR of the first order and therefore TMR can be considered to be 0th order CTMR. Higher orders of CTMR are obtained by multi-stage iterations of these, but this does not

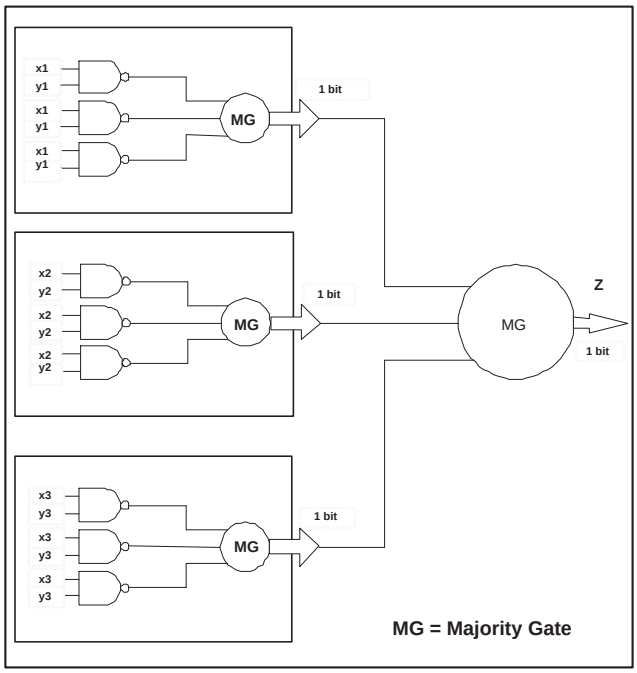


Figure 2.2: Generic Cascaded Triple Modular Redundancy with triple voters: multi-layer voting

mean that the reliability of a system goes up due to this. Increasing the redundancy level also introduces more unreliable devices in the architectural configuration. Figure 2.2 shows a 1st order CTMR configuration where the parallel processing units in each of the three TMR units are NAND gates. This defect-tolerant technique is also called a Cascaded TMR configuration with triple voters (multi-layer voting scheme).

There are many variations to the generic CTMR configuration. Figure 2.3 illustrates a cascading scheme [81, 41] where a logic circuit is divided into smaller units, and TMR is applied to these units. The majority voters are also replicated thrice and placed between the TMR configurations. This architectural configuration can be considered to be a multi-level TMR configuration with triple majority gate logic. The conventional CTMR configuration replicates the Boolean network as a whole (Figure 2.2 illustrates

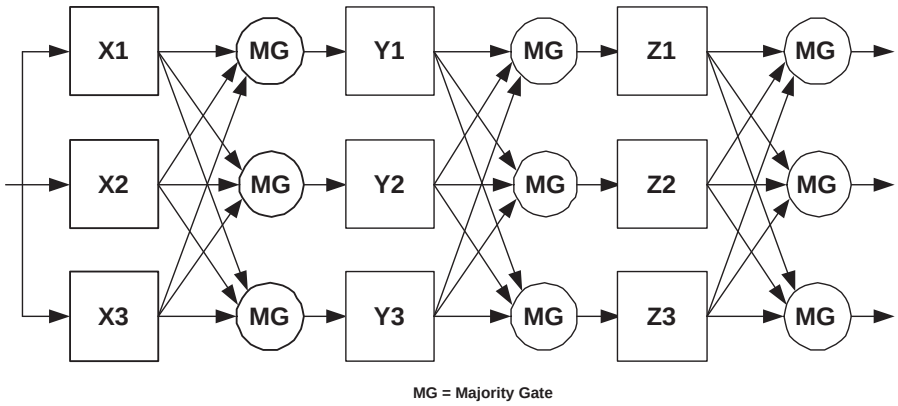


Figure 2.3: Cascaded Triple Modular Redundancy with triple voters: smaller granularity from [41]

a CTMR for a single NAND gate), and the circuit level granularity for the redundant units increases the probability of these failing. Whereas, the multi-level CTMR with triple voters allows a circuit to be apportioned into smaller functional units or modules, and can effectively allow a circuit to withstand more errors across the redundant TMR configurations. Specifically, if the logic network is divided into N functional units, the number of permanent or transient defects that can be tolerated is more than $\lfloor(N/2)\rfloor$. As discussed previously, a simple TMR configuration can provide the same system performance if only one of the redundant units fails. However, by using the cascading technique shown in Figure 2.3, the reliability of the system shown does not degrade even in the presence of errors in the units X1, Y2, and Z3 (more than one erroneous unit).

Let n be the order of the CTMR configuration. The majority logic requires four gates i.e. if a, b and c are inputs, the logic operation is (a∧b)∨(b∧c)∨(c∧a). If F_{n-1} is the total number of devices for a (n-1)th order CTMR, then we can say:

$$F_n = 3F_{n-1} + 5 \tag{2.1}$$

For a single gate, the total number of redundant gates in a nth order CTMR configuration (where n ∈ {0,1,2....}) is:

$$F_n = \frac{21}{2} \cdot 3^n - \frac{5}{2} \tag{2.2}$$

If the total number of gates in a logic circuit is k, then the total number of devices in any nth order configuration will be:

$$F_n = 3 \cdot \frac{2k+5}{3k-1}(3k)^n + \frac{9k^2+6k-20}{3k-1} \tag{2.3}$$

Also, [38] derives the failure rate of a chip with N devices for a RMR defect-tolerant architectural configuration (explained previously) and this can be expressed as:

$$P_{fail}^{chip} = \frac{N}{RN_c + mB}[C(N_c p_f)^{(R+1)/2} + mBp_f] \tag{2.4}$$

In equation 2.4, $C = \begin{pmatrix} R \\ \frac{R-1}{2} \end{pmatrix}$ is the binomial factor, p_f is the probability of an individual device failing, N_c is the total number of devices in one of the R units working in parallel, $N = RN_c$ is the total number of devices and imperfect majority gates have B outputs and mB devices. The probability that an i-th order RMR configuration (cascaded redundancy) works is as follows:

$$P_w^{(i)} = (1 - p_{fail})^{mB}[P_w^{(i-1)^3} + 3P_w^{(i-1)^2}(1 - P_w^{(i-1)})] \tag{2.5}$$

where the majority gate contains mB imperfect devices and $P_w = 1 - P_{fail}$. The authors also show that CTMR is not advantageous when the redundant units have small number of devices and the majority logic is also made from the same devices as the units.

2.2 Majority Multiplexing

Creating computing systems built from devices other than silicon-based transistors is a research area of major interest. A number of alternatives to silicon VLSI have been proposed, including techniques based on molecular electronics, quantum mechanics, and biological processes. One such nanostructure proposed by Lent et al [56, 55] is quantum-dot cellular automata (QCA) that employs arrays of coupled quantum dots to implement Boolean networks. Due to the miniscule size of the quantum dots, extremely high packing densities are possible in CA based architectures. Also, the other advantages are simplified interconnections, and extremely low power-delay product. The fundamental QCA logic device is a three-input majority logic gate consisting of an arrangement of five standard quantum cells: a central logic cell, three inputs and an output cell. A majority gate can be programmed to act as an OR gate or an AND gate, by using one of the three inputs as the control input that determines the AND/OR functionality [83]. This indicates that majority gates are going to play an important role in the future architectural configurations.

This motivates us to consider multiplexing based defect-tolerance (as discussed in Chapter 1) when the logic device is a single majority gate. We, therefore, replace the inputs and output of the gate with N copies and duplicate the majority gate N times in the executive stage, as shown in Figure 2.4. The unit U represents a *random permutation* of the input signals, that is, for each input set to the N copies of the majority gate, three

input signals are randomly chosen from the three separate input bundles respectively. Figure 2.4 also shows the restorative stage which is made using the same technique as the executive stage, duplicating the outputs of this stage to use as inputs to the restorative stage.

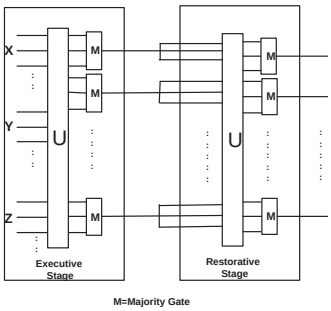


Figure 2.4: A majority multiplexing unit

2.3 Defect-Tolerance through Reconfiguration

A computer architecture that can be configured or programmed after fabrication to implement desired computations is said to be *reconfigurable*. Reconfigurable fabrics such as Field-Programmable Gate Arrays (FPGAs) are composed of programmable logic elements and interconnects, and these can be programmed, or configured, to implement any circuit. Such reconfigurable architectures may mitigate manufacturing defects that will be rampant at the nano substrates. The key idea behind defect tolerance in FPGAs is that faulty components are detected during testing and excluded during reconfiguration. It is expected [59] that reconfigurable fabrics made from next generation manufacturing techniques (CAEN-based technologies where molecular junctions can be made which hold their own state) will go through a post-fabrication testing phase

during which these fabrics will be configured for self-diagnosis. Testing for error-prone devices will not incur either an *area* or a *delay* penalty because the test circuits placed on the fabric during this self-diagnosis phase will utilize resources that will be available later for normal fabric operation (unlike BIST structures). The defect map obtained from this test phase will contain locations of all the defects, and this map can be used by place-and-route tools to layout circuits on the fabric that avoid the defective devices. Thus, the built-in defect-tolerance in such a reconfigurable digital system will ease the requirements on the manufacturing process.

Teramac [50] is an extremely defect-tolerant reconfigurable machine built in HP laboratories. The basic components in Teramac are programmable switches (memory) and redundant interconnections. The high communication bandwidth is critical for both parallel computation and defect tolerance. With about 10% of the logic cells and 3% of the total resources defective, Teramac could still operate 100 times faster than a high-end single processor workstation for some of its configurations. In contrast to the static defect discovery process used in Teramac (test vectors), [59] proposes scalable testing methods that generate defect maps for reconfigurable fabrics in two phases, and dynamic place-and-route techniques for configuartion/programming of the fabric. The reconfigurable architecture particularly targeted by the methodology in [59] is the nanoFabric [39, 40]. The nanoFabric is an example of a possible implementation of a CAEN based reconfigurable device. The post-fabrication testing suggested in [59] comprises of the probability assignment and defect location phases. The probability assignment phase assigns each component a probability of being defective, and discards the components which have a high probability. Thus, this phase identifies and eliminates a large fraction of the defective components. The remaining components are now likely to have a small enough defect rate and can be tested in the defect location phase using simple test-circuit configurations. The authors also point out the

non-adaptiveness of the test-circuit generation. This implies that the results of previous tests are not used to generate new circuits (details in [58]). The Cell Matrix architecture [34, 33] also supports dynamic defect identification and elimination. The Cell Matrix is a fine-grained reconfigurable fabric composed of simple, homogeneous cells and nearest-neighbor interconnect. The cells are programmable, gate-level processors that can be configured by Lookup tables (LUT) [53]. This allows the cells to directly programmed to perform meaningful and extensive logic computations and data processing [32]. The homogeneity of this architecture makes it inherently fault-tolerant. Like Teramac, the Cell Matrix can handle large manufacturing defect rates (permanent faults), and provide high reliability of computation. However, the Cell Matrix architecture also provides defect-tolerance to transient errors (caused due to less amiable operating conditions) due to the inherent self-analyzing, self-modifying, and dynamic local processing capabilities of the architectural configuration. Also, the embryonics architecture [30] is a potential paradigm for future nanoscale computation systems. The objective of developing defect-tolerant and ultra-large integrated circuits capable of self-repair and self-replication makes this architecture viable for future nanoelectronic system design.

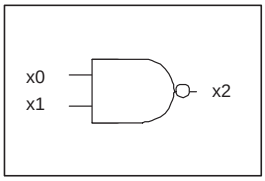


Figure 2.5: A NAND gate

2.4 Markov Random Field based Model of Computation

The advent of non-silicon manufacturing technologies and device miniaturization will introduce uncertainty in logic computation. Newer models of computations [87] have to be designed to incorporate the strong probabilistic notions inherent in nanodevices. Such a probabilistic design methodology has been proposed in [2] that adapts to errors by maximizing the probability of being in valid computational states. This model of computation introduces a new information encoding and computation scheme, where signals are interpreted to be logic low or high over a continuous energy distribution. The basis for this architectural approach is the Markov random network which is based on Markov random fields (MRF) [57]. The Markov random field is defined as a finite set of random variables, $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_k\}$. Each variable λ_i has a neighborhood, N_i, which are variables in $\{\Lambda - \lambda_i\}$.

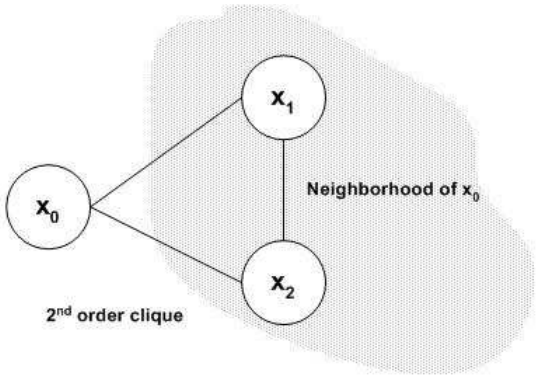


Figure 2.6: The neighborhood of the input of a NAND gate depicted as a network node

The energy distribution of a given variable depends only on a (typically small) neighborhood of other variables that is called a clique. These variables may represent states of nodes in a Boolean network and we might be able to consider effects of noise and

other uncertainty factors on a node by considering the conditional probabilities of energy values with respect to its clique. Due to the Hammersley-Clifford theorem [4],

The conditional probability in Equation 2.6 is Gibbs distribution. Z is the normalizing constant and for a given node i, C is the set of cliques affecting the node i. U_c is the clique energy function and depends only on the neighborhood of the node whose energy state probability is being calculated. KT is the thermal energy (K is the Boltzmann constant and T is the temperature in Kelvin) and is expressed in normalized units relative to the logic energy (clique energy). For example, $KT = 0.1$ can be interpreted as unit logic energy being ten times the thermal energy. The logic energy at a particular node of a Markov network depends only on its neighborhood as discussed earlier. Also, the logic margins of the nodes in a Boolean network decrease at higher values of KT and become significant at lower values. The logic margin in this case is the difference between the probabilities of occurrence of a logic low and a logic high which if high leads to a higher probability of correct computation. This formulation also allows correct analysis of entropy values.

$$P(\lambda_i|\{\Lambda - \lambda_i\}) = \frac{1}{Z} e^{-\frac{1}{KT}\sum_{c\in C} U_c(\lambda)} \tag{2.6}$$

Let us take a specific example to walk through the methodology in [2]. For a two input NAND gate as shown in Figure 2.5, there are three nodes in the assumed logic network: the inputs x_0 and x_1, and the output x_2. Figure 2.6 represents x_0 as a network node, and shows its neighborhood. The energy state of this node depends on its neighboring nodes. The edges in Figure 2.6 depict the conditional probabilities with respect to the other input x_1 and the output x_2 (nodes in the same clique). The operation of the gate is designated by the logic compatibility function $F(x_0,x_1,x_2)$ shown as a truth table in Table 2.1. $F = 1$ when $x_2 = (x_0 \wedge x_1)'$ (valid logic operations). Such a function takes all

i	x_0	x_1	x_2	F
0	0	0	1	1
1	0	0	0	0
2	0	1	1	1
3	0	1	0	0
4	1	0	1	1
5	1	0	0	0
6	1	1	0	1
7	1	1	1	0

Table 2.1: Logic compatibility function for a NAND gate with all possibilities

valid and invalid logic operation scenarios into account so as to represent an energy based transformation similar to the NAND logic. The axioms of the Boolean ring [15] are used to relate F to a Gibb's energy form. Also, the valid input/output states should have lower clique energies than invalid states to maximize the probability of a valid energy state at the nodes. Thus the clique energy (logic energy) is the summation over the minterms of the valid states and is calculated as :

$$U(x_0, x_1, x_2) = -\sum_i \quad F_i(x_0, x_1, x_2) \tag{2.7}$$

where $F_i = 1$ (i is the index for each row of Table 2.1). This clique energy definition reinforces that the energy of invalid logic state is greater than valid state. As shown in [21], for a valid state, the summation of valid states ($F_i = 1$) is '-1' and for any invalid state, this summation value is '0'. The clique energy for the NAND gate is:

$$U(x_0, x_1, x_2) = -x_2 + 2x_0x_1x_2 - x_0x_1 \tag{2.8}$$

The probability of the different energy configurations of x_2 is:

$$p(x_2) = \frac{1}{Z} \sum_{x_0 \in \{0,1\}} p(x_0) \sum_{x_1 \in \{0,1\}} p(x_1) \; e^{-\frac{1}{KT} U(x_0, x_1, x_2)} \tag{2.9}$$

As according to Equation 2.9, the probability of different energy states at x_2 is calculated as a function of x_2 after marginalizing over the distributions of x_0 and x_1. The probability of the energy state configurations at the outputs of any logic gate can be calculated by the methodology above. Given the input probability distributions and logic compatibility functions, using Belief Propagation algorithm [52] it is also possible to calculate entropy and energy distributions at different nodes of the network. Thus, [2] gives a probabilistic model of computation which we exploit to compute reliability-redundancy trade-offs for different nanoscale architectures.

