2006

Reliability-centric probabilistic analysis of VLSI circuits

Thara Rejimon
University of South Florida

Reliability-centric Probabilistic Analysis of VLSI Circuits

by

Thara Rejimon

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Date of Approval:
July 5, 2006

Keywords: Single-Event-Upsets, Soft Errors, Dynamic Errors, Error Modeling, Bayesian Networks

DEDICATION

To God Almighty

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my major professor, Dr. Sanjukta Bhanja, for her continuous guidance and support throughout my doctoral degree program. Her encouragement and moral support was a strong motivating factor for the successful completion of my PhD.

I would also like to thank the members of my PhD committee, Dr. Murali Varanasi, Dr. Srinivas Katkoori, Dr. Wilfrido Moreno and Dr. Manish Agrawal for their timely help and suggestions which improved the quality my dissertation significantly.

My sincere thanks to Dr. Sudeep Sarkar for his valuable suggestions at different stages of this research. I would also like to express my thanks to Dr. Ranganathan for his suggestions and help.

Thanks to all members (present and past) of VLSI design lab in Electrical Engineering for their help and cooperation.

I am thankful to my loving parents whose continuous prayers helped me to complete this challenging task. Thanks to my husband Reji whose constant encouragement was a tremendous help throughout the course of my studies. Special thanks to his parents for their support. Thank you Juvan for your love and patience.

Above all, I thank GOD for His abundant Blessings.

TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT vi

CHAPTER 1 INTRODUCTION 1
- 1.1 Permanent Faults 4
- 1.2 Single-Event-Transients 5
- 1.3 Dynamic Errors 6
- 1.4 Central Modeling Theme 8
  - 1.4.1 Permanent Fault Modeling 9
  - 1.4.2 Single-Event-Transient Modeling 9
  - 1.4.3 Dynamic Error Modeling 11
- 1.5 Bayesian Network Representation of Fault/Error Detection Logic 12
  - 1.5.1 The Bayesian Network Models 13
- 1.6 Bayesian Inference 15
- 1.7 Important Results 15
- 1.8 Contribution of this Dissertation 17
- 1.9 Organization 18

CHAPTER 2 BACKGROUND 19
- 2.1 Fault Detection Probability 19
- 2.2 Transient Faults: Single-Event-Upsets 21
- 2.3 Dynamic Errors 25

CHAPTER 3 MODELING BASED ON BAYESIAN NETWORKS 29
- 3.1 Bayesian Inference 30
- 3.2 Cluster Based Inference 31
- 3.3 Stochastic Inference 35
  - 3.3.1 Probabilistic Logic Sampling (PLS) 35
  - 3.3.2 Evidence Pre-Propagated Importance Sampling 36

CHAPTER 4 LIFE-DAG: AN ACCURATE PROBABILISTIC MODEL FOR ERROR DETECTION PROBABILITIES 39
- 4.1 Motivation 40
- 4.2 LIFE-BN: Fault/Error Model 41
- 4.3 Experimental Results 44

CHAPTER 5 A TIMING-AWARE PROBABILISTIC MODEL FOR SEU SENSITIVITIES 49
5.1 The Proposed Model 53
5.1.1 Timing Issues 53
5.1.2 Delay Modeling Based on Logical Effort 58
5.1.3 TALI: Timing-Aware-Logic-Induced SEU Sensitivity Model 59
5.2 Experimental Results 63
5.2.1 Exact Inference 65
5.2.1.1 Input Space Characterization 65
5.2.2 Larger Benchmarks 66
5.2.3 Results with Delay Model Based on Logical Effort 69

CHAPTER 6 PROBABILISTIC ERROR MODELING OF NANO-DOMAIN LOGIC CIRCUITS 73
6.1 Probabilistic Error Model 75
6.1.1 The Bayesian Network Structure 76
6.1.2 Bayesian Network Quantification 79
6.2 Experimental Results 79

CHAPTER 7 CONCLUSION AND FUTURE WORK 92

REFERENCES 94

ABOUT THE AUTHOR End Page

LIST OF TABLES

Table 1.1. Probabilistic representation of the “truth-table” of a (a) two input error-free AND gate and (b) an AND gate with dynamic error probability p 7

Table 4.1. ISCAS benchmarks and the number of faults 45

Table 4.2. (a) Fault detection probability estimation errors and time for 1000 samples (b) Fault detection probability estimation errors and time for 3000 samples 46

Table 4.3. Comparison with the state of the art 46

Table 5.1. Gate delays based on logical effort 58

Table 5.2. Size of original and time-expanded ISCAS circuits for fanout-dependent delay model 64

Table 5.3. Estimated $P(T_{j\_i})$ values of nodes in benchmark c17 from exact inference 64

Table 5.4. SEU sensitivity estimation errors and time for 9999 samples 68

Table 5.5. Size of TALI model and estimation time for logical-effort based delay model 70

Table 6.1. Probabilistic representation of the “truth-table” of a two input, error-free, AND gate and for an AND gate with dynamic errors 73

Table 6.2. Output error probabilities [from exact inference] 80

Table 6.3. Comparison of modeling using Bayesian networks and probabilistic transfer matrix 81

Table 6.4. Output error probabilities for ISCAS’85 circuits -PLS with 1000 samples 82

Table 6.5. Comparison of Bayesian network modeling and logic simulation 82

Table 6.6. Error sensitivity for c17 84

Table 6.7. c17-Selective redundancy 90

Table 6.8. c432-Selective redundancy 90

LIST OF FIGURES

Figure 1.1.	Three major classes of errors/faults in logic circuits	4
Figure 1.2.	Stuck-at-fault model	9
Figure 1.3.	SET sensitivity model	10
Figure 1.4.	Dynamic error model	11
Figure 1.5.	(a) Conditional probability table of an AND gate (b) A small Bayesian network	12
Figure 2.1.	Related works on FDP analysis	20
Figure 2.2.	SEU propagation	22
Figure 2.3.	Recent works on SET modeling	23
Figure 2.4.	Background works on dynamic errors	26
Figure 2.5.	Modeling capacities for different probabilistic dependency models	27
Figure 3.1.	A small Bayesian network	29
Figure 3.2.	(a) The benchmark circuit c17 (b) Bayesian network representation of c17	32
Figure 3.3.	(a) Moral graph (b) chordal graph	32
Figure 3.4.	Junction tree	33
Figure 4.1.	(a) An illustrative fault detection logic (b) LIFE-DAG model of (a)	41
Figure 4.2.	(a) Estimation time vs. number of samples in the detection logic for benchmark c880 (b) Estimation time vs. number of nodes in the detection logic for benchmark c880	47
Figure 4.3.	Detection probability as SET sensitivity for different nodes in c880	48
Figure 5.1.	SEU Propagation	50
Figure 5.2.	Time-space transformed circuit of benchmark c17, modeling all SEUs	54
Figure 5.3.	Modified time-space transformed circuit of benchmark c17, modeling only the possibly sensitized SEUs	57

Figure 5.4. Time-space transformed circuit of benchmark c17 with logical effort based delay model 60

Figure 5.5. (a) An illustrative SEU sensitivity logic for a subset of c17 (b) Timing-aware-logic-induced-DAG model of the SEU sensitivity logic in (a) 61

Figure 5.6. Input probabilities for achieving zero output errors (at nodes 22 *and* 23 in presence of SEU's: (a) SEU^0 at node 19 (b) SEU^1 at node 19 (c) SEU^0 at node 11 (d) SEU^1 at node 11 for c17 benchmark 67

Figure 5.7. (a) SEU list-fanout dependent delay model (b) SEU sensitivity range-fanout dependent delay model, with delta=1; input bias=0.5 68

Figure 5.8. (a) SEU list-logical effort delay model (b) SEU sensitivity range-logical effort delay model with delta =1 and input bias = 0.5 70

Figure 6.1. (a) Conceptual circuit representation of the logic used to detect errors involving the error-free logic and the unreliable logic components (b) The corresponding Bayesian network representation 76

Figure 6.2. (a) Dynamic error simulation circuit (b) Truth table of a NAND gate in the erroneous logic block 83

Figure 6.3. (a) Sensitivity of output error probability with respect to individual gate errors for c880 (b) Output error profiles of two alternative logic implementation (c499 and c1355) 84

Figure 6.4. c17 - Input space characterization by likelihood ratio for (a) zero error at output node 22 (b) for zero error at output node 23 85

Figure 6.5. Input space characterization by likelihood ratio for benchmarks (a) c432 (b) c1908 87

RELIABILITY-CENTRIC PROBABILISTIC ANALYSIS OF VLSI CIRCUITS

Thara Rejimon

ABSTRACT

Reliability is one of the most serious issues confronted by microelectronics industry as feature sizes scale down from deep submicron to sub-100-nanometer and nanometer regime. Due to processing defects and increased noise effects, it is almost impractical to come up with error-free circuits. As we move beyond 22nm, devices will be operating very close to their thermal limit making the gates error-prone and every gate will have a finite propensity of providing erroneous outputs. Additional factors increasing the erroneous behaviors are low operating voltages and extremely high frequencies. These types of errors are not captured by current defect and fault tolerant mechanisms as they might not be present during the testing and reconfiguration. Hence Reliability-centric CAD analysis tool is becoming more essential not only to combat defect and hard faults but also errors that are transient and probabilistic in nature.

In this dissertation, we address three broad categories of errors. *First*, we focus on random pattern testability of logic circuits with respect to hard or permanent faults. *Second*, we model the effect of single-event-upset (SEU) at an internal node to primary outputs. We capture the *temporal* nature of SEUs by adding timing information to our model. *Finally*, we model the dynamic error in nano-domain computing, where reliable computation has to be achieved with ”systemic” unreliable devices, thus making the entire computation process probabilistic rather than deterministic in nature.

Our central theoretical scheme relies on Bayesian Belief networks that are compact efficient models representing joint probability distribution in a minimal graphical structure that not only uses conditional independencies to model the underlying probabilistic dependence but also uses them for computational advantage. We used both exact and approximate inference which has let us achieve order of magnitude improvements in both accuracy and speed and have enabled us to study larger benchmarks than the state-of-the-art. We are also able to study error sensitivities, explore design space, and characterize the input space with respect to errors and finally, evaluate the effect of redundancy schemes.

CHAPTER 1

INTRODUCTION

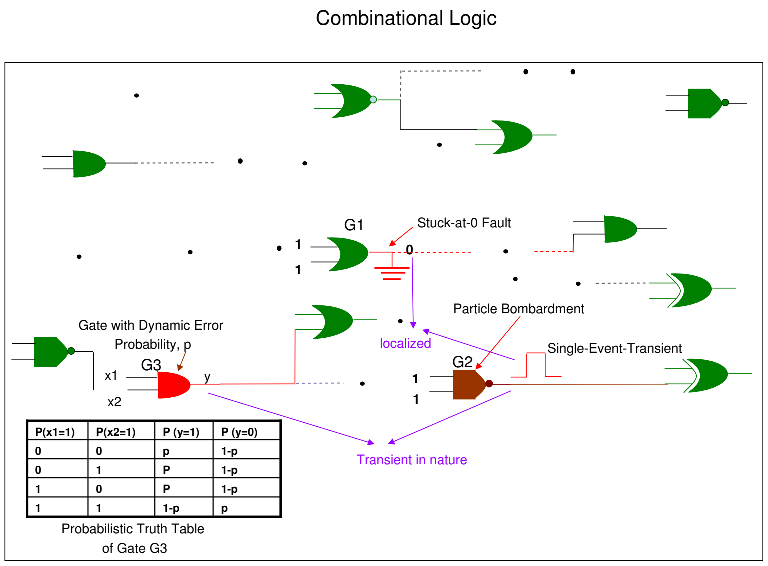
Technology scaling and increased power densities are the major factors affecting reliability of future high performance microprocessors. Reduction in device sizes comes with shrinking of channel length and dielectric thicknesses which increases current density and hence the operating temperature. Transistors in deep submicron and future nano-meter technologies will also have high leakage currents which exponentially rise with temperature. Since devices will operate near thermal limits, the *intrinsic* error rates will increase resulting less reliable circuit operation. The fact that supply voltage and threshold voltage are not scaling appropriately with the scaling of device sizes, results in increased power density which causes further reliability issues. As a result, according to the International Technology Roadmap for Semiconductors (ITRS) [1] reliability of nano-meter circuit is greatly affected due to more transient and permanent failures of signals, logic values, devices and interconnects. Signal integrity issues such as crosstalk and power grid noise will be intensified by manufacturing defects, process variations and temperature fluctuations and hence it is becoming harder to achieve the desired level of noise immunity while maintaining the improvement trends in performance and energy-efficiency [2, 3]. Multiple noise sources and operational variations can dynamically interact with each other to further aggravate error effects [4]. With reduced feature sizes, smaller supply voltages and higher transistor densities, digital logic gates become inherently noisy due to the above mentioned compound noise effects, which we refer to as ”dynamic errors”. The problem of realizing reliable Boolean functions using noisy gates was first introduced by Von Neumann [5] in 1956. Error bounds for achieving reliable computation have been estimated in [5] and [6].

Another major reliability challenge is soft error tolerance. Soft errors result from Single-Event-Upsets caused by radiation induced particle bombardment. High energy neutrons and alpha particles present in atmospheric radiations when bombarded on the active semiconductor regions of CMOS circuits, electron-hole pairs are generated resulting in transients known as single-event-upsets at the gate outputs. In fact, the existence of radiation effects on microchips have been known and addressed since 1970's. Space programs reported on-orbit occurrence of single vent upsets in 70's; IBM reported the use of error correction codes [ECC] in mid-60's; use of ECC to mask the occurrence of SER in SDRAM was the focus in the 1980's [7]. Reduction of soft error rates (SER) in memories was an important research focus in the 1990's. Combinational logic circuits have inherent tendency to mask the effect of single-event-transients, due to three masking phenomena, namely, (1) logical (2) electrical and (3) latching window masking. Due to logical masking, a generated glitch might not propagate to the circuit output. Due to electrical masking, a generated glitch might get attenuated before reaching a primary output and due to latching window masking, a glitch might not be captured by the latch if it reaches the latch input outside the latching window. However, with reduction in circuit size, increased device density and reduction in supply voltages, critical charge per node will reduce which will reduce electrical masking effect [8]. Again due to decreasing number of gates in a pipelined stage, logical masking as well as electrical masking effects are decreasing in future technologies [9]. Latching window masking effect is also reducing because increasing operating frequencies reduce the time period in which latches do not accept data. This will result in unacceptable soft error rates in future technologies. Hence it is important to understand the occurrence and detection of these transients and explore effective mitigation schemes not only for memories but for logic circuits as well.

Due to processing defects, compound noise effects and increased soft error rates, it is almost impractical to come up with error-free circuits in sub-100-nanometer and nanometer technologies. Hence the best feasible solution to this problem is to incorporate efficient error correction schemes in future high performance microprocessors operating under sub-100-nanometer and nanometer technologies. In order to achieve reliable circuit operation, it is important to investigate various types of errors pre-dominant in logic networks with the scaling down of technology and to implement circuit designs with appropriate error mitigation schemes. Application of redundancy is one of the most commonly used error mitigation technique. However, applying redundancy to all gates in a circuit will result in very high area overhead and excess power dissipation in addition to increased cost. Hence it is essential to identify gates that are highly sensitive to errors and apply selective redundancy measures to achieve trade-off between redundancy, reliability, area overhead and cost.

In this dissertation, we investigate three major classes of errors using probabilistic graphical representation for various error/fault models that are relevant in present and future nano-domain CMOS circuits. Specifically, we look into three broad categories of faults/errors: (1) permanent hard faults, (2) single-event-transients - errors caused by radiation and particle bombardment and (3) dynamic errors - errors due to random noise which are predominant in nano-scale computing. Fig. 1.1. depicts the above three types of errors/faults that we analyze in this dissertation. If the output line of gate G1 is shorted to ground due to a fabrication error, it results in a stuck-at-0 fault, which is a permanent and localized fault. A particle bombardment on the active semiconductor area of a transistor within the gate G2 can cause an unwanted 0_1_0 transition resulting in a single-event upset as shown in Fig. 1.1. Note that this will cause a logical error at the gate output only if both inputs of the nand gate G2 are at logic 1. Hence the generation and propagation of single event upsets through a logic gate depends on the gate type and its input signal values. Like stuck-at-faults, single event upsets are also localized events since its occurrence depends on the location of the particle hit. The third type of error - the dynamic error - is illustrated by a signal error at the output line of gate G3 which is assumed to have a dynamic error probability p due to compound noise effects caused by technology scaling factors as discussed before. This type of error can be modeled by means of probabilistic truth tables that capture the gate error probabilities. The truth table of an erroneous *and* gate is shown in figure 1.1. Like single event upsets, dynamic errors are also transient in nature since they are caused by temporary device mal-functions, but are not permanent failures. However, unlike stuck-at-faults and single-event-upsets, they are not localized events. *Every* gate in a logic circuit has a finite error probability.

The fundamental theme of these error modeling is dependency-preserving probabilistic structure. The analysis of these errors are captured in (1) fault/error detection probability, (2) SEU sensitivity and (3) overall output error estimation of logic circuits having unreliable components. We also classify gates in terms of error sensitivity and analyze the effect of selective redundancy application on circuit reliability.



P(x1=1)	P(x2=1)	P (y=1)	P (y=0)
0	0	p	1-p
0	1	P	1-p
1	0	P	1-p
1	1	1-p	p

Figure 1.1. Three major classes of errors/faults in logic circuits

1.1 Permanent Faults

Some of the major causes of permanent faults/defects in VLSI chips are processing defects, material defects and packaging failures. Permanent faults can be classified into stuck-at-faults, transistor open and short faults, bridging faults, functional faults, delay faults, etc. Most of these faults can be covered by single-stuck-at fault tests [10]. We model stuck-at faults in logic circuits for measuring random pattern testability.

Fault Detection Probability (FDP) is an important testability measure that is useful for not only generating test patterns, but also to vet designs for random-input testability. *Fault Detection Probability (FDP) of a stuck-at fault $f \in F$, where F denotes the set of all faults, in a combinational circuit, C, is the probability that f is detected by a randomly chosen equally-likely input patterns*. Signal probabilities, as well as FDP, are affected by spatial correlations induced by circuit re-convergence.

Most of the analysis of FDP was performed in 80's (as proposed by Seth *et al.* [11], Wunderlich [12], etc.) Due to the high computational complexity involved in computing signal and fault detection probabilities they resort to several approximation strategies. In this work, we re-visit this old problem for the following two reasons: (1) State-of-the-art algorithm for computation of *exact* fault detection probabilities, based on Binary Decision Diagrams [13] do not scale well, in terms of time and space requirements, with circuit size. They usually require various heuristic-based approximations of the pure model. In a later development (1988), Bayesian network was introduced for probabilistic reasoning and belief propagation and it has been applied in artificial intelligence and image analysis. We propose an *exact* model based on Bayesian networks, for estimation of FDP and use smart and efficient stochastic inference schemes which has excellent accuracy-time trade-off. (2)Traditionally, FDP has been used for test point insertions, however, it can also be used as an upper bound of random single-event-transient (SET) sensitivity, which is important for characterization of impact of soft errors in logic blocks. Thus we provide a fresher look into an old problem by adopting a novel and efficient scheme.

1.2 Single-Event-Transients

High-energy neutrons present in cosmic radiations and alpha particles from packaging materials give rise to single event upsets (SEUs) resulting in soft errors in logic circuits. When particles hit a semiconductor material, electron-hole pairs are generated, which may be collected by a P-N junction, resulting in a short current pulse that causes logic upset or Single Event Upset (SEU) in the signal value. An SEU may occur in an internal node of a combinational circuit and propagate to an output latch. When a latch captures the SEU, it may cause a bit flip, which can alter the state of the system resulting in a soft error. So far, soft errors are of serious concern in memories, whereas in logic circuits soft error rate is comparatively low due to logical, electrical and temporal masking effects. However, as process technology scales below 100 nanometers and operating frequencies increase, the above masking barriers diminish due to low supply voltages, shrinking device geometry and small noise margin. This will result in unacceptable soft error failure rates in logic circuits even for mainstream applications [14].

Soft Error Sensitivity of a node in a logic block depends on the following four factors: (1) the particle bombardment rate on a chip which is fairly uniform in space and time. (2) Electrical masking probability: probability that a particle hit at the node causes a single-event-transient of a finite duration (it depends on V_{dd}, V_{th} and also on temperature), (3) the SET sensitization probability: the probability of an SET generated an the node is propagated to an output node and (4) latching probability which is a function of latch characteristics and the switching frequency.

In this work, we explore SET sensitivity of individual gates in a circuit by accurately considering the effect of (1) *SEU duration*, (2) *effect of gate delays*, (3) *re-convergence* in the circuit structure and most importantly (4) *inputs*. Several works on soft error analysis estimate the overall output signal errors due to SEUs at the internal nodes [4, 15, 16, 17, 18] . Note that our focus is to identify the SEU locations that cause soft errors at the output(s) with high probabilities and *not* on the overall soft error rates. Knowledge of relative contribution of individual nodes to output error will help designers to apply selective radiation hardening techniques. This model can easily be fused with the modeling of the latches [17, 19] considering parameters such as latching window, setup, hold time, V_{th} and V_{dd} [15, 16, 17] for a comprehensive model capturing processing, electrical and logical effect.

1.3 Dynamic Errors

The ITRS road-map predicts CMOS device dimensions to reach close to the design limit of 50 nm by 2020. Circuits built with such nano-dimensional devices will face design challenges that have not been much of an issue so far. One such challenge involve dynamic errors in the interconnects and gates.

What is a dynamic error? These errors will be transient in nature, occurring anywhere in the circuit, but hard to detect by regular testing methodologies (since they are not permanent damages). They can be characterized only probabilistically. Each device (logic gate/interconnect) will have certain, non-zero, propensity for an output line error. These error probabilities could be as high as 10% [20]. Traditional error masking by extra logic might not be possible since these extra logic will itself be error-prone. *What complicates the picture is that this propensity for errors will, intrinsically, exist at each gate.* Hence, in future, reliable computation has to be achieved with ”systemic” unreliable devices [5]. Thus making the entire computation process probabilistic rather than deterministic in nature. For instance, given inputs 1 and 0, an AND gate will output the state 0, only with probability $1 - p$, where p is the gate error probability. Thus, traditional, deterministic, truth-table based logic representation will not suffice. Instead, the output needs to be specified in terms of probabilities, conditioned on the states of the inputs. Table 1.1. shows such as specification for (a) a 2-input error-free AND gate used in current CMOS designs and (b) 2-input AND gate with dynamic errors for next-generation technologies.

Table 1.1. Probabilistic representation of the "truth-table" of a (a) two input error-free AND gate and (b) an AND gate with dynamic error probability p

Ideal AND gate			
X_{i1}	X_{i2}	$P(X_o \mid X_{i1}, X_{i2})$	
		$X_o = 0$	$X_o = 1$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

(a)

AND gate with dynamic error			
X_{i1}	X_{i2}	$P(X_o \mid X_{i1}, X_{i2})$	
		$X_o = 0$	$X_o = 1$
0	0	1-p	p
0	1	1-p	p
1	0	1-p	p
1	1	p	1-p

(b)

There will be need for formalisms to compare, evaluate, and vet circuit designs with these dynamic error prone gates. Note that *these nano-domain dynamic errors are unlike the traditional permanent hard faults in CMOS circuits due to run-time device failure or manufacturing defects, nor is it similar to the conventional soft errors that can arise due to radiation and device noise, which are localized, purely random, and can be reduced by external means.*

The *sources of dynamic errors* can be different for various emerging technologies. The error probability p (see Table 1.1.) will be dependent on the technology. For instance,

1. In nano-CMOS, dynamic error will arise due to the use of ultra low voltage design, resulting in supply to ground bounce, leakage and coupling noise and due to small device geometry operating with only a handful of electrons.

2. In carbon nano-tube(CNT) [21, 22] and resonant tunnel diodes(RTD) [23], dynamic errors will arise due to their operating conditions near thermal limit. In fact, predictions also suggest that increased clock speed and increased computational requirements would make switching transition energy limits to a few order of magnitude of kT [20], where T is the temperature in Kelvin and k is the Boltzmann's constant.

3. In quantum-dot cellular automata (QCA) based circuits, errors can be classified as static (decay) errors, switching (thermal) errors, dynamic errors, and errors due to background charge fluctuations [24]. Decay errors occur when information stored in a latch is lost before the end of a clock cycle. This can happen when an electron locked at the top dot, tunnels out of it into the bottom dot or vice versa, when the latch is in its locked state. Switching errors occur when a gate switches

into the wrong logic state when clock is applied, because of external thermal excitation. Dynamic errors occur when clock frequency approaches the electron tunneling rate in the device. In addition to dynamic errors, permanent hard faults can occur in QCA designs due to cell displacement, cell misalignment and cell omission [25].

Error probability, p of a gate depends on many factors such as switching at that node, leakage, temperature, V_{th}, etc. For example gates with high switching may produce wrong signals when they operate near their thermal limits due to increased power dissipation. A device under dynamic error at any time instant may operate correctly after a short period. Thus probability of dynamic error in individual devices needs extensive characterization at quantum level and it has to be obtained from some industrial data. Given the p values of the gates, our model accurately estimates the overall output error probabilities. In this work, we assume that all gates have the same dynamic error probability, p, to compare our work with the state-of-the-art. However, it could be easily extended to a more device friendly diverse probabilistic model where each gate would have unique error probability.

1.4 Central Modeling Theme

We model the errors/faults in a logic circuit by comparing the outputs from the ideal logic and from a corresponding error/fault encoded logic. We compare the ideal outputs with the corresponding error/fault encoded outputs by passing these output signals through XOR gates. An XOR output of *one* indicates that error/fault is sensitized to the output. Hence probability of XOR output being equal to *one* gives the error/fault detection probability. Hence our model has three distinct units. (1) ideal logic block (2) error encoded logic block and (3) detection unit. The ideal logic block is the same for all of our fault/error models, whereas the error encoded blocks are unique for each model. The structure of the error encoded logic block depends on the nature of the error/fault and its effect on the primary outputs. The detection unit consists of a set of comparator nodes, the cardinality of which depends on the number of primary outputs and the number of sensitization paths from each fault/error location.

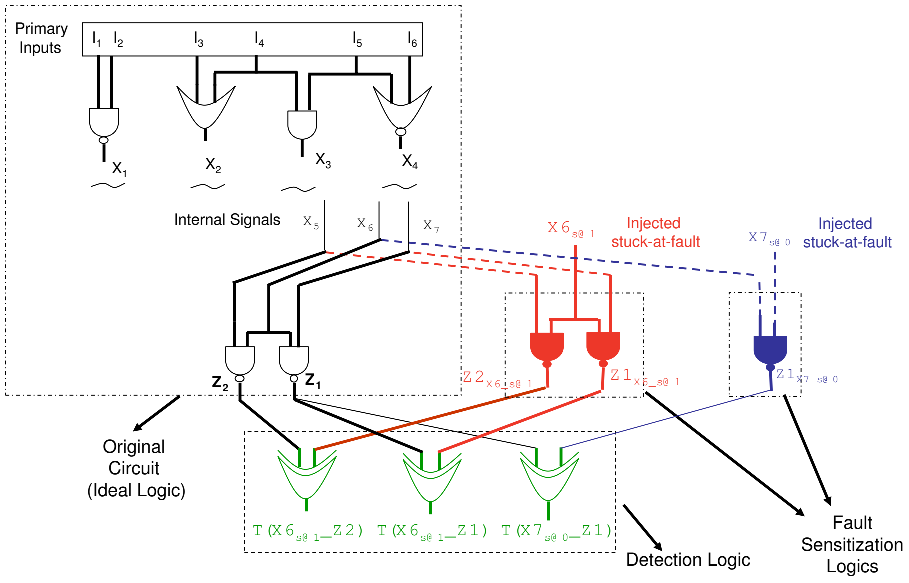


Figure 1.2. Stuck-at-fault model

1.4.1 Permanent Fault Modeling

For modeling the permanent stuck-at faults, we use partial duplication of the original circuit for the fault detection logic. This is shown in Fig. 1.2. Here we duplicate only the sensitized paths from the fault location to primary output(s). To model each fault, the fault detection logic will have one distinct fault encoded logic block. Hence given the circuit structure and the fault list, we are able to model *all* the faults in the fault list by means of a set of fault encoded logic blocks. To simulate the effect of stuck-at-faults, we use an additional input for each fault encoded logic block, that inject the respective fault at the fault location, thus making the faulty node permanently stuck at 1 or 0. We then propagate this fault through the fault encoded logic block until a primary output(s) is reached.

