Reliability Analysis of Fault-Tolerant Reconfigurable Nano-Architectures

(Categories: High Level Design Error Modeling, Fault Tolerance)

Debayan Bhaduri  
ISR-3  
Los Alamos National Laboratory  
Los Alamos, NM  
Email: dbhaduri@lanl.gov

Paul Graham  
ISR-3  
Los Alamos National Laboratory  
Los Alamos, NM  
Email: grahamp@lanl.gov

Sandeep K. Shukla  
Fermat Lab  
Virginia Tech  
Blacksburg, VA 24061  
Email: shukla@vt.edu

Abstract—Manufacturing defects and transient errors will be abundant in high-density reconfigurable nano-scale designs. Recently, we have automated a computational scheme based on Markov Random Field (MRF) and Belief Propagation algorithms in a tool named NANOLAB to evaluate the reliability of nano architectures. In this paper, we show how our methodology can be exploited to design defect- and fault-tolerant programmable logic architectures. The effectiveness of such automation is illustrated by analyzing reconfigurable Boolean networks formed using different industry-based configurable logic blocks (CLBs), both in the presence of thermal perturbations and signal noise.1

Index Terms—CLB, FPGA, Reconfigurable, Gibbs distribution, Nanotechnology, TMR, CTMR, Reliability, Entropy, interconnect, noise, modeling

I. INTRODUCTION

New technologies for building nanometer-scale devices are expected to provide the means for constructing much denser logic and thinner wires. But the economical 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. Thus, designing reliable system architectures that can work around these problems at run-time becomes important.

With the advent of nanotechnology, it is desirable that digital systems exhibit dynamic defect-tolerant attributes such as self-healing, self-replication and self-diagnosis. It has been shown in [10], [13] that reconfigurable FPGA-based architectures may mitigate both manufacturing and transient defects rampant at the nano-substrates. Further, because of their regularity and simplicity, reconfigurable logic architectures may be some of the earliest programmable architectures implemented using nano-scale technology. [11] analyzes the NAND multiplexing [15] and reconfiguration fault tolerant techniques, and presents a defect- and fault-tolerant architecture in which multiplexing (with a low degree of redundancy) is combined with a massively reconfigurable architecture. This points out the fact that different degrees of redundancy need to be applied at different granularity levels (such as gate level, configurable logic block (CLB) level, [6] etc.) to make systems reliable cost-effectively. As a result, in-depth analysis is required to find suitable redundancy levels for specific reliability measures of such reconfigurable architectural configurations.

II. BACKGROUND

In this section, we briefly discuss the probabilistic MRF-based methodology, certain specific FPGAs and a loopy Belief Propagation algorithm.

MRF-Based Methodology: The basis for the approach in [2] is based on Markov Random Fields. An MRF is defined as a finite set of random variables, \( \Lambda = \{ \lambda_1, \lambda_2, \ldots, \lambda_k \} \). Each variable \( \lambda_i \) has a neighborhood, \( N_i \), which has variables from \( \{ \Lambda - \lambda_i \} \). The probability distribution of a given variable depends only on a (typically small) neighborhood of other variables that is called a clique. Due to the Hammersley-Clifford theorem [4],

\[
P(\lambda_i | \{ \Lambda - \lambda_i \} ) = \frac{1}{Z} e^{\sum_{C \in \Lambda} U_c(\lambda)}
\]

The conditional probability in equation 1 is Gibbs distribution. \( Z \) is the normalizing constant and for a given node \( i \), \( C \) is the set of cliques. \( U_c \) is the clique energy function [2] and depends only on the neighborhood of the node whose energy state probability is being calculated. The logic margins of 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. Higher logic margins result in better reliability of computation. This formulation also allows correct analysis of entropy values, since the entropy in the system is proportional to the logic margin. Thus, this methodology 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] where $K$ is the thermal energy ($K$ is the Boltzmann constant and $T$ is the temperature in Kelvin) and $\ln 2$ is expressed in normalized units relative to the logic energy (clique energy). The thermodynamic limit of computation is the thermal energy that is proportional to the minimum entropy loss due to irreversible computation. If we consider energy levels close to these thermal limits, the reliability of computation is likely to be affected. The model of computation in [2] considers thermal perturbations, discrete errors and continuous signal noise [7] 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.