2.4.1 Modeling Noise at Inputs and Interconnects

The probabilistic non-discrete computing scheme described above can be extended to incorporate the impact caused by continuous noise at the inputs and the interconnects of combinational circuits [2]. Let us take the same NAND gate example to illustrate this. For the inputs being logic low or high, the Gaussian noise distribution below zero will be filtered out because negative values are invalid [21]. To make the Gaussian noise symmetrically distributed about the inputs of the gate, the coordinate system has to be shifted such that $x = 0 \rightarrow x' = -1$, and $x = 1 \rightarrow x' = 1$. Thus, x_0, x_1 and x_2 can

be rewritten as :

$$x'_0 = 2(x_0 - \tfrac{1}{2}),\ x'_1 = 2(x_1 - \tfrac{1}{2}),\ x'_2 = 2(x_2 - \tfrac{1}{2})$$

The clique energy in Equation 2.8 for the NAND gate is modified as follows:

$$U(x'_0,x'_1,x'_2) = -\tfrac{1}{2} - \tfrac{1}{4}x'_2 + \tfrac{1}{4}x'_0x'_1 + \tfrac{1}{4}x'_1x'_2 + \tfrac{1}{4}x'_0x'_1x'_2$$

We have modeled noise as a *Gaussian process* with mean μ and variance σ^2. The probability distribution of x'_2 being in different energy configurations $\in$ {-1.0, -0.9, -0.8,...., 0.1, 0.2, 0.3,...., 1.0} is:

$$p(x'_2) = \frac{1}{Z}\int_{-1}^{1}\sum_{x'_1\in\{-1,1\}} e^{-\frac{U}{KT}}\left(\frac{e^{-(x'_0-\mu)^2/2\sigma^2}}{\sqrt{2\pi}\sigma}\right)dx_0 \cdot p(x'_1) \tag{2.10}$$

The energy distribution at x'_2, if uniform distribution is used to model signal noise is given by:

$$p(x'_2) = \frac{1}{Z}\int_{-1}^{1}\sum_{x'_1\in\{-1,1\}} e^{-\frac{U}{KT}}dx_0 \cdot p(x'_1) \tag{2.11}$$

Equation 2.10 or 2.11 is used to compute the probability of x_2 at different energy states, marginalizing over the distributions of x_0 (ranges between −1 and 1) and x_1 [13]. The energy distribution at the output of any logic gate can be calculated in the presence of

noisy inputs and interconnects by the methodology above.

2.5 Granularity and Measure of Redundancy

Redundancy based defect-tolerance can be implemented for Boolean networks at different levels of granularity. For a specific logic circuit, all the gates could be replicated as a particular CTMR configuration and the overall architecture can be some other CTMR configuration. For example, each gate in the circuit could be a kth order CTMR configuration and the overall logic block could be a nth order configuration where $k \neq n$. Redundancy can thus be applied at different levels of granularity, such as gate level, logic block level, logic function level etc. In Chapter 4, we discuss a few experiments that show us that reliability is often dependent on the granularity level at which redundancy is injected in a logic network. NANOPRISM indicates the correct level of granularity beyond which the reliability measures for a system do not improve.

2.6 Probabilistic Model Checking and PRISM

Probability is the measure of level of uncertainty or randomness. It is widely used to design and analyze software and hardware systems characterized by incomplete information, and unknown and unpredictable outcomes. *Probabilistic model checking* is a range of techniques for calculating the likelihood of the occurrence of certain events during the execution of unreliable or unpredictable systems. The system is usually specified as a state transition system, with probability values attached to the transitions. A probabilistic model checker applies algorithmic techniques to analyze the state space and calculate performance measures.

NANOPRISM consists of libraries built on PRISM [54, 73], a probabilistic model checker developed at the University of Birmingham. PRISM supports analysis of three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs). We use DTMCs to develop libraries for generic defect-tolerant architectural configurations. This model of computation specifies the probability of transitions between states, such that the probabilities of performing a transition from any given state sums up to 1. DTMCs are suitable for conventional digital circuits and the fault models considered. The fault models are manufacturing defects in the gates and transient errors that can occur at any point of time in a Boolean network.

The PRISM description language is a high level language based on guarded commands. The basic components of the language are *modules* and *variables*. A system is constructed as a number of modules which can interact with each other by means of the standard process algebraic operations [74]. A module contains a number of variables which represents its state. Its behavior is given by a set of guarded commands of the form:

$$[] \texttt{<guard>} \rightarrow \texttt{<command>};$$

The guard is a predicate over the variables of the system and the command describes a transition which the module can make if the guard is true (using primed variables to denote the next values of variables). If a transition is probabilistic, then the command is specified as:

$$\texttt{<prob>} : \texttt{<command>} + \cdots + \texttt{<prob>} : \texttt{<command>}$$

The properties of a DTMC model can be verified by PCTL model checking techniques. PCTL [49] is a probabilistic temporal logic, an extension of CTL. It can be used to express

properties such as ”termination will eventually occur with probability at least 0.98”. Our work is based on modeling defect tolerant architectures as state machines with probabilistic assumptions about the occurrence of defects in the devices and interconnections, and then use Markov analysis and probabilistic model checking techniques to evaluate reliability measures.

2.7 Emerging Technologies and Techniques

This section has been included to introduce the readers to some novel techniques that are being looked at by the research community to build devices at the nanoscale. Also, some of the emerging technologies that may use such devices are discussed. Their impact on conventional information processing paradigms are also briefly elucidated in this section. These discussions are not directly related to the scope of this thesis.

2.7.1 Brief Introduction to Emerging Technologies

Table 2.2 compares CMOS and a set of emerging technologies in terms of speed, size, power consumed and cost incurred per device. In Table 2.2, T refers to a single delay, CD refers to critical dimension, Energy is the intrinsic operational energy (Joules/operation), and cost is defined as $ per gate. A number of assumptions are made to estimate these parameters for these non-silicon manufacturing technologies in the absence of firm empirical data. For emerging technologies in the concept stage with no observed data, the underlying physical principles impose constraints on the parameters. If there is availability of some measured data, the assumptions involve a scalability estimation of these techniques. Also, some of these technologies are particularly effective for cer-

tain specific application areas. For instance, the applicability of theoretical quantum computing was shown by P. Shor in 1994 [19]. He devised the first quantum algorithm to find the prime factors of large numbers in polynomial time, that is, considerably faster than any classical algorithm and exploits the parallelism intrinsic in quantum computing. But quantum computing is much less efficient for other applications. In this case, the time required by a classical device to perform an operation using a classical algorithm is defined as "effective" time per operation. The different parameters for quantum computing (shown in Table 2.2) are determined by calculating the "effective" time per operation for different algorithms. A similar approach is used for neuromorphic and optical computing. The importance of such a discussion is to elucidate that few of the new technologies are directly competitive with scaled CMOS and most are highly complementary, and to point out the advantages of heterogeneous integration of the non-silicon manufacturing technologies with silicon CMOS.

Technology	T_{min} sec	T_{max} sec	CD_{min} m	CD_{max} m	Energy	Cost min	Cost max
Si CMOS	3E-11	1E-6	8E-9	5E-6	4E-18	4E-9	3E-3
RSFQ	1E-12	5E-11	3E-7	1E-6	2E-18	1E-3	1E-2
Plastic	1E-4	1E-3	1E-4	1E-3	1E-24	1E-9	1E-6
Optical (digital)	1E-16	1E-12	2E-7	2E-6	1E-12	1E-3	1E-2
NEMS	1E-7	1E-3	1E-8	1E-7	1E-21	1E-8	1E-5
Neuromorphic	1E-13	1E-4	6E-6	6E-6	3E-25	5E-4	1E-2
Quantum Computing	1E-16	1E-15	1E-8	1E-7	1E-21	1E3	1E5

Table 2.2: Estimated Parameters for Emerging Techniques from [18]

Recently, there has been significant advancements in non-silicon manufacturing techniques. We will briefly discuss a few methodologies in this section to illustrate their

impact in future digital systems. [24, 25] discuss a new technique for silicon transistor fabrication on plastic substrates. Such transistors are called thin-film transistor (TFT) devices, wherein the active layer of these devices can be amorphous or polycrystalline-silicon as well as organic semiconductors. These devices can be used in combination with organic light-emitting devices (OLED) [26, 27] for the development of high-resolution, flexible flat-panel displays, that would enable highly portable information devices that are rugged and lightweight. Another important criterion for the maturity of an emerging manufacturing methodology is the cost incurred as compared to existing techniques. Currently, the most common type of flat panel display is the Liquid Crystal Display (LCD) made on glass substrates. A large percentage of the manufacturing cost of such a display comes from the material cost of the glass panels used for the front and back display surfaces, the driver electronics used to address the display, and display breakage. High quality TFTs on plastic substrates could eliminate costs incurred due to the glass and the driver IC's, and thus provide highly flexible and cost-effective displays. However, this novel technique is only in its nascency and considerable research is being done to perfect this manufacturing technology.

Another interesting computational paradigm is optical computing [28]. The concept of optical computing is almost 100 years old, and is based on a theory proposed by Albert Einstein in the early 20th century about the mechanics of light molecules. Information processing is dependent on light transmission and interaction with solids. An optical logic gate is a switch that controls one light beam with another. The device is considered to be "on" when it transmits light, and "off" when it blocks the light. Digital optical computers have certain advantages, and these are due to certain characteristics of light that is used as an information carrier. First, optical information-processing functions can be performed in parallel, and second, optical beams do not cause any interference with each other. Lastly, optical signals can be propagated at the speed of light in a

media.

Nanoelectromechanical systems (NEMS) consist of integrated electromechanical actuators of nanometer-scale dimensions that are driven by electrical energy. These systems have initiated several researchers to look at the possibility of developing fast logic gates, switches, and even computers that are entirely mechanical. For instance, this methodology applied to logic gates entails mechanical digital signals represented by displacements of solid rods, wherein signal propagation speed is limited to the speed of sound [18]. Optimistic empirical estimates predict that the switching speed of NEMS logic gates will be around 0.1ns, and these devices will dissipate less than 10^{-21} Joules. The idea of electromechanical actuators and mechanical computers dates back to the 1820s [72], when Charles Babbage designed the first mechanical computer, viewed as the forerunner to the modern computer. In the 1960s, electronic logic gates and integrated circuits vastly outperformed moving elements due to which the idea of mechanical computers was dropped. But with the rapid developments in nanotechnology, it may be possible to manufacture complex molecular-scale mechanical elements that will move on timescales of a nanosecond or less.

On the other hand, quantum computing exploits physical phenomenon unique to quantum mechanics to realize a fundamentally new mode of information processing. At the quantum level, the values of certain observable quantities are restricted to a discrete finite set (Quantization). This is to ensure that each classical bit can be stored as a stable state of the system. A binary bit is a fundamental unit of information, represented as a 0 or 1 in digital systems. The fundamental unit of information in a quantum computer is called a quantum bit or qubit [68]. A qubit can be a 1 or a 0, or it can exist in a superposition that is simultaneously both 1 and 0 or somewhere in between. This implies that quantum computers are not limited to only two states as compared to classical digital systems. The massive parallelism intrinsic in quantum

computing is due to such superposition of different states [66, 67]. This allows a quantum computer to work much faster than conventional digital systems. Quantum computers also utilize another aspect of quantum mechanics known as entanglement. Two spatially separated and non-interacting quantum systems are said to be entangled if they have some information (due to previous interactions) that can only be accessed by observing the states of the systems together, and not by performing experiments on either of them alone. Quantum cryptography uses this characteristic of quantum mechanics [19].

At the extreme end of the spectrum of these developing technologies are neuromorphic systems, which are silicon implementations of sensory and neural systems. The architectural configurations and designs of such systems are inspired by neurobiology [69]. This is because the human brain is a classical neuromorphic information processing system, and can be considered as a motivation for future technological advancements. The different parameters (shown in Table 2.2) for the human brain are approximate estimations. For example, the critical dimension of each neuron is computed by estimating the volume of the brain and the number of neurons. Similarly, the single delay T_{max} is the experimentally observed time for opening and closing of synapses. Also, each neuron is connected to 100 to 10,000 synapses, and this high interconnect density differentiates the human brain from any silicon-based system [18]. This developing technological area offers exciting possibilities, such as sensory systems that can compete with human senses and pattern recognition systems that can run in real time.

2.7.2 Techniques to Build Nano-Scale Devices

The current techniques to manufacture silicon-based devices such as casting, grinding, milling and even lithography provide only approximate control over the molecular

structures of the devices being manufactured, and form structures that are composed of billions of atoms. The statistical nature of these large groups of atoms guarantee the correct behavior of the device being built. Thus, these methodologies employ the *Law of Large Numbers* (LLN) [45] at the device scale to provide higher level of reliability. LLN is said to hold when the mean of a sequence of random variables with a common distribution converges to their common expectation as the size of the sequence goes to infinity.

In accordance to Moore's law [70], the device density on a chip is increasing, necessitating the scaling down of device size. The current techniques used in mass-production of silicon devices provide phenomenal reliability in both manufacture and operation of these devices, however, putting a limit on the ability to scale down their size. Thus, such techniques are not viable for emerging computing systems that incorporate nano-scale devices. André DeHon proposes the application of LLN at levels of abstraction higher than device level [42], and suggests that a large number of such nanodevices can be used at the architectural or micro-architectural level, for instance, to design redundancy based defect-tolerant architectures. This may circumvent the reliability issues associated with devices built with small number of atoms, and assembled as well-defined molecular structures. Techniques to build devices at the nano-scale are still in their infancy, but it is worth noting that recently there has been major advancements in some of these areas. In this subsection, we discuss some these methodologies in brief.

Self-assembly is a spontaneous process by which atoms and molecules form organized aggregates or networks, typically by interacting with a solution or gas phase [43]. This methodology involves a process known as convergent synthesis [79], that allows assembling atomically precise devices. Structures formed by this process are covalently bonded, well-defined and stable. The components of such self-assembled structures

find their appropriate locations based on their structural properties (or chemical properties for atomic or molecular self-assembly). This methodology is not limited only to the nanoscale, and is considered a very generic and powerful bottom-up device manufacturing technique. Self-assembly is still in its nascent stages, but researchers have succeeded in building some primitive nanodevices with this technique. This has created a lot of interest in the industry as well as the academia.

Positional assembly [44] is not the same as self-assembly. The main difference between the two processes is as follows: positional assembly provides a higher degree of control on positional placements of atoms and molecules to form arbitrary stable structures, however, self-assembly does not do so. On the other hand self-assembly is largely a natural process, that only needs an external initiation. Positional assembly requires a designer to observe and analyze the device manufacturing process (perform nanoanalysis), and this requires the usage of precision observation equipments and the ability to manipulate atoms. Researchers are trying to conglomerate these two processes to form novel and cost-effective manufacturing techniques for nano-scale devices.

Lithography is the process of imprinting patterns on semiconductor materials that are used as integrated circuits. In the past, electron-beam (e-beam) lithography [80] has been used to fabricate nanoscale molecular devices. This process is impractical for commercial applications because of the sequential nature of the methodology that slows down writing speed. High-energy electron beams can also damage the active molecules in a circuit. In contrast, a new technique called imprint lithography [17] has been introduced, and is well suited to transfer patterns from micrometers to the nanometers range. This methodology entails the placement of a thin polymer film on top of the substrate, and a pre-patterned mold is brought into soft contact with the film. The mold pattern is transferred to the polymer by applying pressure and increasing the temperature. The residual polymer is removed from the feature bottom and further

processing may be done (e.g. metal deposition, etching). *Nanoimprint* Lithography has high throughput due to parallel processing, and is cost-efficient since no sophisticated tools are required.

Chapter 3

NANOLAB: A MATLAB Based Tool

In this chapter, we discuss how information theoretic entropy coupled with thermal entropy can be used as a metric for reliability evaluation [7]. We also present the automation methodology for the NANOLAB tool with a detailed example and code snippets.