1.4.2 Single-Event-Transient Modeling

The major difference between permanent fault modeling and SEU modeling is in that for permanent fault modeling, we model logical masking effect whereas for SEU modeling we model both logical and *temporal* masking effect. Due to the temporal nature of SEUs, only those SEUs which reach the

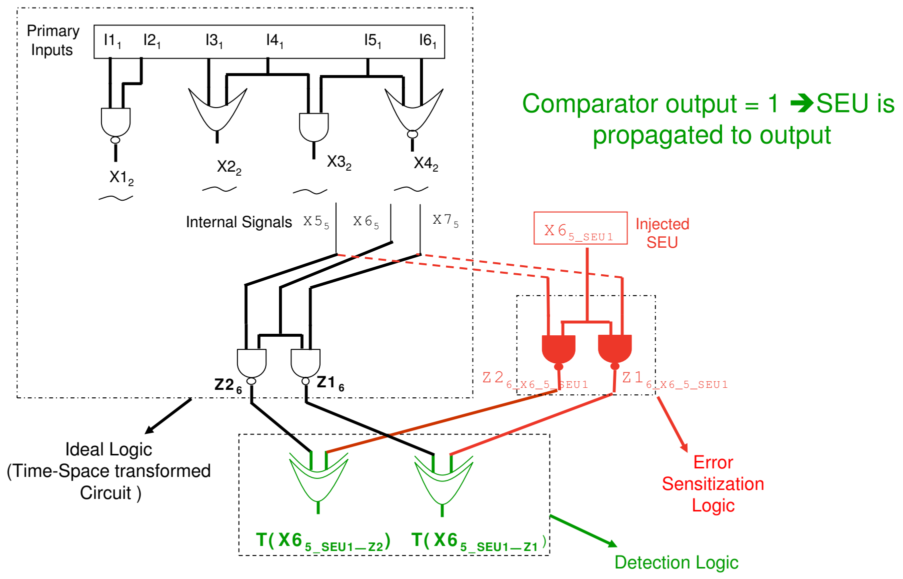

Primary Inputs
I1₁ I2₁ I3₁ I4₁ I5₁ I6₁
X1₂
X2₂
X3₂
X4₂
Internal Signals X5₅ X6₅ X7₅
Z2₆
Z1₆
Comparator output = 1 ➔SEU is propagated to output
X6₅_SEU1
Injected SEU
Z2₆_X6_5_SEU1
Z1₆_X6_5_SEU1
Ideal Logic (Time-Space transformed Circuit)
Error Sensitization Logic
T(X6₅_SEU1—Z2)
T(X6₅_SEU1—Z1)
Detection Logic

Figure 1.3. SET sensitivity model

primary output during the latching window are captured by the output latch, which can cause a bit-flip at the output and hence soft error. Hence to model single-event-transients, we need to take into account of the timing aspects such as gate delays, path delays and width of the SEU. We incorporate these timing features by performing a time-space transformation of the original circuit. This is done by replicating gates in the circuits several times depending on the time frames at which new gate output signals are evaluated. To model SEUs, we need to consider only those gates in the fan-in cones of the primary outputs that are evaluated during latching window. Hence from the expanded (time-space transformed) circuit, we derive a sub-circuit that propagates SEUs to output latches. From this circuit, we identify those SEUs that might be possibly sensitized to at least one primary output and derive a *reduced* SEU list. From the time-space transformed circuit and the reduced SEU list, we construct the SEU detection logic exactly in the same manner as the stuck-at-fault detection logic which is shown in Fig. 1.3. In this figure, we represent signals as subscripted variables, where the subscript denotes the time frame at which signals are evaluated. Effect of an SEU is modeled by injecting a faulty input signal. Each SEU has a unique error encoded logic block to propagate the SEU effect to primary output(s). Thus we model *all* SEUs in the reduced SEU list simultaneously by means of a single comprehensive model.

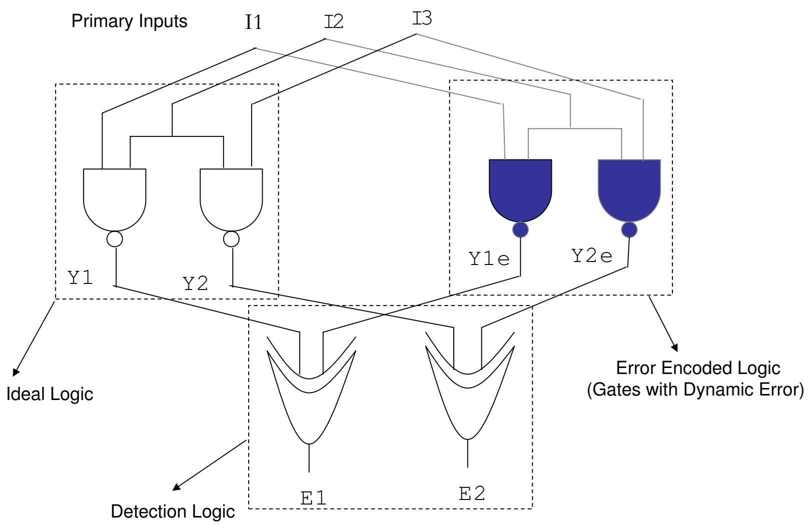
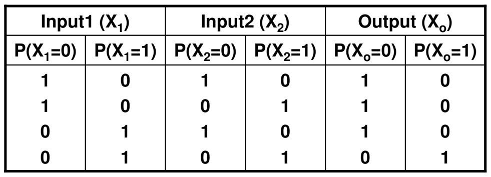


Figure 1.4. Dynamic error model

In this work, we use two types of gate delay models: (1) fanout dependent delay model where gate delays are assumed to be equal to its fanouts and (2) logical effort based delay model where gate delays depend not only on fan-out but also on input capacitance as well as parasitic capacitance.

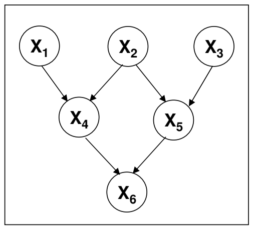
1.4.3 Dynamic Error Modeling

In nano-domain computing, gates in logic networks have an intrinsic propensity to operate erroneously. Hence gates are to be represented by *probabilistic* truth tables similar to that shown in Table 1.1.. In this work, we assume that dynamic error probability of of a gate is p. To model these errors, we duplicate the *whole* circuit. Gates in the ideal logic block are represented by the ideal truth table and the corresponding gates in the error encoded logic block are represented by respective probabilistic truth table. Both ideal and error encoded blocks are fed by the same primary inputs. As in the other two models discussed above, the detection logic compares output signals from the ideal block and the error encoded block. Fig. 1.4. shows the dynamic error detection logic. Dynamic error model is to be distinguished from the other two previously discussed models in that, here we duplicate the *whole* circuit to capture the overall effect of individual gate errors on the circuit output.



Input1 (X_1)		Input2 (X_2)		Output (X_o)	
$P(X_1=0)$	$P(X_1=1)$	$P(X_2=0)$	$P(X_2=1)$	$P(X_o=0)$	$P(X_o=1)$
1	0	1	0	1	0
1	0	0	1	1	0
0	1	1	0	1	0
0	1	0	1	0	1

(a)



$$\begin{tikzcd}
X_1 \arrow[dr] & & X_2 \arrow[dl] \arrow[dr] & & X_3 \arrow[dl] \\
& X_4 \arrow[dr] & & X_5 \arrow[dl] & \\
& & X_6 & &
\end{tikzcd}$$

(b)

Figure 1.5. (a) Conditional probability table of an AND gate (b) A small Bayesian network

1.5 Bayesian Network Representation of Fault/Error Detection Logic

Bayesian Networks [26] are graphical probabilistic models representing the joint probability function over a set of random variables. A Bayesian Network is a directed acyclic graphical structure (DAG), whose nodes describe random variables and node to node arcs denote direct causal dependencies. A directed link captures the direct cause and effect relationship between two random variables. Each node has a conditional probability table (CPT) except the root nodes. Each root node has a prior probability table. A logic circuit can be transformed into an equivalent Bayesian network by representing the state of a line (logic zero or one) with a node and the conditional probability table of each gate by directed edges that quantifies the conditional probability of the state of a node *given* the states of its parents or its direct causes. Figure 1.5.(a) gives the CPT of an AND gate derived from its truth table and Figure 1.5.(b) is a small Bayesian network structure. In a Bayesian network, the state of a node is independent of the states of all other nodes in the network, *given* the states of its parent nodes. This unique property of Bayesian network is termed as *conditional independence*. Exploiting this conditional independence among the variables in a Bayesian network, the complete joint probability distribution over the entire network can be simplified into a minimal factored representation.

The attractive feature of this graphical representation of the joint probability distribution is that not only does it make conditional independency relationships among the nodes explicit but it also serves as a computational mechanism for efficient probabilistic updating. Probabilistic Bayesian Networks can be used not only to infer effects due to known causes (predictive problem) but also to infer possible causes for known effects (the backtracking or diagnosis problem).

1.5.1 The Bayesian Network Models

Logic Induced Fault Encoded Directed Acyclic Graph (LIFE-DAG): We model single stuck-at-faults (errors) in large combinational circuits using a Logic Induced Fault Encoded Direct Acyclic Graph (LIFE-DAG) graph structure by transforming the fault detection logic into a Bayesian Network as discussed in the previous section. LIFE-DAG consists of nodes representing (1) primary input signals (2)injected faults (3) internal signals in the ideal logic (4) corresponding internal signals in the fault encoded logic (5)ideal output signals from the ideal logic (6) corresponding faulty output signals from the fault encoded logic and (7) fault detection signals which are outputs from detection logic. Note that only those nodes descending from the fault location are replicated in the fault encoded logic. Some of the nodes in the fault encoded logic derive all of their parent signals from the same fault encoded logic whereas some nodes have their parents both from the ideal circuit and the fault encoded logic block. Size of the LIFE-DAG structure is proportional to (1) the number of nodes in the ideal logic block, determined by the circuit size (2) number of *hard* faults, for which the detection probabilities are to be estimated (3) number of nodes in the fault encoded logic block, which depends on the length of sensitization paths from each fault location and (4) number of detection nodes (comparator nodes) cardinality of which is determined by the number of testable outputs from each fault location.

Timing-Aware Logic Induced SET Sensitivity Model (TALI-SES): We model Single Event Transients in large combinational circuits using a Timing aware Logic induced Soft Error Sensitivity model (TALI-SES), which is a complete joint probability model, represented as a Bayesian Network. This network is derived from the SET detection logic that we discussed previously. In TALI-SES, the graph structure corresponding to the ideal logic consists of signals in the time-space transformed circuit. Here nodes represent signals that carry timing information in addition to the signal location. Signal are represented in the form of composite random variables. For example (i,t) is a signal evaluated at the output of gate i at time t. Effect of each SEU in the SEU list is modeled by a sub-graph that contains nodes representing injected SEU, nodes representing signals descending from the SEU location. Signals in the error encoded logic are also in the form of composite random variables carrying both time and space information. Size of the TALI-SES Bayesian network structure is proportional to (1) the number of nodes in the ideal logic block, determined by the size of the time-space transformed circuit (2) number of SEUs in the SEU list, (3) number of nodes in the error encoded logic block, which depends on the length of sensitization paths from each SEU location to primary output(s) which are evaluated during the latching window and (4) number of detection nodes(comparator nodes) cardinality of which is determined by the number of testable outputs (outputs evaluated during the latching period) from each SEU location. Size of the time-space transformed circuit (ideal logic block) is determined by the number of gates in the fanin cones of circuit outputs evaluated during the latching window. Hence size of the ideal logic is determined by the gate and path delays and SEU duration. Assuming fanout dependent delay model and SEU duration of one time unit, size of the ideal logic (time-space transformed circuit) is double that of the original circuit, in the worst case. We explain this in detail in Chapter 5.

Logic Induced Probabilistic Error Model (LIPEM): We model the effect of dynamic errors in nano devices by estimating the overall error probability at the output of a logic block where individual logic elements are subject to error with a finite probability, p. We construct the overall BN representation based on logic level specifications by coupling gate level representations. Each gate is modeled using a conditional probability table (CPT) which models the probability of gate output signal being at a logic state, given its input signal states. An ideal logic gate with no dynamic error has its CPT derived from the gate truth table, whereas the CPT of a gate with error is derived from its truth table and the error probabilities, which can be input dependent. The overall joint probability model (LIPEM) is constructed by networking these individual gate CPTs. This model captures all signal dependencies in the circuit and is a minimal representation, which is important from a scalability perspective. In LIPEM, the original circuit is duplicated to model the effect of dynamic errors. Hence both ideal and error encoded logic have the same size. Size of the detection is determined by the number of primary outputs. Hence size of the LIPEM structure is a function of the circuit size and the number of primary outputs. From Fig. 1.4. it can be seen that the same primary inputs are going to both ideal and error encoded logic blocks and they are re-converging to the comparator gates in the detection unit. Hence the LIPEM DAG structure has more re-convergence compared to the other two models discussed before. Hence computational complexity associated with Bayesian inferencing in this case is higher than that of LIFE-DAG and TALI-SES. However, complexity of other two models deepens on the number of faults to be modeled and their location as well. Hence the overall computational complexity for each model will vary depending on the circuit structure, number of outputs, number of faults to be detected and their location.

1.6 Bayesian Inference

Bayesian inference is the process of updating signal probabilities based on the network structure and the knowledge of a few *evidence*. In predictive problems, the evidence is the set of input probabilities whereas in diagnostic problems, it can be the probabilities of any set of nodes in the network. In this work, we explore three inference schemes: (1) The exact inference scheme, which is also known as clustering technique [27], where the original DAG is transformed into special tree of cliques such that the total message passing between cliques will update the overall probability of the system. The computational complexity of the exact method is exponential in terms of number of variables in the largest clique. Since exact inference in expensive in terms of time, it is suitable for only small circuits. To handle larger circuits, we explore two stochastic sampling algorithms, namely (2) Probabilistic Logic Sampling (PLS) and (3) Evidence Pre-propagated Importance Sampling (EPIS) [28]. These algorithms are scalable, pattern insensitive, anytime method with excellent accuracy-time trade-off.

The space requirement of the Bayesian network representation is determined by the space required to store the conditional probability tables at each node. For a node with n_p parents, the size of the table is 2^{n_p+1}. The number of parents of a node is equal to the fan-in at the node. In the LIFE-BN model we break all fan-ins greater than 2 into a logically equivalent, hierarchical structure of 2 fan-in gates. For nodes with fan-in f, we would need $\lceil f/2 \rceil$ extra Bayesian network nodes, each requiring only 2^3 sized conditional probability table.

1.7 Important Results

FDP: We developed an accurate and efficient technique for estimation fault detection probabilities using LIFE-DAG which is an *exact* model, unlike several other existing techniques for FDP analysis. We outperform the state-of-the-art BDD based technique in both time and space complexity. Moreover, our model handles all benchmarks uniformly whereas the BDD based model require special heuristics and decomposition techniques to handle individual circuits.

SET Modeling and Analysis: We use both exact and approximate inference schemes for updating signal probabilities. For small circuits, we use the exact inference to predict the effect of inputs and SEU at a node on the outputs. we also exploit the unique backtracking (diagnostic) feature of Bayesian networks to answer queries like: "What input behavior will make SEU at node j definitely causing a bit-flip the at circuit outputs?" or "What input behavior will be more conducive to no error at output given that there is an SEU at node j?" For handling larger circuits, we use the approximate Bayesian inference scheme, namely, Probabilistic Logic Sampling (PLS) and compute the SEU sensitivities of individual gates in the circuit. We classify gates in the order of SEU sensitivity values. These results can be used for determining the gates to be chosen for selective redundancy application (or any other soft error hardening techniques). Note that our model is the *only non-simulative model* that captures logical and temporal masking effects. We compare our estimation results with logic simulation and find that our modeling technique is highly accurate and has less time and space complexity compared to simulative method.

Dynamic Error Modeling and Analysis: We use the exact inference scheme to compute the overall output error probabilities of small benchmarks and show that our model is extremely time and space efficient by comparing with the state-of-the-art technique which is a Probabilistic Transfer Matrix [PTM] based model. In the absence of medium sized benchmarks for nano-domain circuit, we use ISCAS'85 benchmarks to demonstrate the scalability and accuracy of our modeling scheme. The PTM technique can handle only small circuits, due to extremely high computational complexity, whereas ours is the *unique* probabilistic model that can handle mid-size circuits. For handling these larger circuits, we use two approximate inference schemes, Evidence Pre-propagated Importance Sampling (EPIS) and Probabilistic Logic Sampling (PLS). Our model can be used for ranking equivalent designs in terms of propensity of dynamic errors. We use the diagnostic aspects of Bayesian networks to characterize the input space which results in a desired output behavior even in the presence of dynamic errors. Finally, our modeling tool can be used for choosing the gates to be selected for application of error-tolerant techniques based on their dynamic error sensitivity and also for determining the amount of such techniques required for achieving a desired error tolerance.

1.8 Contribution of this Dissertation

The major contributions of this research can be summarized as follows:

1. An accurate and efficient non-simulative probabilistic model (LIFE-DAG) for estimation of fault detection probabilities which is a measure of random pattern testability. We use using two close-to-exact inference schemes. Our model advances the state-of-the-art BDD based techniques in terms of time and space complexity and in providing a uniform model.

2. Timing-Aware Logic-Induced Single-Event-Upset Sensitivity Model(TALI-SES): A comprehensive model for the underlying error framework, which is a graphical probabilistic that is causal, minimal and exact. Using this model we are able to accurately estimate the SEU sensitivities of individual nodes in a circuit capturing spatial and temporal signal correlations, effect of inputs, gate delay, SEU duration and re-convergence in the circuit. We use this model (1) to infer error probabilities using an exact inference algorithm that transforms the graph into a special junction tree and relies on local message passing scheme and smart stochastic non-simulative inference algorithms that have the features of any-time estimates and excellent accuracy-time trade-off for larger circuits (2) to classify nodes with respect to SEU sensitivities for application of selective redundancy schemes for soft error hardening and (3) to generate the input space for which an SEU occurring at an input node have no impact on the output by exploiting the unique backward reasoning property of Bayesian networks.

3. A Logic-Induced Probabilistic Error Model (LIPEM) which is an exact probabilistic representation of the dynamic error model that captures the circuit topology and individual gate error probability for estimating the overall output error probabilities of logic circuits given dynamic error probabilities in each gate. We use both exact and stochastic inference schemes for propagation of probabilities and show that exact inference is faster that the state-of-the-art PTM technique. To handle larger circuits we use two stochastic sampling algorithms and demonstrate the efficiency and accuracy of these schemes by comparing with simulation results.

4. We also use the LIPEM model to (a) compute dynamic error sensitivities of individual gates in a circuit, which is useful for selective redundancy application (b) to compare and vet equivalent designs with respect to dynamic errors (c) to characterize the input space for achieving a desired output behavior in terms of error tolerance by utilizing the unique backtracking feature of Bayesian Networks and (d) to classify nodes with respect to dynamic error sensitivity and analyze the effect of selective redundancy techniques on circuit reliability.

1.9 Organization

The remainder of this dissertation is organized as follows. Chapter 2 is a summary of the prior works done on FDP modeling, soft error modeling and analysis and on inherent errors. In Chapter 3, Bayesian Networks fundamentals and different types of inference schemes are discussed. Chapter 4 describes modeling of Stuck-at-Faults using Logic Induced Fault Encoded Directed Acyclic Graph (LIFE-DAG) model. In Chapter 5 we discuss the modeling of Single-Event-Upsets incorporating timing issues. Here we describe how Timing-Aware-Logic-Induced Soft Error Sensitivity (TALI-SES) model can be used to estimate the SEU sensitivities of individual nodes in logic circuits. Chapter 6 deals with modeling of inherent/dynamic errors in nano-domain logic circuits. The proposed LIPEM (Logic-Induced-Probabilistic Error Model) can be used for (a) estimation of overall output error probabilities due to dynamic errors, (b) error sensitivity of individual gates in logic circuits (c) input space characterization and (d) application of selective redundancy measures. In Chapter 7 we discuss results and future research directions.

CHAPTER 2

BACKGROUND

With increase in the complexity of digital VLSI circuits, design errors and/or manufacturing defects can occur in a chip. We broadly classify errors/faults into three categories: *Permanent faults* caused by manufacturing defects, material defects and packaging failures. (Examples of such faults are stuck-at-faults, open and short faults, bridging faults, ect.), *transient errors* caused by atmospheric radiation, known as soft errors and *dynamic errors* which are inherent in nano-domain devices due to technology scaling and devices operating near thermal limits.

2.1 Fault Detection Probability

Fault Detection Probability (FDP) is an important testability measure that is useful for not only generating test patterns, but also to vet designs for random-input testability. Most of the permanent faults can be covered by single stuck-at-fault tests [10]. In this dissertation, we model single stuck-at-faults by estimating fault detection probability. Traditionally, FDP has been used for test point insertions, however, it can also be used as random single-event-transient (SET) sensitivity measure, which is important for characterization of impact of soft errors in logic blocks.

Fault Detection Probability (FDP) of a stuck-at fault $f \in F$, where F denotes the set of all faults, in a combinational circuit, C, is the probability that f is detected by a randomly chosen equally-likely input patterns. Signal probabilities, as well as FDP, are affected by spatial correlations induced by circuit re-convergence.

Figure 2.1. enlists some of research works done on FDP analysis. Due to the high computational complexity involved in computing signal and fault detection probabilities, several approximation strategies have been developed in the past [12, 29, 30, 31]. The cutting algorithm [31], computes lower bounds of fault detection probabilities by propagating signal probability values. This algorithm delivers loose

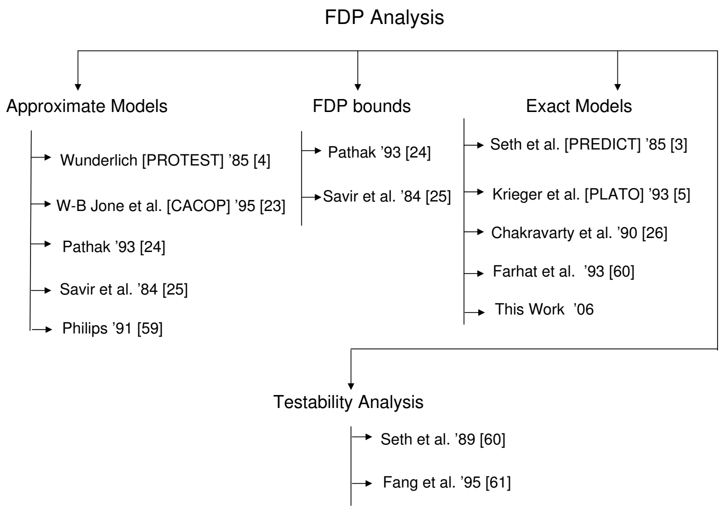


Figure 2.1. Related works on FDP analysis

bounds, which may lead to unacceptable test lengths. Also, computing complexity of this algorithm is $O(n^2)$. Lower bounds of fault detection probability were also derived from controllability and observability measures [30]. This method do not account for the component of fault detection probability due to multiple path sensitizations. The above mentioned methods are satisfactory only for faults that have single sensitizing path for fault propagation to an output and hence will not give good results for highly re-convergent fan-out circuits that have multiple path sensitizations.

PREDICT [11] is a probabilistic graphical method to estimate circuit testability by computing node controllability and observability using Shannons expansion. The time complexity of exact analysis by this method is exponential in the circuit size. PROTEST [12], which is a tool for probabilistic testability analysis, calculates fault detection probabilities and optimum input signal probabilities for random test pattern, by modeling the signal flow. Fault detection probabilities, which are computed from signal probability values, are underestimated due to the fact that the algorithm does not take into account multiple path sensitization. Another method (CACOP) [29] is a compromise between the full range cutting algorithm and the linear time testability analysis, like the controllability and observability program. This is also an approximate scheme.

[32] uses super gate decomposition to compute exact fault detection probabilities of large circuits. PLATO (Probabilistic Logic Analyzing Tool) [13] is a tool to compute exact fault detection probabilities using reduced ordered binary decision diagrams (ROBDD)s. Space requirement for constructing the ROBDD of large circuits is very large. Shannon decomposition and divide-and-conquer strategies are used to reduce large circuits into small sub-circuits. Computing complexity of these decomposition methods are quite high. Another BDD based algorithm [33] computes exact random pattern detection probabilities. However, this is not scalable for large circuits.

State-of-the-art algorithm for computation of *exact* fault detection probabilities, based on Binary Decision Diagrams [13] do not scale well, in terms of time and space requirements, with circuit size. They usually require various heuristic-based approximations of the pure model.

Most of the analysis of FDP was performed in 80's (as proposed by Seth *et al.* [11], Wunderlich [12], etc.) In a later development (1988), Bayesian network was introduced for probabilistic reasoning and belief propagation and it has been applied in artificial intelligence and image analysis. Recently, in [34, 35], switching probabilities and signal arrivals in VLSI circuits have been modeled using a Bayesian Network, however their use in estimation of error detection probabilities in digital logic is new. In this dissertation, we provide a fresher look into an old problem by adopting a novel and efficient scheme. However, this probabilistic FDP analysis technique can be applied to measure Single Event Transient (SET) sensitivity and soft error susceptibility [14] of logic circuits which are future nanotechnology challenges.

2.2 Transient Faults: Single-Event-Upsets

High-energy neutrons present in cosmic radiations and alpha particles from packaging materials give rise to single event upsets (SEUs) resulting in soft errors in logic circuits. When particles hit a semiconductor material, electron-hole pairs are generated, which may be collected by a P-N junction, resulting in a short current pulse that causes logic upset or Single Event Upset (SEU) in the signal value. An SEU may occur in an internal node of a combinational circuit and propagate to an output latch. When a latch captures the SEU, it may cause a bit flip, which can alter the state of the system resulting in a soft error. In current technology, soft errors are of serious concern in memories, whereas in logic circuits

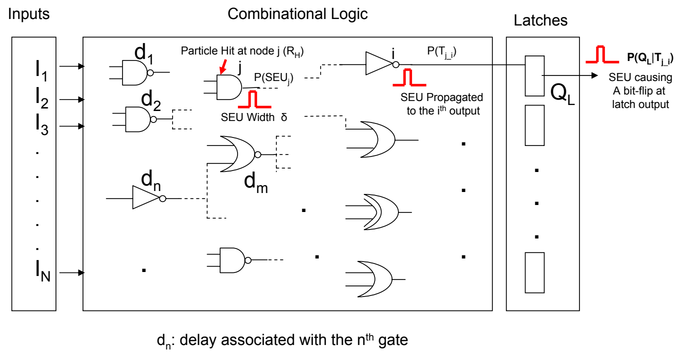

Inputs
Combinational Logic
Latches
I_1
I_2
I_3
I_N
d_1
d_2
d_n
d_m
Particle Hit at node j (R_H)
j
$P(SEU_j)$
SEU Width δ
i
$P(T_{L,i})$
SEU Propagated to the i^{th} output
Q_L
$P(Q_L|T_{L,i})$
SEU causing A bit-flip at latch output
d_n: delay associated with the n^{th} gate

Figure 2.2. SEU propagation

soft error rate is comparatively low due to logical, electrical and temporal masking effects. However, as process technology scales below 100 nanometers and operating frequencies increase, the above masking barriers diminish due to low supply voltages, shrinking device geometry and small noise margin. This will result in unacceptable soft error failure rates in logic circuits even for mainstream applications [14].