Defect- and Fault-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 (often referred to as CLBs) and interconnects, and these can be programmed, or configured, to implement any circuit. Defect-tolerance can be achieved in FPGA-based architectures by detecting faulty components during testing and excluding them during reconfiguration. It is expected [13] 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 testing can be done with massive parallelism, drastically reducing test time. While such reconfigurable architectures may aid in circumventing manufacturing defects at the nano-scale, architectures such as the Cell Matrix [10] have been proposed to support 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). The Cell Matrix can handle large manufacturing defect rates (permanent faults), and provide defect-tolerance to transient errors due to the inherent self-analyzing, self-modifying, and dynamic local processing capabilities of the architectural configuration.

In this study, we are analyzing some simple configurable logic block structures for their defect and fault tolerance in nano-scale implementations. The specific two CLBs analyzed for this study are based on logic blocks found commercial Actel FPGAs, as shown in Figure 1. The C-cell in Figure 1(a) is from the Axcelerator (AX) anti-fuse FPGA family while the core tile shown in Figure 1(b) is from the ProAsic PLUS flash FPGA family (see [1] for more information on these FPGAs). Our analysis of these architectures are not concerned with a specific memory technology for holding the programming data but, rather, how the basic logic cells can be made more reliable for nano-scale implementation using redundancy at various architectural levels.

Loopy Belief Propagation: The computation of posterior marginals on nodes in an arbitrary Bayesian or Markov Random network is a NP-hard problem [9]. Different Belief Propagation algorithms and approximation schemes have
been proposed in the past, and attempts have been made to categorize different types of networks for which each algorithm works best. In this work, we have implemented a technique called Unwrapped tree [8] to analyze loops in the sequential portions of the ProAsicPLUS core logic tile (shown in Figure 1(b)). The computation of the marginal probabilities is similar to the Pearl polytree algorithm [14]. To illustrate the technique, let us walk through an example. Figure 2 shows a network G that forms an undirected cycle composed of four nodes \{1, 2, 3, 4\}. To analyze the marginal probability values at each of these nodes, G is translated to T which is the corresponding unwrapped tree. T is an acyclic graph that is locally equivalent to the original graph, G.

The unwrapping technique is as follows: choose an arbitrary node r and initialize T = r (in Figure 2 node 1 is chosen). For each leaf node m of T, find the neighbors of the corresponding node in G, other than the parent of m in T. Add these nodes to the tree. The probability distributions at the nodes of both G and T remain same. The probability values at the leaves of T are compared. When they become equivalent (approximation thresholds may be defined), the beliefs are said to have converged. The cycle is unwrapped till convergence is achieved, but it has also been observed in existing literature [9] and this work that when loops are present in networks, messages may circulate indefinitely around the loops and the process may not converge to a stable equilibrium. Thus, a restriction may be imposed on the depth of the tree T such that the algorithm does not form an infinite chain for the analysis.

![Unwrapped Tree - Technique for Analyzing Loopy Belief Propagation](image)

**III. NANOLAB AND OUR METHODOLOGY**

**NANOLAB** [5] is a MATLAB based reliability evaluation tool, that uses entropy as the reliability metric. Our tool automates the MRF-based methodology briefly discussed in Section II. It consists of a library of functions and a Belief Propagation algorithm [12] that can compute energy distribution and entropy at the primary/intermediate outputs and interconnects of arbitrary Boolean networks, given discrete or continuous (signal noise) energy distributions at the primary inputs and interconnects of the circuits. These functions work for any generic one-, two- and three- input logic gates and can be extended to handle n-input logic gates, and take in as inputs the logic compatibility function (similar to truth table) [2] and the initial energy distribution for the inputs of a gate. Energy distributions are returned as vectors by these functions and indicate the probability of the output of a gate 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. NANOLAB also consists of functions that can model noise either as an uniform or Gaussian distributions or combinations of these, depending on the user specifications. Arbitrary Boolean networks in any redundancy-based fault-tolerant architectural configuration can be analyzed by writing simple MATLAB scripts that use these NANOLAB library functions.