3.1 Reliability, Entropy and Model of Computation

[2] not only provides a different non-discrete model of computation, in fact, it relates information theoretic entropy and thermal entropy of computation in a way so as to connect reliability to entropy. It has been shown that the thermodynamic limit of computation is $KT \ln 2$ [3]. What the thermodynamic limit of computation means is that the minimum entropy loss due to irreversible computation amounts to thermal energy that is proportional to this value. If we consider energy levels close to these thermal limits, the reliability of computation is likely to be affected, and if we can keep

our systems far from the temperature values that might bring the systems close to this amount of entropy loss, the reliability is likely to improve. The model of computation in [2] considers thermal perturbations, discrete errors and continuous signal noise as sources of errors. The idea is to use a Gibbs distribution based technique to characterize the logic computations by Boolean gates and represent logic networks as MRFs, and maximize probability of being in valid energy configurations at the outputs.

3.2 Automation Methodology

NANOLAB consists of a library of functions implemented in MATLAB [1]. The library consists of functions based on the probabilistic non discrete model of computation discussed earlier (details in [2]), and can handle discrete energy distributions at the inputs and interconnects of any specified architectural configuration. We have also developed libraries that can compute energy distribution at the outputs given continuous distributions at the inputs and interconnects, introducing the notion of signal noise. Therefore, this tool supports the modeling of both discrete and continuous energy distributions.

The library functions work for any generic one, two and three input logic gates and can be extended to handle n-input logic gates. The inputs of these gates are assumed to be independent of each other. These functions are also parameterized and take in as inputs the logic compatibility function (Table 2.1) and the initial energy distribution for the inputs of a gate. If the input distribution is discrete, the energy distribution at the output of a gate is computed according to Equation 2.9, by marginalizing over the set of possible values of the nodes that belong to the same clique. In the case of generic gates these nodes are their inputs. These probabilities are returned as vectors by these functions and indicate the probability of the output node being at different energy

levels between 0 and 1. These probabilities are also calculated over different values of KT so as to analyze thermal effects on the node. The Belief Propagation algorithm [52] is used to propagate these probability values to the next node of the network to perform the next marginalization process. The tool can also calculate entropy values at different nodes of the logic network. It also verifies that for each logical component of a Boolean network, the valid states have an energy level less than the invalid states as shown theoretically in [2].

Our tool also consists of a library of functions that can model noise either as an uniform or gaussian distributions or combinations of these, depending on the user specifications. The probability of the energy levels at the output of a gate is calculated by the similar marginalizing technique but using Equation 2.10 or 2.11 or both depending on the characterization of signal noise. Similar to the library functions discussed previously, the signal library returns output energy distributions as vectors but the energy levels are between -1 and 1 due to rescaling of the logic level for reasons discussed in Chapter 2. Entropy values at the primary outputs of Boolean networks are also computed for different thermal energy levels. Arbitrary Boolean networks in any redundancy based defect-tolerant architectural configuration can be analyzed by writing simple MATLAB scripts that use the NANOLAB library functions. Also, generic fault tolerant architectures like TMR, CTMR are being converted into library functions such that these can be utilized in larger Boolean networks where more than one kind of defect-tolerant scheme may be used for higher reliability of computation.

3.2.1 Detailed Example

We now discuss a detailed example to indicate the power of our methodology. Figure 2.2 shows a CTMR configuration with three TMR blocks working in parallel with a

majority gate logic. The code listing shown in Figure 3.1 is a MATLAB script that uses NANOLAB functions and Belief Propagation algorithm to evaluate the probability of the energy configurations at the output of the CTMR.

The probability distributions for x_1, y_1, x_2, y_2, x_3 and y_3 for the NAND gates in Figure 2.2 are specified as vectors. These vectors specify the probability of the inputs being a logic low or high (discrete). The input probability distributions for all the TMR blocks are the same in this case but these can be varied by having separate input vectors for each TMR block.

z=0.0	z=0.2	z=0.5	z=0.8	z=1.0
0.809	0.109	0.006	0.025	0.190
0.798	0.365	0.132	0.116	0.201
0.736	0.512	0.324	0.256	0.263
0.643	0.547	0.443	0.379	0.356

Table 3.1: Probability of the output z of a logic gate being at different energy levels for values of $KT \in \{0.1, 0.25, 0.5, 1.0\}$

The NANOLAB functions return vectors similar to the one shown in Table 3.1. These indicate the probability of the output of a logic network being at specified energy levels for different *KT* values. In the CTMR configuration, *for each TMR block*, the energy configurations at the outputs of each of the three NAND gates are obtained from the function for a two input gate. Then these probabilities are used as discrete input probability distributions to the function for a three input gate. This computes the energy distribution at the output of the majority logic gate. Similarly, the probabilities of the

```
1
2   no_of_blocks = 3;% number of TMR blocks
3   prob_input1 = [0.1 0.9];% prob distbn of input1 of NAND gate
4   prob_input2 = [0.1 0.9];% prob distbn of input2 of NAND gate
5   BT_Values = [0.10.25 0.5 1.0];% different kbT values
6
7   for TMR_block = 1:no_of_blocks
8       counter = 1;
9       % energy_2_input_gates_function is a NANOLAB function and takes in as parameters
10      % the logic compatibility function, input prob distbns and the kbT values.
11      % The output gives the state configurations of the primary output of the logic
12      % gate at different kbT values.
13
14      prob1 = energy_2_input_gates_function(input1,prob_input1,prob_input2,BT_Values);
15      prob2 = energy_2_input_gates_function(input1,prob_input1,prob_input2,BT_Values);
16      prob3 = energy_2_input_gates_function(input1,prob_input1,prob_input2,BT_Values);
17      [a,b] = size(prob1);
18
19      % req_pb1, req_pb2 , req_pb3 are vectors which contain probabilities
20      % of the output being a 0 or a 1 for a particular kbT value for Belief
21      % Propagation.
22      for i = 1:a
23          req_pb1 = [prob1(i,1) prob1(i,b)];
24          req_pb2 = [prob2(i,1) prob2(i,b)];
25          req_pb3 = [prob3(i,1) prob3(i,b)];
26          BT = BT_Values(i);
27
28          % energy_3_input_gates_function is a part of NANOLAB and takes in input
29          % and output parameters similar to the previous 2 input gates function.
30          t_p = energy_3_input_gates_function(input2,req_pb1,req_pb2,req_pb3,BT_Values(i));
31          prob(TMR_block,counter) = t_p(1,1);
32          counter = counter + 1;
33          prob(TMR_block,counter) = t_p(1,b);
34          counter = counter + 1;
35      end
36  end
```

Figure 3.1: Script for 1st order CTMR with discrete input distribution

final output of the CTMR is calculated. It can be seen that our methodology provides a very convenient way of evaluating this CTMR configuration and only requires minor modifications to be extended to any i-th order CTMR. Additionally, it is desired that valid input/output states should have lower clique energies than invalid states [2]. We have checked for the conformance to this property for the different clique energy functions which are generated by NANOLAB.

Similarly, MATLAB scripts can be written to model signal noise, and evaluate energy distribution and entropy at the output of the CTMR shown in Figure 2.2. The NANOLAB library functions can be used to interject unform or gaussian noise or combinations of these. Our tool provides a wide range of modeling options to system architects, and expedites reliability analysis of defect tolerant architectural configurations for combinational logic circuits.

3.3 Shortcomings of NANOLAB

NANOLAB libraries can be used to inject random noise at the inputs and interconnects of logic circuits. Signal noise can be modeled as discrete and continuous distributions. We have experimented with different such combinations at the primary inputs of a Boolean network as well as interjected such noise distribution at the interconnects of a logic circuit. While trying to specify two noise spikes at the inputs of a NAND gate as independent gaussian distributions, the expression for the energy distribution at the output of the NAND gate turns out to be as follows [13]:

$$p(output) = \frac{1}{Z}\int_{-1}^{1}\int_{-1}^{1} e^{-\frac{U}{KT}}\left(\frac{e^{-(x_0-\mu_0)^2/2\sigma_0^2}}{\sqrt{2\pi}\sigma_0}\right)\left(\frac{e^{-(x_1-\mu_1)^2/2\sigma_1^2}}{\sqrt{2\pi}\sigma_1}\right)dx_0dx_1 \tag{3.1}$$

The two gaussian processes have mean μ_0 and μ_1 which may or may not be equal, and variance σ_0 and σ_1. Equation 3.1 does not evaluate to an explicit integral and probability values for the output being at different energy levels cannot be computed. To tackle this scenario, we are developing a procedure in MATLAB to approximate the multiplication of two gaussian distributions. Note that specifying noise as a bivariate gaussian leads to an explicit integral as the probability density function of a bivariate gaussian leads to a less complex integrand. Also, the accuracy of the probabilities computed by our libraries and the Belief Propagation algorithm [52] are limited by the floating point accuracy of MATLAB.

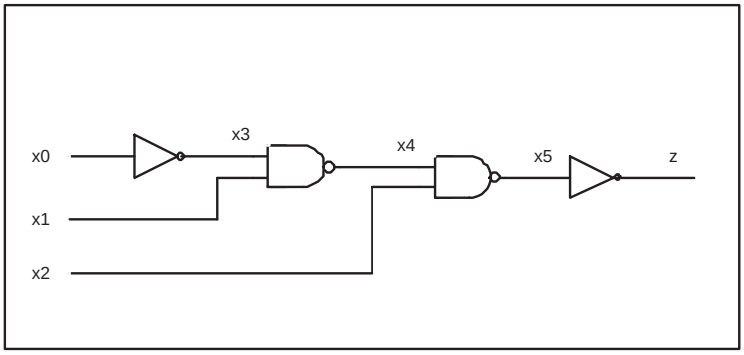


Figure 3.2: A Boolean network having the logic function z = $x_2 \wedge (x_0 \vee x_1\prime)$

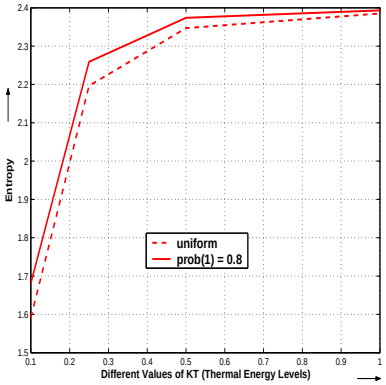
3.4 Reliability Analysis of Boolean Networks using NANOLAB

We analyze different defect-tolerant architectural configurations of arbitrary Boolean networks starting from single gates to logic blocks with NANOLAB. The entropy values and logic margins are observed and these determine interesting facts [5]. The next few subsections discuss in details the different experimental results. The combinational circuit we refer to is shown in Figure 3.2.

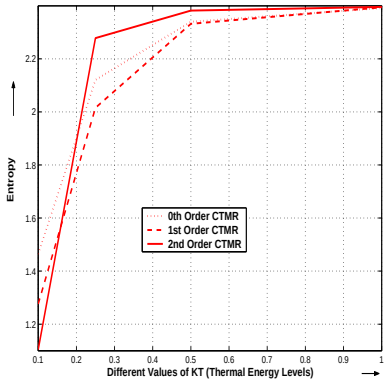
3.4.1 Reliability and Entropy Measures of a NAND Gate

Figure 3.3 shows the entropy curves at the outputs of a single NAND gate and different orders of a NAND CTMR configuration at different KT values. The output probability distribution of a NAND gate is asymmetrical. This should be expected since only one input combination produces a logic low at the output. Figure 3.3 (a) indicates that the entropy is lower when the inputs of the single NAND gate are uniformly distributed. At higher KT values, in both the uniform and non-uniform probability distribution cases, the entropy increases, resulting in the logic margins (probability of being in valid energy configurations) being reduced. This indicates that the logic margins in both scenarios of distribution almost approach zero as thermal energy levels increase. When thermal energy becomes equal to logic energy ($KT = 1$), the entropy values in the case of the uniform distribution is lower implying that the logic margins are better than when the inputs have a higher probability of being logic high.

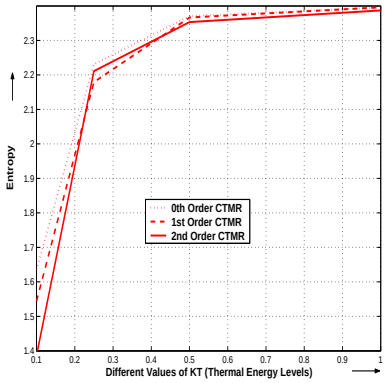
Figure 3.3 (b) shows the entropy curves for different orders of a NAND CTMR configuration at different KT values. The inputs in this case are uniformly distributed. It can



(a) Entropy of a NAND gate for different input distributions



(b) Entropy of a CTMR NAND configuration when inputs are uniformly distributed



(c) Entropy of a CTMR NAND configuration when inputs are logic high with 0.8 probability

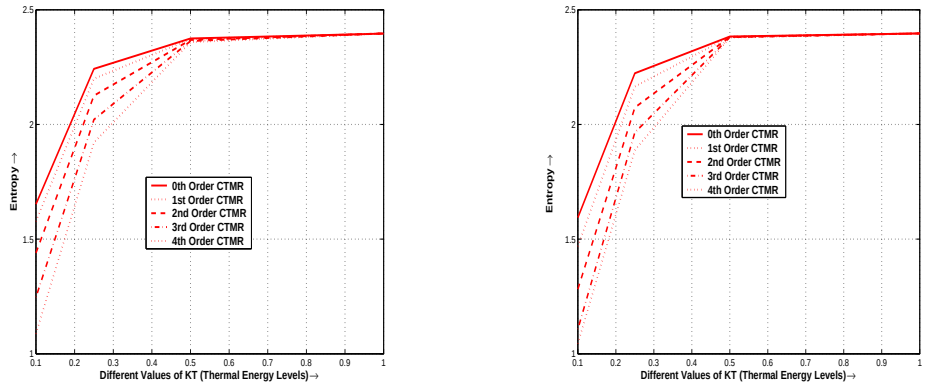
Figure 3.3: Entropies for the output of a Single NAND gate and CTMR configurations of the NAND gate at different *KT* values

be observed that the 0th order CTMR (TMR) has higher entropy value i.e. less logic margin than the 1st order CTMR at lower thermal energy levels and in both cases the entropy values almost converge at $KT = 0.5$. At this point (at close proximities of the thermodynamic limit of computation) logic margins at the output of the CTMR configurations become insignificant and there is total disorder, and computation becomes unpredictable. The interesting plot is for the 2nd order CTMR. The entropy shoots up at a KT value of 0.25 indicating that the 2nd order CTMR degrades the reliability of computation. This aptly shows that every architectural configuration has reliability/redundancy trade-off points and even if redundancy levels are increased beyond these, the reliability for the architecture either remains the same or worsens.

Figure 3.3 (c) indicates the entropy values when the inputs are non-uniformly distributed with a probability of 0.8 to be logic high. In this case, it is observed that the 2nd order CTMR degrades reliability of the system, but does better than the 0th order CTMR configuration. This result demonstrates that varying energy distributions at the inputs of a Boolean network may result in different reliability-redundancy trade-off points. Also, the entropy values are higher than Figure 3.3(b) at lower KT values. This is because the inputs being at a higher probability of being a logic high implies that the output of the NAND gate has a higher probability of being non-stimulated, and only one input combination can cause this to happen i.e. when both inputs are high. Thus, the logic margins are a bit reduced due to convergence towards a valid output energy state, and the entropy has higher values than the previous experiment.

3.4.2 Reliability and Entropy Measures for the Logic Block

Figure 3.4 shows the entropy curves at the outputs of the CTMR configurations for the logic block. Figure 3.4 (a) shows the entropy when the input distribution is uniform.



(a) Inputs are uniformly distributed

(b) Inputs are logic high with 0.8 probability

Figure 3.4: Entropy for different orders of CTMR configurations for the Boolean network given in Figure 3.2 at different KT values

It can be observed that as the CTMR orders are increased, the entropy decreases (logic margin increases) at lower KT values, and the entropy converges at $KT = 0.5$ for all the CTMR orders. This indicates that as the system approaches the thermal limit of computation ($KT \ln 2$) [3], increase in redundancy level does not improve the logic margins. The reliability measures remain the same. But at lower thermal energy levels, increasing the redundancy level (orders of the CTMR) results in improvement of reliability. This is because the 4th order CTMR has entropy values less than the other lower orders at thermal energy levels below 0.5. Whereas, in the NAND CTMR configuration, the 2nd order CTMR has higher entropy at lower KT values indicating degradation in reliability of computation. This experiment illustrates that each specific Boolean network has an *unique* reliability-redundancy trade-off point. It may also be inferred that the redundancy level is proportional to the device density of a logic network.