There have been several analytical and experimental works on soft error rate analysis and error hardening techniques, at different levels of abstraction, such as architectural level, RTL, logic and circuit level. Use of Laser Fault Injection (LFI) technique for soft error tolerant design verification on FPGAs and ISICs has been proposed in works done by Wiley *et al.* [36] and Falquez *et al.* [37]. Research done by Wiley *et al.* [36] demonstrate that LFI can repeatedly induce single event upsets and latchups in FPGA designs without causing permanent damage to the chip.

Mitra *et al.* [38] gives a comparison of several system level soft error mitigation schemes in terms of soft error rate reduction, power consumption, area overhead, performance and cost. The method presented in this dissertation models logical and temporal masking at gate level and it can be integrated with the electrical and latching window masking models existing in current literature to get a comprehensive model to capture the effect of supply voltage, threshold voltage, set up, hold time, etc.

Soft error susceptibility of a node j with respect to a latch L, $SES_{j\_Q_L}$ is the soft error rate at the latch output Q_L, contributed by node j. The propagation of an SEU generated due to a particle hit at an internal node j to an output i which causes a bit flip at the output of a latch L is depicted in Fig. 2.2.

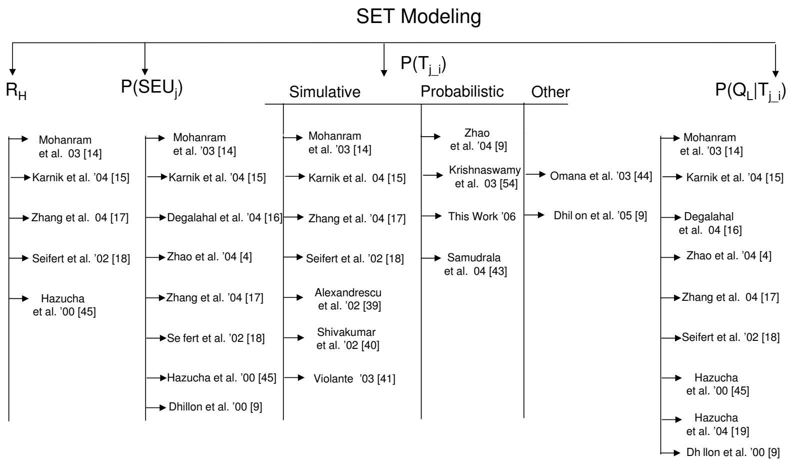


Figure 2.3. Recent works on SET modeling

We model the SEU propagation as follows: Let T_{j_i} be a Boolean variable which takes logic value 1 if an SEU at a node j causes an error at an output node i. Then $P(T_{j\_i})$ (measured as the probability of T_{j_i} being equal to 1) is the conditional probability of occurrence of an error at output node i given an SEU at node j. Let $P(SEU_j)$ be the probability that a particle hit at node j generates an SEU of sufficient strength and let $P(Q_L|T_{j\_i})$ be the probability that an error at output node i causes an erroneous signal at latch output Q_L. Mathematically $SES_{j\_Q_L}$ is expressed by Eq. 5.1.

$$SES_{j\_Q_L} = R_H P(SEU_j) P(T_{j\_i}) P(Q_L|T_{j\_i}) \tag{2.1}$$

where R_H is the particle hit rate on a chip which is fairly uniform in space and time. $P(SEU_j)$ depends on V_{dd}, V_{th} and also on temperature. $P(Q_L|T_{j\_i})$ is a function of latch characteristics and the switching frequency.

Figure 2.3. gives a list of the recent works done on soft error analysis. An estimation method for soft error failure rates resulting from Single Event Upsets proposed in [14] computes soft error susceptibility of a node based on the rate at which a Single Event Upset (SEU) occurs at the node (R_{SEU}), the probability that it is sensitized to an output $(P_{sensitized})$ and the probability that it is captured by a latch$(P_{latched})$. In [39], Alexandrescu *et al.* present a SET fault simulation technique to evaluate the soft error probability caused by transient pulses. A model that captures the effects of technology trends in the Soft Error failure Rates (SER), considering different types of masking phenomena such as electrical masking, latching window masking and logical masking, is presented in [40]. Another model to analyze Single Event Upsets with *zero-delay* logic simulation, which is accurate and faster than timing simulators, is presented in [41]. As discussed in the previous section, this model uses a circuit expansion algorithm to incorporate gate delays and a fault list generation algorithm to get a reduced list of SETs. All of the above methods use simulation techniques which are highly input pattern dependant.

Zhao *et al.* proposes a methodology to evaluate softness or vulnerability of nodes in a circuit due to compound noise effects by considering the effects of electrical, logical and timing masking [4]. They use a probabilistic approach to estimate the effect of logical masking. However this method can not be used for circuits with re-convergent paths and will not be able to handle larger circuits. Also, this method doesn't capture the effect of gate delays.

A selective triple modular redundancy technique (STMR) for achieving radiation tolerance in FPGA designs is discussed in [43]. With the method presented in their work, they achieve a reduction in area overhead by 2/3rd of the area required by ordinary Triple Modular Redundancy (TMR). Another cost-effective radiation hardening technique based on gate (transistor) sizing technique has been discussed in [42].

Karnik *et al.* suggests that soft error rate should be considered as a design parameter along with power, performance and area due to its increasing impact on circuits and systems with the scaling of process technology [15]. They propose a methodology to quantify the impact of supply voltage, transistor size, circuit topology, doping as well as circuit structure on the Soft Error Rate (SER) of a chip. effect of threshold voltage on SER of memories and combinational logic has been studied in [16]. Zhang *et al.* in [17] proposed a composite soft error rate analysis method (SERA) to capture the effect of supply voltage, clock period, latching window, logic depth, circuit topology and input vector on soft error rate. However, they resort to logic and circuit level simulation to capture these probabilities. This method uses a conditional probability based parameter extraction technique obtained from device and logic simulation. In their work, combinational circuits are assumed to have unbalanced re-convergent paths. However, other design considerations usually drive optimal circuit design to have balanced paths by adding buffers wherever re-convergence is necessary. For circuits with balanced paths, soft error analysis based on approximations given in [17] might not be the best choice.

Dhillon *et al.* [9] has proposed a MATLAB based model for soft error tolerance analysis and optimization of nano-meter circuits by studying the glitch generation and propagation characteristics of gates in logic blocks. Another mathematical model to estimate the possible propagation of glitches due to transient faults has been presented in [44].

Seifert *et al.* discusses the importance of latch design on the soft error rate (SER) of core logic [18]. It also analyses the impact technology scaling on SER at device, circuit and chip level. Relation between soft error rate and technology feature size based on device level simulation has been studied in [45].

Since all the state-of-the-art techniques have resorted to simulation for logical and device level effects (known to be expensive and pattern-sensitive especially for low probability events), we felt the need to explore the input data-driven uncertainty in a comprehensive manner through a probabilistic model to capture the effect of primary inputs, the effect of gate delay and the effect of SEU duration on the logical masking. There is future scope for these kinds of models to be fused with other models [15, 16, 17, 18, 45] for capturing device effects such as electrical masking, threshold voltage and supply voltage. Our model can be used to identify gates in a logic circuit that are highly sensitive to single-event-upsets and classify gates with respect to SEU sensitivity. Application of selective redundancy techniques to these gates will enhance circuit reliability. A study of reliability/redundancy trade-off can then be done using our TALI-SES model.

2.3 Dynamic Errors

As mentioned in Section 1.1., this type of errors will be pre-dominant in nano-domain computing. They are caused by temporary device malfunction due to reduced feature size and operation near thermal limits. Hence one of the major challenges will be reliable computation with unreliable devices. This issue has been addressed in many previous researches. Fig. 2.4. is a list of related works done on dynamic error modeling and reliability estimation. Indeed the basic methods of Triple Modular Redundancy (TMR) and NAND Multiplexing were proposed in the 1950s [5, 46]. These ideas have been adapted for nano-computing architectures such as N-tupple Modular Redundancy [47], or NAND Multiplexing

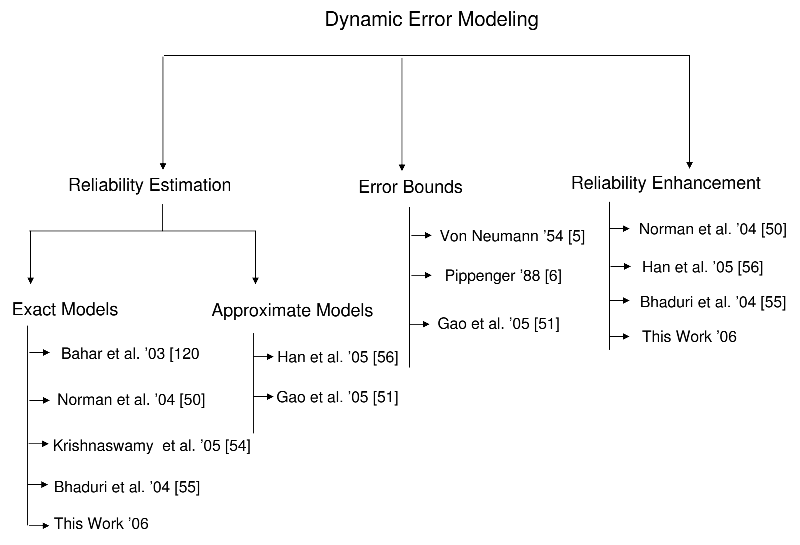


Figure 2.4. Background works on dynamic errors

and reconfiguration [48]. A recent comparative study of these methods [49], indicates that a 1000-fold redundancy would be required for a device error (or failure) rate of 0.01[1]. Probabilistic model checking for reliability-redundancy trade-off was introduced in [50]. These techniques of providing error and fault tolerance methods, rarely utilized the topology of a circuit and dependency of errors on the circuit topology.

The analysis technique based on elementary bifurcation theory proposed in [51] for computing the exact error threshold values for noisy gates is applicable for building future fault-tolerant nano-domain networks. H. J. Gao *et. al* in [52] gives a comparison of different redundancy schemes such as Von Neumann's multiplexing logic, N-tuple modular redundancy and interwoven redundancy in terms of degree of redundancy and reliability by using markov chain models and bifurcation analysis. Reliability of nand-multiplexed nanoelectronic systems is characterized using markov chains and probabilistic computation schemes in [53].

In a recent work, S. Krishnaswamy *et al.* [54] provided a probabilistic transfer matrix-based (PTM) formalism for modeling dynamic errors at logic level. Since computational complexity of this method

[1]Note that this does *not* mean 1 out of 100 devices will fail, it indicates the devices will generate erroneous output 1 out of 100 times.

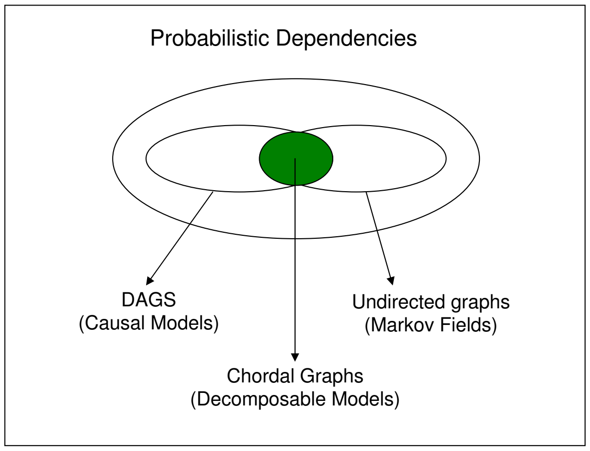


Figure 2.5. Modeling capacities for different probabilistic dependency models

is very high, this method is useful reliability estimation of very small circuits. Moreover, the time and space complexity associated with this modeling scheme varies with changes in gate error probabilities.

In [55], Bhaduri *et al.* propose a MATLAB based tool to evaluate and compare the reliability of various fault tolerant designs employing TMR redundancy, cascaded TMR, etc. In this probabilistic modeling tool, boolean logic is replaced by energy distribution function based on Gibbs distribution. Probabilities of energy levels at gate inputs and outputs are propagated to the circuit output through belief propagation. This model is used for analysis of reliability-redundancy tradeoffs for alternative defect tolerant architectures in the presence of signal noise. However, results for known benchmark circuits are not given in this paper.

Han *et al.* in [56], discuss an approximate method based on Probabilistic Gate Model (PGM) for reliability evaluation, where gate input/output signals are assumed to be statistically independent which is not true with circuits having re-convergent fanouts. However this model has less time and space complexity, as compared to PTM method and the reliability estimates obtained from the PGM based model was found to be very close to the PTM estimates.

Bayesian Networks and Markov Random Fields: Belief propagation for nano-computing was introduced by Bahar *et al.* [20] using Markov Random Fields (MRF), but demonstrated only for simple circuits. Bayesian networks(BN) and Markov Random Fields (MRF) are two probabilistic modeling tools. MRFs model dependencies using *undirected* graph structure, which might be suitable for non-causal logical models such as in molecular computing. Bayesian networks model dependencies using *directed* graphical structures, which are minimal for causal models. Traditional computing is and future computing, based on RTDs, CNTs, or clocked-QCA's, will be causal, i.e. there is flow of information from input to output. MRFs are, however, inefficient for modeling such causal computing. Specifically, two conceptually important dependencies exhibited by causal models, i.e. induced and non-transitive dependencies, are not captured by MRF's [26]. Figure 2.5. (taken from [26] chapter 3) shows the limitations of different probabilistic models; all *causal* dependencies cannot be represented by Markov networks. Another reason for our choice of Bayesian network based models, is that although the MRF computing model has the ability to incorporate dynamic (signal) errors, the errors are only implicitly captured in the choice of the polynomial coefficients of the energy potential expressions. For certain computing elements, such as those based on molecules, these errors in these coefficients might be directly related to physical phenomenon. However, for others, such as RTD, QCA, or nano-CMOS, the coefficients will be complex functions of the underlying error causes, making it harder to specify them directly. For instance, uniform errors across gates will not translate into uniform errors over the coefficients.

The model proposed in this dissertation is the first step towards understanding the errors specific to a circuit. We know that error probability inference is NP-hard, however, using conditional independence and smart approximation, reasonable estimates can be found. In this work, we present an exact algorithm for error probability computation of small circuits and an approximate algorithm for the mid-sized ISCAS benchmarks. Note that a simulative approach is not selected as it is not only driven by input pattern dependence but also separate simulation and input trace generation schemes have to be adopted for biased inputs.

In the following chapter we discuss the fundamentals of Bayesian networks, the graphical probabilistic modeling tool we used for modeling stuck-at-faults, single-event-upsets and dynamic errors. We also discuss the computational mechanisms used of efficient updating of signal probabilities in a Bayesian network.

CHAPTER 3

MODELING BASED ON BAYESIAN NETWORKS

A Bayesian network [26] is a Directed Acyclic Graph (DAG) in which the nodes of the network represent random variables and a set of directed links connect pairs of nodes. The links represent causal dependencies among the variables. Each node has a conditional probability table (CPT) except the root nodes. Each root node has a prior probability table. The CPT quantifies the effect the parents have on the node. Bayesian networks compute the joint probability distribution over all the variables in the network, based on the conditional probabilities and the observed evidence about a set of nodes.

Fig. 3.1. illustrates a small Bayesian network. The exact joint probability distribution over the variables in this network is given by Eq. 3.1.

$$\begin{aligned} P(x_6,x_5,x_4,x_3,x_2,x_1) &= P(x_6|x_5,x_4,x_3,x_2,x_1) \\ &\quad P(x_5|x_4,x_3,x_2,x_1)P(x_4|x_3,x_2,x_1) \\ &\quad P(x_3)P(x_2)P(x_1). \end{aligned} \tag{3.1}$$

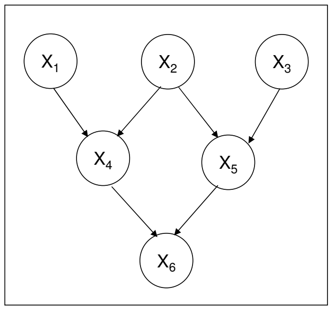


Figure 3.1. A small Bayesian network

In this BN, the random variable, X_6 is independent of X_1, X_2 and X_3 given the states of its parent nodes, X_4 and X_5. This *conditional independence* can be expressed by Eq. 3.2.

$$P(x_6|x_5,x_4,x_3,x_2,x_1) = P(x_6|x_5,x_4) \tag{3.2}$$

Mathematically, this is denoted as $I(X_6,\{X_4,X_5\},\{X_1,X_2,X_3\})$. In general, in a Bayesian network, given the parents of a node *n*, *n* and its descendents are independent of all other nodes in the network. Let U be the set of all random variables in a network. Using the conditional independencies in Eq. 3.2, we can arrive at the minimal factored representation shown in Eq. 3.3.

$$\begin{aligned} P(x_6,x_5,x_4,x_3,x_2,x_1) &= P(x_6|x_5,x_4)P(x_5|x_3,x_2) \\ &\quad P(x_4|x_2,x_1)P(x_3)P(x_2)P(x_1). \end{aligned} \tag{3.3}$$

In general, if x_i denotes some value of the variable X_i and $pa(x_i)$ denotes some set of values for X_i's parents, the minimal factored representation of exact joint probability distribution over m random variables can be expressed as in Eq. 3.4.

$$P(X) = \prod_{k=1}^{m} P(x_k|pa(x_k)) \tag{3.4}$$

In Fig. 3.1., it can be seen that nodes x_4 and x_5 are dependent since they have a common parent. Even though this dependency is not shown in the initial Bayesian net graph structure, during Bayesian inference process these dependencies are taken care of by a process known as moralization where each pair of unconnected nodes having a common child node are connected by an undirected graph, making every parent child set a complete sub graph. We explain this aspect of Bayesian inference in detail under Section 3.1.

3.1 Bayesian Inference

We explore three method for computing probabilities in Bayesian networks. One is an exact scheme that uses local message passing, but places high demands on memory for circuits with lots of loops in the underlying undirected graph structure; we call this the cluster based inference scheme. The second one is an approximate algorithm based on importance sampling. And, the third one is also an approximate algorithm based on sampling but exploiting loopy, local message passing. These approximate schemes can be proven to converge to the correct probability estimates [28, 57] and these are scalable any-time algorithms with excellent accuracy-time trade-off. An elaborate discussion on this is given in [58].

3.2 Cluster Based Inference

We demonstrate this inference scheme using a small example, the c17 benchmark circuit (See Fig 3.2.). The combinational circuit is shown in Fig. 3.2.(a) and its corresponding Bayesian network representation is shown in Fig 3.2.(b).

The first step of the exact inference process is to create an undirected graph structure called the *moral graph* (denoted by D^m) given the Bayesian network DAG structure (denoted here by D). The moral graph represents the Markov structure of the underlying joint function [59]. The dependencies that are preserved in the original DAG are also preserved in the moral graph [59]. From a DAG, which has the structure of a Bayesian network, a moral graph is obtained by adding undirected edges between the parents of a common child node and dropping the directions of the links. Fig. 3.2(a) shows the undirected moral graph and the dashed edges are added at this stage. This step ensures that every parent child set is a complete sub graph. Moral graph is undirected and due to the added links, some of the independencies displayed in DAG will not be graphically visible in moral graph. Some of the independencies that are lost in the transformation contributes to the increased computational efficiency but does not affect the accuracy [59]. The independencies that are graphically seen in the moral graph are used in inference process to ensure local message passing.

The moral graph is said to be triangulated if it is chordal. The undirected graph G is called chordal or triangulated if every one of its cycles of length greater than or equal to 4 possesses a chord [59] that is we add additional links to the moral graph, so that cycles longer than 3 nodes are broken into cycles of three nodes. The chordal graph derived from the moral graph is shown in Fig. 3.2(b) Here we can see that an additional link between nodes X_{10} and X_{11} has been added so that the cycle consisting of nodes X_3 , X_{10} ,X_{16} and X_{11} is broken down to two cycles each having nodes 3. Once the chordal graph

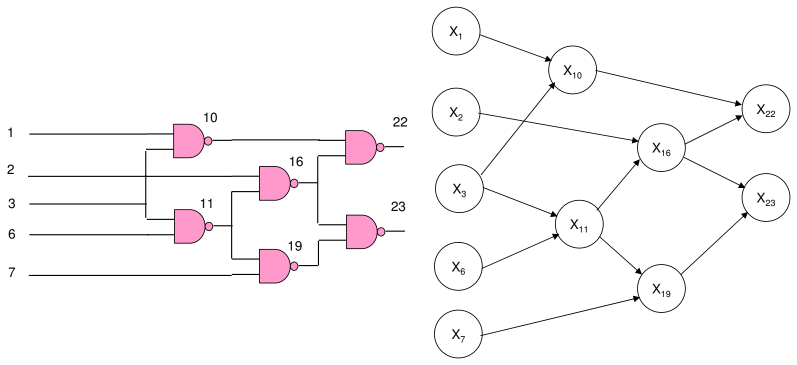


Figure 3.2. (a) The benchmark circuit c17 (b) Bayesian network representation of c17

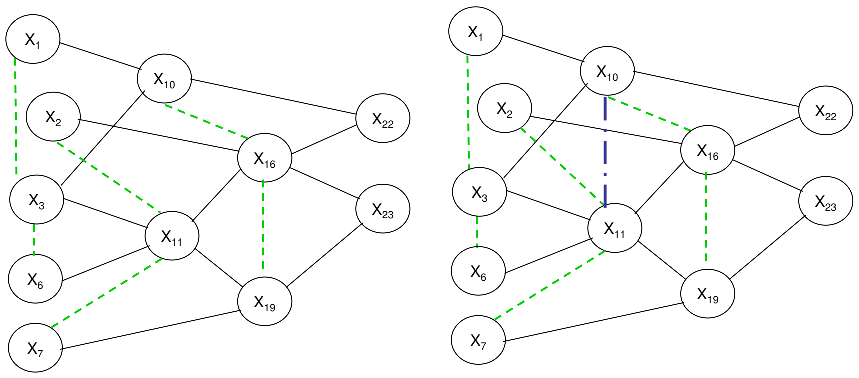


Figure 3.3. (a) Moral graph (b) chordal graph

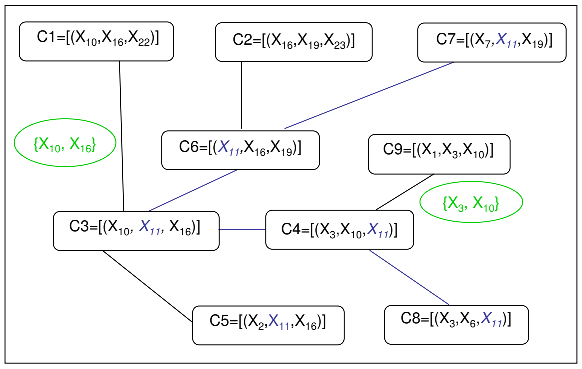


Figure 3.4. Junction tree

is constructed, the next step is creation of junction tree. The junction tree is defined as a tree with nodes representing cliques (collection of completely connected nodes) of the choral graph and between two cliques in the tree T there is a unique path. Junction tree possesses a property called running intersection that ensures that if two cliques share a common variable, the variable should be present in all the cliques that lie in the unique path between them. Fig 3.2 shows the junction tree. Note that every clique in the moral graph is a node (example $C1 = [X_{10}, X_{16}, X_{22}]$) in the junction tree. Also, C7 and C8 have node X_{11} in common, hence X_{11} is present in all the cliques namely C4,C3, and C6 that lie between the unique path between C7 and C8. This property of junction tree is utilized for probabilistic inference so that local operation between neighboring cliques guarantees global probabilistic consistency.

Chordal graphs are essential as they guarantee the existence of at least one junction tree. Hence chordalization is a necessary step. There are many algorithms to obtain junction tree from chordal graph and we use a tool HUGIN [27] that uses minimum-fill-in heuristics to obtain a minimal chordal and junction tree structure.

Since every child parent team is present together in one of the cliques, we initialize the clique joint probabilities by the original joint probability of a child parent team. We then use a message passing scheme to have consistent probabilities. Suppose we have two leaf clique in the junction tree, say in our example in Fig 3.2, C8 and C9. Both the cliques are initialized based on the child parent team (C8 by nodes 3, 6 and 11 and C9 by node 1, 3, 10). Similarly, other leaf nodes C1, C2, C5 and C7 are initialized. The initial clique probability of clique Ci is termed as ϕ_{Ci} and is also called potential of a clique.

Let us now consider two neighboring cliques to understand the key feature of the Bayesian updating scheme. Let two cliques Cl and Cm have probability potentials ϕ_{Cl} and ϕ_{Cm}, respectively. Let S be the set of nodes that separates cliques A and B (Example: $S = \{X_{10}, X_{16}\}$ between cliques C1 and C3 in Fig. 3.2) The two neighboring cliques have to agree on probabilities on the node set S which is their separator. To achieve this we first compute the marginal probability of S from probability potential of clique Cl and then use that to scale the probability potential of Cm. The transmission of this scaling factor, which needed in updating, is referred to as message passing. New evidence is absorbed into the network by passing such local messages. The pattern of the message is such that the process is multi-threadable and partially parallelizable. Because the junction tree has no cycles, messages along each branch can be treated independently of the others.

Note that since junction tree has no cycle and it is also not directional, we can propagate evidence from any node at any clique and the propagate the evidence in any direction. It is in sharp contrast with simulative approaches where flow of information always propagate from input to the outputs. Thus, we would be able to use it for input space characterization for achieving zero output error due to SEUs. We would instantiate a desired observation in an output node (say zero error) and backtrack the inputs that can create such a situation. If the input trace has large distance from the characterized input space, we can conclude that zero error is reasonably unlikely. Note that this aspect of probabilistic backtracking is already used in medical diagnosis but are new in the context of input space modeling for errors in VLSI circuits.

Complexity The computational complexity of the exact method is exponential in terms of number of variables in the largest cliques. Space complexity of the exact inference is $n.2^{|Cmax|}$ [60], where n is the number of nodes in the Bayesian Network, and $|Cmax|$ is the number of variables in the largest clique. The time complexity is given by $p.2^{|Cmax|}$ [60] where p is the number of cliques.

Due to the extremely high space complexity, the exact inference scheme can handle only small circuits. Hence we explore two approximate inference schemes based on stochastic sampling, namely

Probabilistic Logic Sampling (PLS) and Evidence Pre-Propagated Importance Sampling (EPIS). These schemes are proven to be any-time scalable sampling techniques with excellent accuracy-time trade-off.

3.3 Stochastic Inference

Stochastic sampling algorithms are the prominent subclass of approximate inference schemes. They generate randomly selected instantiations of the network variables in topological order according to probabilities in the model, and then calculate frequencies of instantiations of interest to obtain the approximate inference. These algorithms use a sampling technique known as *Importance Sampling*.