We have enhanced the capabilities of our tool by developing libraries for the core CLBs of the Actel FPGAs (Figure 1). The dynamic programming of these core logic blocks are controlled by configuration files that are given as inputs to these libraries along with energy distributions at the inputs of these CLBs. Errors can also be introduced in these configuration files either interactively or by using some specific error distribution. Entropy values and energy distributions at the outputs of the CLBs are returned by the libraries. For implementing the sequential part of the core logic tile shown in Figure 1(b), we have used the Unwrapped tree loopy Belief Propagation algorithm discussed in Section II. But due to external posterior marginal probabilities (dependencies) in the loop of the logic tile, the Belief Propagation algorithm sometimes does not converge within the threshold number of iterations. We are looking at approximation schemes to solve this problem. We have also used loop unrolling to implement the same sequential portion of the logic block, and this technique seems to work better but with lesser degree of probabilistic accuracy.

With such a framework, we expect to analyze different reliability-redundancy trade-off points at different levels of granularity [6] such as at gate level, CLB level or application level. As our tool can handle sequential circuits, we can analyze complex systems with NANOLAB. Both thermal perturbations and signal noise can be introduced in the FPGA-based models, and different hardware redundancy based techniques may be adopted within the reconfigurable architectural framework.

**IV. EXPERIMENTS AND RESULTS**

We have conducted some experiments with the CLBs shown in Figure 1. The entropy values and logic margins of some defect-tolerant reconfigurable architectural configurations are observed and these determine interesting facts. Note that these CLBs are examples of reconfigurable core logic, and eventually we plan to support more of these.

**Reliability and Entropy Measures of Axcelerator CLB:**

Figure 3 (a) indicates the entropy values when the C-cell is configured to perform a two-input OR function. The entropy values are plotted till the 4th order CTMR for different KT values. It can be observed that as redundancy is increased by adding more CTMR orders, the entropy decreases (logic margin and reliability increases) at lower
NANOLAB by augmenting capabilities to analyze reliability. Reliability can no longer be improved if the system’s reliability for a given configuration reaches a steady state. Any further increase in redundancy may either marginally improve the reliability or even worsen it.

Energy distributions at the Output of the ProAsic CLB: NANOLAB can also be used to compute the probability of different energy configurations at the primary outputs of a Boolean network. Reliability measures of logic circuits can also be analyzed from these probability distributions. Figure 3 (b) and (c) show the energy distributions at the outputs of a TMR and a 6th order CTMR configuration applied to the ProAsic core logic tile respectively. Note that the probability values are based on bin sizes of 0.1. It can be seen that the logic margins for the output (z) at KT values of 0.1, 0.25 and 0.5 are higher for the higher CTMR orders. Also, the probability of z (p(z)) being at logic low is higher than being at one because of the configuration and the input distribution for the CLB. It is also observed that at a KT value of one, the logic margin for any CTMR configuration becomes really small (output energy distribution becomes almost uniform), and remains the same even with an increase of redundancy resulting in unreliable computation. Comparing these different orders of CTMR in Figure 3, 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 further experimental results show that 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.

In summary, here we show how we enhance our tool NANOLAB by augmenting capabilities to analyze reliability-trade-offs of reconfigurable FPGA-based architectural configurations. Also, NANOLAB can be used to model sequential circuits due to the implementation of a loopy Belief Propagation algorithm. This makes our tool more effective in analyzing reliability measures of different reconfigurable Boolean networks.

REFERENCES