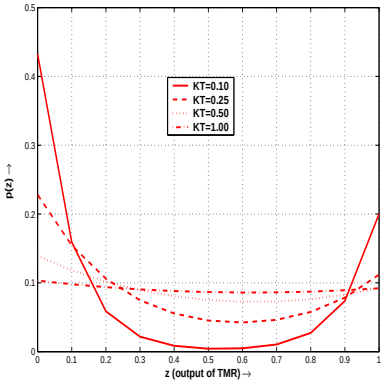
Figure 3.4 (b) indicates the entropy values when the inputs have a non-uniform prob-

ability distribution i.e the probability of being logic high is 0.8. The same facts are observed in this case as in Figure 3.4 (a). But, the entropy is higher than the previous experiment at lower thermal energy levels. This is because the inputs being at a higher probability of being logic high means that the output of the logic block will be stimulated and only a few input combinations can cause this to happen. Thus, the logic margins are slightly reduced as compared to when the inputs are uniformly distributed. Also, the entropy curves for the 3rd and 4th order CTMR configurations are in very close proximity to each other. This indicates that the degree of reliability improvement depletes as more redundancy is augmented to the architecture.

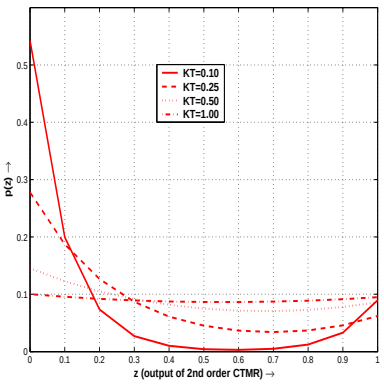
We conduct this experiment to explore the flexibility and robustness of our tool in evaluating any arbitrary Boolean network. Note that the Boolean network shown in Figure 3.2 has been used only for illustration purposes and reliability/redunancy analysis of larger and more complex combinational circuits have been performed.

3.4.3 Output Energy Distributions for the Logic Block

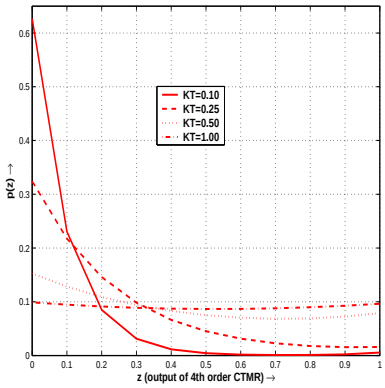
NANOLAB has another perspective to it: the probability of being in different energy configurations at the primary outputs of a Boolean network can also be computed. Reliability measures of logic circuits can also be analyzed from these probability distributions. Figure 3.5 shows the energy distributions at the outputs of the different CTMR orders applied to the logic circuit when the inputs have uniform energy distribution. Note that the probability values are based on bin sizes of 0.1. As the order of the CTMR is increased, it can be seen that the logic margins for the output (z) at KT values of 0.1, 0.25 and 0.5 keep on increasing. Due to the asymmetrical nature of the logic network, the probability of z ($p(z)$) being at energy level zero is almost always higher than being at one and hence such plots are obtained. It is also observed that at



(a) The output distributions of a TMR configuration



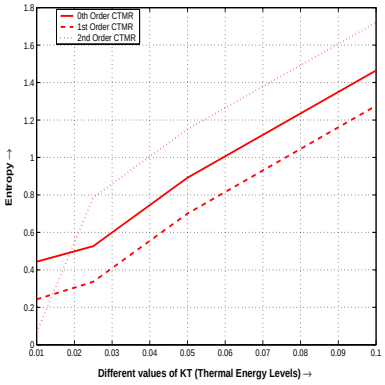
(b) The output distributions of a 2nd Order CTMR configuration



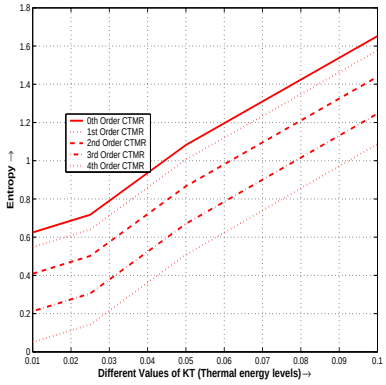
(c) The output distributions of a 4th Order CTMR configuration

Figure 3.5: Energy distributions at the output of the Boolean network for different orders of CTMR configuration at different *KT* values. Inputs are uniformly distributed

a *KT* value of one, the logic margin for any order of the CTMR configuration becomes considerably small (output energy distribution becomes almost uniform), and remains the same even with an increase of redundancy resulting in unreliable computation. Comparing the different orders of CTMR in Figure 3.5, we infer that for lower thermal energy levels, the probability of being in a valid energy configuration increases as more redundancy is added to the architecture. But this increase in probability slows down as higher orders of CTMR are reached. This can be understood as follows: the logic margin of the system reaches a saturation point after which reliability can no longer be improved. The experimental results for single NAND gate CTMR configurations show similar plots with the difference that the output of the 2nd order CTMR configuration is non-stimulated (valid state) with a higher probability at $KT = 0.1$. This can be attributed to the intuitive fact that if a unit being duplicated has a lesser number of devices, a stable logic margin is reached with lower redundancy levels than a unit which has higher number of components. Such observations give a clear notion of the optimal redundancy points for specific logic networks.



(a) CTMR orders of single NAND gate



(b) CTMR orders of logic circuit shown in Figure 3.2

Figure 3.6: Entropy for different logic networks at *KT* values $\in [0.01, 0.1]$

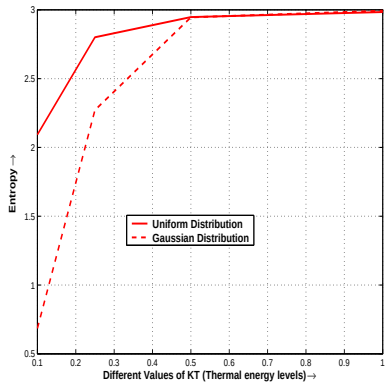
3.4.4 Reliability Analysis of Different Circuits for Smaller KT Values

Reliability analysis of real systems at thermal energy levels below 0.1 are of practical importance. Figure 3.6 presents computational entropy values for a single NAND gate and the Boolean network. These results are similar to the ones indicated by Figures 3.3 and 3.4. It is interesting to note that in this case, there is a linear increase in the entropy plots for both the circuits with increase in KT values. The previous experimental results had a convergence of the entropy values at around $KT = 0.5$ and remained steady for all thermal energy levels beyond that point. Similar reliability/redundancy trade-off points are observed in both the logic networks. The 2nd order CTMR configuration for the NAND gate does worse than the lower orders, whereas increasing the redundancy level for the logic block yields better reliability.

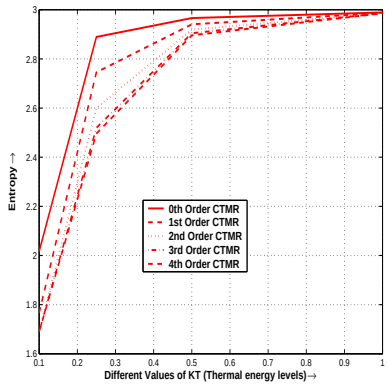
3.4.5 Reliability Analysis in the Presence of Noisy Inputs and Interconnects

We have also analyzed reliability of different logic networks in the presence of random noise at the inputs and interconnects. Entropy and energy distributions at the outputs are computed automatically and reliability/redundancy trade-off points are inferred. Logic margins in the presence of signal noise is the difference between the probabilities of occurrence of -1 (logic low) and 1 (logic high) due to rescaling the energy levels as discussed in Chapter 2. The Boolean network we consider here is the one given in Figure 3.2.

Reliability and Entropy Measures of a NAND Gate: Figure 3.7 shows the entropy values at the outputs of different NAND gate configurations in the presence of noisy



(a) Entropy at the output of a NAND gate



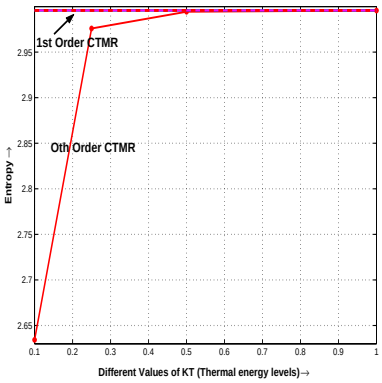
(b) Entropy for different CTMR orders of NAND gate

Figure 3.7: Entropy at the outputs of different NAND gate configurations in the presence of noisy inputs and interconnects

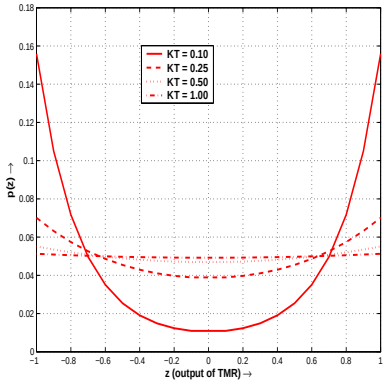
inputs and interconnects. The plots in Figure 3.7 (a) are obtained when (i) one of the inputs of the gate has equal probability of being at logic low or high and the other is subjected to a gaussian noise spike centered around 1 with a variance of 2 and (ii) when the gaussian noise signal is changed to an uniformly distributed one ranging from $\{-1, 1\}$. The plots show that the entropy has lower values in the presence of gaussian noise as compared to uniformly distributed noise at lower KT values. This is because a gaussian noise spike centered around logic high alleviates the probability of the output being in a valid energy state and decreases the degree of randomness the system is in (entropy). Normally gaussian noise spikes have mean at invalid energy levels, and thus prevents the system from converging to a valid logic state. Such noise signals have also been modeled for complex logic circuits.

The entropy curves for different CTMR configurations of a NAND gate are indicated in Figure 3.7 (b). The entropy values for the NAND gate are plotted till the 4th order CTMR for different thermal energy levels. The inputs to the logic block and the in-

terconnects are subjected to both uniformly distributed and gaussian noise. We have assumed two gaussian noise spikes, one centered around 1 with variance of 2 and the other centered around 0.5 with variance of 0.2. It can be observed that as redundancy is increased by adding more CTMR orders, the entropy decreases (logic margin and hence reliability increases) at lower *KT* values. However, the rate of improvement in reliability decreases as 3rd order CTMR is reached. This result emphasizes that beyond a certain redundancy level, the system's reliability for a given defect-tolerant configuration reaches a steady state and cannot be improved substantially.



(a) Entropy for different CTMR orders of the logic block



(b) Output distribution for the TMR of the Boolean network

Figure 3.8: Entropy and Energy distribution at the outputs of different CTMR configurations of the logic block in the presence of noisy inputs and interconnects

Entropy and Energy Distribution for the Boolean Network: The entropy and energy distribution at the output of different CTMR configurations of the logic block are presented in Figure 3.8. The entropy values for the different CTMR orders of the Boolean network are given in Figure 3.8 (a). The experimental setup is as follows: the inputs and interconnects are subjected to both uniformly distributed and gaussian noise. Two gaussian noise signals are considered, with equal means at 0.5 energy

level and variances 2 and 0.2 respectively. It can be inferred from the results that TMR (0th order CTMR) has low entropy values at KT values less than 0.5. The higher CTMR orders have very high entropy irrespective of the thermal energy. We have only indicated entropy for the 1st order CTMR because higher orders do not improve the reliability. Considering this result, we can state that due to a very noisy and error-prone environment, the augmentation of any level of redundancy beyond the TMR configuration causes a steep degradation of reliability. Comparing these results with Figure 3.7(b), we surmise that reliability/redundancy trade-off points are not only dependent on specific logic networks, these vary with the normal operation scenarios (such as random noise) of the circuits.

The energy distribution at the output of the TMR configuration for the circuit at different thermal energy levels is presented in Figure 3.8 (b). As discussed earlier, these probability values can also indicate reliability measures. Noise characterizations at the inputs and interconnects are similar to the previous experiment. Due to the asymmetrical nature of the logic network, the probability of z being at energy level zero is almost always higher than being at one when the inputs have equal probability of being logic low or high. But due to the uncertainty interjected in the form of noise, the logic margins at the output of the logic block becomes very sensitive to thermal fluctuations. At $KT = 0.25$, z has high probabilities of being in any of the valid logic states, and as the thermal energy increases, the probability of occurrence of the invalid states is almost equal to that of valid states.

Chapter 4

NANOPRISM: A Tool Based on Probabilistic Model Checking

The advent of nanoscale devices necessitates architectures to be defect-tolerant through redundancy. Redundancy may be applied at different levels of granularity, such as gate level, logic block level, logic function level, unit level etc. The trade-offs between reliability-redundancy and the level of granularity and reliability must be evaluated rapidly for computer architects to find suitable design pareto points. This chapter explains how we use a probabilistic model checking framework, in particular PRISM [73] to model defect-tolerant architectures such as TMR, CTMR and multi-stage iterations of these. These probabilistic models that represent different defect-tolerant architectural configurations are integrated to form a library that composes NANOPRISM [8, 12]. We also report interesting results that indicate the effectiveness of our tool.

4.1 Modeling Single Gate TMR, CTMR

In this section, we explain the PRISM model construction of a single gate TMR, CTMR and multistage iterations of these. The first approach is directly modeling the systems as given in Figures 2.1 and 2.2. For each redundant unit and the majority voting logic, we construct separate PRISM modules and combine these modules through synchronous parallel composition. However, such an approach leads to the well know state space explosion problem. At the same time, we observed that the actual values of the inputs and outputs of each logic block is not important, instead one needs to keep track of only the total number of stimulated (and non-stimulated) inputs and outputs. Furthermore, to compute these values, without having to store all the outputs of the units, we replace the set of replicated units working in *parallel* with ones working in *sequence*. This folds space into time, or in other words reuse the same logic unit over time rather than making redundancy over space. This approach does not influence the performance of the system since each unit works independently and the probability of each gate failing is also independent.

The different orders of CTMR configurations are built incrementally from the models of the previous iterations. In this case too, two approaches seem to emerge. One of the approaches is incorporating PRISM modules of the previous CTMR iteration as the redundant functional units for the current order and adding a majority voting logic. This causes the model to grow exponentially as the higher orders of configuration are reached. The other approach is to use already calculated probability values (probability of being a logic low or high) at the output of the last CTMR configuration as the output probability distributions of the three redundant functional units of the current order of a CTMR configuration.

The DTMC model of the TMR configuration of a NAND gate is shown in Figure 4.1

```
1
2       prob p_err = 0.1; //probability that gate has error
3       prob p_in=0.9; //probability an input is logic high
4
5       const R=3;  //number of redundant processing units const
6       const R_limit=1;
7
8       module TMR
9
10          x : bool;
11          y : bool;
12          s : [0..3] init 0; // local state
13          z : [0..R] init 0; // number of outputs that are stimulated
14          z_output : [0..1] init 0; // output of majority logic
15          c : [0..4] init 0;    // count of the redundant unit being processed
16
17          [] s=0 & c>R -> (s'=0);
18
19          // processed all redundant units
20          [] s=0 & c=R -> (s'=3)&(c'=c+1);
21
22          // initial choice of x and y
23          [] s=0 & c<R -> p_in : (x'=1)&(s'=1)&(c=c+1) + (1-p_in) : (x'=0)&(s'=1)&(c=c+1);
24          [] s=1 -> p_in : (y'=1)&(s'=2) + (1-p_in) : (y'=0)&(s'=2);
25
26          // NAND operation
27          [] s=2 -> p_err : (z'=z+(x&y))&(s'=0) + (1-p_err) : (z'=z+(!(x&y)))&(s'=0);
28
29          // majority logic
30          [] s=3 & z>=0 & z<=R_limit -> (s'=0) & (z_output'=0);
31          [] s=3 & z>R_limit & z<=R -> (s'=0) & (z_output'=1);
32
33      endmodule
```

Figure 4.1: PRISM description of the TMR configuration of a single NAND gate

for illustration purposes. We assume in this case that the inputs X and Y have identical probability distribution (probability of being logic high is 0.9), and the failure (inverted output) probability of NAND gates is 0.1. However, the input probability distributions and failure distribution of the NAND gates can be changed easily by modification of the constants given at the start of the description. The probabilistic state machine for this DTMC model built by PRISM has 115 states and 182 transitions. Also, model checking is performed to compute the probability distribution of the TMR configuration's output being in an invalid state for different gate failure probabilities. Furthermore, since PRISM can also represent non-deterministic behavior, one can set upper and lower bounds on the probability of gate failure and then obtain (best and worst case) reliability characteristics for the system under these bounds. As discussed before, the CTMR configuration uses three TMR logic units and majority voter. The probability distribution obtained for the TMR block of a single NAND gate can be used directly in the CTMR configuration, thus reducing the state space.