Given the states of certain nodes in the network (evidence) and the conditional probability table of each node, this algorithm generates sample instantiations of the network (i.e., defines the state of each node in the network). From these sample instantiations, it can find approximate probabilities of each state for each node in the network. This sampling is done according to an optimum importance function $g(x)$, that closely resembles the underlying joint probability distribution $P(x)$ for which the variance of $\hat{I}_N$ is *zero* where $\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{P(s_i)}{g(s_i)}$ for N samples $s_i \forall i = 1, \cdots, N$ obtained by sampling the importance function $g(x)$ [58]. It is proven that in a Bayesian network, the product of the conditional probability functions at all nodes form the optimal importance function [57, 58]. Let $X = \{X_1, X_2, \cdots, X_m\}$ be the set of variables in a Bayesian network, $Pa(X_k)$ be the parents of X_k, and E be the evidence set. Then, the optimal importance function is given by

$$P(X|E) = \prod_{k=1}^{m} P(x_k|Pa(x_k, E) \tag{3.5}$$

3.3.1 Probabilistic Logic Sampling (PLS)

Probabilistic logic sampling is the earliest and the simplest stochastic sampling algorithms proposed for Bayesian Networks [28]. Probabilities are inferred by a complete set of samples or instantiations that are generated for each node in the network according to local conditional probabilities stored at each node. The advantages of this inference are that: (1) its complexity scales linearly with network size, (2) it is an any-time algorithm, providing adequate accuracy-time trade-off, and (3) the samples are not based on inputs and the approach is input pattern insensitive. The salient aspects of the algorithm are as follows.

1. Each sampling iteration stochastically instantiates all the nodes, guided by the link structure, to create a network instantiation.

2. At each node, x_k, generate a random sample of its state based on the conditional probability, $P(x_k|Pa(x_k))$, where $Pa(x_k)$ represent the states of the parent nodes. This is the local, importance sampling function.

3. The probability of all the query nodes are estimated by the relative frequencies of the states in the stochastic sampling trace.

4. If states of some of the nodes are known (evidence), such as in diagnostic backtracking, network instantiations that are incompatible with the evidence set are disregarded.

5. Repeat steps 1, 2, 3 and 4, until the probabilities converge.

In predictive inference, logic sampling generates precise values for all the query nodes based on their frequency of occurrence but with diagnostic reasoning, this scheme fails to provide accurate estimates because of large variance between the optimal importance function and the actual underlying joint probability distribution function. The disadvantage of this approach is that in case of unlikely evidence they disregard most samples and thus the performance of the logic sampling approach deteriorates.

3.3.2 Evidence Pre-Propagated Importance Sampling

To overcome the problem of backtracking in the light of our on-going BN based input characterization problem, we explore another stochastic algorithm that uses Loopy Belief Propagation to update the optimal importance function such that the variance of $\hat{I}_N$ is extremely small. This method scales well with circuit size and is proven to converge to correct estimates. Let $X = \{X_1, X_2, \cdots, X_m\}$ be the set of variables in a Bayesian network, $Pa(X_k)$ be the parents of X_k, and E be the evidence set. Then, the optimal importance function given by Eq. 3.5. can be approximated as

$$P(X|E) = \prod_{k=1}^{m} \alpha(Pa(X_k))P(x_k|Pa(X_k))\lambda(X_k) \tag{3.6}$$

Yuan *et al.* has proved that in poly-trees we can calculate them directly. The EPIS algorithm uses loopy belief propagation [61, 62] to pre-compute the approximate importance function and then use ε-cutoff heuristics to adjust small probabilities in the importance function.

Loopy Belief Propagation: The belief propagation algorithm computes the posterior belief of each node X in a Bayesian network as $BEL(x) = P(X = x|E)$ where E denotes the set of evidence. This is done by combining messages from its children and messages from its parents. $E = \{E^+, E^-\}$ where E^+ is the set of evidence connected to the parents of X and E^- is the set of evidence connected to its child nodes. Any node X d-separates E into two subsets E^+ and E^-, or in other words, given the state of X, the two subsets $itfE^+$ and E^- are independent. The message from E^+ is defined as

$$\pi(x) = P(x|E^+) \tag{3.7}$$

and the message from E^- is defined as

$$\lambda(x) = P(E^-|x) \tag{3.8}$$

The posterior belief of node X is computed as

$$BEL(X) = \alpha\lambda(x)\pi(x); \tag{3.9}$$

where α is a normalizing constant. Based on the messages received from its neighboring nodes, each node passes new messages back to its parent and child nodes . This way, messages are propagated throughout the network. In [61], Murphy *et al.* explains how each node computes new λ and π messages based on the received massages. The loopy belief propagation algorithm normalizes both λ and π messages after each iteration. This normalization doesn't make any difference in the final beliefs, but avoids numerical underflow.

In networks with loops, getting exact λ messages for all variables is an NP-hard problem. Loopy belief propagation will provide us with good approximate λ messages which can be used to obtain a good approximate importance function.

This stochastic sampling strategy works because in a Bayesian Network the product of the conditional probability functions for all nodes is the optimal importance function. Because of this optimality, the demand on samples is low. We have found that just thousand samples are sufficient to arrive at good estimates for the ISCAS'85 benchmark circuits.

In the next chapter we discuss Bayesian network based modeling of single stuck-at-faults which is useful for estimation of fault detection probabilities in combinational circuits. We use both Probabilistic Logic Sampling and Evidence Pre-propagated Importance Sampling for Bayesian inference. Results from our model shows that BN-based modeling is computationally more efficient compared to the state-of-the-art BDD based technique.

CHAPTER 4

LIFE-DAG: AN ACCURATE PROBABILISTIC MODEL FOR ERROR DETECTION PROBABILITIES

Permanent faults such as stuck-at-faults, stuck-open fault and bridging faults can occur in a logic circuit due to fabrication errors or on-line device failures. Fault modeling is important for random pattern testability analysis. Most of the permanent faults can be covered by single stuck-at-fault tests [10]. In this chapter, we describe the modeling of stuck-at-faults in combinational circuits for the estimation of fault detection probability which is an important random pattern testability measure.

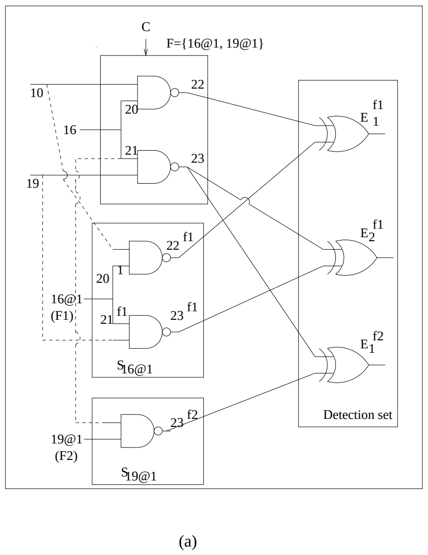
We model single stuck-at-faults (errors) in large combinational circuits using a Logic Induced Fault Encoded Direct Acyclic Graph (LIFE-DAG) graph structure. We prove that such a DAG is a Bayesian Network. Bayesian Networks are graphical probabilistic models representing the joint probability function over a set of random variables. A Bayesian Network is a directed acyclic graphical structure (DAG), whose nodes describe random variables and node to node arcs denote direct causal dependencies. A directed link captures the direct cause and effect relationship between two random variables. In a LIFE-DAG, each node describes the state of a line and a directed edge quantifies the conditional probability of the state of a node *given* the states of its parents or its direct causes. Figure 1.5.(a) gives the conditional probability table of an AND gate. The attractive feature of this graphical representation of the joint probability distribution is that not only does it make conditional dependency relationships among the nodes explicit but it also serves as a computational mechanism for efficient probabilistic updating. Probabilistic Bayesian Networks can be used not only to infer effects due to known causes (predictive problem) but also to infer possible causes for known effects (the backtracking or diagnosis problem). The diagnostic aspects of BNs make it suitable for further use in test-pattern generators. We used two stochastic algorithms based on Importance sampling to compute the fault/error detection probability using the LIFE-DAG model. An importance sampling algorithm generates sample instantiations of the *whole* DAG network, i.e. for all lines in our case. These samples are then used to form the final estimates. At each node, this sampling is done according to an importance function that is chosen to be close to the actual joint probability function. Specifically, we explore two stochastic inference strategies: Probabilistic Logic Sampling (PLS) [28] and Evidence Pre-propagated Importance Sampling (EPIS) algorithm [57], which are discussed in section 3.1. It is worth pointing out that unlike simulative approaches that sample the inputs, importance sampling based procedures generate instantiations for the whole network, not just for the inputs.

We advance the state-of-the-art in fault analysis in terms of space and time requirements and in providing a uniform model. A detailed discussion about time and space complexity is given in section 3.1.

4.1 Motivation

When high-energy neutrons present in cosmic radiations and alpha particles from impurities in packaging materials hit a semiconductor device, they generate electron-hole pairs and cause a current pulse of very short duration, termed as Single Event Transient (SET). The effect of these SETs may be propagated to an output latch and cause a bit flip in latch, resulting in a *soft error*. As the stored charge in each logical node decreases with decreased dimensions and decreased voltages in nanometer technology, even weak radiations can cause disturbance in the signals, which results in increased soft error failure rate.

SET sensitivity of a node depends on the probability that there is a functionally sensitized path from the node to an output latch (which is the same as the fault detection probability of the node), probability that the generated SET is propagated to the latch and the probability that the latch captures the transitions arriving at its input [14]. If the last two probabilities are assumed to be one, FDP of a node is an accurate measure of the SET sensitivity. In nano-domain circuits, these probabilities are very close to one due to very high operating frequencies, reduced voltage, diminishing dimensions and reduced propagation delays. Hence FDP is a tight upper bound of SET sensitivity in nano-domain circuits. Designers can selectively apply corrective measures to nodes with higher FDP than the ones with lower FDP to reduce soft error failure rates.



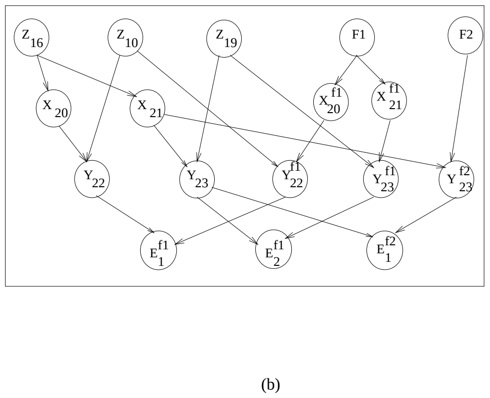


Figure 4.1. (a) An illustrative fault detection logic (b) LIFE-DAG model of (a)

4.2 LIFE-BN: Fault/Error Model

We first discuss the basics of fault/error detection probabilities for random-pattern testability analysis. Note that the probabilistic modeling does not require any assumption in the input patterns and can be extended to biased target workload patterns. Next, we sketch the concept of Logic-induced-Fault-Encoded (LIFE) Directed Acyclic Graph that represents the underlying fault/error model. In this probabilistic framework, we use partial duplication of the original circuit for the fault detection logic. Only the sensitized paths from the fault/error are duplicated. A set of comparator nodes compares the ideal and error-sensitized logic. A logic one in such comparator outputs indicates the occurrence of an error. Fault/Error detection probability of faults/errors that affect multiple outputs, is the maximum probability of the sensitized comparator outputs. These detection probabilities depend on the circuit structural dependence, the inputs and the dependencies amongst the inputs. In this work, however, we assume random inputs for validation and experimentation of our model.

We follow our discussion by proving that such a Logic-induced-Fault-Encoded (LIFE) Directed Acyclic graph is indeed the minimal I-map of the fault model for the $< C, F >$ and, hence, is a Bayesian Network.

Definition: The Logic Induced Fault Encoded Direct Acyclic Graph (LIFE-DAG) corresponding to the fault detection model $< C, F >$ where C represents the combinational circuit an F represents the fault/error set, can be constructed as follows (We illustrate the ideas with the help of a small fault detection circuit and the LIFE-DAG constructed from it, as shown in Figure 4.1.):
The model consists of the combinational circuit C (fault/error-free) and a set of fault/error sensitized logic blocks, $S^f \ \forall f \in F$, which propagate the effect of the faults.

- Nodes of LIFE-DAG are random variables with two possible values, 0 or 1. There are 7 types of nodes.

  1. $\{Z\}$: Primary inputs.
  2. $\{F\}$: Nodes representing injected faults.
  3. $\{X\}$: Internal nodes in the fault-free circuit C. Example: X_{21}.
  4. $\{X^f \ \forall f \in F\}$: Corresponding internal nodes in the duplicated circuit (descendents of $\{F\}$). Example: $X_{21}^{f_1}$
  5. $\{Y\}$: Primary outputs of fault-free circuit C. Example: Y_{22}.
  6. $\{Y^f \ \forall f \in F\}$: Corresponding faulty outputs. Example: $Y_{22}^{f_1}$.
  7. $\{E_o^f \ \forall o \in \ Output-set, \ \forall f \in F\}$: Set of all the detection nodes (comparator outputs). Cardinality of this set is determined by the number of testable outputs for each fault in the fault set. Example: $E_1^{f_1}$

- Edges of LIFE-DAG are directed and denoted as ordered pair $(u \rightarrow v)$, denoting u causes v. The edge set can be classified as follows:

  1. $\{Z \rightarrow X\}$: Edges between primary inputs and gates in C, which are directly fed by the primary inputs. These edges are part of the original circuit C.
  2. $\{F \rightarrow X^f\}$: Edges from the induced faulty inputs to the duplicate gates in the fault sensitized logic block, S^f.
  3. $\{Z \rightarrow X^f\}$: Edges from the primary inputs to the duplicate gates in S^f. Note that this edge indicates that there must be at least one parent of X^f that is in $\{F\}$ or in $\{X^f\}$.

4. $\{X \to X\}$, $\{X^f \to X^f\}$: Edges between the nodes representing the internal signals (Edges from the input of a gate to the corresponding output).

5. $\{X \to X_f\}$, $\{X \to Y^f\}$: Edges such that child node v is in S^f(in $\{X^f\}$ or in $\{Y^f\}$) and the parent u is in the original circuit C. These edges are called bridge edges. Note that this edge indicates that there must be at least one parent of v that is in $\{X^f\}$ or in $\{F\}$. Example: $X_{21} \to Y_{23}^{f_2}$.

6. $\{X \to Y\}$,$\{X^f \to Y^f\}$: Edges where parent node u is an internal signal feeding an output gate and the child node v represents the corresponding output signal. Example: $X_{21} \to Y_{23}$ and $X_{21}^{f_1} \to Y_{23}^{f_1}$

7. $\{Y \to E_o^f\}$ and corresponding $\{Y^f \to E_o^f\}$: Edges where u is the output node in Y (or Y^f) and v is the detection node E_o^f. These edges are quantified by a xor gate. Example: $Y_{22} \to E_1^{f_1}$ and $Y_{22}^{f_1} \to E_1^{f_1}$

Theorem: The LIFE-DAG structure, corresponding to the combinational circuit C and a Fault set F is a minimal I-map of the underlying dependency model and hence is a Bayesian network.

Proof: Markov Boundary of a variable v in a probabilistic framework, is the minimal set of variables that make the variable v conditionally independent of all the remaining variables in the network.

Let us order the random variables in the node set, $\{\{Z\},\{F\},\{X\},\{X^f\},\{Y\},\{Y^f\},\{E_o^f\}\}$ such that for every edge (u,v) in LIFE-DAG, u appears before v.

With respect to this ordering, the Markov boundary of any node, $v \in \{\{Z\},\{F\},\{X\},\{X^f\},\{Y\},$ $\{Y^f\},\{E_o^f\}\}$ is given as follows. If v represents an input signal line, then its Markov boundary is the null set. If v is a fault node f in the fault set F, then also its Markov boundary is the null set (since this is treated as a primary input with a particular value). And, since the logic value of an output line is just dependent on the inputs of the corresponding gate (whether v is in $\{X\},\{X^f\},\{Y\},\{Y^f\} or \{E_o^f\}$), the Markov boundary of a variable representing an output line consists of just those that represent the inputs to that gate. Thus, in LIFE structure the parents of each node are its Markov boundary elements hence the LIFE is a boundary DAG. Note that LIFE is a boundary DAG because of the causal relationship between the inputs and the outputs of a gate that is induced by logic. It has been proved in [26] that if a graph structure is a boundary DAG D of a dependency model M, then D is a minimal I map of M. This theorem along with definitions of conditional dependencies in [26] (we omit the details) specifies the structure of the Bayesian network. Thus LIFE is a minimal I-map and thus a Bayesian network (BN).

Quantification of LIFE-BN: LIFE-BN consists of nodes that are random variables of the underlying probabilistic model and edges denote direct dependencies. All the edges are quantified with the conditional probabilities making up the joint probability function over all the random variables of LIFE-DAG. The overall joint probability function that is modeled by a Bayesian network can be expressed as the product of the conditional probabilities. Let us say, $X^{'} = \{X^{'}_1, X^{'}_2, \cdots, X^{'}_m\}$ are the node set in LIFE-DAG, then we can say

$$P(X^{'}) = \prod_{k=1}^{m} P(x^{'}_k | Pa(X^{'}_k)) \tag{4.1}$$

where $Pa(X^{'}_k)$ is the set of nodes that has directed edges to $X^{'}_k$. A complete specification of the conditional probability of a two input AND gate output will have 2^3 entries, with 2 states for each variable. These conditional probability specifications are determined by the gate type. By specifying the appropriate conditional probability we ensure that the spatial dependencies among sets of nodes (not only limited to just pair-wise) are effectively modeled.

4.3 Experimental Results

We demonstrate the ideas using ISCAS benchmark circuits. The logical relationship between the inputs and the output of a gate determines the conditional probability of a child node, given the states of its parents, in the LIFE-BN. Gates with more than two inputs are reduced to two-input gates by introducing additional dummy nodes, without changing the logic structure and accuracy.

The LIFE-BN based model is capable of detecting stuck-at-faults as well as the soft errors caused by single-event-transients. In this work, we present results for a set of stuck-at-faults (*hard* faults), which shows the efficiency of our model, in time and space requirements, compared to the BDD based model. First, we determine the hard faults using 1024 random input vectors [13]. Faults that are not detected by these vectors are hard faults. We tabulate the hard faults in Table 4.1. for all the benchmark circuits. Accurate detection probabilities are needed for these hard faults.

Table 4.1. ISCAS benchmarks and the number of faults

Circuits	Faults	Red. flt	Hard FLT
c432	524	4	5
c499	758	8	9
c880	942	0	32
c1355	1574	8	32
c1908	1879	9	114
c2670	2747	117	435
c3540	3428	137	218
c5315	5350	59	78
c6288	7744	34	34
c7552	7550	131	586

We performed experiments using both PLS and EPIS inference schemes. Recall that the Probabilistic Logic-Sampling (PLS) scheme is simple and time-efficient, but cannot handle diagnostic evidence efficiently. However, the Evidence Pre-propagated Importance Sampling Algorithm (EPIS) [57]) can efficiently handle diagnostic evidence, but with increased time for cases when there is evidence. We performed an in-house logic simulation with $500,000$ random vectors to detect the exact fault detection probability of all the faults and used these probabilities to check the accuracy of our model.

The results of detection probabilities computed by Probabilistic Logic Sampling (PLS) [28], and Evidence Pre-propagated Importance Sampling EPIS [57] are shown in Table 4.2.. Results are reported for 1000 samples and 3000 samples. In this table, μ_E is the accuracy of modeling in terms of average error in FDP over all the non-redundant hard faults compared to simulation results. $T(s)$ is the total elapsed time, *including memory and I/O access*. This time is obtained by the ftime command in the WINDOWS environment on a Pentium-4 2.0 GHz PC.. We report the μ_E and $T(s)$ for EPIS in columns 2 and 3, respectively, and the μ_E and $T(s)$ for PLS in columns 4 and 5, respectively.

We partitioned the faults in circuits c3540, c6288 and c7552 into three subsets and determined the detection probabilities in each set by parallel running of the circuits for all the fault sets.

We empirically demonstrate the linear dependence of estimation time on number of samples in Figure 4.2.(a) for benchmark c880. We also present empirical evidence of linear dependence of FDP estimation time on the number of nodes in the fault detection logic for c880 benchmark in Figure 4.2.(b). We considered different subsets of faults in the fault list of the circuit. Figure 4.3. shows the FDP, which can also be used to characterize single-event-transient (SET) error sensitivity, for the nodes in c880.

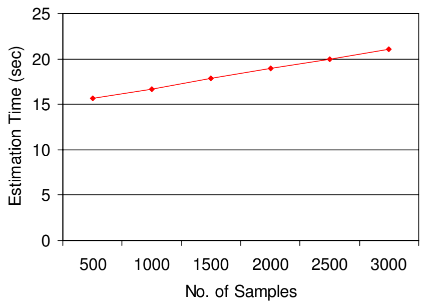
Table 4.2. (a) Fault detection probability estimation errors and time for 1000 samples (b) Fault detection probability estimation errors and time for 3000 samples

	Bayesian Networks (1000 samples)			
	EPIS		PLS	
	μ_E	T(s)	μ_E	T(s)
c432	0.00012	1.09	0.00032	0.24
c499	0.00064	5.11	0.00131	1.00
c880	0.00042	16.70	0.00070	2.00
c1355	0.00059	27.28	0.00093	3.00
c1908	0.00071	61.39	0.00079	5.00
c2670	0.00003	145.19	0.00029	28.00
c3540	0.00034	807.77	0.00215	34.00
c5315	0.00024	186.88	0.00036	12.00
c6288	0.00000	1779.93	0.00929	65.00
c7552	0.00034	863.73	0.00069	37.24

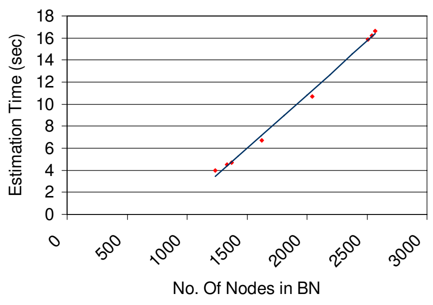
	Bayesian Networks (3000 samples)			
	EPIS		PLS	
	μ_E	T(s)	μ_E	T(s)
c432	0.00012	2.17	0.00039	1.26
c499	0.00012	7.52	0.00094	3.48
c880	0.00033	21.04	0.00051	6.58
c1355	0.00032	32.76	0.00083	8.50
c1908	0.00052	71.28	0.00048	13.60
c2670	0.00001	615.88	0.00026	52.12
c3540	0.00020	848.69	0.00163	64.69
c5315	0.00013	205.42	0.00031	27.00
c6288	0.00000	1823.04	0.00874	109.87
c7552	0.00030	902.24	0.00081	69.14

Table 4.3. Comparison with the state of the art

	BDD [13]		BN	Ratio	
	PSG	SG	PLS		
	T_{CPU} (s) (t_1) (Y93)		T_{total}(s) (t_2) Y(04)	R (t_1/t_2)	R/16
c432	139	71	0.24	295.83	18.49
c499	80	44	1.00	44.00	2.75
c880	328	1132	2.00	164.00	10.25
c1355	157	79	3.00	26.33	1.65
c1908	686	288	5.00	57.60	3.6
c2670	2051	–	28.00	73.25	4.58
c3540	23630	1732	34.00	50.94	3.18
c5315	82	31	12.00	2.58	0.16
c6288	–	–	65.00	–	–
c7552	1281	–	37.24	34.34	2.15
Average				83.22	5.20



(a)



(b)

Figure 4.2. (a) Estimation time vs. number of samples in the detection logic for benchmark c880 (b) Estimation time vs. number of nodes in the detection logic for benchmark c880

Note that some nodes that have high detection probabilities which should have been captured by the initial simulation, however was undetected. Hence, characterization based on simulation suffers from pattern-dependence of the simulative methods.

In Table 4.3., we compare LIFE-BN fault modeling with the performance of approaches based on the Binary Decision Diagram (BDD) model, as reported by Krieger *et al.* [13] for these same circuits. They reported results using four type of fault partitioning. We compare our time (column 4) with the time taken by their two best methods, namely and PSG (column 2) and Super gate SG (column 3). In column 5, we report the ratio between the minimum time taken by the BDD based method and the time taken by our Bayesian network based approach. The average improvement seems to be 83 times. Taking into account of the improvement in processor speed as compared to the year 1993, we assumed a scaling factor of 16 and report the scaled time performance ratio (Actual Ratio/16) in column 6 of Table 4.3..

Even though [13] explains BDD based method of FDP computation, the complete algorithm for different types of decomposition techniques could not be obtained from this paper. Hence we could not give a direct comparison of cpu time between the BDD based model and the LIFE-DAG model by re-implementing their algorithm in the same computer we used for experimenting our model.

Using LIFE-DAG, we model the logical masking effect capturing circuit structure, re-convergence and the inputs. In the following chapter, we discuss another class of errors, known as Single-Event-

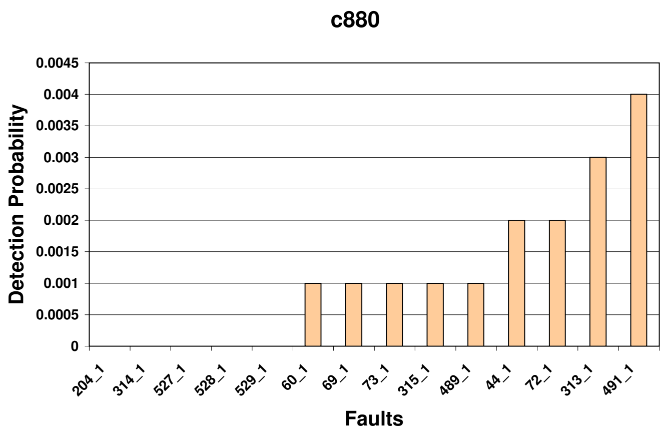


Figure 4.3. Detection probability as SET sensitivity for different nodes in c880

Upsets. To model this type of errors, we need to take into account of their *temporal* nature. In Chapter 5, we describe the modeling of Single-Event-Upsets by capturing logical and temporal masking effects.

CHAPTER 5

A TIMING-AWARE PROBABILISTIC MODEL FOR SEU SENSITIVITIES

Alpha particles and heavy neutrons present in cosmic radiations cause *soft errors* in microelectronic chips. Due to reduction in device sizes and scaling down of operating voltages, stored charges per node in logic circuits will reduce considerably in future technologies. Hence even a weak radiation can cause single-event-upset (SEU) within a circuit. The SEU when propagated to a circuit output and captured by an output latch during the latching window will cause soft error. Hence in deep submicron and future nano-meter technologies, the soft error failure rate will be a major reliability challenge not only in embedded memories but also in logic blocks. Hence it is important to analyze the effect of single event upsets on the circuit output and identify gates that are highly susceptible to soft errors. By doing so, designers can apply selective mitigation schemes targeting only those gates which are highly sensitive to single-event-upsets. in this chapter, we describe a probabilistic model for estimation of single-event-upset sensitivities of gates in large combinational circuits.

Soft error susceptibility of a node j with respect to a latch L, $SES_{j\_Q_L}$ is the soft error rate at the latch output Q_L, contributed by node j. The propagation of an SEU generated due to a particle hit at an internal node j to an output i which causes a bit flip at the output of a latch L is depicted in Fig. 5.1.

We model the SEU propagation as follows: Let T_{j_i} be a Boolean variable which takes logic value 1 if an SEU at a node j causes an error at an output node i. Then $P(T_{j\_i})$ (measured as the probability of T_{j_i} being equal to 1) is the conditional probability of occurrence of an error at output node i given an SEU at node j. Let $P(SEU_j)$ be the probability that a particle hit at node j generates an SEU of sufficient strength and let $P(Q_L|T_{j\_i})$ be the probability that an error at output node i causes an erroneous signal at latch output Q_L. Mathematically $SES_{j\_Q_L}$ is expressed by Eq. 5.1.

$$SES_{j\_Q_L} = R_H P(SEU_j) P(T_{j\_i}) P(Q_L|T_{j\_i}) \tag{5.1}$$

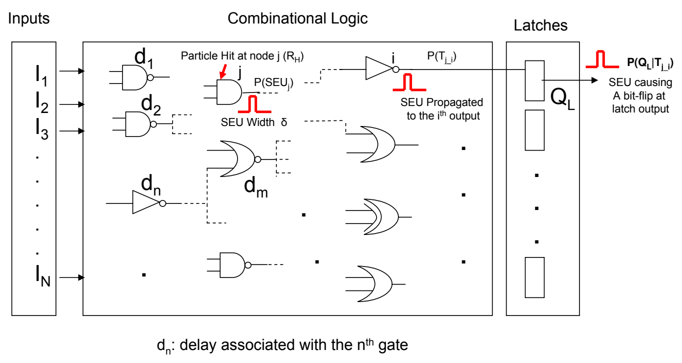

Inputs
Combinational Logic
Latches
I_1
I_2
I_3
I_N
d_1
d_2
d_n
d_m
Particle Hit at node j (R_H)
j
P(SEU_j)
SEU Width δ
i
P(T_{j_i})
SEU Propagated to the i^{th} output
Q_L
P($Q_L|T_{j\_i}$)
SEU causing A bit-flip at latch output
d_n: delay associated with the n^{th} gate

Figure 5.1. SEU Propagation

where R_H is the particle hit rate on a chip which is fairly uniform in space and time. $P(SEU_j)$ depends on V_{dd}, V_{th} and also on temperature. $P(Q_L|T_{j\_i})$ is a function of latch characteristics and the switching frequency.