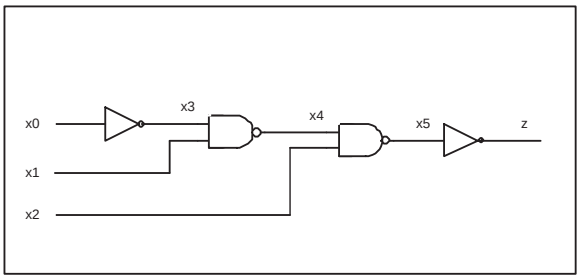


Figure 4.2: A Boolean network having the logic function z = $x_2 \wedge (x_0 \vee x_1\prime)$

4.1.1 Modeling Block Level TMR, CTMR

We explain the PRISM model construction for different orders of block level CTMR configurations. Each unit as shown in Figure 2.1 is no longer composed of a single

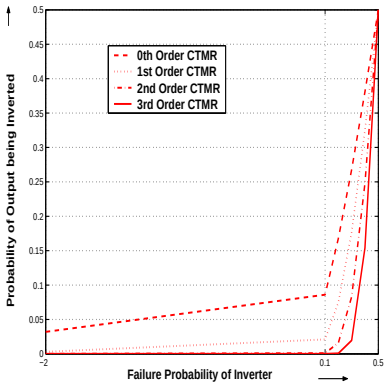
gate, but contains a logic circuit such as the one shown in Figure 4.2. As discussed earlier, we replace the set of replicated units working in *parallel* with ones working in *sequence*, thus folding space into time. The DTMC model for the TMR configuration of the logic block in Figure 4.2 has 1407 states with this approach of modeling. The models for different orders of CTMR configurations are built using similar techniques that are used for single gate CTMR.

We have also incorporated different levels of granularity in these redundancy based architectural configurations. Let us walk through an example to explain this concept clearly. For the logic circuit under discussion, we model the different levels of CTMR wherein each redundant unit contains the logic block itself. We consider this redundancy at the logic block granular level. Next , for each redundant logic block, we further replicate each gate so as to form different orders of CTMR configurations at the gate level of granularity. Our experimental results indicate the intuitive fact that CTMR configurations at different levels of granularity and lower gate failure probabilities give better reliability than the conventional flat architectural configuration.

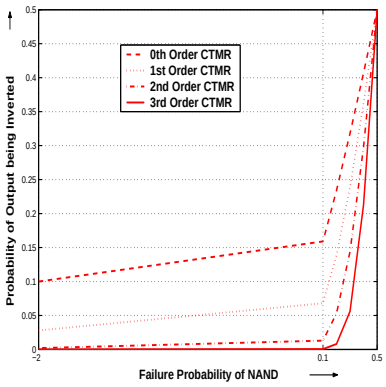
4.2 Reliability Analysis of Logic Circuits with NANOPRISM

NANOPRISM has been applied for reliability analysis of different combinational circuits starting from single gates to logic blocks. The output error distributions for different granularity levels and failure distributions of the gates are observed, and these demonstrate certain anomalous and counter-intuitive facts. The next subsections discuss in details the different experimental results [8]. Figure 4.2 shows the Boolean network used for the illustration of the effectiveness of NANOPRISM in evaluating

arbitrary logic circuits.



(a) Inverter CTMR Configurations



(b) NAND CTMR Configurations

Figure 4.3: Error Distribution at the Outputs for the different Orders of CTMR Configurations of Single Gates

4.2.1 Reliability Measures of Single Gate CTMR Configurations

Figure 4.3 shows the error distribution at the output of CTMR configurations for the Inverter and NAND gates. The input distribution in both the cases is assumed to be logic high with 0.9 probability. The probability of the gates being in error $\in \{10^{-2}, 0.1, 0.2, ..., 0.5\}$. Figure 4.3 (a) shows the probability of the output being in error (inverted with respect to the expected output) for the inverter CTMR configurations upto order three. It is observed that as the order increases, the probability of having the output in error goes down. But at lower gate error probabilities, it is clearly seen that the 2nd order and 3rd order configurations provide the same reliability measures. Thus increasing redundancy beyond this point does not make sense. At higher gate failure rates, there is a difference between the reliability obtained by the 2nd and 3rd order CTMR architectures. We extended the number of orders further

and observed that at the 5th CTMR order when gate failure distributions are high, the difference in reliability measures obtained from this iteration and the previous one reduces but never becomes equivalent. This is intuitive because at higher device failure rates, introducing more redundancy causes more error prone computation and at same point the reliability can not be improved anymore and may sometimes degrade as we will observe shortly. Also, for each CTMR configuration, at lower gate failure distributions a plateau is reached i.e. lowering the device failure rate does not lower the probability of an erroneous output. This means that a redundancy based architecture cannot provide a reliability level lower than a certain "point", and NANOPRISM can analyze such reliability measures accurately.

Figure 4.3 (b) shows the error distribution at the output of the different CTMR configurations of a NAND gate. This graph indicates certain observations that are already seen in the previous plot. In this case, the 1st order configuration provides a major improvement in reliability as compared to the 0th order CTMR. This observation is of interest and can be interpreted as follows: due to the asymmetrical nature of the NAND gate, there are ceratin probabilistic combinations of the inputs that will cause the output to be in a logic state that is an invalid state in the context of the model. For example, if one of the inputs of the NAND gate are logic low and the gate is not in error, the output is logic high. This scenario occurs when the inputs have higher probability of being 1 and a logic high is expected at the output if there is an error thus mildly elevating the probability of errors at the output. Also, the 2nd and 3rd order CTMR configurations provide almost the same reliability measures for lower gate failure rates.

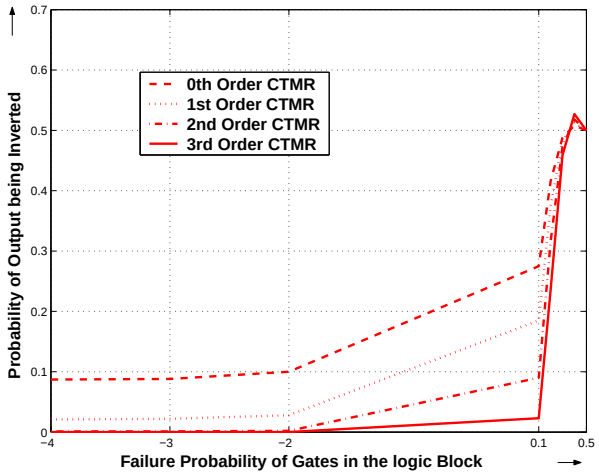
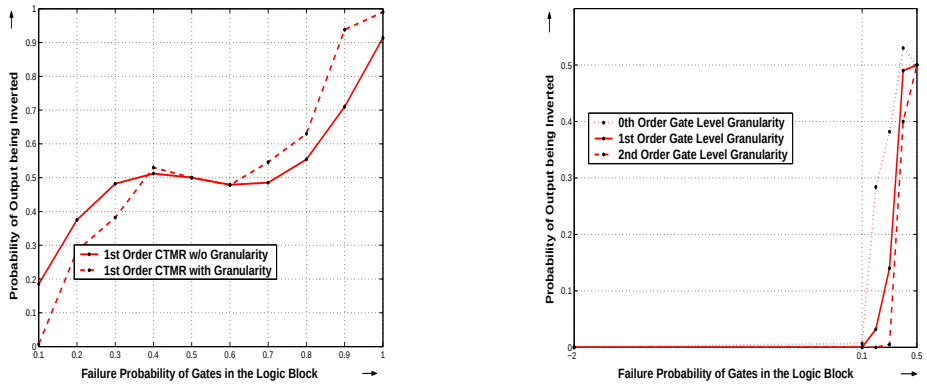


Figure 4.4: Error distribution at the outputs of the different CTMR orders for the Boolean network given in Figure 4.2

4.2.2 Reliability Measures for the Logic Block CTMR Configurations

We also analyze the combinational circuit given in Figure 4.2. This is to illustrate that our tool can be used to compute reliability measures of any arbitrary logic circuit. Figure 4.4 shows the output error distributions for different orders of CTMR configurations for the aforesaid logic network. The probability of the inputs being stimulated (logic high) is assumed to be 0.9. Also, the component inverter and NAND gates are assumed to have same failure distribution. These failure probability values $\in \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.2, ..., 0.5\}$. The plots observed in Figure 4.4 are similar to the previous experiment. It can be seen that as the CTMR orders increase, the probability of having the output in error goes down. Also, at lower gate error probabilities, it is clearly seen that the 2nd and 3rd order configurations provide the same reliability measures. Interestingly, in this case, at higher gate failure rates, the rate of improvement in the reliability of the system slows down steadily as the redundancy (in terms of CTMR orders) is increased. This is due to the augmentation of unreliable devices to

the architecture. Note that this degree of slow down is higher than what is observed in the case of single gate logic networks. This important observation can be interpreted as follows: for Boolean networks with higher number of unreliable component gates, increasing redundancy factor increases the probability of erroneous outputs and these probability values are higher than for single gate logic networks.



(a) 1st Order CTMR with and w/o Granularity

(b) Comparison of different Gate Level Granularities

Figure 4.5: Granularity Analysis for the Logic Block shown earlier

4.2.3 Reliability vs. Granularity Trade-offs for CTMR Configurations

We discuss the impact of applying reliability at different granularity levels for specific Boolean networks. We determine interesting facts and anomalies while experimenting with this form of redundancy based architectures. Figure 4.5 shows error distributions at the outputs of 1st order CTMR configurations with and without granularity. For illustration purposes, the granularity considered here is at the gate level but our generic libraries can handle granularity based redundancy at logic block level, logic function level, unit level etc. In Figure 4.5 (a), we compare the reliability measures of two different defect-tolerant architectural configurations of the logic circuit shown in Figure 4.2.

One of these is a 1st order CTMR configuration with redundancy at the logic block level (only the logic circuit is duplicated), whereas the other has gate level and logic block level redundancy and we call this a granularity based architectural configuration. In this experiment, the logic circuit is configured as a 1st order CTMR and the component gates are individually configured as TMR configurations. It is observed that at lower device failure rates, the reliability of the system for the granularity based architecture is better. This is intuitive because introduction of more redundant devices causes better reliability. But a counter-intuitive phenomena is observed at higher gate failure distributions. At the 0.4 failure probability mark, the probability of an erroneous output for the granular architecture becomes more than the one without granularity. This shows that for the CTMR configuration with gate level granularity, there is a degradation of the reliability of the system. The reason for this anomaly is that introduction of redundancy at any level of granularity entails addition of more unreliable devices to a specific Boolean architecture. Thus, this interesting finding ascertains the fact that there are indeed trade-off points between reliability and granularity of redundant systems that may impact the design of a defect-tolerant architecture.

We also model different orders of CTMR for the component gates (gate level granularity) while having an overall 1st order CTMR configuration for the logic block used in the previous experiment. The plots in Figure 4.5 (b) show that as the CTMR levels at the gate level of granularity increase, the reliability of the system improves steadily. Our experiments show that the reliability measures of the system remain almost the same beyond the 2nd order CTMR configuration. At lower gate failure distributions, the 1st order CTMR configuration at the gate level can be considered to be the optimal redundancy level of the overall architecture as increasing the level of redundancy further at the gate level does not yield much improvement in the overall reliability.

Chapter 5

Reliability Evaluation of Multiplexing Based Majority Circuits

In this emerging era of nanotechnology, non-silicon manufacturing methodologies such as quantum dots [84], quantum cellular automata (QCA) [11, 55] use majority logic as the fundamental building blocks for implementing Boolean computation. In this chapter, we analyze reliability measures of multiplexing based majority circuits by building a generic multiplexing library. To achieve these trade-off figures, and comparison of these figures against previously reported figures for NAND-multiplexing [62], we have enhanced our probabilistic model checking based tool NANOPRISM (discussed in details in Chapter 4). We outline the enhancements needed in our tool for evaluating majority multiplexing configurations. Other than the interesting reliability trade-off measures, the library based enhancement of the NANOPRISM tool is a major contribution by itself, especially to designers building QCA based architectures that work around probable defects, and in need of a design automation tool to quickly evaluate the trade-offs in their designs. Our NANOPRISM libraries can be applied to model any

arbitrary boolean circuit or a portion of a large Boolean network and also at different levels of granularity, such as gate level, logic block level, logic function level, unit level etc. Our attention to the importance of majority gates in the nanotechnology context was drawn by [76, 75], where analytical results for similar trade-off evaluations were presented.

5.1 Main Results

We have determined different reliability measures of multiplexing architectures based on majority circuits. We have also compared the results of different configurations for majority and NAND multiplexing. Our observations show that:

- the probability of errors at the output is higher for large gate failure probabilities.
- the probability of errors at the output decreases if more redundancy is incorporated in the majority multiplexing architecture, provided the probability of the gates failing is small.
- for large gate failure probabilities, due to augmentation of more redundant devices, either the reliability of the system does not improve or may even degrade.
- for the same redundancy factors, the majority multiplexing system is less reliable than NAND multiplexing when probability of the gates failing is small.
- the majority multiplexing provides better reliability as compared to multiplexing systems based on NAND gates when the gate failure probabilities are large.

Such observations could be of significant consequence in determining redundancy levels for systems with majority gates as the basic building blocks.

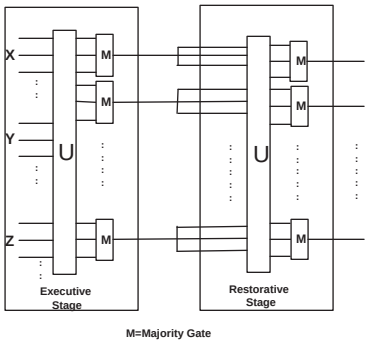


Figure 5.1: A majority multiplexing unit

5.2 Model Construction of Majority Multiplexing System

In this section, we explain the PRISM model of a majority gate multiplexing configuration. The first approach is directly modeling the system as shown in Figure 5.1. A PRISM module is constructed for each multiplexing stage comprising N majority gates and these modules are combined through synchronous parallel composition. However, such a model construction scheme leads to the well know state space explosion problem. Due to the observations stated earlier in Chapter 4, a model construction technique similar to the TMR and CTMR architectural configurations is adopted. The set of N majority gates working in *parallel* is replaced by N majority gates working in *sequence*. The same methodology is applied to the multiplexing stages of the system so as to reuse the same module for each of the stages while keeping a record of the outputs from the previous stage. This approach does not influence the performance of the system since each majority gate works independently and the probability of each gate failing is also independent.

The unit U in Figure 5.1 performs random permutation. Consider the case when k outputs from the previous stage are stimulated for some $0 < k < N$. Since there are k

stimulated outputs, the next stage will have k of the inputs stimulated if U performs random permutation. Therefore, the probability of either all or none of inputs being stimulated is 0. This implies that each of the majority gates in a stage are dependent on one other, for example, if one majority gate has a stimulated input, then the probability of another having the same input stimulated decreases. It is difficult to calculate the reliability of a system by means of analytical techniques for such a scenario. To change the number of restorative stages, bundle size, input probabilities or probability of the majority gates failing requires only modification of parameters in the PRISM model description. Since PRISM can also represent non-deterministic behavior, one can set upper and lower bounds on the probability of gate failure and then obtain best and worst case reliability characteristics for the system under these bounds.

Figure 5.2 shows the DTMC model of the von Neumann majority multiplexing system. We assume in this case that the inputs X and Y have identical probability distribution (probability of being logic high is 0.9), and the input Z can be non-stimulated with the same probability value. The failure (inverted output) probability of the majority gates is 0.01. However, these parameters can be modified by changing the value of the constants (*const*) given at the start of the model . The probabilistic state machine built by PRISM for the multiplexing configuration shown in Figure 5.2 has 1227109 states and 1846887 transitions. This indicates the complexity of analyzing such architectural configurations. The models for the different multiplexing configurations are verified for PCTL [49] properties such as ”probability of having less than 10% errors at the primary output”.

```
1   const N = 20;  // number of inputs in each bundle
2   const M = 3;   // number of restorative stages equals (M-1)
3   const low_lim = 1;  // the higher limit for logic low at majority gate output
4   const high_lim = 3; // the higher limit for logic high at majority gate
5
6   prob p_err = 0.01;  // probability that majority gate has von Neumann fault
7   prob p1_in=0.9;  // probability one of the inputs is logic high
8   prob p2_in=0.1;  // probability one of the inputs is logic high
9
10  module majority_multiplex
11      u : [1..M]; //current stage
12      c : [0..N]; // counter (number of copies of the majority gate done)
13      s : [0..5]; // local state
14      // number of stimulated inputs (outputs of previous stage)
15      nx : [0..N]; ny : [0..N]; nz : [0..N];
16      x : [0..1];  y : [0..1];  z : [0..1]; // value of inputs
17      k : [0..3]; // value of addendum of all inputs
18      out : [0..N];  // number of stimulated outputs
19
20      [] s=0 & c<N -> (s'=1); // next majority gate of current stage
21      // move on to next stage
22      [] s=0 & c=N & u<M -> (s'=1)&(nx'=out)&(ny'=out)&(nz'=out)&(u'=u+1)&(c'=0);
23      // finished (reset variables)
24      [] s=0 & c=N & u=M -> (s'=0)&(nx'=0)&(ny'=0)&(nz'=0)&(x'=0)&(y'=0)&(z'=0)&(k'=0);
25      // choose x (initially random)
26      [] s=1 & u=1 -> p1_in : (x'=1)&(s'=2) + (1-p1_in) : (x'=0)&(s'=2);
27      // permute
28      [] s=1 & u>1 -> nx/(N-c) : (x'=1)&(s'=2)&(nx'=nx-1) + (1-(nx/(N-c))) : (x'=0)&(s'=2);
29      // choose y (initially random)
30      [] s=2 & u=1 -> p1_in : (y'=1)&(s'=3) + (1-p1_in) : (y'=0)&(s'=3);
31      // permute
32      [] s=2 & u>1 -> ny/(N-c) : (y'=1)&(s'=3)&(ny'=ny-1) + (1-(ny/(N-c))) : (y'=0)&(s'=3);
33      // choose z (initially random)
34      [] s=3 & u=1 -> p2_in : (z'=1)&(s'=4) + (1-p2_in) : (z'=0)&(s'=4);
35      // permute
36      [] s=3 & u>1 -> nz/(N-c) : (z'=1)&(s'=4)&(nz'=nz-1) + (1-(nz/(N-c))) : (z'=0)&(s'=4);
37      // add values for the inputs to check for majority
38      [] s=4 -> (k'=x+y+z) & (s'=5);
39      // decide majority logic low or high
40      [] s=5 & k>=0 & k<=low_lim ->  (1-p_err) : (out'=out+0)&(s'=0)&(c'=c+1)&(k'=0) +
41                                     p_err : (out'=out+1)&(s'=0)&(c'=c+1)&(k'=0);
42      [] s=5 & k>low_lim & k<=high_lim -> (1-p_err) : (out'=out+1)&(s'=0)&(c'=c+1)&(k'=0) +
43                                          p_err : (out'=out+0)&(s'=0)&(c'=c+1)&(k'=0);
44  endmodule
```

Figure 5.2: PRISM description of the von Neumann majority multiplexing system

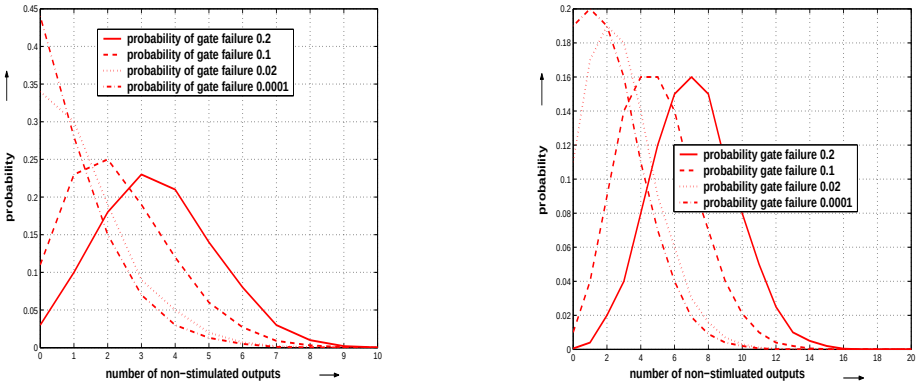
5.3 Reliability Evaluation of Multiplexing based Majority Systems

In this section, we compute the reliability measures of multiplexing based majority systems both when the I/O bundles are of size 10 and 20. These bundle sizes are only for illustration purposes and we have investigated the performance of these systems for larger bundle sizes. In all the experiments reported here, we assume that the inputs X, Y and Z are identical (this is often true in circuits containing similar devices) and that two of the inputs have high probability of being logic high (0.9) while the third input has a 0.9 probability of being a logic low. Thus the circuit's correct output should be stimulated. Also, it is assumed that the gate failure is a von Neumann fault, i.e. when a gate fails, the value of its output is inverted.

The PRISM model in Figure 5.2 is verified for properties such as the probability of having k non-stimulated outputs which in terms of Figure 5.2 corresponds to the probability of reaching the state where out $= N - k$, u $=$ M and c $=$ N, for $k = 0, \ldots, N$ where N is the I/O bundle size. This is the computation of the error distribution at the output of the majority gate architectural configuration. Hence any measure of reliability can be calculated from these results. PRISM can be also be used to directly compute other reliability measures such as, the probability of errors being less than 10% and the expected number of incorrect outputs of the system. Our analysis of reliability for the majority multiplexing system using the NANOPRISM generic multiplexing library concentrates on the effects of the failure probabilities of the majority gates and the number of restorative stages. The results we present show:

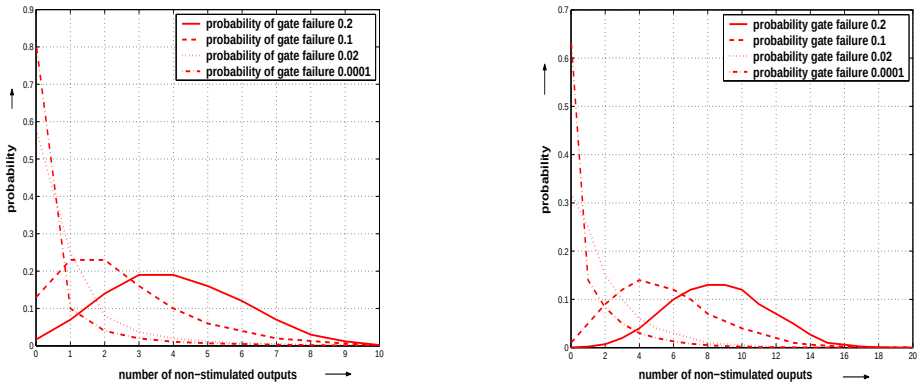
- the error distribution at the output for different gate failure probabilities (Figure 5.3).

- the error distribution at the output for different gate failure probabilities when additional restorative stages are augmented (Figure 5.4).
- reliability measures in terms of the probability that at the most 10% of the outputs are erroneous, as the probability of gates failure varies (Figure 5.5).
- reliability almost reaches steady state for small gate failure probabilities and can be improved marginally once a certain number of restorative stages have been augmented to the architecture. (Figure 5.5).
- the maximum probability of gate failure allowed for the system to function reliably by comparing the probability that at most 10% of the outputs are incorrect and the expected percentage of incorrect outputs for different numbers of restorative stages (Figures 5.6 and 5.7).



(a) I/O bundle size equals 10 (b) I/O bundle size equals 20

Figure 5.3: Error distributions at the output with 1 restorative stage under different gate failure rates and I/O bundle sizes



(a) I/O bundle size equals 10 (b) I/O bundle size equals 20

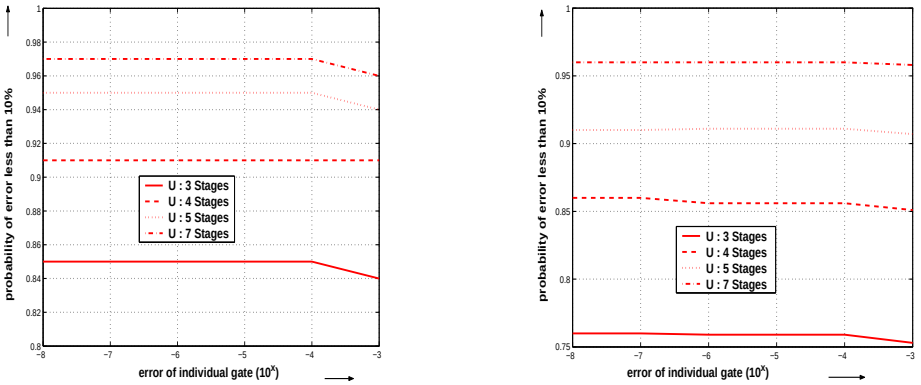
Figure 5.4: Error distributions at the output with 3 restorative stages under different gate failure rates and I/O bundle sizes

5.3.1 Error Distribution at the Outputs of Different Configurations

We consider a majority multiplexing system as shown in Figure 5.1, where the I/O bundle size equals 10 and 20. We first investigate the effect of varying the failure probabilities of the majority gates on the reliability of the system. Figure 5.3 shows the error distribution at the output of the system in the cases when the probability of gate failure equals 0.2, 0.1, 0.02 and 0.0001. The experimental setup is such that two of the inputs of the majority gate are stimulated when working correctly, and the correct output of the majority gate should be logic high. Hence, the more outputs that are non-stimulated, the lesser the reliability of the system.

As expected, the distributions given in Figure 5.3 show that as the probability of a gate failure decreases, the reliability of the multiplexing system increases, i.e. the probability of the system returning incorrect results diminishes. If Figure 5.3 (a) and (b) are compared, it can be determined that the increase in I/O bundle size increases the reliability of the system as the probability of having erroneous outputs decreases.

Also, additional restorative stages are augmented to the architecture and the change in reliability is observed. Figure 5.4 presents the error distributions at the output of the system with 3 restorative stages. The gate failure probabilities are similar to Figure 5.3. Comparing these distributions with those presented in Figure 5.3, we observe that, when the gate failure probability is sufficiently small (e.g. 0.0001), augmenting additional restorative stages results in increase of reliability i.e. the probability of non-stimulated outputs is small. On the other hand, in the cases when the gate failure probability is sufficiently large, adding additional stages does not increase reliability and, in fact, can decrease the reliability of the system (compare the distributions when the failure probability equals 0.2 for each bundle size).

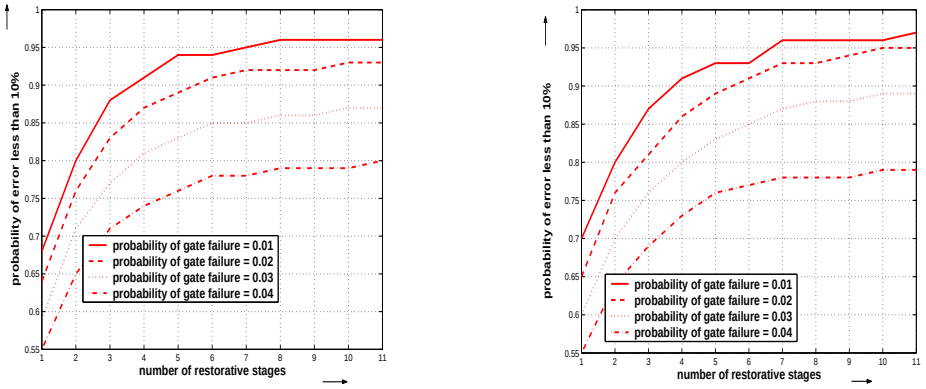


(a) I/O bundle size equals 10 (b) I/O bundle size equals 20

Figure 5.5: Probability that atmost 10% of the outputs being incorrect for different I/O bundle sizes (small probability of failure)

5.3.2 Small Gate Failure Probabilities

In Figure 5.5, we have plotted the probability that less than 10% of the outputs are incorrect against small gate failure probabilities for multiplexing stages of 3, 4, 5 and 7. These plots show that for small gate failure probabilities, the reliability of the



(a) I/O bundle size equals 10 (b) I/O bundle size equals 20

Figure 5.6: Probability that atmost 10% of the outputs of the overall system are incorrect for different I/O bundle sizes (large probability of failure)

multiplexing system almost reaches a steady state. Comparing Figure 5.5(a) and (b), it can be inferred that increasing the bundle size results in improvement of the reliability of the system. We have also experimented with higher I/O bundle sizes such as 40 and 45, and the results from these multiplexing configurations show that the rate of increase in reliability of the system decreases as the bundle size increases.

Also, these results demonstrate that increasing the number of stages can greatly enhance the reliability of the system. However, the degree of improvement slows down as more restorative stages are added. Moreover, there exists an optimal redundancy level (in terms of number of restorative stages) beyond which the reliability improvement is marginal. For example, let us consider the plots presented in Figure 5.5 that indicate probability of erroneous outputs being less than 10% when the number of restorative stages equals 5 and 7. These show that the probability values increase marginally for different gate failure probabilities as compared to when the architecture has lower number of restorative stages. We should also mention that this result corresponds to the observation made in [47] that, as the number of stages increases, the output distribution

of the system will eventually become stable and independent of the number of stages.

5.3.3 Large Gate Failure Probabilities

In this subsection, we consider the case when the probability of gate failure of the majority gates becomes too large for the multiplexing system to function reliably. Figure 5.6 plots the probability that less than 10% of the outputs are incorrect against large gate failure probabilities (between 0.01 and 0.04) for different number of multiplexing stages $\in \{1, 2, \cdots, 10, 11\}$. As can be seen from the results, when the probability of gate failure is equal to 0.04 increasing the I/O bundle size does not improve the reliability of the system. Comparing the same plots in Figure 5.6 (a) and (b), it can be observed that augmenting additional restorative stages and increasing the I/O bundle size of the architecture tends to cause a degradation in reliability of computation. This anomalous behavior can be understood as follows: when the failure rate is 0.04 (or higher), the restorative stages are sufficiently affected by the probability of gate failures to be unable to reduce the degradation, and hence increasing the number of stages in this case makes the system more amenable to error.

We can infer from these observations that for gate probabilities of 0.04 and higher, increasing bundle size or addition of more restorative stages cannot make the system more reliable and may even cause degradation. On the other hand, in the case when the gate failure probability is less than 0.04, the results demonstrate that the system can be made reliable once a sufficient number of restorative stages have been added.

In Figure 5.7, we have plotted the expected percentage of incorrect outputs when I/O bundle size equals 10. The gate failure probabilities and restorative stages are configured similar to Figure 5.6. The plots in Figures 5.7 and 5.6 determine similar

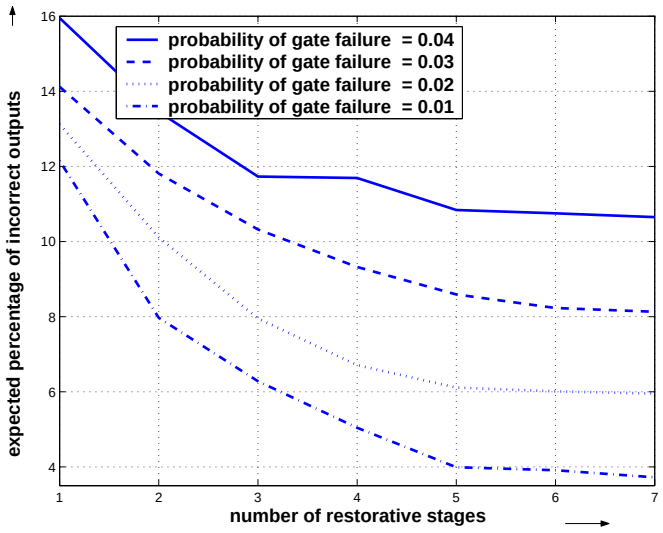


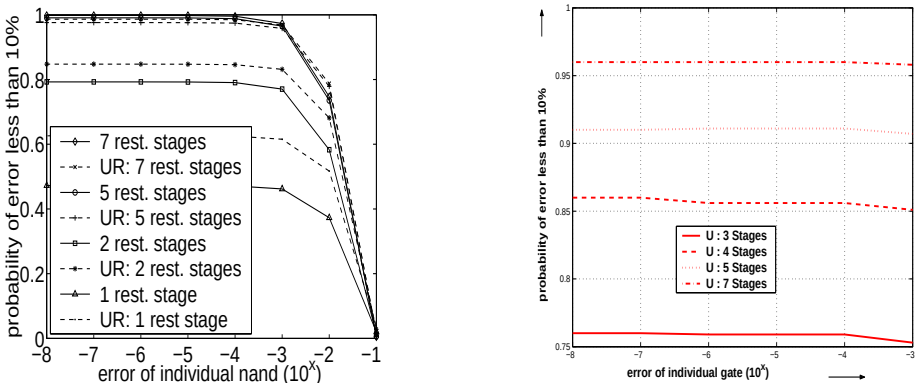
Figure 5.7: Expected percentage of incorrect outputs for I/O bundle size of 10 (large probability of failure)

intuitive results but have different perspectives to reliability evaluation of the system. By similar facts, we mean that for gate failure probabilities of 0.04 and higher, increasing the number of restorative stages cannot improve reliability of the system to a very high level. In fact, when the multiplexing system is configured to have certain specific number of restorative stages, the reliability may decrease as compared to a system with less redundancy (less number of multiplexing stages). It can also be observed from Figure 5.7 that for all gate failure probabilities, the reliability of the architecture reaches a steady state once a sufficient number of restorative stages have been added. Such reliability-redundancy trade-off points may mitigate challenges in design and evaluation of defect-tolerant nano-architectures.