In this work, we explore $P(T_{j\_i})$ by accurately considering the effect of (1) *SEU duration*, (2) *effect of gate delay* and (3) *timing*, (4) *re-convergence* in the circuit structure and most importantly (5) *inputs*. Several works on soft error analysis estimate the overall output signal errors due to SEUs at the internal nodes [15, 16, 4, 17, 18] . Note that our focus is to identify the SEU locations that cause soft errors at the output(s) with high probabilities and *not* on the overall soft error rates. Knowledge of relative contribution of individual nodes to output error will help designers to apply selective radiation hardening techniques. This model can easily be fused with the modeling of the latches [17, 19] considering parameters such as latching window, setup, hold time, V_{th} and V_{dd} [15, 16, 17] for a comprehensive model capturing processing, electrical and logical effect.

We model internal dependency of the signals taking into consideration timing issues so that the SEU sensitization probability ($P(T_{j\_i})$) captures the effect of circuit structure, circuit path delay and also the input space. We use a circuit expansion algorithm similar to that presented in [41, 63] to embed time-related information in the circuit topology without affecting its original functionality. A fan-out dependent delay model is assumed where gate delay of each node is equal to its fan-out. We also use logical effort based delay model where gate delays are dependent not only on fan-out but also on input capacitance as well as parasitic capacitance. Due to the temporal nature of SEUs, not all of the SEUs cause soft errors. From the expanded circuit, we generate a list of SEUs (possible SEU list) that are possibly sensitized to the circuit outputs at the time frame when output signals are latched. From the expanded circuit and the possible SEU list, we construct an error detection circuit and model SEU in large combinational circuits using a Timing aware Logic induced Soft Error Sensitivity model (TALI-SES), which is a complete joint probability model, represented as a Bayesian Network.

We model the effect of single event upsets produced at an internal node of a circuit on the circuit output, by computing the joint probability distribution described by Eq. 5.2.

$$P(T_{j\_i}) = \sum_{j,\{I_l\},X_k,k\neq j} P(T_{j\_i}, I_0, \cdots I_l, \cdots I_N, X_1, \cdots, X_k, \cdots X_M) \tag{5.2}$$

where $P(T_{j\_i})$ is the probability that an SEU generated at an internal node j causes an erroneous signal at output i. T_{j_i} is a test signal which compares the ideal output signal at the i^{th} output with the corresponding error-sensitized output caused by an SEU at the j^{th} node. If $T_{j\_i} = 1$, it indicates the occurrence of an error at output i due to an SEU at j. $P(T_{j\_i})$ depends on the N input signals $I_0, \cdots, I_N$, M internal signals $X_1, \cdots, X_M$, and the type of SEU at j (SEU^1 caused by 0-1-0 transition or SEU^0 caused by 1-0-1 transition). Ideally, the real effect of SEU at i^{th} output is product of $P(T_{j\_i})$ and $P(SEU_j)$, where $P(SEU_j)$ is the probability that a particle hit at a node j produces an SEU at that node and it depends on process parameters such as V_{dd} and V_{th} and also depend dynamically on temperature. With reduced supply voltages and diminishing dimensions, this probability will be very close to one. In this work, we assume that a particle hit occurring at a node generates an SEU and hence $P(SEU_j) = 1$. In Eq. 5.2, the probability $P(T_{j\_i})$ does not consider the transient nature of SEU. For example, the SEU effect may reach the output for a short time span, but the output signal can be reinforced to its correct value before it is sampled by the latch. SEU propagation depends on the gate delays and SEU duration. Let t_h be the time when an SEU originates at a node, δ be the SEU duration, t_s be the time when outputs are sampled and Π be the set of propagation delays (t_d) of sensitized paths from the node to the circuit outputs. Nodes satisfying the following conditions do not cause soft error [41]:

$$t_h + \delta + t_d < t_s \quad \forall t_d \in \Pi. \tag{5.3}$$

Even though the above empirical formula doesn't take into account of set up and hold time requirements which affect latching window masking, we use this equation for our modeling because this is pretty accurate as far as logical masking effect, circuit structure and gate delays are concerned.

To capture the effect of gate delays and SEU duration, we do a time-space transformation of the original circuit, by means of a circuit expansion algorithm similar to that presented in [41]. Our model captures not only the effect of gate delays, but also effect of difference in path delays (arrival times) between the input signals of gates assuming gate delay is equal to its fan-out. In the expanded circuit, each gate is replicated several times corresponding to the time instants at which the gate output is evaluated. The circuit outputs are also replicated.

Thus each of the random variables in Eq. 5.2 represent a set of variables at different time frames. $I_i = \{I_{i,0}, I_{i,1}\}$ where $I_{i,0}$ is the input signal value of I_i at time instant just before the occurrence of a clock cycle and $I_{i,1}$ is the new input signal after the clock pulse is applied. Signal $\{I_{i,1}\}$ remains the same throughout the clock cycle. $X_i = \{X_{i,t_k}\} \forall t_k$ where t_k is the signal evaluation time.

Only the final output values - output signals arriving at the latching window - are captured by the latch. An SEU is effect-less if it doesn't cause a bit flip in the final outputs. We arrive at a reduced SEU list by considering only those SEUs whose effect reach the final outputs - outputs at the sampling time frames. We also modify the expanded circuit by removing parts of the circuit which do not generate and propagate soft-error-causing-SEUs (discussed in Section 5.1.1).

TALI-SES is a Directed Acyclic Graph we build from the expanded circuit and the reduced SEU list to capture the effect of each SEU at a node to the output. This model consists of the ideal time-space transformed circuit without any SEUs and a set of duplicate logic blocks to propagate the SEU effects. Outputs from the SEU sensitized duplicate blocks are compared with corresponding outputs of the ideal circuit. If those signal values are not the same, it indicates that the SEU causes an error at the output. We discuss TALI-SES construction in Section 5.1.3. The salient features of modeling SEU by Bayesian Network are as follows.

1. We provide a comprehensive model for the underlying error framework using a graphical probabilistic Bayesian Network based model TALI-SES that is causal, minimal and exact.

2. We can model the effect of timing and transient nature of the SEU's along with the accurate modeling of re-convergence in the circuit.

3. This model captures the data-driven uncertainty in the modeling of soft error that can be used where exact input patterns are not known apriori and also can be used by building a probabilistic model in case data traces are available by learning algorithm [65, 66].

4. We infer error probabilities by (1) exact inference that transforms the graph into a special junction tree structure and relies on local message passing scheme and also by (2) smart stochastic non-simulative inference algorithms that have the feature of any-time estimates and generates excellent accuracy time trade-off for larger circuits.

5. Bayesian Networks are unique tool where effect of an observation at a child node can be used to get a probability space of the parents. This is called backward reasoning. Our model can be used to generate input space for which the SEU occurring at a particular node j might have no impact on the outputs. Note that in such case, hardening techniques will not be needed for node j. Similarly, we can find input space for which SEU at a node j cause high error probability at outputs. If the data trace is similar to the second type of input space, extensive hardening techniques need to be applied to j.

5.1 The Proposed Model

In this section, we first focus on handling the timing aware feature of our probabilistic model, followed by the fault list construction. We conclude the section with discussion about the model itself, given the timing-aware graph and the fault list.

5.1.1 Timing Issues

We first expand the circuit by time-space transformation of the original circuit, without changing its functionality. The approach is similar to the method discussed in [41, 63]. Fig. 5.2. is the expanded circuit of benchmark $c17$. A gate in the original circuit C will have many replicate gates in the expanded circuit C', corresponding to different time-frames at which the gate output is evaluated. The output

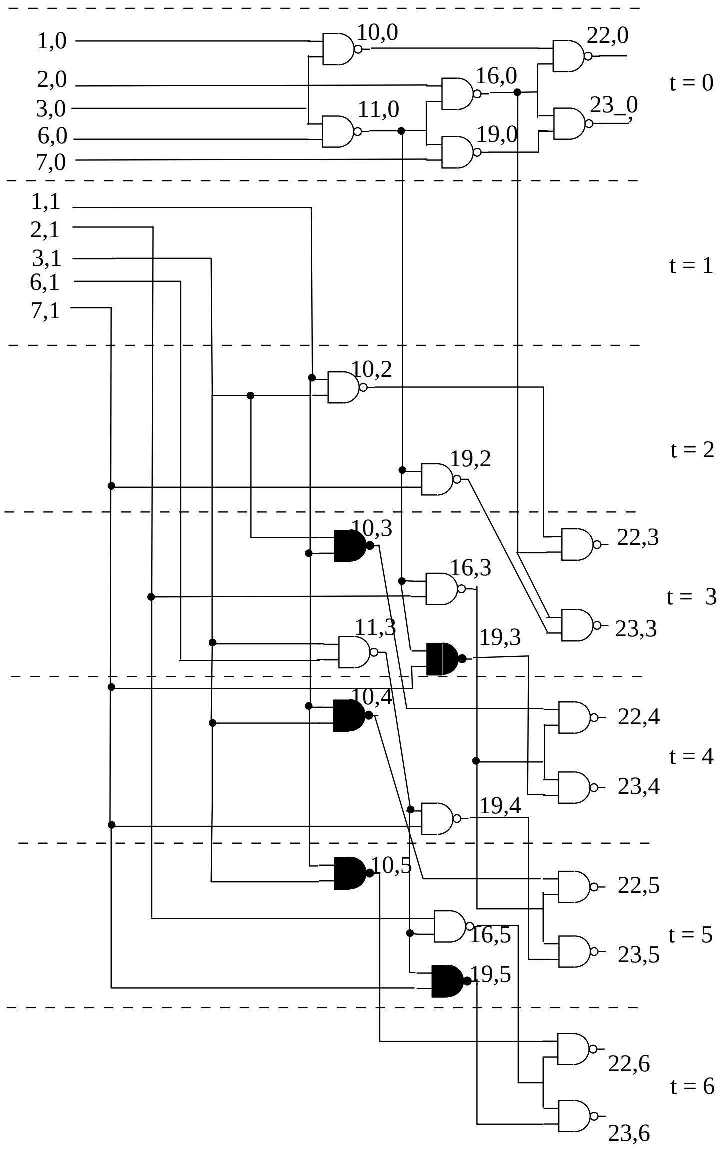


Figure 5.2. Time-space transformed circuit of benchmark c17, modeling all SEUs

evaluation time $\{T\}$ of each gate in the circuit is calculated based on variable delay model. We assume that the delay associated with a gate is equal to its fan-out. For each gate g whose output is evaluated at time $t \in \{T\}$ a replicate node g,t is constructed. In addition to these replicate gates, we insert some duplicate gates (shown by filled gate symbols in Fig. 5.2.) We explain the reasons for adding these duplicate gates later in this section.

The inputs of g,t are the replicate nodes of the gates, which are the inputs of g in the original circuit and belongs to the time-frames $t' < t$. We consider the value of signal i at time t by (i,t). Now the random variable that represents the value of a signal i at time t is denoted by $X_{i,t}$. The circuit outputs reach steady state values, $X_{22,0}$ and $X_{23,0}$ at $t = 0$, after the application of the previous inputs, $\{X_{1,0}, X_{2,0},\ X_{3,0},\ X_{6,0},\ X_{7,0}\}$. Let the new inputs $\{X_{1,1}, X_{2,1},\ X_{3,1},\ X_{6,1},\ X_{7,1}\}$ be applied at $t = 1$. $X_{10,2}$ is the signal value at the output of gate 10 at time instant 2.

We insert a few duplicate gates (example: $(10,4)$, $(10,5)$, $(19,5)$, etc. shown by filled gate symbols) due to the following reasons:

- Input signals of certain gates in the circuit might have different arrival time due to the difference in path delays. In order to model the effect of any SEU generated at the junction of the gates at time instants, later than the signal's arrival time, we insert additional duplicate nodes for those internal signals with less path delay. For example, in Fig. 5.2., input signals to gate 22 have path delays 2 and 5 respectively. The final output signal $(22,6)$ is evaluated with input signals $(16,5)$ and $(10,5)$. If no SEUs originated at the output of gate 10 between time instants 2 and 5, $(10,2)$ and $(10,5)$ would be the same. However, in the event an SEU occurs at node 10 at $t = 5$, $(10,2)$ and $(10,5)$ may differ depending on the inputs, which can cause a wrong output signal at $(22,6)$. We model the effect of SEU at $(10,5)$ by introducing a duplicate gate $(10,5)$ whose inputs are $(1,1)$ and $(3,1)$. Similarly, $(10,3)$, $(10,4)$, $(19,4)$and $(19,5)$ are other duplicate gates.

- Duplicate gates also model the masking effect of some of the SEUs generated in the signal path of the input having lesser path delay. Example: Duplicate gate $(10,5)$ mask the effect of an SEU originated at the output of gate 10, at time $t = 2$. Thus we can arrive at a reduced SEU list which is further explained later in this section.

Steps for constructing the timing-aware expanded circuit, based on fan-out dependent delay model are the following:

1. Arrange gates in the order of levels, with the level of input gates equal to zero.

2. Include all gates that are present in the original circuit. Output signals of these gates represent the steady state signal values at $t = 0$, before the application of new inputs.

3. Add additional input nodes representing new input signal values at $t = 1$;

4. For each level of the circuit starting from level $l_i = 1$, repeat the following step:

   For each gate g in level l_i, create replicate gates at time frame $t = t_p + f_g$, where t_p is the maximum time frame of the previously inserted parent gates of g and f_g is the fan-out of gate g. Update time frames of gate g.

Output signals of a circuit are sampled at $t = t_s$, where t_s is the maximum of the latest signal arrival times of the output signals. SEUs which do not satisfy Eq. 5.3 affect circuit outputs resulting in soft errors. These SEUs are the upsets generated at the output of gates, which are in the fan-in cones of final outputs (outputs evaluated at time t_s). SEUs occurring at certain other gates, which are not in the fan-in cones of the final outputs, may also affect circuit outputs. These nodes arise due to the SEU duration time δ. For example in Fig. 5.2., we see that the final outputs are generated at time instant t=6. If an SEU occurs at signal 19 at 4 ns and lasts for one time unit, it will essentially be capable of tampering the value of node 23 at 6 ns. Note that we assume that δ is one time unit. The fault list will be different if we change the value of δ. Thus we can see that SEUs which are sensitized to outputs at time frames between t_s and $t_s - \delta$ may cause soft errors, depending on the input signals and circuit structure.

Considering the above factors, we modify the expanded circuit by including only those gates that propagate SEUs to the outputs between time instants, t_s and $t_s - \delta$. Thus we get a considerable reduction in the circuit size. Fig. 5.3. is the modified expanded circuit of c17, which models all SEUs possibly sensitized to a final output.

Next, we discuss how to generate a list of possible SEUs affecting the circuit outputs. Not all gates in Fig. 5.3. are SEU sensitive. As discussed above, a duplicate node introduces an additional delay of

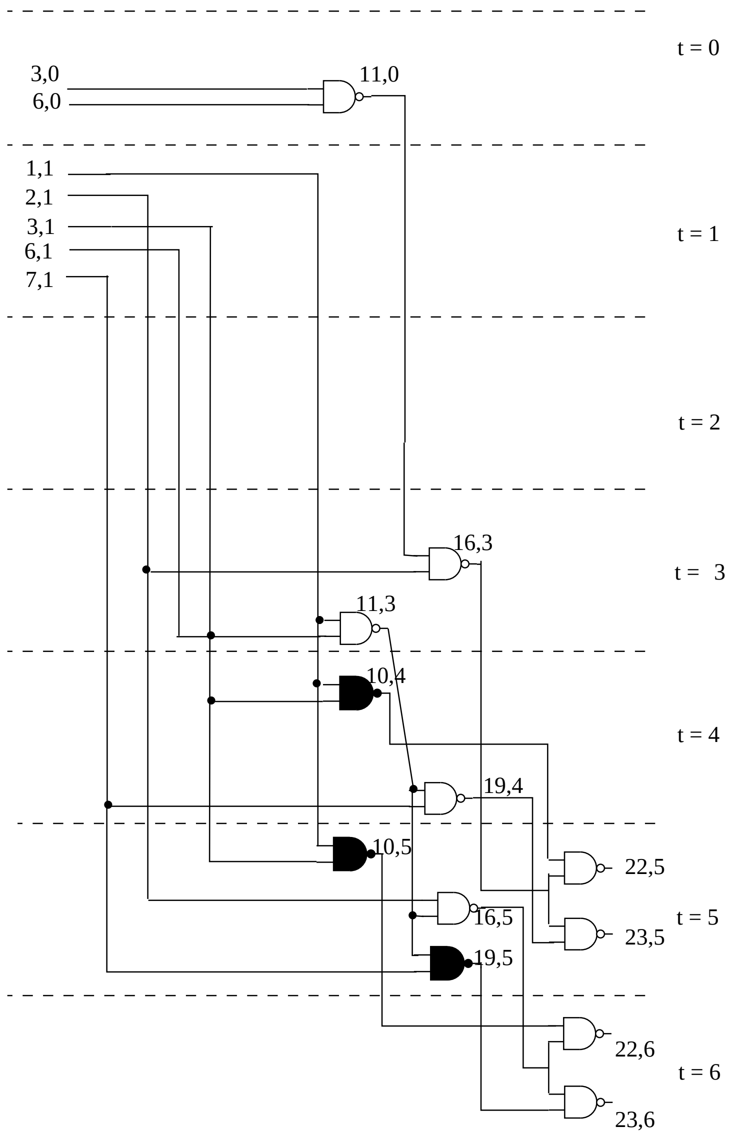


Figure 5.3. Modified time-space transformed circuit of benchmark c17, modeling only the possibly sensitized SEUs

at least one time unit. If the delay introduced by a duplicate gate is greater than or equal to δ, the SEU duration time, the effect of SEUs originated at any of the gates in the fan-in cones of the duplicate gate is nullified and correct signal value is restored at the output of the duplicate gate, and hence those SEUs are effect-less. Thus we create a reduced list of SEUs by traversing the modified extended circuit from each of the circuit outputs at time instants between t_s and $t_s - \delta$, until a duplicate gate or an input node is reached.

5.1.2 Delay Modeling Based on Logical Effort

We extend this work by using logical effort based model which is dependent on fan-out, input capacitance as well as parasitic delay. In this section we explain how gate delays are calculated based on logical effort [64]. Delay of a logic gate can be expressed as the sum of two components, effort delay and parasitic delay. effort delay is the product of logical effort and electrical effort, where logical effort is defined as the relative ability of a gate topology to deliver current and electrical effort is the ratio of output capacitance to input capacitance. Electrical effort is sometimes called fan-out. Mathematically, gate delay is expressed as $d = f + p = gh + p$ where f is effort delay, p is the parasitic delay, g is the logical effort and h is electrical effort. Logical effort is defined to be 1 for an inverter. Hence logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter delivering the same output current. It can be estimated counting capacitance in units of transistor width. Parasitic delay represents delay of a gate driving no load and it depends on diffusion capacitance. parasitic delay of an inverter, $P_{inv} \approx 1$. From the above considerations, we compute basic CMOS gate delays and use these delay values in our model. Table 5.1. shows the delay expressions for basic gates.

Table 5.1. Gate delays based on logical effort

Gate Type	Delay
Inverter	$fanout + P_{inv}$
n-input NAND	$\frac{n+2}{3} * fanout + nP_{inv}$
n-input NOR	$\frac{2n+1}{3} * fanout + P_{inv}$
2-input XOR	$4 * fanout + 4nP_{inv}$

Circuit expansion is performed in a similar way as explained in the above section. Each gate is replicated several times corresponding to the time frames at which new gate output signals are evaluated. Here, gate output evaluation time is based on delay values calculated as above. This is illustrated in Figure 5.3. which shows how benchmark circuit $c17$ is expanded with logical effort based gate delay model. Delay of a 2-input nand gate with one fan-out is calculated as 3.33 time units and that of a 2-input gate nand gate with 2 fan-out is 4.67 time units. Final output is evaluated at time unit $T_s = 13.67$. From this expanded circuit, we arrive at a reduced circuit by traversing backward from outputs evaluated at T_s and $T_s - \delta$ until a duplicate gate or an input is reached, thereby modeling only the possibly sensitized SEUs.

5.1.3 TALI: Timing-Aware-Logic-Induced SEU Sensitivity Model

In this section, we first describe the proposed Bayesian network based model, which can be used to estimate the soft error sensitivity of logic blocks. This model captures the dependence of SEU sensitivity on the input pattern, circuit structure and the gate delays. Note that this probabilistic modeling does not require any assumptions on the inputs and can be used with any biased workload patterns. The proposed model, Timing-Aware-Logic-Induced-Soft-Error-Sensitivity (TALI-SES) Model is a Directed Acyclic Graph (DAG) representing the time-space transformed, SEU-encoded combinational circuit, $< C', J >$ where C' is the expanded circuit created by time-space transformation as discussed in section. 5.1.1 and J is the set of possible SEUs (also discussed in section 5.1.1). The error detection circuit consists of the expanded circuit $C^{'}$, an error sensitization logic for each SEU and a detection unit T consisting of several comparator gates. We explain it with the help of a small example shown in Fig 5.5.(a), which is the error detection circuit for a small portion of benchmark c17. The error sensitization logic for an SEU at node j consists of the duplicate descendant nodes from j. In Fig. 5.5.(a), the block with the dotted square is the sensitization logic for $16, 5_s^1$ [An SEU^1 at node 16 at time $t = 5$]. It consists of nodes $22, 5_s$ and $22, 6_s$ descending from node $16, 5$ of the time-space transformed circuit. For simplicity, we show the modeling of only one SEU in this example. Our model can handle any number of SEUs simultaneously. Each SEU sensitization logic has an additional input to model the SEU. Example: input $SEU_{16,5}^1$. This

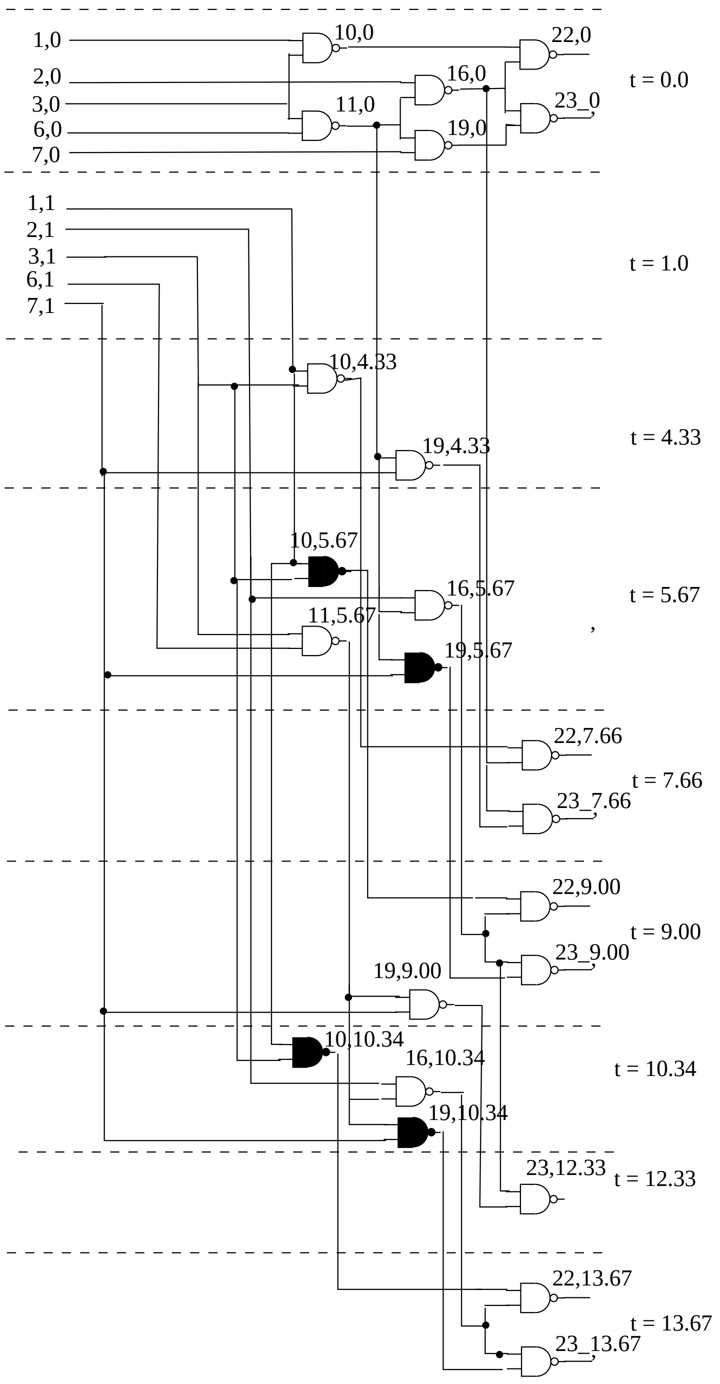


Figure 5.4. Time-space transformed circuit of benchmark c17 with logical effort based delay model

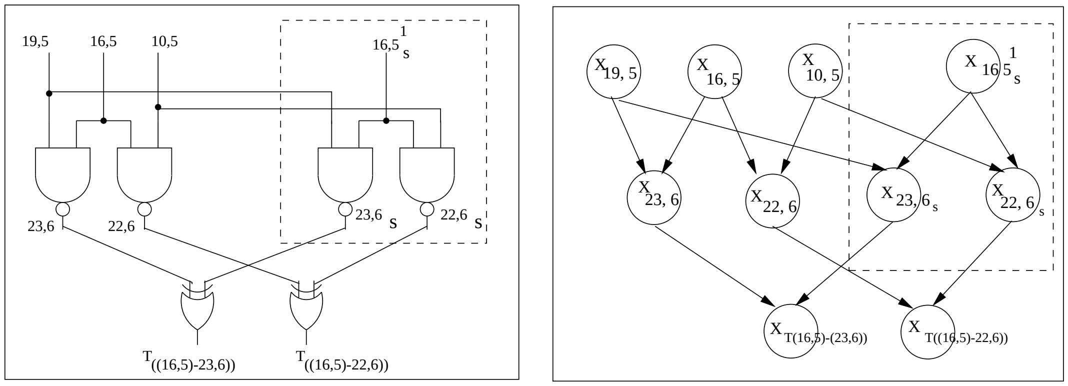


Figure 5.5. (a) An illustrative SEU sensitivity logic for a subset of c17 (b) Timing-aware-logic-induced-DAG model of the SEU sensitivity logic in (a)

input signal value is set to logic one in order to model the effect of a 0-1-0 SEU occurring at node 16 at time frame 5.