It is important to note that there is a difference between the bounds computed by our NANOPRISM tool on the probability of gate failure required for reliable computation and the theoretical bounds presented in the literature. This difference is to be

expected since our methodology evaluates the performance of systems under fixed configurations, whereas the bounds presented in the literature correspond to scenarios where different redundancy parameters can be increased arbitrarily in order to achieve a reliable system.

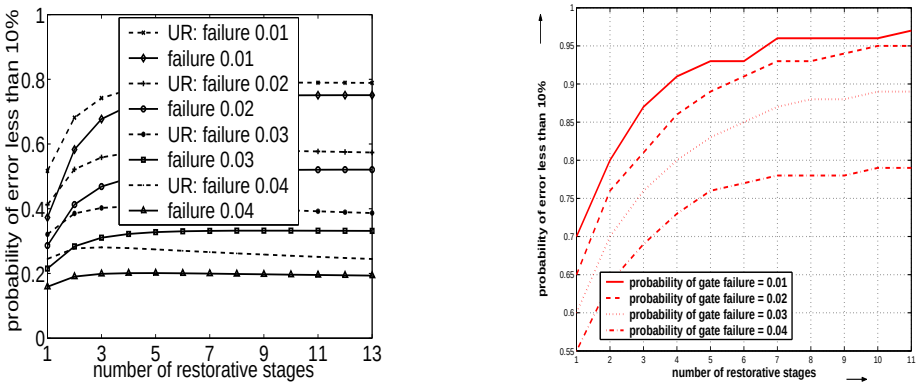
5.4 Comparison of Reliability Measures for NAND and Majority Multiplexing Systems



(a) NAND multiplexing [62] (b) Majority multiplexing

Figure 5.8: Probability of atmost 10% of the outputs being incorrect for the multiplexing systems when I/O bundle size equals 20 and gate failure probabilities are small

In this section we compare reliability measures of the multiplexing architectural configuration for the NAND and majority gates. The input distributions for the majority multiplexing system (shown in Figure 5.1) are identical to the ones assumed in Section 5.3 i.e. two of the inputs have high probability of being logic high (0.9) while the third input has a 0.9 probability of being a logic low. Thus the normal operation of the system should return a stimulated output. The reliability results for the NAND multiplexing configuration (Figure 1.1) reported in [62] assume that the inputs X and Y



(a) NAND multiplexing [62] (b) Majority multiplexing

Figure 5.9: Probability of atmost 10% of the outputs being incorrect for the multiplexing systems when I/O bundle size equals 20 and gate failure probabilities are large

are identical and that initially 90% of the inputs are stimulated. It is also assumed that for these multiplexing configurations, the gate failure is a von Neumann fault (inverted output error).

Figure 5.8 compares the reliability of the majority and NAND multiplexing systems in terms of probability of atmost 10% of the outputs being incorrect, for I/O bundle size of 20 and small gate failure probabilities. The number of restorative stages varies between 3 and 7. We only consider the plots for the unit U (random permutation of the input signals) in Figure 5.8 (a). Comparing these results, we can infer that for both the multiplexing systems, increasing the number of stages enhances the reliability measures and that the rate of increase in reliability decreases as more restorative stages are added to the configurations. It can also be observed that NAND multiplexing provides better reliability than its majority gate counterpart, and such an observation is independent of the number of restorative stages.

Also, Figure 5.9 gives a comparison of the majority and NAND multiplexing systems in terms of reliability, for I/O bundle size of 20 and when probability of gate failure is large.

The gate failure probabilities vary between 0.01 and 0.04. In Figure 5.9 (a), we only observe the curves for U since our majority multiplexing model only performs random permutation of the signals. Figure 5.9 (a) and (b) show that for both the multiplexing systems as the probability of gate failure increases, high reliability of computation cannot be achieved even by augmenting additional restorative stages. This is because the reliability of the system reaches a saturation point at an optimal redundancy factor (number of restorative stages). The results reported also indicate that the majority multiplexing scheme provides better reliability than NAND multiplexing, and such an observation is independent of the number of restorative stages.

Chapter 6

Conclusion and Future Work

This thesis proposes two automation methodologies that can expedite reliability evaluation of defect-tolerant architectures for nanodevices. We have developed tools that can determine optimal redundancy and granularity levels of any specific logic network, given the failure distribution of the devices and the desired reliability of the system. These tools can also be used to inject signal noise at inputs and interconnects and create practical situations the circuits are subjected to during normal operation.

We have developed NANOLAB, a tool that is based on a non-discrete probabilistic design methodology proposed by Bahar et al. This computational scheme has two aspects. First, the gates are assumed to be defect free, and the model of computation based on Markov Random Fields correlates information theoretic entropy and the thermal entropy of computation. Since at the reduced voltage level in nano-architecture this issue can become significant, reliability may suffer when the computation is carried out close to the thermal limit of computation. However, we show that by considering various defect-tolerant architectural techniques such as TMR, CTMR and multi-stage

iterations of these, the effects of carrying out computation within close thermal limits can be reduced substantially. Second, signal noise can be injected at the inputs and interconnects of a logic network. We have introduced the effects of signal noise in our automation methodology. Noise is modeled using Gaussian and uniform probability distributions, and this goes beyond the thermal aspects as described above. NANOLAB automatically computes reliability measures of systems in terms of energy distributions and entropy in the presence of discrete input distributions and random noise. This tool consists of MATLAB based libraries for generic logic gates that can calculate the probability of the outputs being in different energy levels, and Belief Propagation algorithm that can compute such distributions at the primary outputs of arbitrary Boolean networks (given as inputs to the tool). It also supports various interconnect noise models, and can be very useful in modeling transient faults and provide the designers a way to figure out the configuration that is best in terms of reduced entropy (which in turn means higher reliability). This is indeed an effective tool and technique for evaluating reliability and redundancy trade-offs in the presence of thermal perturbations and interconnect noise models. Our tool can also inject error probabilities at the gates and do similar trade-off calculations.

We have reported on the results obtained for single gate and logic block level architectural configurations and investigated the performance of the system in terms of entropy and energy distribution, as the number of CTMR orders vary. Using NANOLAB, we were able to compute the energy distributions at the outputs of systems, and hence construct a complete picture of the reliability of the systems under study. We chose to analyze TMR and CTMR for illustration purposes as these are canonical fault tolerant architectures standard in the literature, and since these enabled us to compare the results with others. However, as explained, this approach can equally be applied to alternative fault-tolerant architectures for nanotechnology.

This thesis also presents NANOPRISM, a reliability evaluation tool based on probabilistic model checking. This tool consists of libraries for different redundancy based defect-tolerant architectural configurations. We have analyzed reliability measures of specific logic circuits with this tool, and shown that for a given probability of gate failure, we can find the minimum level of redundancy and *granularity* that enables reliable computation. The NANOPRISM libraries support implementation of redundancy at different levels of granularity, such as gate level, logic block level, logic function level, unit level etc. We illustrate the proficiency of our methodology by modeling von Neumann multiplexing systems, and different orders of CTMR for arbitrary logic circuits. We also describe a methodology that reduces the effect of the well known state space explosion problem, and hence allows for the analysis of larger configurations. The experimental results from our tool show anomalous and counter-intuitive phenomena that may not be detected quickly by analytical methods.

NANOLAB applies an approach to reliability analysis that is based on Markov Random Fields as the probabilistic model of computation. Whereas, NANOPRISM applies probabilistic model checking techniques to calculate the likelihood of occurrence of probabilistic transient defects in devices and interconnections. Thus, there is an inherent difference in the model of computation between these two approaches. Although these two methodologies are different, they offer complementary alternatives to analytical methodologies and allow system architects to obtain sharp bounds and study anomalies for specific architectures. Future work includes extending these tools to support reliability analysis of sequential circuits. We are also in the process of building libraries for other frequently used fault tolerance schemes.

Bibliography

[1] Web Page: www.mathworks.com.

[2] R. I. Bahar, J. Mundy, and J. Chen, *A probability-based design methodology for nanoscale computation*, in *International Conference on Computer-Aided Design* (IEEE Press, San Jose, CA, November 2003).

[3] C. H. Bennett, 'The thermodynamics of computation–a review', *International Journal of Theoretical Physics* **21** (1982.), no. 905-940.

[4] J. Besag, 'Spatial interaction and the statistical analysis of lattice systems', *Journal of the Royal Statistical Society* **Series B** (1994), no. 36(3), 192–236.

[5] D. Bhaduri and S. K. Shukla, 'Nanolab: A tool for evaluating reliability of defect-tolerant nano architectures', *Tech. Report* (Fermat Lab, Virginia Tech, 2003). Available at `http://fermat.ece.vt.edu/Publications/pubs/techrep/techrep0309.pdf`.

[6] D. Bhaduri and S. K. Shukla, 'Nanoprism: A tool for evaluating granularity vs. reliability trade-offs in nano architectures', *Tech. Report* (Fermat Lab, Virginia Tech, 2003). Available at `http://fermat.ece.vt.edu/Publications/pubs/techrep/techrep0318.pdf`.

[7] D. Bhaduri and S. K. Shukla, *Nanolab: A tool for evaluating reliability of defect-tolerant nano architectures*, in *IEEE Computer Society Annual Symposium on VLSI* (IEEE Press, Lafayette, louisiana, February 2004). `http://fermat.ece.vt.edu`.

[8] D. Bhaduri and S. K. Shukla, *Nanoprism: A tool for evaluating granularity vs. reliability trade-offs in nano architectures*, in *GLSVLSI* (ACM, Boston, MA, April 2004).

[9] D. Bhaduri and S. K. Shukla, 'Reliability evaluation of multiplexing based defect-tolerant majority circuits', to appear in the proceedings of *IEEE Conference on Nanotechnology* (Munich, Germany August 2004).

[10] P. Beckett and A. Jennings. Towards nanocomputer architecture. In *ACS Conferences in Research and Practice in Information Technology (CRPIT)*, volume 6 of *Computer Systems Architecture*. Australian Computer Society Inc, 2002. Available at `http://www.cellmatrix.com/entryway/products/pub/publications.html`

[11] I. Amlani, A. O. Orlov, G. Toth, G. H. Bernstein, C. S. Lent, and G. L. Snider, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284, no. 289-291, April 1999, available at `http://www.nd.edu/~qcahome/reprints/Amlani2.pdf.`

[12] D. Bhaduri and S. K. Shukla, *Tools and techniques for evaluating reliability of defect-tolerant nano architectures*, in *International Joint Conference on Neural Networks* (IEEE Press, Budapest, Hungary, July, 2004).

[13] D. Bhaduri and S. Shukla, 'Reliability analysis in the presence of signal noise for nano architectures', to appear in the proceedings of *IEEE Conference on Nanotechnology* (Munich, Germany August 2004).

[14] F. Buot, 'Mesoscopic physics and nanoelectronics: nanosicence and nanotechnology', *Physics Reports* (1993), 173–174.

[15] S. Burris, *Boolean algebra* (March 2001). Available at `http://www.thoralf.uwaterloo.ca/htdocs/WWW/PDF/boolean.pdf`.

[16] J. Chen, M. Reed, and A. Rawlett, 'Observation of a large on-off ratio and negative differential resistance in an electronic molecular switch', *Science* **286** (1999), 1550–2.

[17] Yong Chen, Douglas A. A. Ohlberg, Xuema Li, Duncan R. Stewart, R. Stanley Williams, Jan O. Jeppesen, Kent A. Nielsen, J. Fraser, Deirdre L. Olynick and Erik Anderson, 'Nanoscale molecular-switch devices fabricated by imprint lithography', in *Applied Physics Letters* **82** (2003), no. 10, 1610–1612.

[18] James A. Hutchby, George I. Bourianoff, Victor V. Zhirnov and Joe E. Brewer, 'Extending The Road beyond CMOS', in *IEEE Circuits and Devices Magazine* (March 2002), 28–39.

[19] P. W. Shor, 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer', in *SIAM Journal for Computing* **26** (1997), no. 5, 1484–1509

[20] J. Chen, W. Wang, M. Reed, M. Rawlett, D. W. Price, and J. Tour, 'Room-temperature negative differential resistance in nanoscale molecular junctions', *Appl. Phys. Lett.* **1224** (2000), 77.

[21] J. Chen, J. Mundy, Y. Bai, S.-M. Chan, P. Petrica, and I. Bahar, *A probabilistic approach to nano-computing*, in *IEEE non-silicon computer workshop* (IEEE Press, San Diego, June 2003).

[22] R. Chen, A. Korotov, and K. Likharev, 'Single-electron transistor logic', *Apll. Phys. Lett.* (1996), 68:1954.

[23] C. Collier, E. Wong, M. Belohradsky, F. Raymo, J. Stoddart, P.J.Kuekes, R. Williams, and J. Heath, 'Electronically configurable molecular-based logic gates', *Science* **285** (1999), 391–3.

[24] Y. J. Tung, P. G. Carey, P. M. Smith, S. D. Theiss, X. Meng, R. Weiss, G. A. Davis, V. Aebi, and T. J. King, 'An Ultra-Low-Temperature-Fabricated Poly-Si TFT with Stacked Composite ECR-PECVD Gate Oxide', *SID Int. Symp. Digest of Technical Papers, Anaheim* (CA, May 1998).

[25] S. D. Theiss, P. G. Carey, P. M. Smith, P. Wickboldt, T. W. Sigmon, Y. J. Tung, and T. J. King, 'Polysilicon Thin Film Transistors Fabricated at 100degC on a Flexible Plastic Substrate', *Int. Electron Devices Mtg* (San Francisco, CA, December 1998).

[26] Webster E. Howard, 'Better Displays with Organic Films', *Scientific American* (Feb. 2004), p. 76.

[27] Joseph Shinar, *Organic Light-Emitting Devices: A Survey* (Springer-Verlag, NY 2004). ISBN 0-387-95343-4.

[28] Broome, 'How to Design Optical Systems Using Cost-Effective Manufacturing Technologies', in *Optical Society of America* (1990), Vol. 15, p. 204.

[29] Y. Cui and C. Lieber, 'Functional nanoscale electronic devices assembled using silicon nanowire building blocks', *Science* **291** (2001), 851–853.

[30] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti, 'Toward robust integrated circuits: the embryonics approach', in *IEEE*, **88**, 516–41.

[31] R. Dobrushin and E. Ortyukov, 'Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements', *Problems of Information Transmission* **13** (1977), no. 3, 203–218.

[32] L. Durbeck, *Underlying future technology talk on computing: Scaling up, scaling down, and scaling back* (American Nuclear Society's International Meeting on Mathematical Methods for Nuclear Applications, September 2001). Available at `http://www.cellmatrix.com/entryway/products/pub/ANS_Abstract.html`.

[33] L. Durbeck and N. Macias, 'The cell matrix: An architecture for nanocomputing', in *Nanotechnology*, **12** (Institute of Physics Publishing, 2001), 217230.

[34] L. J. K. Durbek, *An approach to designing extremely large, extremely parallel systems*, in *Conference on High Speed Computing* (Oregon, USA, April 2001).

[35] J. C. Ellenbogen and J. C. Love, 'Architectures for molecular electronic computers: Logic structures and an adder designed from molecular electronic diodes', in *IEEE*, *3* **88** (2000), 386–426.

[36] W. Evans and N. Pippenger, 'On the maximum tolerable noise for reliable computation by formulas', *IEEE Transactions on Information Theory* **44** (1998), no. 3, 1299–1305.

[37] D. F, S. M. G, and S. R, 'Fault-tolerance and reconfigurability issues in massively parallel architectures', in *Computer Architecture for Machine Perception* (IEEE Computer Society Press, Los Alamitos, CA, 1995), 3409.