As discussed previously in section 5.1.1, an SEU lasting for a duration δ can cause an erroneous output if its effect reaches the output at any instant between the sampling time t_s and time frame $t_s - \delta$. In this work we assume δ to be one. Hence we get error sensitized outputs at time frame t_s and for some SEUs at $t_s - 1$ also, if there exist re-convergent paths between SEU location and an output. We need to compare the SEU-free output signals evaluated at the sampling time, t_s with the corresponding SEU-sensitized output signals arriving at $t_s - 1$ and t_s. Hence these signals are sent to a detection unit T. The comparators in the detection unit compare the ideal and error sensitized outputs with the corresponding error-free outputs and generate test signals. For example, the test signals for an SEU at node j at time t are $T_{(j,t)\_(i,t_s)}$ and $T_{(j,t)\_(i,t_s-1)}$. If any of these the test signal value is 1, it indicates the occurrence of an error. The probability $P(T_{(j,t)\_i})$, which is a measure of the effect of SEU $(j,t)_s$ on the output node i is computed as a joint probability which is explained below:

Let A be an event that an SEU at node j causes a bit-flip at output i at time t_s and let B be an event that an SEU at node j causes a bit-flip at output i at time $t_s - 1$. $P(A = 1)$ is the probability of occurrence of error and at time t_s. $P(A = 0)$ is the probability that SEU doesn't cause an error at t_s. $P(B)$ can be explained in a similar way. The Error probability due to an SEU at node j at time t w.r.t. output i is the joint probability

$$P(A \cup B) = P(A=1, B=0) + P(A=0, B=1) + P(A=1, B=1) \tag{5.4}$$

which is expressed as:

$$P(T_{(j,t)\_i}) = P(T_{(j,t)\_(i,t_s)} \cup T_{(j,t)\_(i,t_s-1)}). \tag{5.5}$$

An SEU can have effect on more than one output. The overall effect of an SEU $(j,t)_s$ on the outputs is computed as $P(T_{(j,t)}) = \max_{\forall i}\{P(T_{(j,t)\_i})\}$. In the example the SEU $(16,5)_s$ is sensitized to outputs 22,6 and 23,6. Hence the two test signals for this SEU are $T_{(16,5)\_(22,6)}$ and $T_{(16,5)\_(23,6)}$.

An SEU occurring at node j at time t, which is either SEU^1 or SEU^0 *(but not both)*,can cause a bit-flip at the output with probability $P(T^1_{j,t})$ or $P(T^0_{j,t})$. In order to compute the SEU sensitivity of a node, we take the worst case probability, which is the maximum of the above two probabilities. $P(T_{j,\,t}) = \max\{P(T^1_{(j,t)}), P(T^0_{(j,t)})\}$

More than one SEU can originate at a node at different time frames. Considering the effect of SEUs at node j at all time frames, we compute the worst case output error probability due to node j as $P(T_j) = \max_{\forall t}\{P(T_{(j,\,t)})\}$, which is the maximum probability over all time frames.

These detection probabilities depend on the circuit structural dependence, the inputs, dependencies amongst the inputs, gate delays and the SEU duration. In this work we assume random inputs for experimentation and validation of our model.

We construct the TALI-SES Bayesian Network of the SEU detection circuit by nodes which are random variables representing signal values of the SEU detection circuit. A signal i in the detection circuit is represented by the random variable X_i in the Bayesian Network.

In TALI-SES DAG structure the parents of each node are its Markov boundary elements. Hence the TALI-SES is a boundary DAG. For definition of Markov Boundary and boundary DAG, please refer to [26]. Note that TALI-SES is a boundary DAG because of the causal relationship between the inputs and the outputs of a gate that is induced by logic. It has been proven in [26] that if graph structure is a boundary DAG D of a dependency model M, then D is a minimal I-map of M ([26]). This theorem along with definitions of conditional independencies, in [26] (we omit the details) specifies the structure of the Bayesian network. Thus TALI-SES DAG is a minimal I-map and thus a Bayesian network (BN).

Quantification of TALI-SES-BN: TALI-SES-BN consists of nodes that are random variables of the underlying probabilistic model and edges denote direct dependencies. All the edges are quantified with the conditional probabilities making up the joint probability function over all the random variables. The overall joint probability function that is modeled by a Bayesian network can be expressed as the product of the conditional probabilities. Let us say, $X^{'} = \{X_1^{'}, X_2^{'}, \cdots, X_m^{'}\}$ are the node set in TALI-SES Bayesian Network, then we can say

$$P(X^{'}) = \prod_{k=1}^{m} P(x_k^{'}|Pa(X_k^{'})) \tag{5.6}$$

where $Pa(X_k^{'})$ is the set of nodes that has directed edges to $X_k^{'}$. A complete specification of the conditional probability of a two input AND gate output will have 2^3 entries, with 2 states for each variable. These conditional probability specifications are determined by the gate type. By specifying the appropriate conditional probability we ensure that the spatial dependencies among sets of nodes (not only limited to just pair-wise) are effectively modeled.

We demonstrate the modeling of SEU based on TALI-SES using ISCAS benchmark circuits. The logical relationship between the inputs and the output of a gate determines the conditional probability of a child node, given the states of its parents, in the TALI-DAG.

5.2 Experimental Results

In Table 5.2. we report the total number of gates in the actual circuit (column 2), total number of gates in the modified expanded circuit (column 3), and the total number of nodes in the resulting TALI-SES (column 4). Column 5 lists the maximum time-frames of the circuits.

We compute the SEU sensitivity of an individual node $P(T_j)$ in a circuit as follows:

1. Compute the output error probability at output node *i* due to an SEU at node j at time t by taking the joint probabilities as discussed in section 5.1 5.1.3.

$$P(T_{(j,t)\_i}) = P(T_{(j,t)\_(i,t_s)} \cup T_{(j,t)\_(i,t_s-1)}) \tag{5.7}$$

Table 5.2. Size of original and time-expanded ISCAS circuits for fanout-dependent delay model

	Gates	Gates expanded	# of nodes (TALI)	Time frames
c432	196	476	1989	55
c499	243	464	1596	30
c880	443	729	2552	51
c1355	587	1440	3388	55
c1908	913	1524	18118	79
c2670	1426	2584	4097	81
c3540	1719	3795	15670	93
c5315	2485	4887	13228	90
c6288	2448	30113	31157	263
c7552	3719	10006	45907	88

Table 5.3. Estimated $P(T_{j\_i})$ values of nodes in benchmark c17 from exact inference

node j	SEU^1		SEU^0	
	$P(T_{j\_22})$	$P(T_{j\_23})$	$P(T_{j\_22})$	$P(T_{j\_23})$
10	0.2813	0	0.4375	0
11	0.0625	0.2344	0.3125	0.6563
16	0.3125	0.1875	0.4375	0.4375
19	0	0.375	0	0.4375
22	0.4375	0	0.5625	0
23	0	0.4375	0	0.5625

2. Considering the effect of all SEUs at node j at all possible time frames, compute the probability of occurrence of an error at the i^{th} output due SEUs at node j by Eq. 5.8.

$$P(T_{j\_i}) = \max_{\forall t}\{P(T_{(j,t)\_\,i})\} \tag{5.8}$$

3. Compute the worst case SEU sensitivity of a node j due to an SEU^1 and SEU^0 and all for outputs by Eq. 5.9

$$P(T_j) = \max_{\forall i}\{P(T^1_{j\_i}), P(T^0_{j\_i})\} \tag{5.9}$$

5.2.1 Exact Inference

In this section, we explore a small circuit c17, with exact inference where we transform the original graph into junction tree and compute probabilities by local message passing between the neighboring cliques of the junction tree as outlined in section 3.13.2. Note that this inference is proven to be exact [26, 59](zero estimation error).

Table 5.3. tabulates the results of the TALI-SES of benchmark c17 using the exact inference. In this table, we report the probabilities of error at output nodes 22 and 23 due an SEU at each node j (column 1) namely $(10,\ 11,\ 16,\ 19,\ 22\ and\ 23)$. Column 2 and 3 of Table 5.3. give error probabilities due to SEU^1 (0-1-0 transition) at output nodes 22 *and* 23 respectively. Similarly 4 and 5 give error probabilities due to SEU^0 (1-0-1 transition) at output nodes 22 *and* 23 respectively. We compare the error-free outputs at 22 and 23 at sampling time t_s with corresponding error sensitized outputs arriving at time frames $t_s - 1$ and t_s due to SEUs generated at a node at all possible time frames (as discussed in section 5.1 5.1.3). Columns 2, 3, 4 and 5 of Table 5.3. reports the maximum of error probabilities due to SEUs originated at individual nodes at all time frames. From this table it can be seen that for this benchmark circuit SEU^0s have high impact on the output error probabilities than SEU^1s. Error probability at output node 22 due to an SEU^1 at node 11, is very low (0.0625) whereas error probability at output 22 due to SEU^0 at 11 is 0.3125. It also shows that the effect of SEUs are not the same over all outputs. For example, an SEU^1 at node 19 causes no error at output 22 whereas error probability due to this SEU at output node 23 is 0.4375. Note that nodes 22 and 23 are the output nodes. SEUs occurring at these nodes at sampling time t_s or time $t_s - 1$ will be latched by an output latch, and are expected to cause very high error probability. However from Table 5.3., it is observed that probability of occurrence of an error due to SEU^1 at node 23 is only 0.4375. Similarly, probability of occurrence of an error due to SEU^1 at node 22 is also 0.4375. This is due to the type of input pattern. In this work, we assume random inputs. This result shows the dependence of input pattern on $P(T_{j\_i})$.

5.2.1.1 Input Space Characterization

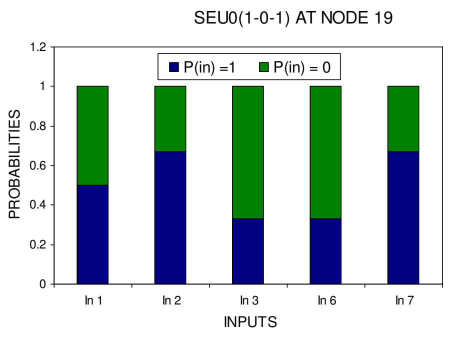
In this section, we describe the input space characterization for a particular observation exploring the diagnostic (backtracking) feature of the TALI-SES model. Note that this feature makes it really unique as instead of predicting the effect of inputs and SEU at a node on the outputs, we try to answer queries like “What input behavior will make SEU at node j definitely causing a bit-flip the at circuit outputs?” or “What input behavior will be more conducive to no error at output given that there is an SEU at node j?” Resolving queries like this, aids the designer in observing the input space and helps perform input clustering or modeling.

Let us take an example of c17 benchmark. We explore the input space for studying the effect of SEU^0 and SEU^1 at node 19 on errors on both the outputs (22 and 23). One can characterize input space for any one of the outputs (or in general effect of SEU at any node on any other subset of nodes). Fig 5.6.a characterizes the input space for an SEU^0 at node 19 such that no bit-flip occur at the outputs. This is done by setting the output error probability at zero (by giving “evidence” to the detection nodes in the Bayesian Network) and then back propagating the probabilities. We plot the probabilities of each inputs $1, 2, 3, 6$ *and* 7 that gives no output error for an SEU^0 at 19. Each column in the plot represents an input. The lighter color represents the probability of that input $= 0$ and the black color represents the probability of input $= 1$ (sum of these two part should always be *one*). One can see that for obtaining zero output error with an SEU^0 at 19, input 1 can be random, input 2 and 7 have 65% probability of being at logic one and node 3 and 6 has probability of 30% for logic 1. Note that the input space is nearly random (p(1)=p(0)=0.5) when SEU^1 at node 19 produces zero output error at both the outputs. Similar characteristics are shown in Fig. 5.6.c, 5.6.d for characterizing the input space with respect to output errors while SEU^0 or SEU^1 occurs at node 11. Once again it can be seen that zero output error for SEU^1 can be more likely by a random inputs than for SEU^0.

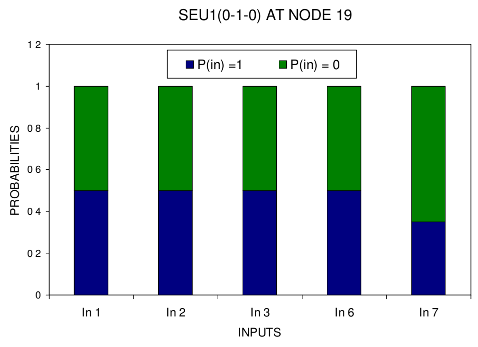
5.2.2 Larger Benchmarks

We use approximate inference for larger circuit using Probabilistic Logic sampling [28] which is pattern independent random markov chain sampling and has shown good results in many large industry-size applications.

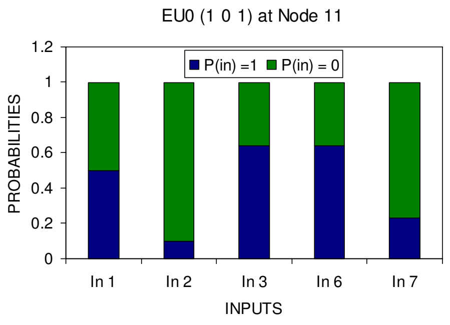
In Fig. 5.7.(a), we plot the number of gates and the number of possibly sensitized SEUs for ISCAS benchmarks. This reduced SEU list was created based on fanout-dependent delay model and assuming an SEU duration δ equal to one time unit. We get a considerable reduction in the number of listed



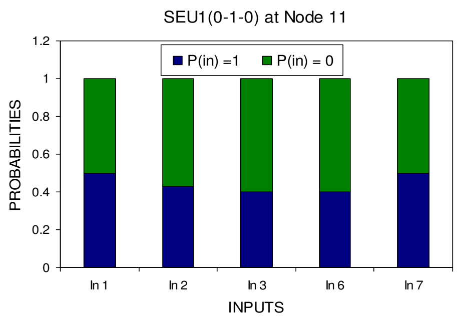
(a)



(b)



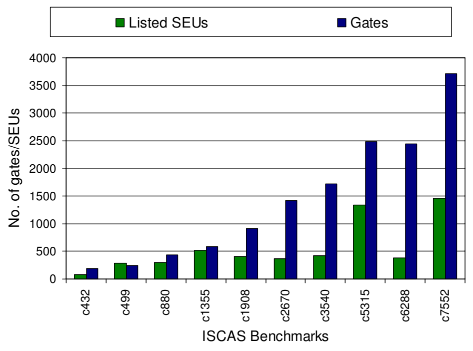
(c)



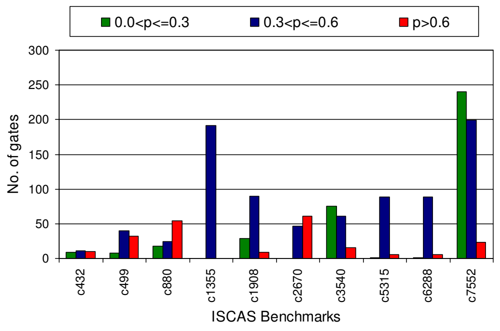
(d)

Figure 5.6. Input probabilities for achieving zero output errors (at nodes 22 *and* 23 in presence of SEU's: (a) SEU^0 at node 19 (b) SEU^1 at node 19 (c) SEU^0 at node 11 (d) SEU^1 at node 11 for c17 benchmark

SEUs compared to the number of gates in a circuit. This is because reduced SEU list is generated by traversing backward from the final outputs evaluated at sampling time t_s and $t_s - 1$ and only those gates that lie between the final outputs and duplicate gates need to be considered for SEU sensitivity analysis. Depending on the input pattern and the circuit structure, only a few of these SEUs actually cause soft errors. Based on the estimated SEU sensitivity $P(T_j)$ calculated as in Eq. 5.9 we classify the SEU sensitive gates in a circuit into three categories, gates where $P(T_j)$ is (i) less than or equal to 0.3 (ii) between 0.3 and 0.6 and (iii) above 0.6. This is plotted in Fig. 5.7.(b) These results are helpful to apply selective redundancy measures or to modify $P(SEU_j)$ (by changing device features) by giving higher



(a)



(b)

Figure 5.7. (a) SEU list-fanout dependent delay model (b) SEU sensitivity range-fanout dependent delay model, with delta=1; input bias=0.5

Table 5.4. SEU sensitivity estimation errors and time for 9999 samples

	(E_{mean})	(E_{max})	$T_{bn}(sec)$
c432	0.0031	0.0069	18.57
c499	0.0024	0.0198	13.43
c880	0.0027	0.0090	27.58
c1355	0.0027	0.0120	28.84
c1908	0.0028	0.0120	176.63
c2670	0.0034	0.0130	34.70
c3540	0.0023	0.0101	148.07
c5315	0.0045	0.0112	121.62
c7552	0.0035	0.0100	513.05

priority to nodes those are in the high sensitivity range than those in the lower sensitivity ranges. From Fig. 5.7.(b), it can be seen that the SEU sensitive nodes of circuit c432 are equally distributed within the three probability ranges (i), (ii) and (iii), whereas all the SEU sensitive nodes in circuit c1355 lie within the middle range where $P(T_j)$ is between 0.3 and 0.6. Results of c7552 shows that $P(T_j)$ of most of the SEU sensitive nodes is in the lowest range (less than or equal to 0.3), which indicates that gates in this circuit do not require extensive hardening techniques, whereas majority of SEU sensitive gates in c2670 requires extensive hardening techniques since $P(T_j)$ is very high (above 0.6) for these nodes.

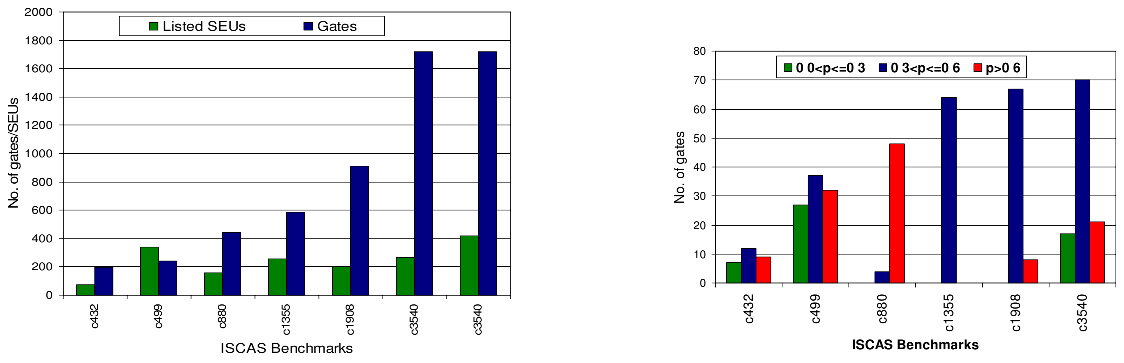
We implemented the SEU simulator based on the work done in [41] with a fanout-dependent delay model for the ground truth. We performed the simulation with $500,000$ random vectors obtained by changing seed after every 50000 vectors to get the ground-truth SEU probabilities. For our probabilistic framework, we use Probabilistic Logic Sampling [28] inference scheme. We compute the SEU sensitivities P_j of gates in ISCAS benchmark circuits using Probabilistic Logic Sampling (PLS) [28] with 9999 samples and compare our results with ground-truth simulation results. Table 5.4. gives the average estimation error E_{mean} [in column 2] and maximum estimation errors E_{max} [in column 3]. Here E_{mean} of a circuit is the average of difference between the *SEU* detection probabilities (or *SEU* sensitivities) obtained from simulation and estimated probabilities from PLS sampling over all possible SEU sensitive nodes in the circuit. Similarly E_{max} of a circuit is the maximum of difference between the *SEU* sensitivities obtained from simulation and estimated *SEU* sensitivities from PLS sampling over all possible SEU sensitive nodes in the circuit. Estimation time, T_{bn} [column 4] is the time taken by the PLS scheme for belief propagation. We estimated the SEU sensitivities all the ISCAS'85 benchmarks with an average belief propagation time of 140.49 sec, whereas the average time taken for logic simulation of these circuits is 33 hours. Estimation error over all benchmarks is below 0.0034 which shows excellent accuracy-time trade-off. $T_{bn}$is the total elapsed time, *including memory and I/O access*. This time is obtained by the ftime command in the WINDOWS environment on a Pentium-4 2.0 GHz PC. It is evident from the results that using a graph-based causal, compact probabilistic framework, Bayesian Network, we are able to accurately model the Single-event-upset (SEU) sensitivities of logic circuit signals accounting for temporal and spatial dependencies. The exciting feature of this stimulus-free approach is that it uses conditional independencies in modeling spatial correlations and time-space transformation for capturing temporal dependencies.

5.2.3 Results with Delay Model Based on Logical Effort

In this section we give estimation results from our model with logical effort based gate delay modeling. In Table 5.5., we list the number of nodes in TALI Bayesian network and the estimation time in seconds for some of the ISCAS benchmarks. Number of TALI nodes depends on the SEU list as well as the circuit size, whereas estimation time directly depends on the number of nodes and the number of samples. We show results for Probabilistic Logic Sampling (PLS) with 9999 samples.

Table 5.5. Size of TALI model and estimation time for logical-effort based delay model

	# of nodes (TALI)	Estimation Time(s)
c432	2390	22.32
c499	7814	65.75
c880	1097	12.49
c1355	1773	15.092
c1908	2279	22.22
c3540	14370	135.79



(a) (b)

Figure 5.8. (a) SEU list-logical effort delay model (b) SEU sensitivity range-logical effort delay model with delta =1 and input bias = 0.5

Figure 5.8.(a) shows the number of possibly sensitized SEUs vs. the number of gates in ISCAS benchmarks. From this graph, it can be seen that the number of SEUs in the reduced SEU list is low compared to fanout dependent delay model. This is due to high gate delay values with logical effort based delay modeling since we take into account the input capacitance as well as parasitic delay in addition to fanout. Due to increased gate delays the relative effect of an SEU at an internal gate on a primary output during latching period is less since most of the signals get enough time to restore to their ideal values. Figure 5.8.(b) shows the SEU sensitivity ranges of gates in the circuits, with an input bias of 0.5 and SEU width equal to one time unit. As with fanout-dependent delay modeling, here also we classify the SEU sensitive gates in a circuit into 3 categories. Gates with estimated sensitivity values (1) less than 0.3, (2) between 0.3 and 0.6 and (3) above 0.6. Given any delay library for a logic circuit, our model can be used to classify the gates in the circuit in the order of their SEU sensitivity values capturing logical masking effect, circuit structure, input pattern and SEU duration.

Please note the above estimated probability values are relatively high when we consider the overall soft error susceptibility of individual gates. To get a comprehensive model, the electrical masking effect, latching window masking effect and also the SEU generation and propagation characteristics of individual gates are to be incorporated with our model. Modeling electrical masking effect needs circuit level simulation techniques, which we are trying to integrate with our current approach as a future direction.

We are able to effectively model Single-event-Upsets in logic circuits (ISCAS benchmarks) to estimate the SEU sensitivity of individual nodes in a circuit capturing spatial and temporal signal correlations, specially emphasizing the effect of inputs, gate delay, SEU duration and circuit structure. We show results with exact and approximate inferences. Using exact inference we characterize input space which gives zero output error even in the presence of some SEUs. Results from approximate inference shows excellent accuracy-time trade-offs. Future effort includes modeling with biased input patterns, for different SEU width δ and also for other delay models, to study the effect of these factors on SEU sensitivities. We are also investigating on the the effect of threshold voltage and supply voltage on the electrical masking effect on transient pulses caused by particle bombardment.

In the following chapter, we discuss the modeling and analysis of the third class of errors, known as dynamic errors which will be predominant in nano-meter technologies. Similar to Single-Event-Upsets, dynamic errors are also transient in nature. These are caused by temporary mal-function of devices since they are forced to operate near their thermal limits. However, dynamic errors are not localized events as SEUs. They exists everywhere in the circuits, since every gate will operate with a finite error probability. Hence in future technologies, computing will become probabilistic rather than deterministic. In Chapter 6, we discuss the modeling of dynamic errors using Bayesian networks and analyze the effect of individual gate errors on the overall circuit outputs.

CHAPTER 6

PROBABILISTIC ERROR MODELING OF NANO-DOMAIN LOGIC CIRCUITS

As technology scales down from deep submicron to nano-meter levels, circuit reliability is greatly affected due to various noise sources. Devices will operate erroneously due to supply voltage bounds, high sub threshold leakage currents, high switching, high operating temperatures, etc. We refer to this compound noise effects as ”dynamic errors”. Every gate in a logic circuit will have a finite dynamic error probability. We propose a novel technique to model the effect of dynamic errors in scaled nano-meter circuits and to estimate the dynamic error sensitivity of individual gates in the circuit.

We estimate the overall error probability at the output of a logic block where individual logic elements are subject to error with a finite probability, p. We construct the overall BN representation based on logic level specifications by coupling gate level representations. Each gate is modeled using a conditional probability table (CPT) which models the probability of gate output signal being at a logic state, given its input signal states. An ideal logic gate with no dynamic error has its CPT derived from the gate truth table, whereas the CPT of a gate with error is derived from its truth table and the error probabilities, which can be input dependent. The overall joint probability model is constructed by networking these individual gate CPTs. This model captures all signal dependencies in the circuit and is a minimal representation, which is important from a scalability perspective.

Table 6.1. Probabilistic representation of the “truth-table” of a two input, error-free, AND gate and for an AND gate with dynamic errors

Ideal AND gate			
X_{i1}	X_{i2}	$P(X_o \mid X_{i1}, X_{i2})$	
		$X_o = 0$	$X_o = 1$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

AND gate with dynamic error			
X_{i1}	X_{i2}	$P(X_o \mid X_{i1}, X_{i2})$	
		$X_o = 0$	$X_o = 1$
0	0	1-p	p
0	1	1-p	p
1	0	1-p	p
1	1	p	1-p

We measure the error with respect to the ideal logical representation. Suppose, we have logic block with inputs $Z_1, \cdots Z_N$, internal signals $X_1, \cdots X_M$, and outputs $Y_1, \cdots Y_K$. Let the corresponding versions of the internal lines and the outputs with dynamic errors be denoted by $X_1^e, \cdots X_M^e$ and $Y_1^e, \cdots Y_K^e$, respectively. Thus, the error at the ith output can be mathematically represented as the XOR of the error-free and the output with error.

$$E_i = Y_i^e \oplus Y_i \tag{6.1}$$

We propose the output error probability $P(E_i = 1) = P(Y_i^e \bigoplus Y_i = 1)$ as a design metric, in addition to other traditional ones such as area or delay, to vet different designs in the nano-domain. Note that the probability of output error is dependent on the individual gate error probability p and also on the internal dependencies among the signals that might enhance or reduce the overall output error based on the circuit structure. In this work,*we prove that for causal logical circuits, these dependencies can be modeled by a Bayesian Network, which is known to be the exact, minimal probabilistic model for the underlying joint probability density function (pdf)*. Probabilistic belief propagation on these Bayesian Networks can then to be used to estimate this probability.

Contributions of this work are summarized as follows:

1. We propose an exact probabilistic representation of the error model that captures the circuit topology for estimating the overall exact output error probabilities for small circuits

2. We use scalable pattern insensitive stochastic inference schemes for estimation of approximate output error probabilities of medium sized benchmarks (ISCAS'85). These inference schemes give excellent accuracy-time trade-off.

3. To compute sensitivities of individual gate errors to the overall output error of a circuit, which is useful for selective redundancy application.

4. We use the estimated overall output error probabilities as a design matrix for comparing equivalent designs.

5. To characterize the input space for achieving a desired output behavior in terms of error tolerance by utilizing the unique backtracking feature of Bayesian Networks.

6. For reliability enhancement by applying selective redundancy techniques and to evaluate reliability/redundancy trade-off.