[38] M. Forshaw, K. Nikolic, and A. Sadek, 'EC answers project (Melari 28667)', *Third Year Report*. Available at http://ipga.phys.ucl.ac.uk/research/answers.

[39] S. C. Goldstein and M. Budiu, 'Nanofabrics: Spatial computing using molecular electronics', in *Annual International Symposium on Computer Architecture (ISCA)* (July 2001), 178–191.

[40] S. C. Goldstein and D. Rosewater, 'Digital logic using molecular electronics', in *IEEE International Solid-State Circuits Conference (ISSCC)* (San Francisco, CA, Feb 2002), 204–205.

[41] P. Graham and M. Gokhale, 'Nanocomputing in the Presence of Defects and Faults: A Survey', in *Nano, quantum and molecular computing: Implications to high level design and validation* (Kluwer Academic Publishers, June 2004), 39–72. To be Published.

[42] André DeHon, 'Law of large numbers system design', in *Nano, quantum and molecular computing: Implications to high level design and validation* (Kluwer Academic Publishers, June 2004), 213–244. To be Published.

[43] J. C. Huie, 'Guided molecular self-assembly: a review of recent efforts', *Smart Materials and Structures* **12** (2003), no. 2, 264–271.

[44] R. C. Merkle, 'Molecular manufacturing: adding positional control to chemical synthesis', *Chemical Design Automation News* **8** (1993), no. 9 and 10, p 1.

[45] Alvin W. Drake, *Fundamentals of Applied Probability Theory* McGraw-Hill (1988).

[46] S. C. H, 'A new statistical approach for fault-tolerant vlsi systems', in *Int. Symp. on Fault-Tolerant Computing* (IEEE Computer Society Press, Los Alamitos, CA, 1992), 35665.

[47] J. Han and P. Jonker, 'A system architecture solution for unreliable nanoelectronic devices', *IEEE Transactions on Nanotechnology* **1** (2002), 201–208.

[48] J. Han and P. Jonker, 'A defect- and fault-tolerant architecture for nanocomputers', *Nanotechnology* (2003), 224230. IOP Publishing Ltd.

[49] H. Hansson and B. Jonsson, 'A logic for reasoning about time and probability', *Formal Aspects of Computing* **6** (1994), no. 5, 512–535.

[50] J. Heath, P. Kuekes, G. Snider, and R. Williams, 'A defect tolerant computer architecture: Opportunities for nanotechnology', *Science* **80** (1998), 1716–1721.

[51] K. I and K. Z, 'Defect tolerance in vlsi circuits: techniques and yield analysis', in *IEEE*, **86** (1998), 181936.

[52] J.Yedidia, W.Freeman, and Y.Weiss, *Understanding belief propagation and its generalizations*, in *Proc. International Joint Conference on AI* (2001). Distinguised Lecture.

[53] S. Knapp, *Fpga lookup tables build flexible pattern matchers* (EDA Expert Column, May 1998). Available at `http://www.optimagic.com/acrobat/pein_0598.pdf`.

[54] M. Kwiatkowska, G. Norman, and D. Parker, 'Prism: Probabilistic symbolic model checker', in *TOOLS 2002*, *LNCS* **2324** (Springer-Verlag, April 2002), 200–204.

[55] C. Lent, 'A device architecture for computing with quantum dots', in *Porceedings of the IEEE*, **541** (April 1997), 85.

[56] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, 'Quantum cellular automata', *Nanotechnology* **4** (1993), 49–57.

[57] S. Li, *Markov random field modeling in computer verification* (Springer-Verlag, 1995).

[58] M. Mishra and S. C. Goldstein, 'Defect Tolerance at the End of the Roadmap', in *Nano, quantum and molecular computing: Implications to high level design and validation* (Kluwer Academic Publishers, June 2004), 73–108 To be Published.

[59] M. Mishra and S. C. Goldstein, *Defect tolerance at the end of the roadmap*, in *International Test Conference (ITC)* (Charlotte, NC, Sep 30-Oct 2 2003).

[60] K. Nikolic, A. Sadek, and M. Forshaw, 'Architectures for reliable computing with unreliable nanodevices', in *Proc. IEEE-NANO'01* (IEEE, 2001), 254–259.

[61] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, 'Evaluating reliability of defect tolerant architecture for nanotechnology using probabilistic model checking', *Tech. Report* (Fermat Lab, Virignia Tech, 2003). Available at http://fermat.ece.vt.edu/Publications/pubs/techrep/techrep0314.pdf.

[62] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, *Evaluating reliability of defect tolerant architecture for nanotechnology using probabilistic model checking*, in *IEEE VLSI Design Conference* (IEEE Press, Mumbai, India, January 2004).

[63] J. R. Norris, *Markov chains*, in *Statistical and Probabilistic Mathematics* (Cambridge University Press, October 1998).

[64] C. P. Collier, G. Mattersteig, E. W. Wong, Y. Luo, K. Beverly, J. Sampaio, F. Raymo, J. Stoddart and J. R. Heath, 'A [2] catenane-based solid state electronically reconfigurable switch', *Science* **280** (2000), 11725.

[65] N. Pippenger, 'Reliable computation by formulas in the presence of noise', *IEEE Transactions on Information Theory* **34** (1988), no. 2, 194–197.

[66] R. Feynman, 'Simulating physics with computers', *International Journal of Theoretical Physics* **21** (1982), no. 6/7, 467–488.

[67] R. Feynman, 'Quantum mechanical computers', *Optics News* **11** (1985), 11–20.

[68] C. P. Williams and S. H. Clearwater, *Explorations in quantum computing*, **TELOS** (Springer-Verlag New York, 1998).

[69] L. S. Smith and A. Hamilton, *Neuromorphic systems engineering silicon from neurobiology*, in *Progress in Neural Processing* **10** (World Scientific, May 1998).

[70] Gordon Moore, *Cramming more Components onto Integrated Circuits* (1965), **38** , no. 8.

[71] D. B. Pollard, *Hammersley-clifford theorem for markov random fields* (2004). Handouts, available at `http://www.stat.yale.edu/~pollard/251.spring04/Handouts/Hammersley-Clifford.pdf`.

[72] M. Roukes, 'Nanoelectromechanical systems face the future', *Physics World* (2001), 25.

[73] PRISM Web Page: `www.cs.bham.ac.uk/~dxp/prism/`.

[74] W. Roscoe, *Theory and practice of concurrency* (Prentice Hall, 1998).

[75] S. Roy, V. Beiu, and M. Sulieman, *Majority multiplexing: Economical redundant fault-tolerant designs for nano architectures*.

[76] S. Roy, V. Beiu, and M. Sulieman, *Reliability analysis of some nano architectures*.

[77] N. Saxena and E. McCluskey, 'Dependable adaptive computing systems', in *IEEE Systems, Man, and Cybernetics* (San Diego, CA, Oct 11-14 1998), 2172–2177. The Roar Project.

[78] Semiconductor Industry Association (SIA), *International roadmap for semiconductors 2001 edition* , Ausitn, TX: International SEMATECH, 2001. Available at `http://public.itrs.net`.

[79] K. Mikami, K. Takahashi, T. Nakai, and T. Uchimaru, 'Asymmetric Tandem Claisen-Ene Strategy for Convergent Synthesis of (+)-9(11)-Dehydroestrone Methyl Ether : Stereochemical Studies on the Ene Cyclization and Cyclic Enol Ether Claisen Rearrangement for steroid Total Synthesis' in *Journal of the American Chemical Society* (1994), pp. 116.

[80] Henderson Research Group, School of Chemical Engineering, Georgia Tech, Web Site: `http://dot.che.gatech.edu/henderson/`

[81] D. P. Siewiorek and R. S. Swarz, *Reliable computer systems: Design and evaluation* (Digital Press, Burlington, MA, 2nd ed., 1992).

[82] E. J. Siochi, P. T. Lillehei, K. E. Wise, C. Park, and J. H. Rouse, 'Design and characterization of carbon nanotube nanocomposites', *Tech. Report* (NASA Langley Research Center, Hampton, VA, 2003). Available at http://techreports.larc.nasa.gov/ltrs/PDF/2003/mtg/NASA-2003-4ismn-ejs.pdf.

[83] G. L. Snider, A. O. Orlov, I. Amlani, G. H. B. adn Craig S. Lent, J. L. Merz, and W. Porod, 'Quatum-dot cellular automata: Line and majority logic gate', *Japanese Journal of Applied Physics* **38** (1999), no. 12B, 7227–7229.

[84] R. Turton, *The quantum dot: A journey into the future of microelectronics* (Oxford University Press, U.K, 1995).

[85] J. von Neumann, 'Probabilistic logics and synthesis of reliable organisms from unreliable components', *Automata Studies* (1956), 43–98.

[86] T. G. Y and E. J. C, 'Toward nanocomputers', *Science* **294** (2001), 12934.

[87] L. Lavagno, A. Sangiovanni-Vincentelli, and E. Sentovich, Models of Computation for Embedded System Design, Website: `citeseer.nj.nec.com/lavagno98model.html`, 1998.

Vita

Debayan Bhaduri

Personal Data

Date of Birth	:	July 30, 1978
Marital Status	:	Single
Visa Status	:	F-1

Office Address:

FERMAT Research Lab.
340 Whittemore Hall
Blacksburg, VA 24061
Email: dbhaduri@vt.edu
URL: http://www.fermat.ece.vt.edu/dbhaduri
Tel: (617) 519-5346

Research Interests

(a) Embedded Systems Design

(b) Formal Verification of Probabilistic Systems

(c) Defect Tolerant Computing for Nano Architectures

(d) Modeling and Validation of Network Protocols

(e) Software Design

(f) High Level Specification of Complex Control Systems

Education

(a) Ph.D., Computer Engineering, (Expected May 2007), Virginia Polytechnic Institute and State University.

(b) M.S., Computer Engineering, (Expected May 2004), Virginia Polytechnic Institute and State University.

M.S. Thesis Title: "Tools and Techniques for Evaluating Reliability Trade-offs for Nano-Architectures"

(c) B.S., Computer Engineering, June 2001, Manipal Institute of Technology, Manipal, India.

Current Work

(a) Applying Probabilistic Approaches for the Analysis of Reliability-Redundancy Trade-offs for Nano-scale Architectures

Applying probabilistic approaches for evaluation of reliability measures for different redundancy based Defect-Tolerant architectures for nano-scale devices. Developing a MATLAB based tool (code named **NANOLAB**) that implements a novel non-discrete probabilistic model of computation suited for unreliable nanoscale computation. Also developing a probabilistic model checking based tool called **NANOPRISM** that implements the conventional Boolean model of computation and can automatically evaluate reliability vs. redundancy vs. granularity trade-offs.

(b) Using Quantum Model of Computation for Reliability Evaluation of Defect Tolerant Nano-architectures

Applying the succinctness and parallelism of the Feynman's quantum model of computation to encode multiple possible input patterns into one superposed input, encode circuit computation as unitary evolution operations, and read multiple outputs by un-entangling the superposed output. Computing the probability values of various possible states at the output for various possible input probability distributions, and evaluating reliability of the circuit being modeled in terms of the entropy at the entangled output.

(c) High Level Specifications of Systems using XUML

Modeling high level specifications of complex control systems and network protocols in Executable Unified Modeling language (XUML). Simulation of these systems to check for conformance of certain properties like deadlock.

Skills

– C, C++, SystemC, Matlab, Mathematica, Java, JDBC, Winsock API, OPNET, NS2, Ptolemy II, X86 Assembly, JSP, ASP, JDBC, UML, XUML, PRISM.

– Linux and Microsoft Windows NT administration.

– Languages: English, Hindi and Bengali.

Experience

(a) January 2003 to present: Graduate Research Assistant, Fermat Lab, Virginia Tech., Blacksburg, VA.

- Investigating different probabilistic approaches to evaluate reliability/redundancy trade-offs of defect-tolerant nano architectures. Working on the development of two tools: a MATLAB based tool called NANOLAB, and a library package for generic fault tolerant architectures in a probabilistic symbolic model checking based framework called NANOPRISM.
- Developing a tool for evaluating reliability measures of Boolean networks using succinct and novel quantum models of computation.

- Developed a methodology to automate conversion of arbitrary MTG (Multi threaded Graph) models to SystemC models.
- Developed systematic abstraction mechanisms for RTL models of microprocessors written in SystemC to provide simulation efficiency.
- Applied UML and XUML to design and model complex systems and network protocols, and verified for conformance to requirement specifications.

(b) May 2003 to August 2003: Summer Intern, BBN Technologies, Boston, MA.

Worked for the DARPA Next Generation MAC (XG) Project. Participated in the development of an EM spectrum meta-language framework, modeling and design of XG network protocols.

- Aided in expanding the knowledge base for UML.
- Developed a console front-end for the DARPA query language (DQL) tool.
- Explored different XUML tools for project specific network protocol modeling and simulation requirements.
- Modeling and simulation of network protocols using Ptolemy II.
- Developed mapping Specifications from Ptolemy II to UML so as to expedite conversion of Ptolemy II models to UML diagrams and vice versa.
- Worked on the different design alternatives of the XG Opportunity Information Dissemination protocol.

(c) July 2001 to August 2002: Software Engineer, Axes Technologies (Subsidiary of Alcatel USA Inc), Bangalore, India

Worked on the Motorola ENHANCED TRUNK AUDITING Project for the Alcatel Electronic Mobile Exchange (EMX) 2500/5000. Participated in the implementation and testing of an enhancement to the pre-existing audit feature in the EMX.

- Implementation and testing of the call processing manager and line-trunk manager interfaces.
- Development of new man-machine interfaces (MMI).

Publications

[1] S. Sharad, D. Bhaduri, M. Chandra, H. D. Patel, and S. Syed, *Systematic abstraction of microprocessor rtl models to enhance simulation efficiency*, in *the proceedings of Microprocessor Test and Verification 2003* (2003).

[2] D. Bhaduri and S. K. Shukla, *Nanolab: A tool for evaluating reliability of defect-tolerant nano architectures*, in *IEEE Computer Society Annual Symposium on VLSI* (IEEE Press, Lafayette, louisiana, February 2004). Available at http://fermat.ece.vt.edu/Publications/pubs/techrep/techrep0309.pdf.

[3] D. Bhaduri and S. K. Shukla, *Nanolab: A tool for evaluating reliability of defect-tolerant nano architectures*, in *IEEE Transactions on Nanotechnology* (2004). Being revised.

[4] D. Bhaduri and S. K. Shukla, *Nanoprism: A tool for evaluating granularity vs. reliabilty trade-offs in nano architectures*, in *Great Lakes Symposium on VLSI* (ACM, Boston, MA, April 2004). Available at http://fermat.ece.vt.edu/Publications/pubs/techrep/techrep0318.pdf.

[5] D. Bhaduri and S. K. Shukla, 'Reliability analysis in the presence of signal noise for nano architectures', to appear in the proceedings of *IEEE Conference on Nanotechnology* (Munich, Germany August 2004).

[6] D. Bhaduri and S. K. Shukla, 'Reliability evaluation of multiplexing based defect-tolerant majority circuits', to appear in the proceedings of *IEEE Conference on Nanotechnology* (Munich, Germany August 2004).

[7] D. Bhaduri and S. K. Shukla, *Tools and techniques for evaluating reliability of defect-tolerant nano architectures*, in *International Joint Conference on Neural Networks* (IEEE Press, Budapest, Hungary, July 2004).

[8] D. Bhaduri and S. K. Shukla, 'Tools and techniques for evaluating reliability trade-offs for nano-architectures', in *Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation* (Kluwer Academic Publishers, June 2004), 157–212. To be Published.

[9] D. Bhaduri and S. K. Shukla, 'Using quantum model of computation for reliability evaluation of defect tolerant nano-architectures', to appear in the proceedings of *IEEE Conference on Nanotechnology* (Munich, Germany August 2004).

Selected Projects

(a) Implementation of a Markov Random Field based model of computation for evaluating reliability measures of defect-tolerant architectures.

(b) Development of generic libraries for redundancy based architectural configurations in a probabilistic model checking framework.

(c) Performance Analysis of a Wide Area Network Model using Opnet.

(d) Implementation of a HTTP/1.1 server using Winsock libraries.

(e) Implementation of a peer-peer resource discovery application based on IP Multicast using WINSOCK libraries.

(f) Automated conversion of arbitrary MTG (Multi threaded Graph) models to SystemC models.

(g) Implementation of real time scheduling algorithm like RMA, EDF and DASA in the ucLinux kernel.

(h) Implementation of a real time Air traffic control system using POSIX system calls.

(i) Implementation of an Online Computer Adaptive testing agent using JDBC, JSP and ASP.