Rational for using Bayesian Model:

We propose to use Bayesian modeling and inferencing for the following reasons

- Conventional logic computing is inherently causal. For example outputs of a gate is influenced by the inputs where as inputs are not changed by the outputs. This paradigm of conventional computing translates into a causal probabilistic model where the causality forces the probabilistic model to a *directed graph structure*.
- Nano-CMOS, CNT, RTD and Clocked QCA do exhibit causal flow in processing information.
- Certain dependencies observed in causal networks induced and non-transitive dependencies (more on theoretical formalism section), are not captured most effectively By Bayesian Networks [26] and makes it a superior model for causal networks over MRF and other non graph-based algorithms.
- As we know all these probabilistic models are NP-hard, average case complexity of Bayesian Network is the least because it not only models the conditional independencies but exploits it to its advantage in probabilistic inference. More interestingly, the approximate BN models have found faster and easier convergence and has potential to infer for realistic network sizes.
- Last but not the least Bayesian Network are uniquely capable to solve the inverse problem and are capable of answering questions like "What are probabilistic input profile that guarantees zero error at the output?". This makes the most dominant probabilistic tool for diagnosis.

6.1 Probabilistic Error Model

We compute the error probability $P(e_i) = P(Y_i^e \bigoplus Y_i = 1)$ by marginalizing the joint probability function over the inputs, internal lines, and the outputs[1].

$$P(e_i) \quad = \sum_{z_1,\cdots z_N} P(e_i|z_1,\cdots z_N)P(z_1)\cdots P(z_N) \tag{6.2}$$

[1]In this work, $P(x)$ denotes the probability of the event $X = x$, i.e. P(X=x).

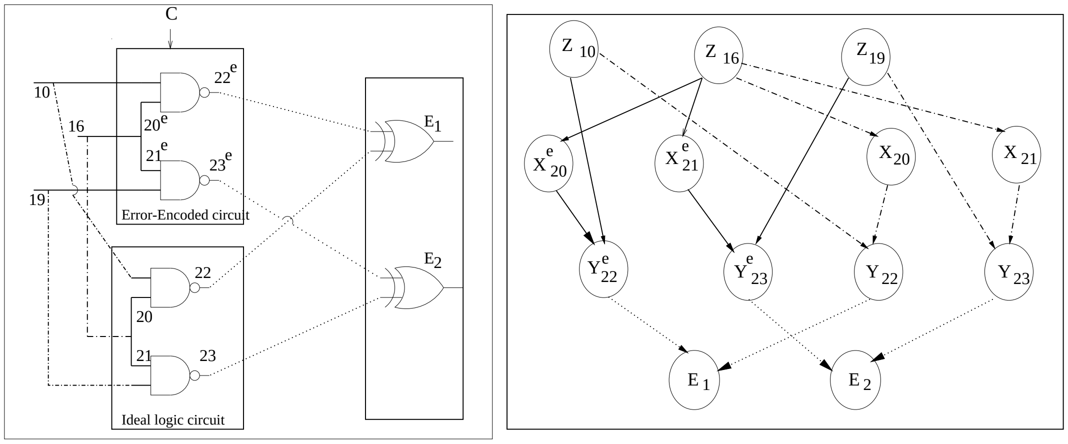


Figure 6.1. (a) Conceptual circuit representation of the logic used to detect errors involving the error-free logic and the unreliable logic components (b) The corresponding Bayesian network representation

$$= P(z_1)\cdots P(z_N)\sum_{z}\sum_{x,x^e} P(e_i|z_1,\cdot z_N) \tag{6.3}$$

$$= P(z_1)\cdots P(z_N)$$

$$\sum_{\forall z}\sum_{\forall x,\forall x^e} P(e_i,y_i,y_i^e,z_1,\cdot z_N,x_1,\cdots x_M,x_1^e,\cdots x_M^e) \tag{6.4}$$

where Eq. 6.4 shows that the joint density function that is necessary to compute the dynamic error exactly. Summing over all possible values of all the involved variables is computationally expensive (NP-hard), hence we require a graphical model that would use the causal structure and conditional independences to arrive at the minimal optimally factorized representation of this joint probability function as a Bayesian network.

6.1.1 The Bayesian Network Structure

We model, both the error free logic model and the one with dynamic errors, as a Directed Acyclic Graph (DAG). These two models, which we will refer to as the ideal logic model and the error-encoded model, are then connected at the outputs by comparators. The comparator output is the variable $E_i = Y_i^e \oplus Y_i$ in Eq. 6.1. The comparator output of logic 1 indicates that the ideal logic model and error-encoded model outputs are different. The probability of the comparator outputs being in state *"1"* provides the overall circuit error probability, $P(E_i = 1)$.

Figure 6.1.(a) shows the (conceptual) representation of a error detection circuit for a simple logic involving two NAND gates, represented by block C. The other block involves the same logic, but built with unreliable components. These gates are assumed to have gate error probability of p. The inputs to both the blocks are the same. The two outputs are connected to two comparators. The output of the comparator represents error in computation. Note that this is just a conceptual representation, we do not actually propose synthesizing the circuit. From the conceptual circuit design, we can construct the Bayesian network representation, which we call the LIPEM-DAG model. Each node in the LIPEM is a line in circuit and the links denote a connection between the lines via gates. Figure 6.1.(b) shows the LIPEM corresponding to the circuit in Figure 6.1.(a). In the rest of this section, we present a more formal definition and prove that the representation, thus obtained, is minimal.

Definition: The Logic Induced Probabilistic Error Model (LIPEM) corresponding to a combinational circuit $< C, \{p\} >$, where C is the circuit and $\{p\}$ are individual gate error probabilities, can be constructed as follows. Nodes are random variables with two values: 0 or 1. There are 6 types of nodes.

- $\{Z\}$: Primary inputs
- $\{X^e\}$: Internal signals with error probability p.
- $\{X\}$: Internal signals under ideal logical condition.
- $\{Y^e\}$: Erroneous output signals.
- $\{Y\}$: Primary output under ideal logical condition.
- $\{E\}$: Error nodes representing the error at each output.

Edges of LIPEM are directed and denoted by the ordered pair $(u \rightarrow v)$ where u causes a v. The edge set can be classified as follows:

- $(Z \rightarrow X^e)$: Edges between nodes representing primary inputs and outputs of erroneous gates that are directly fed by the primary inputs. $(Z \rightarrow X)$: Edges between nodes representing the same primary inputs and outputs of corresponding ideal logic gates.

- $(X_i^e \rightarrow X_j^e)$, $(X_i \rightarrow X_j)$: Edges between random variables representing internal signals (Edges from the input of a gate to the corresponding output). If there is an edge $(X_i \rightarrow X_j)$, then there must be a mirror edge $(X_i^e \rightarrow X_j^e)$. These two edges differ in the conditional probability discussed later during the quantification of LIPEM.

- Edges $(X \rightarrow Y)$ and $(X^e \rightarrow Y^e)$

- Edges $(Y \rightarrow E)$ and corresponding $(Y^e \rightarrow E)$

Theorem: The LIPEM-DAG structure, corresponding to the combinational circuit C is a minimal I-map of the underlying dependency model and hence is a Bayesian network.

Proof: Markov boundary of a random variable v in a probabilistic framework, is the minimal set of variables that make the variable v conditionally independent of all the remaining variables in the probabilistic network.

Let us order the random variables in the node set, such that for every edge $(u,\ v)$ in LIPEM-DAG, u appears before v. With respect to this ordering, the Markov boundary of any node, $v \in \{\{Z\}, X, Y, \{\{X^e\}, \{Y^e\}, \{E\}\}$ is given as follows. If v represents a primary input signal line, then its Markov boundary is the null set. And, since the logic value of an output line is just dependent on the inputs of the corresponding gate (whether v is in $\{\{X\}$, or $\{X^e\}$, or $\{\{Y\}$, $\{Y^e\}$, or $\{\{E\}$) the Markov boundary of a variable representing an output line consists of just those variables that represent the inputs to that gate. Thus in LIPEM structure the parents of each node are its Markov boundary elements. Hence the LIPEM is a boundary DAG. Note that LIPEM is a boundary DAG because of the causal relationship between the inputs and the outputs of a gate that is induced by logic. It has been proven in [26] that if graph structure is a boundary DAG D of a dependency model M, then D is a minimal I-map of M ([26]). This theorem along with definitions of conditional independencies, in [26] (we omit the details) specifies the structure of the Bayesian network. Thus LIPEM is a minimal I-map and thus a Bayesian network (BN).

6.1.2 Bayesian Network Quantification

LIPEM-BN thus constructed, consists of nodes that are random variable of the underlying probabilistic model and edges denote direct dependencies. All the edges are quantified with the corresponding conditional probabilities of the form $p(x_v|x_{\text{parent}(v_i)})$, where $parent(v_i)$ is the set of nodes that has directed edges to v_i. These conditional probability specifications are determined by the gate type. A complete specification of the conditional probability of a two input AND gate output will have 2^3 entries since each variable has 2 states. The edges in the error-encoded part would be quantified by logic function allowing for some randomness. The conditional probability of an error-free AND gate and an unreliable AND gate with gate error probability p are shown in Table 6.1.(a) and (b), respectively.

6.2 Experimental Results

We demonstrate the experimental results using LGSynth'93 and ISCAS'85 benchmark circuits. Even though these benchmarks are for nano-CMOS, logical structures namely re-convergence and input-output behaviors are fundamental to most nano devices. Also, we do not have medium size standard designs to show the efficiency of our model for most of the emerging devices. Hence to study the scalability and time-accuracy trade-off, we choose these benchmark circuits that are used in many previous researches and are well understood.

We use exact inference scheme to estimate output error probabilities of small circuits from LGSynth '93 benchmarks and show that the runtime and memory requirement for our model is orders of magnitude less in comparison with PTM based model [54]. The approximate computation of the LIPEM using Bayesian Networks was done by a tool named "GeNIe" [67]. The tests were performed on a Pentium IV, 2.00GHz, Windows XP computer.

Small Circuits: In table 6.2., we report the maximum output error probabilities of benchmark circuits for different gate error probabilities. Column 2, 3 and 4 give the maximum output error probabilities when gate error probabilities are 0.005, 0.05 and 0.1 respectively. In column 5 we report the elapsed time for the error estimation. From this table, we can see that the decoder circuit has the lowest output error probability and circuits like, xor5, malu4 and parity have the worst error characteristics. For the

Table 6.2. Output error probabilities [from exact inference]

	Nodes in BN	Maximum Output error Probability for individual gate error probability p			time(s)
		=0.005	=0.05	=0.1	
c17	19	0.0148	0.1342	0.2398	0.0
parity	47	0.0699	0.3971	0.4824	0.0001
pcle	116	0.0179	0.1560	0.2702	0.07
decod	129	0.0068	0.0654	0.1251	0.14
cu	210	0.0232	0.1969	0.3327	0.56
pm1	225	0.0331	0.2627	0.4141	0.44
xor5	118	0.0925	0.4336	0.4900	0.26
alu4	222	0.0676	0.3906	0.4816	1.87
b9	392	0.0315	0.2475	0.3867	2.49
comp	415	0.0733	0.3828	0.4683	0.66
count	404	0.0203	0.1613	0.2632	1.14
malu4	280	0.0845	0.4253	0.4903	1.94
max_flat	70	0.0296	0.2151	0.3234	0.02
pc	261	0.0377	0.2794	0.4161	0.41
voter	134	0.0299	0.2178	0.3294	0.08

xor circuit, even a gate error probability of $p = 0.05$ is not acceptable, because with this gate error probability, the output error probability is 0.4336, which is the highest over all circuits.

In Table 6.3., we compare the time and space complexity of our model with those of Probabilistic Transfer Matrix [PTM] based method proposed in [54]. Column 2 and 3 of this table give the runtime and memory requirement of PTM model, where simulations were performed using a 3GHz Pentium 4 processor. Column 4 and 5 give the runtime and memory requirement of our model. We used a 2GHz Pentium 4 processor. These results show the effectiveness of our model in terms of estimation time and memory usage. The time and space complexity of BN based model does not depend on gate error probability values, whereas experimental results form [54] show that the runtime and memory requirement for PTM based modeling are not the same for different gate error probabilities.

Han *et al.* [56] has given a comparison between their PGM model(Probabilistic Gate Model) and the PTM model. The PGM model, which is an approximate method for reliability modeling in nano-circuits, gives reliability estimates very close to the results from the exact PTM model. Computational complexity analysis given in [56] shows that time and space complexity with PTM models is exponen-

Table 6.3. Comparison of modeling using Bayesian networks and probabilistic transfer matrix

	PTM [54]		BN	
	time(s)	memory(MB)	time(s)	memory(MB)
c17	0.076	0.003	0.0	0.096
parity	0.35	0.144	0.0001	0.14
pcle	74.9	24.2	0.07	0.34
decod	56.9	11.8	0.14	0.42
cu	93.87	10	0.56	2.28
pm1	7169	160	0.44	1.38
xor5	1337	57.3	0.26	2.48

tial with respect to the number of inputs *and* outputs, whereas complexity of PGM method is exponential *only* to the number of circuit inputs and it grows linearly with the number of outputs.

Scalability of the Error Model: We show that the error model and associated computations scales extremely well with circuit size by showing results with circuits of varying sizes. Table 6.4. shows the error probabilities for various ISCAS'85 benchmark circuits for four different gate errors of $p = 0.01, 0.001$ and 0.0001. Reported results were obtained from PLS with 1000 samples. As expected, all circuits exhibit higher overall error as individual gate error increases. It can be observed that c499 has lower error growth over all the other circuits. For many of the benchmark circuits (namely c1908, c2670, c6288, c7552), the output error exceeds 0.2 for gate error probability 0.01. This indicates that 0.01 is unacceptable gate error probability for these circuits. Additional gate level redundancy may be required for these circuits.

To validate our model we performed an in-house logic simulation of the benchmarks with 500,000 random vectors obtained by changing seed after every 50,000 vectors and obtained the average output error probabilities for different gate error probabilities. Simulation is done by comparing the outputs from the ideal logic block with corresponding outputs from an erroneous logic block. Both ideal and erroneous logic blocks are fed by the same primary inputs. To simulate the effect of erroneous operation, we inject an additional input, I_f to each gate in the faulty logic blocks. If this input to a gate in the erroneous logic block holds logic value one, the gate is outputs a wrong signal. If I_f to a gate is logic zero, it behaves as an ideal gate. Probability ofI_f being at logic one is the gate error probability p. Note that in a single simulation run, the input I_f to any gate in the erroneous logic block can be either zero

Table 6.4. Output error probabilities for ISCAS'85 circuits -PLS with 1000 samples

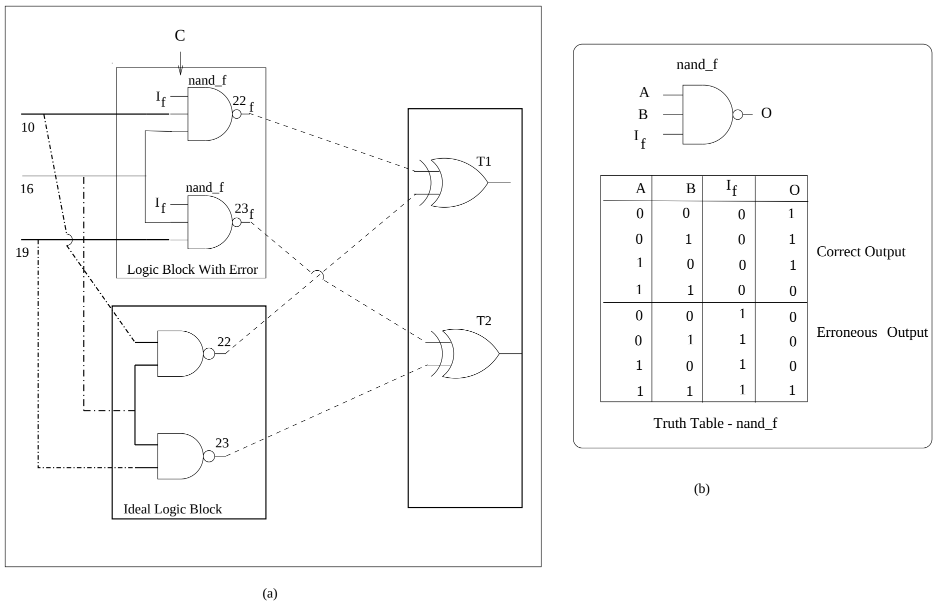
	No. of gates	Maximum Output error Probability for individual gate error probability p		
		=0.01	=0.001	=0.0001
c432	160	0.149	0.021	0.004
c499	202	0.04	0.006	0.002
c880	383	0.205	0.03	0.003
c1355	546	0.077	0.014	0.003
c1908	880	0.331	0.14	0.15
c2670	1193	0.395	0.27	0.237
c3540	1669	0.503	0.502	0.535
c5315	2307	0.389	0.643	0.011
c6288	2416	0.523	0.202	0.037
c7552	3512	0.5	0.15	0.031

Table 6.5. Comparison of Bayesian network modeling and logic simulation

	Nodes in BN	1000 Samples		10000 samples	
		μ_e	T1(s)	μ_e	T2(s)
c432	475	0.0020	0.234	0.0020	2.140
c499	565	0.0010	0.344	0.0003	3.260
c880	956	0.0010	0.844	0.0004	7.876
c1355	1253	0.0020	1.141	0.0007	10.454
c1908	2172	0.0070	2.187	0.0060	18.984
c2670	3097	0.0104	3.205	0.0009	26.457
c3540	4038	0.0700	4.547	0.0700	36.249
c5315	6247	0.0077	7.844	0.0160	56.971
c6288	4896	0.0065	5.672	0.0022	43.062
c7552	8398	0.0650	11.64	0.0640	78.688

or one, depending on the gate error probability p and the output of the random number generator which determines the state of this input. Fig. 6.2.(a) is a small dynamic error simulation circuit. Fig. 6.2.(b) is the truth table of a nand gate in the erroneous logic block of the simulation model.

In Table 6.5., we compare simulation results with Bayesian network results. We report the accuracy of our model in terms of average error, μ_e , between the exact output error probabilities and the estimated output error probabilities obtained from PLS with 1000 and 10,000 samples. Column 2 of this table gives the number of nodes in the Bayesian network. We list μ_e and elapsed time for PLS with 1000 samples in column 3 and 4 respectively. Column 5 and 6 of this tables gives μ_e and elapsed time with 10,000 samples. Column 7 gives the logic simulation time. Mean estimation error with 1000 samples is 0.017



A	B	I_f	O
0	0	0	1
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

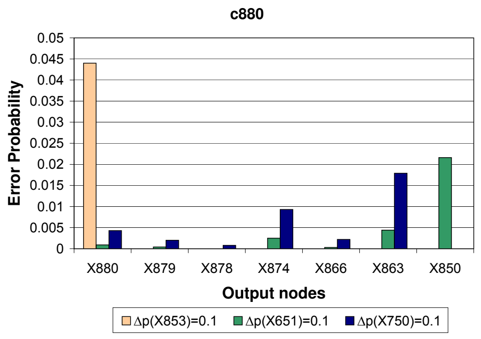
Figure 6.2. (a) Dynamic error simulation circuit (b) Truth table of a NAND gate in the erroneous logic block

and with 10000 samples is 0.016. Average estimation time with 1000 samples and 10,000 samples are 3.76 seconds and 28.41 seconds respectively, whereas average simulation time is 161.43 seconds. These results show the effectiveness of our model in terms of accuracy and estimation time. We see that estimation time varies linearly with number of samples, whereas average estimation error is almost the same with 1000 and 10000 samples for most of the circuits. With PLS 1000 samples, circuit c2670 gave the lowest estimation error of 0.01 and circuits c2670 and c5315 gave the highest estimation error of 0.07 over all circuits. With PLS 10000 samples, again c2670 gave lowest μ_e of 0.0009 and c3540 gave highest μ_e of 0.07.

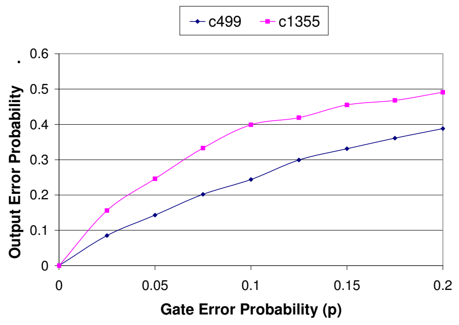
Error Sensitivity: We compute sensitivity of output error to a particular gate with a Δp change in gate error, rest of the gates having zero gate error probability. This is useful for determining the application of redundancy in achieving nano-domain fault tolerance. In Table 6.6., we tabulate the output error probability corresponding to gate 22 and 23 of benchmark $c17$, when the error probabilities of gates $10, 11, 16,$ *and* 19 change by 0.1. It is clear that output error at gate 22 is most sensitive to gate 16. We also plot sensitivity of a few individual gates for benchmark c880 in Figure 6.3.(a). As it can be seen

Table 6.6. Error sensitivity for c17

	Error Probability at outputs	
Gate δp	22	23
Δp(X10)=0.1	0.0625	0
Δp(X11)=0.1	0.0375	0.075
Δp(X16)=0.1	0.075	0.0625
Δp(X19)=0.1	0	0.0625



(a)



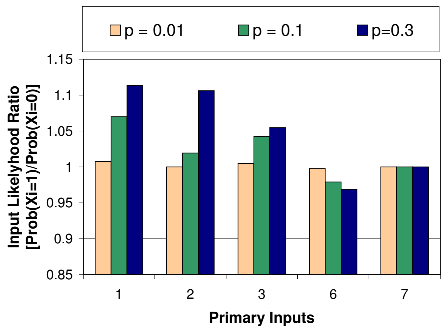
(b)

Figure 6.3. (a) Sensitivity of output error probability with respect to individual gate errors for c880 (b) Output error profiles of two alternative logic implementation (c499 and c1355)

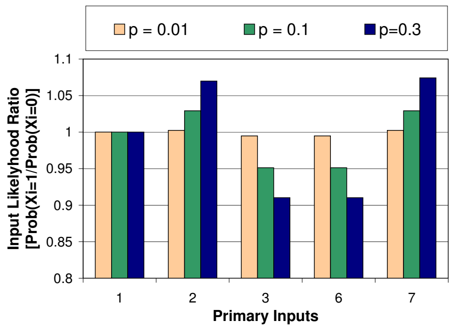
that output error at node 880 is sensitive only to gate 853 out of the three intermediate gates shown here and gate output error at node 863 is reasonably sensitive to both node 651 and node 750.

Design Space Exploration: Figure 6.3.(b) shows the variation in output error with gate error p for two ISCAS benchmark c499 and c1355 that are logically equivalent. We see that c499 is clearly a better design of the logic for nano-domain in terms of resistance to dynamic errors. The circuit c1355 is more sensitive to dynamic-error almost for all individual gate error probabilities, specially when p=0.1, the output error probability become 0.4 as opposed to 0.25 for c499. The expected output dynamic error can be used, along with other design measures such as power and area for nano-domain circuits.

Input Space Exploration: In this section, we explore input characteristics for a desired output behavior. Namely, what should be input probabilities, entropy, likelihood etc for such a desired behavior.



(a)



(b)

Figure 6.4. c17 - Input space characterization by likelihood ratio for (a) zero error at output node 22 (b) for zero error at output node 23

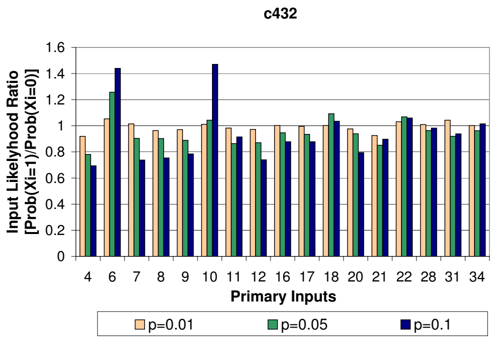
Many of the above results in the previous subsection is conducted under random inputs. However, inputs themselves might aggravate or eliminate some of the serious problems that we face. There are two key components for such characterization. How do we propagate probabilistically from a desired output characteristic (can be any node really)? Which input characteristics would be beneficial for the designers? The answer to the question lies in the unique back-tracking capability of Bayesian network based models. In fact, BN models are unique for not only predictive inference but their main application is in diagnosis. We show that exact inference scheme can be used for input space characterization of small circuits and approximate inference scheme for medium sized benchmark circuits. The logic sampling provides inaccurate estimates for backtracking as discussed in Chapter 3.1. Hence we resort to EPIS based inference for probabilistic diagnosis. We use likelihood of logic one as an input behavior. Designers are free to use any other metrics to characterize the inputs.

Input Space - Exact inference

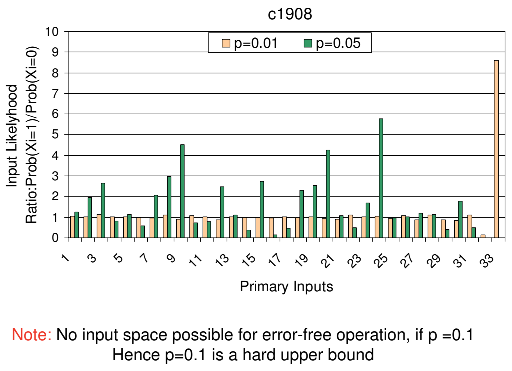
We characterize the input space of c17 benchmark for obtaining some specific output behavior given the dynamic error probability of individual gates. We set some specific output error probabilities as evidence in the BN model and back propagate these probabilities to obtain the corresponding input probabilities. We use Hugin tool for exact inference. In Fig 6.4.(a) and (b), we plot the input space in terms of likelihood ratio $Prob(X_i = 1)/Prob(X_i = 0)$ for primary inputs 1, 2, 3, 6 and 7 by setting (a) error probability of output 22 at zero and (b) error probability of output 23 at zero, respectively. We plot the graphs for three different gate error probabilities, $p = 0.01$, $p = 0.1$, and $p = 0.3$. Likelihood ratio is equal to one if the input probability is 0.5 which indicates randomness of the input. From Fig. 6.4.(a) and (b), it can be seen that, with individual gate error probability $p = 0.01$, all the inputs have likelihood ratio equal to, or close to one. In other words, random inputs are likely to produce zero output error probability, if individual gate error probability is $p = 0.01$. Thus it can be concluded that if we have no information about the inputs, i.e. if the actual input space of this circuit is random, we don't have to apply any dynamic error mitigation technique to the gates in the circuit since the circuit has in-built error tolerance. As the gate error probability increases, we can see that input likelihood moves further away from one, indicating that the circuit requires some error mitigation schemes if the actual input pattern is different from the pattern shown in the graph. For example, with gate error probability $p = 0.3$, in order to have no error at output 22, input 1 should have more ones than zeros. If the actual input pattern has more zeros than ones for this input, we should definitely apply additional redundancy techniques to make the circuit error-free. From Fig. 6.4.(a), it is evident that inputs 1 and 2 follow almost same pattern as gate error probability increases, whereas input 6 follows an opposite pattern. Likelihood of input 6 being at logic 0 increases as the gate error probability increases from 0.01 to 0.3 in order to get zero output error at node 22. Hence in the actual input space of the circuit, if input 6 has more 1 values than 0's, circuit requires more stringent redundancy techniques and vice-versa. From Fig. 6.4.(b), it can be seen that likelihood of inputs 2 and 7 being at logic one increase with increase in gate error probability, whereas inputs 3 and 6 exhibit the opposite behavior. Input 1 has likelihood ratio equal to one for all gate error probabilities which shows that the logic level of this input does not influence the output error probability at node 23 for any gate error probabilities. Since $c17$ is a very small benchmark circuit, the effect of inputs on the overall output error is not very important. To demonstrate the usefulness of our model, we experiment input space characterization on mid-size benchmarks, which is explained below.

Input Space - Approximate Inference

To handle medium sized benchmark circuits, we use the approximate inference scheme, namely EPIS, for input space characterization. In the following three set of experiments on c432 and c1908, we



(a)



(b)

Figure 6.5. Input space characterization by likelihood ratio for benchmarks (a) c432 (b) c1908

make the gate-error probability of all gates in the circuit to 0.01 and forced all the output error probabilities to zero (also termed evidence), then we back-track and find out the input probabilities that will give the desired output characteristic of zero error. These input probabilities are then used to calculate input likelihood ratios. In Figure 6.5.(a), we plot the likelihood ratios $Prob(X_i = 1)/(Prob(X_i = 0)$ of selected input of benchmark c432. As it can be seen that for this circuit, with gate error probability $p = 0.01$, the likelihood is close to one for most of the inputs. This implies that to have no output error, primary inputs would have equal probability of being at logic 0 or at logic 1, if the gate error probability $p = 0.01$. We then repeat this by increasing the individual gate error probability values to p=0.05 and then to p=0.1. It is clear that input likelihoods are further away from one in both directions, for gate error probability $p = 0.1$. For example, inputs 6 and 10 have likelihood ratio, close to 1.5 for achieving zero output error with a gate error probability $p = 0.1$, indicating that probability of these inputs being at logic one should be higher than that of being at logic zero. Hence it is clear that if these inputs have more 0's than 1's in the actual input pattern, then the circuit is less resistant to dynamic errors and needs mitigation schemes to make it error-tolerant. However, inputs 4 and 12 have low likelihood ratios, close to 0.7, for gate error probability $p = 0.1$. Circuit behavior with respect to these inputs are exactly opposite to that with respect to inputs 6 and 10. We can also observe that some inputs like, 4, 8, 9, 11, 12, 20 and 21 have likelihood ratio less than one for all three gate error probabilities whereas some other inputs like, 6, 10 and 22 always have likelihood ratio greater than one. From these observations, it can be concluded that if inputs in the *first* category have more number of *zeros* than *ones* and inputs in the *second* category have more number of *ones* than *zeros*, in the input pattern, this circuit possesses in-built error tolerance with respect to dynamic errors and doesn't require severe redundancy techniques. However, if the input pattern is different from above, the circuit gives erroneous outputs and designers have to incorporate error tolerant schemes for reliable operation.

These experiments are repeated for c1908 and we observe that most inputs have a likelihood of one for p=0.01 as seen in Fig. 6.5.(b) For p=0.05, we have many inputs with likelihood much higher than one indicating that logic one at those inputs is more probable to cause zero output errors. If the actual input patterns have more zeros, the circuit will give erroneous outputs and hence gates are to be targeted for redundancy application. We can see another important result from Fig. 6.5.(b) With p=0.1, no input vectors can account for a zero output error, which means that no input combination can give zero output error if the gate error probability reaches 0.1. Hence $p = 0.1$ is an upper bound on gate error probability, beyond which reliable computation is impossible. We need to have effective error mitigation schemes for circuit c1908 to achieve reliable computing if the individual gate error probability is 0.1 or above.

Reliability Enhancement: The effects of dynamic errors in logic circuits can be mitigated by application of various error-tolerant techniques thereby achieving a given level of reliability. Application of redundancy to all gates in a circuit will result in very high area overhead and excess power dissipation in addition to increased cost. We need to choose gates that are highly sensitive to dynamic errors and apply selective redundancy measures to achieve trade-off between redundancy, reliability, area overhead and cost.

We first estimate dynamic error sensitivities of selected gates in a circuit. Gates which are logically closer to the primary outputs are considered to be more sensitive to dynamic errors than those gates which are at higher logical depths from primary outputs. Hence we choose gates which are within a specific logical depth from primary outputs. We compute error sensitivities of each of these selected gates and then compute percentage reduction in overall output error probabilities by applying selective redundancy to highly sensitive nodes. This is explained below:

- Step 1: Estimate the overall output error probabilities with a given dynamic error probability (say $p = 0.01$) for ALL gates in the circuit $(\{P_{E_1}, P_{E_2}, \cdots, P_{E_N}\})$.

- Step 2: Give a higher error probability (say $p = 0.05$) for the gate whose error sensitivity is to be determined keeping all other gate error probabilities same as before (0.01) and estimate the overall output error probabilities $(\{P'_{E_1}, P'_{E_2}, \cdots, P'_{E_N}\})$.

- Step 3: Compute the difference in estimated output error probabilities from step 1 and step 2 $(\Delta P = \{|P_{E_1} - P'_{E_1}|, |P_{E_2} - P'_{E_2}|, \cdots, |P_{E_N} - P'_{E_N}|\})$.

- Step 4: Determine the set of gates $\{S\}$ which give max$(\Delta P) \geq$ a threshold value (δp_{th}). These gates are to be selected for application of redundancy. Value of δp_{th} is chosen depending on the desired error tolerance.

- Step 5: We estimate overall output error probabilities by giving a low dynamic error probability (say $p = 0.001$) for gates selected for redundancy $(gates \in \{S\})$ and original error probability $p = 0.01$ for all other gates ($gates \notin \{S\}$) in the circuit. We compare these values with output error probabilities for 100% redundancy (by giving $p = 0.001, \forall gates$) and compute reliability/redundancy trade-off in terms of percentage reduction in average output error probabilities.

Results for circuits c17 and c432 in Table 6.7. and Table 6.8. respectively. Column 1 of these tables gives the δp_{th} values and column 2 gives the percentage of gates selected for redundancy for each δp_{th}. In column 3, we report the percentage reduction in output error probabilities by applying selective redundancy. Number of gates to be selected for redundancy application depends on the error threshold δp_{th}. With decrease in δp_{th}, more number of gates are selected for redundancy, thus reducing the output error probability. For example, from Table 6.8., it can be seen that by decreasing δp_{th} from 0.020 to 0.001, we achieve reliability improvement from 11% to 60% measured in terms of reduction in output error probability. Thus our modeling tool can be used for choosing the gates to be selected for application of error-tolerant techniques and also for determining the amount of such techniques required for achieving a desired error tolerance.

We presented an exact probability model, based on Bayesian networks, to capture the inter-dependent effects of dynamic errors at each gate. Dynamic error at each gate is modeled through the conditional probability specifications in the Bayesian network. The expected output error can be used to select

Table 6.7. c17-Selective redundancy

δp_{th}	% of Gates Selected for Redundancy	Percentage Reduction in Average Output Error Probability
0.025	60%	69.91
0.000	100%	89.80

Table 6.8. c432-Selective redundancy

δp_{th}	% of Nodes Selected for Redundancy	Percentage Reduction in Average Output Error Probability
0.020	8%	11%
0.015	14%	17%
0.010	24%	32%
0.005	37%	50%
0.001	59%	60%
0.000	100%	88%

between different implementation of the same logical function, so as to result in reliable nano-level circuits.

We theoretically proved that the dependency model for probabilistic gate errors is a Bayesian Network and hence is an exact, minimal representation. We used exact inference scheme for small circuits and show that our model is extremely time and space efficient compared to existing PTM based technique. To handle even larger benchmarks, we used two stochastic sampling algorithms, PLS and EPIS. Even though many of the futuristic techniques do not yet offer benchmarks, we demonstrate scalability of our estimation tool by using the logic level specifications of the ISCAS'85 benchmarks, which would be valid for uncertainty modeling in nano-CMOS. We studied the input characteristics of logic circuits for a desired output behavior by exploring the unique back-tracking feature of Bayesian networks. By conducting a sensitivity analysis of gates in the circuits, we were able to identify gates that affect the overall circuit behavior due to dynamic errors and present a technique to determine the amount of redundancy to be applied for achieving a desired improvement in circuit performance.

We are currently working on modeling dynamic error tolerant designs by applying TMR redundancy on selected nodes having high dynamic error sensitivities based on their switching characteristics and other device features. We intend to pursue the dynamic error for QCA logic blocks and model interconnect gate error in future. A relative study on various technologies like CNT, RTD with respect to dynamic error would be easily extended by this framework. The other more difficult task would be to handle sequential logic in terms of dynamic errors.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this dissertation, we model and analyze three major classes of faults/errors which are significant in logic networks in deep-submicron, sub-100-nanometer and future nanometer regimes. We also study how reliability can be enhanced by the application efficient error mitigation schemes. We developed probabilistic models based on Bayesian Networks for the estimation and analysis of different types of errors. We summarize below the important results from this dissertation:

1. LIFE-DAG: Logic Induced Fault Encoded Directed Acyclic Graph which is an accurate and efficient model for estimation of Fault Detection Probabilities.

2. TALI-SES: A Timing Aware Probabilistic Model for Single Event Upset Sensitivity, which is an exact probabilistic model for the estimation of single event upset sensitivities of individual gates in logic circuits that captures the effect of logical masking, inputs, gate delays, SEU duration and circuit re-convergence.

3. LIPEM: Logic Induced Probabilistic Error Model for accurate and efficient estimation of overall circuit output error due to dynamic errors/inherent errors in logic gates.

4. Input space characterization of logic circuits for a desired output behavior by exploring the unique backtracking feature of Bayesian networks.

5. Analysis of the effect of selective redundancy techniques on circuit reliability and calculation of reliability/redundancy trade-off.

It is one of the future research directions to develop a comprehensive model for soft error analysis by integrating the electrical masking effect, latching window masking effect and also the SEU generation and propagation characteristics of individual gates with our current approach. Another future work is to model dynamic error tolerant designs by applying TMR redundancy on selected nodes having high dynamic error sensitivities based on their switching characteristics and other device features. We intend to pursue the dynamic error for QCA logic blocks and model interconnect gate error in future. A relative study on various technologies like CNT, RTD with respect to dynamic error would be easily extended by this framework. The other more difficult task would be to handle sequential logic in terms of dynamic errors. There is future scope for developing a macro model from our gate level models that can be used at higher levels of abstraction. Another research direction is the estimation of error bounds that are circuit-specific.

REFERENCES

[1] Semiconductor Industry Association, International Technology Roadmap for semiconductors, 2005.

[2] C. Zhao, Y. Zhao and S. Dey, “Constraint-aware robustness insertion for optimal noise-tolerance enhancement in VLSI circuits”, *Design Automation Conference*, pp. 190–195, June 2005.

[3] N. Shanbhag, K. Soumyanath and S. Martin “ Reliable low power design in the presence of deep submicron noise”, *International Symposium on Low Power Electronics and Design* , pp. 295 – 302, 2000.

[4] C. Zhao, X. Bai and S. Dey, “A scalable soft spot analysis methodology for compound noise effects in nano-meter circuits,” *Proceedings of Design Automation Conference*, pp. 894–899, Jun. 2004.

[5] J. von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” in *Automata Studies* (C. E. Shannon and J. McCarthy, eds.), pp. 43–98, Princeton Univ. Press, Princeton, N.J., 1954.

[6] N. Pippenger, “Reliable Computation by Formulas in the Presence of Noise”, *IEEE Trans on Information Theory*, vol. 34(2), pp. 194-197, 1988.

[7] W. Heidergott, “SEU tolerant device, circuit and processor design”, *Design Automation Conference*, pp. 5–10, June 2005.

[8] S. Borkar, T. Karnik and Vivek De, ”Design and reliability challenges in nanometer technologies”, *D*esign Automation Conference, 2004.

[9] Y. S. Dhillon, A. U. Diril and A. Chatterjee, “Soft-Error Tolerance Analysis and Optimization of nanometer circuits,” *Proceedings of Design, Automation and Test in Europe*, Volume: 1, pp. 288–293, Mar. 2005.

[10] M. L. Bushnell and V. D. Agrawal, ”Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal Vlsi Circuits,” Boston:Springer, 2005.

[11] S. C. Seth, L. Pan, and V. D. Agrawal, “Predict - probabilistic estimation of digital circuit testability,” *IEEE International Symposium on Fault-Tolerant Computing*, pp. 220–225, Jun. 1985.

[12] H. J. Wunderlich , “PROTEST: A Tool for probabilistic testability Analysis,” *Proceedings of the 22nd IEEE Design Automation conference*, vol. 14-3, pp. 204–211, 1985.

[13] R. Krieger, B. Becker and R. Sinkovic, “A BDD-based Algorithm for computation of exact fault detection probabilities,” *Digest of Papers, Fault-Tolerant Computing*, vol. 22-24, pp. 186–195, June 1993.

[14] K. Mohanram and N. A. Touba, ”Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits,” *International Test Conference*, pp. 893–901, 2003.

[15] T. Karnik and P. Hazucha, ”Characterization of soft errors caused by single event upsets in CMOS processes,” *IEEE Transactions on Dependable and Secure Computing*, Volume: 1-2, pp. 128–143, Apr-Jun. 2004.

[16] V. Degalahal, R. Rajaram, N. Vijaykrishan, Y. Xie and M. J Irwin, ”The effect of threshold voltages on soft error rate,” *5th International Symposium on Quality Electronic Design*, March 2004.

[17] M. Zhang and N. R. Shanbhag, “A Soft Error Rate Analysis (SERA) Methodology” *International Conference on Computer Aided Design*, November, 2004.

[18] N. Seifert *et al.* “Impact of Scaling on Soft-Error Rates in Commercial Microprocessors” *IEEE Transactions on Nuclear Science*,Volume: 49, No. 6, pp. 3100–3106, Dec. 2002.

[19] P. Hazucha *et al.* “Measurement and Analysis of SER-Tolerant Latch in a 90-nm Dual-$V-T$ CMOS Process” *IEEE Transactions on Solid-State Circuits*,Volume: 39, No. 9, pp. 1536–1543, Sept. 2004.

[20] R. Iris Bahar, J. Mundy, and J. Chan, “A Probabilistic Based Design Methodology for Nanoscale Computation”, *International Conference on Computer Aided Design*, pp 480 – 486, 2003.

[21] R. Martel, V. Derycke, J. Appenzeller, S. Wind, and Ph. Avouris, “Carbon Nanotube Field-Effect Transistors and Logic Circuits,” *Proceedings of the 39th Conference on Design Automation*, 2002.

[22] J. Chen, J. Mundy, I. Bahar, and J. M. Xu, “Fault-tolerance Carbon Nanotube-based Computer Logic Design,” *Foresight Conference on Molecular Nanotechnology*, Oct. 2003.

[23] J. P. Sun, G. I. Haddad, and P. Mazumder, “Resonant Tunneling Diodes: Models and Properties” *Proceedings of the IEEE*, vol. 86-2, pp. 641–661, April 1998.

[24] R. K. Kummamuru, A. O. Orlov, R. Ramasubramanyam, C. S. Lent, G. H. Bernstein, and G. L. Snider, “Operation of Quantum-Dot Cellular Automata (QCA),Shift Registers and Analysis of Errors” *IEEE Transactions on Electron Devices*, vol. 50-59, pp. 1906–1913, June 1993.

[25] M. B. Tahoori, M. Momenzadeh, and J. Huang, and F. Lombardi, “Defects and Faults in Quantum Cellular Automata at Nano Scale” *Workshop on Fault Tolerance in Parallel and Distributed Systems*, 2004.

[26] J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference,” Morgan Kaufmann Publishers, Inc., 1988.

[27] URL `http://www.hugin.com`.

[28] M. Henrion, “Propagation of uncertainty by probabilistic logic sampling in Bayes’ networks,” *Uncertainty in Artificial Intelligence*, 1988.

[29] W-B. Jone and S. R. Das, “ CACOP-a random pattern testability analyzer,” *IEEE Transactions on Man and Cybernetics*, vol. 25-5 pp. 865–871, 1995.

[30] R. Pathak, “A generalized algorithm for bounding fault detection probabilities in combinational circuits,” *AUTOTESTCON, Proceedings of IEEE Systems Readiness Technology Conference*, vol. 20-23, pp. 683–689, Sep. 1993.

[31] J. Savir and G. S.Ditlow, and P. H. Bardell, “Random pattern testability,” *IEEE Transactions on Computers*, vol. C-33, pp. 79–90, Mar. 1984.

[32] S. Chakravarty and H. B. hunt, III, “On computing signal probability and detection probability of stuck-at faults,” *IEEE Transactions on Computers*, vol. 39-11, pp. 1369–1377, Nov. 1990.

[33] H. Farhat, A. Lioy and M. Pocino, “Computation of exact random pattern detection probability,” *Proceedings of IEEE Custom Integrated Circuits conference*, vol. 9-12, pp. 26.7.1–26.7.4, May 1993.

[34] S. Bhanja and N. Ranganathan, “Dependency preserving probabilistic modeling of switching activity using Bayesian networks” *Design Automation Conference*, June 2001.

[35] S. Bhardwaj, S.B.K. Vrudhula and D. Blaauw, “*t*AU: Timing analysis under uncertainty” *International Conference on Computer Aided Design*, Nov. 2003.

[36] P. D. Wiley, ”Fault Tolerant Design Verification Through the Use of Laser Fault Injection”, *Master’s Thesis*, University of South Florida, Feb. 2004.

[37] F. J. Falquez, ”Fault Tolerance and Testability Design With Validation by Laser fault Injection To Improve the Performance and Testability of Advanced VLSIC”, *PhD Dissertation*, University of South Florida, Dec. 1998.

[38] S. Mitra, T. Karnik, N. Seifert and M. Chang, “Logic soft errors in sub-65nm technologies design and CAD challenges”, *Design Automation Conference*, pp. 2–4, June 2005.

[39] D. Alexandrescu, L. Anghel and M. Nicolaidis, “New Methods for Evaluating the Impact of Single Event Transients in VDSM ICs,” *Proc. Defect and Fault Tolerance Symposium*, pp. 99–107, 2002.

[40] P. Shivakumar, et al.,“Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic,” *Proc. International Conference on Dependable Systems and Networks*, pp. 389–398, 2002.

[41] M. Violante, “Accurate Single-Event-Transient Analysis via Zero-Delay Logic Simulation,” *IEEE Transactions on Nuclear Science*, Vol. 50, No. 6, pp. 2113–2118, 2003.

[42] Q. Zhou and K. Mohanram, “Cost-effective radiation hardening technique for combinational logic”, *International Conference on Computer Aided Design*, pp. 100 – 106, Nov. 2004.

[43] P. K. Samudrala, J. Ramos and S. Katkoori, “Selective Triple Modular Redundancy (STMR) Based Single-Event-Upset (SEU) Tolerant Synthesis for FPGAs,” *IEEE Transactions on Nuclear Science*, Vol. 51, No. 5, Oct. 2004.

[44] M. Omana, G. Papasso, D. Rossi, C. Metra,“A Model for Transient Fault Propagation in Combinatorial Logic,” *On-Line Testing Symposium*, pp. 111-115, 2003.

[45] P. Hazucha and C. Stevenson, ”Impact of CMOS technology scaling on the atmospheric neutron soft error rate,” *IEEE Transactions on Nuclear Science,* Volume: 47-6 , pp. 2586–2594, Dec. 2000.

[46] S. Winograd and J. D. Cowan, *Reliable Computation in the Presence of Noise*. The MIT Press, 1963.

[47] S. Spagocci and T. Fountain, “Fault rates in nanochip devices,” in *Electrochemical Society*, pp. 582–593, 1999.

[48] J. Han and P. Jonker, “A defect- and fault-tolerant architecture for nanocomputers,” *Nanotechnology*, vol. 14, pp. 224–230, 2003.

[49] K. Nikolic, A. Sadek, and M. Forshaw, “Fault-tolerant techniques for nanocomputers,” *Nanotechnology*, vol. 13, pp. 357–362, 2002.

[50] G. Norman, D. Parker, M.Kwiatkowska and S. K. Shukla, “Evaluating the reliability of defect-tolerant architectures for nanotechnology with probabilistic model checking”, *International Conference on VLSI Design*, 2004.

[51] J. B. Gao, Yan Qi and J.A.B. Fortes, “Bifurcations and Fundamental Error Bounds for Fault-Tolerant Computations” *IEEE Transactions on Nanotechnology*, vol. 4-4 pp. 395–402, July 2005.

[52] J. Han, J. B. Gao, P. Jonker, Yan Qi and J.A.B. Fortes, “Toward hardware-Redundant Fault-Tolerant Logic for Nanoelectronics” *IEEE Transactions on Design and Test of Computers*, vol. 22-4 pp. 328–339, July-Aug. 2005.

[53] J. B. Gao, Y. Qi and J.A.B. Fortes, “Markov Chains and Probabilistic Computation - A general Framework for Multiplexed Nanoelectronic Systems” *IEEE Transactions on Nanotechnology*, vol. 4 pp. 194–205, July 2005.

[54] S. Krishnaswamy, G. S. Viamontes, I. L. Markov, and J. P. Hayes, “Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices”, *Design Automation and Test in Europe (DATE)*, March 2005.

[55] D. Bhaduri and S. K. Shukla, “NANOLAB: A Tool for Evaluating Reliability of Defect-Tolerant Nanoarchitectures”, *IEEE Transactions on Nanotechnology,* 2005.

[56] J. Han, E. Taylor, J. Gao and J. Fortes, ‘”Reliability Modeling of Nanoelectronic Circuits” *IEEE Conference on Nanotechnology,* July 2005.

[57] C. Yuan and M. J. Druzdzel, “An Importance Sampling Algorithm Based on Evidence Pre-propagation,” *Proceedings of the 19th Annual Conference on Uncertainty on Artificial Intelligence*, pp. 624-631, 2003.

[58] J. Cheng, “Efficient Stochastic Sampling Algorithms for Bayesian Networks,” *Ph.D Dissertation, University of Pittsburgh,* 2001.

[59] R. G. Cowell, A. P. David, S. L. Lauritzen, D. J. Spiegelhalter, “Probabilistic Networks and Expert Systems”, Springer-Verlag New York, Inc., 1999.

[60] S. Bhanja and N. Ranganathan, “Cascaded Bayesian inferencing for switching activity estimation with correlated inputs,” *Accepted for publication in IEEE Transaction on VLSI*, 2004.

[61] K.P. Murphy, Y. Weiss and M.I. Jordan “Loopy belief propagation for approximate inference: an empirical study,” *In Proceedings of Uncertainty in AI*, pp. 467–475, 1999.

[62] Y. Weiss, “Correctness of local probability propagation in graphical models with loops,” *Neural Computation*, vol.12-1, pp. 1-41, Aug. 2000.

[63] S. Manich and J. Figueras,“Maximizing the weighted switching activity in combinational CMOS circuits under the variable delay model,” *European Design and Test Conference*, pp. 597–602, 1997.

[64] I. Sutherland, R. Sproull and D. Harris, “Logical Effort: Designing Fast CMOS Circuits”,Morgan Kaufmann, February 1999.

[65] N. Ramalingam and S. Bhanja, “Causal Probabilistic Input Dependency Learning for Switching Model in VLSI Circuits”, *A*CM GLSVLSI, 2005.

[66] R. Marculescu, D. Marculescu, M. Pedram, ”Sequence Compaction for Power Estimation: Theory and Practice”, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.18, No.7, pp. 973-993, July 1999.

[67] "GeNie", URL http://www.sis.pitt.edu/~genie/genie2.

[68] B. P. Philips, “On computing the detection probability of stuck-at faults in a combinational circuit,” *IEEE system readiness Technology Conference*, vol. 24-26, pp. 301–305, Sep. 1991.

[69] S. C. Seth, V. D. Agrawal and H. Farhat, “A theory of testability with application to fault coverage analysis,” *Proceedings of the 1st Europian test conference*, vol. 12-14, pp. 139–143, April 1989.

[70] Y. Fang and A. Albicki, “Efficient testability enhancement for combinational circuits,” *Proceedings of the International Conference on Computer Design*, vol. 2-4, pp. 168–172, 1995.

[71] T. Rejimon and S. Bhanja, “An Accurate Probabilistic Model for Error Detection ,” *Proc. IEEE International Conference on VLSI Design*, pp. 717–722, Jan. 2005.

[72] M. Sonza Reorda and M. Violante, “Fault List Compaction through Static Timing Analysis for Efficient Fault Injection Experiments,” *Proc. Defect and Fault Tolerance Symposium*, pp. 263–271, 2002.

[73] T. Rejimon and S. Bhanja, “A Stimulus-Free Probabilistic Model for Single-Event-Upset Sensitivity,” *Proc. IEEE International Conference on VLSI Design*, Jan. 2006.

[74] M. Nicolaidis,“Time Redundancy based Soft-Error Tolerance to Rescue nanometer Technologies,” *VLSI Test Symposium*, pp. 86–94, 1999.

[75] P. Robinson, W. Lee, R. Aguero and S. Gabriel,“Anomalies due to single event upsets,” Journal of Spacecraft and Rockets,” *Journal of Spacecraft and Rockets*, vol. 31, no. 2, pp. 166–171, Mar-Apr 1994.

[76] J.T. Wallmark and S.M. Marcus,“Minimum size and maximum packaging density of non-redundant semiconductor devices,” *Proceedings of IRE*, vol. 50, pp. 286–298, March 1962.

[77] G.H. Johnson, J.H. Hohl, R.D. Schrimpf and K.F. Galloway,“Simulating Single-event burnout in n-channel power MOSFETs,” *IEEE Transactions on Electron Devices*, vol. 40, pp. 1001–1008, 1993.

[78] J. Srinivasan, S. V. Adve, P. Bose and J. A. Rivers, ”The Case for Lifetime Reliability-Aware Microprocessors ”, *International Conference on Computer Architecture*, 2004.

[79] T. Rejimon and S. Bhanja, “Scalable Probabilistic Computing Models using Bayesian Networks,” *IEEE Midwest Symposium on Circuits and Systems*,pp.712–715,July 2005.

[80] Y. Monnet, M. Renaudin and R. Leveugle, “Asynchronous circuits transient faults sensitivity evaluation”, *Design Automation Conference*, pp. 863–868, June 2005.

[81] G. Asadi and M. B. Tahoori, “An accurate SER estimation method based on propagation probability [soft error rate]”, *Design Automation and Test in Europe*, pp. 306–307, 2005.

[82] C. Lopez-Ongil, M. Garcia-Valderas, M. Portela-Garcia and L. Entrena-Arrontes, “Techniques for fast transient fault grading based on autonomous emulation [IC fault tolerance evaluation]”, *Design Automation and Test in Europe*, pp. 308–309, 2005.

[83] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir and Y. Xie, “Reliability-centric high-level synthesis”, *Design Automation and Test in Europe*, pp. 1258–1263, 2005.

[84] J. S. Hu, F. Li, V. Degalahal, M. Kandemir, N. Vijaykrishnan and M. J. Irwin, “Compiler-directed instruction duplication for soft error detection ”, *Design Automation and Test in Europe*, pp. 1056–1057, 2005.

[85] S. Srinivasan, A. Gayasen, N. Vijaykrishnan, M. Kandemir, Y. Xie and M. J. Irwin, “Improving soft-error tolerance of FPGA configuration bits”, *International Conference on Computer Aided Design*, pp. 107 – 110, Nov. 2004.

[86] V. De and S. Borkar, “Technology and design challenges for low power and high performance”, *International Symposium on Low Power Electronics and Design* , pp. 163 – 168, 1999.

[87] R. Hegde and N.R. Shanbhag, ”Toward achieving energy efficiency in presence of deep-submicron noise”, *IEEE Transactions on VLSI Systems*, vol. 8, no.4, pp. 379 – 391, Aug. 2000.

ABOUT THE AUTHOR

Thara took her B.Tech in Electrical and Electronics Engineering from University of Kerala, India in 1992 and M.Tech in Power Electronics from University of Calicut, Kerala, India in 1995. From 1995 to 1997, she worked as an instructor in the Electrical Engineering departments of two engineering colleges in India. From 1997 to 2000 she served as an Electrical Engineer in Kerala State Electricity Board, India. In the year 2002 she received Masters Degree in Electrical Engineering majoring in VLSI circuits design from the University of South Florida, after which she continued her studies towards doctoral degree. She has co-authored in two journal publications (IEEE Transactions on VLSI Systems and IEE journal on Computers and Digital Techniques). Her work on dynamic error modeling and reliability estimation is under review in IEEE transactions on Nanotechnology. She has co-authored in four peer-reviewed IEEE conference publications and two other publications. Her paper on Single-Event-Upset modeling and analysis was nominated for the ”Best Paper Award” and received ”Honorable Mention Award” in the International Conference on VLSI Design, 2006, where she presented the paper. She has also presented her work on SEU analysis in a symposium on VLSI design sponsored by NASA.