LAYOUT-BASED FAULT INJECTION FOR COMBINATIONAL LOGIC IN NANOMETER TECHNOLOGIES

By

BRADLEY KIDDIE

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

May, 2012

Nashville, Tennessee

Approved:

Professor William H. Robinson

Professor Bharat L. Bhuva

ACKNOWLEDGEMENTS

I would like to thank my adviser Dr. William H. Robinson for his guidance and support along the way with this research as well as in my pursuit of graduate study in general. I would also like to thank Dr. Bharat L. Bhuva for his helpfulness in this research and in providing advice. The Radiation Effects and Reliability group at Vanderbilt has been instrumental in sparking my interest in this field and providing feedback on the progress of my work, and I have my family and friends as well to thank for continued support and encouragement throughout the process.

This material is based upon work supported by the National Science Foundation under Grant No. CCF-0747042. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As integrated circuits (ICs) utilize nanometer fabrication technologies, concern has grown for design reliability in the presence of radiation-induced faults. Research has been conducted to better understand the sensitivity of ICs to the effects of radiation [1, 2]. This sensitivity is not confined to just memory but can also affect the logic components of the circuit. In particular, recent experimental results have demonstrated the susceptibility of combinational logic to radiation-induced faults and provided a comparison of the error rates to sequential logic [3]. With the complexity of modern digital circuits, combinational and sequential logic are typically synthesized using standard library cells [4]. With the size of these library cells in nanometer technologies, a single particle strike can create charge collection inside multiple cells [5-7], which leads to multiple transients within the logic of the design. As these transients propagate, they can create complex responses to affect the overall soft error rate (SER).

Typically, fault injection studies rely upon the insertion of a bit flip in a single storage element, using the assumption that a particle strike causes one fault [8-10]. However, these studies have not accounted for the spatial relationship of the layout (i.e., library cells) in nanometer technologies. Multiple transients caused by a single particle strike will originate in a single location in the layout of a circuit. Physically adjacent logic cells will simultaneously produce faults, which may then separately propagate through the circuit or may experience reconverging fan-out before arriving at a storage element where it can be detected. This thesis describes the coupling of layout information directly with the circuit behavior to conduct fault injection to nearest-neighbor logic cells. By assessing the logical propagation of these faults, the probability of error can be determined for designs synthesized with standard library cells. Several benchmark circuits were selected and synthesized with a 90-nm cell library, and the results show that spatially injecting multiple faults can produce significantly more complex behavior than when only a single fault is considered.

When using only a single injected fault (i.e., assuming that a particle strike only affects one cell), the output error percentage can be underestimated. Reconverging fan-out will limit the effects of the faults within the circuit. This thesis shows that in most cases, the probability of errors occurring at the output of a combinational circuit increases with an increasing number of radiation-affected cells, and that in some cases, recombination helps mitigate this effect and contribute fewer errors. These more complex cases will show that multiple-transient analysis is a unique challenge for the field of radiation effects, but the data presented herein will expose a number of trends which can be used to help design for better reliability in combinational logic.

The rest of this thesis is organized as follows. Chapter 2 covers previous and related work as a springboard for layout-based fault injection. Chapters 3 and 4 describe the design choices and methodology for this work. Chapter 5 discusses the results for the selected benchmark circuits. Chapter 6 concludes the thesis and offers some future avenues for study.

CHAPTER II

MULTIPLE-FAULT ANALYSIS

Related Work

The research area of radiation-induced multiple-fault analysis in combinational logic has been the result of gradual changes in the whole field of reliability analysis. The primary area of study in this field is of single-event upsets (SEUs). Much work has been put into understanding how charge-carrying particles strike a circuit, interact with nodes, and affect the charge stored at a point in a logic design [11, 12]. Research also examines how single faults dissipate or propagate through a circuit; this topic has been extended for multiple faults. The goal is to determine if there are ways to prevent fault propagation, through selective gate hardening [13, 14] or other specific design methods [15, 16]. Several works regarding faults in memory were referenced as a basis of general understanding of reliability analysis for this work [3, 11, 17]. As frequency rates increase in developing technology, combinational circuits as compared to sequential circuits become increasingly vulnerable to logic errors [3, 18].

In the study of fault propagation, several barriers present themselves to make a thorough understanding difficult. Early work that attempts to take timing and charge deposition quantities into account found that analyzing large circuits are nearly impossible with such detail [19]. Using such an analysis for multiple faults rather than just SEUs would be intractable in terms of required simulation time. Other approaches attempted a more symbolic, mathematical analysis [20, 21]. These analyses can be effective to produce values describing the circuit in general, but are difficult to visualize and use in practical application.

Miskov-Zivanov and Marculesu focus on a unified treatment of three masking factors: logical, electrical, and latching-window masking [22]. This analysis was applied separately to combinational logic and sequential logic, so this framework required timing information on latches included in the circuits or appended to the inputs/outputs of the combinational logic sectors. Their method of analysis was heavily mathematical, but managed to decrease simulation time by first analyzing small components of the circuit and then merging these results to form a picture of the overall circuit.

The key takeaway from observing the several methods attempted already is that, without some sort of approximation or the use of a supercomputer, it is necessary to either make some assumptions or to study only limited aspects of reliability; a 100% accurate and precise study of anything but the smallest circuits is too resource-intensive.

When examining combinational logic, an understudied characteristic is reconverging fan-out, which describes how a fault may propagate through a circuit to produce several effects. The faults may remain separated or potentially join together at a later gate, which could mitigate errors or magnify them. The level of possible reconvergence depends on each circuit, but an accurate analysis requires simulation of the circuit as a whole rather than as individual components (e.g., individual gates).

In terms of studying multiple upsets stemming from a single strike, a few different methods have been attempted. The simplest would be to simulate multiple faults over an entire circuit – any two nodes could be selected to receive upsets during a given run [23], and all of the results could be averaged together for a snapshot of the circuit as a whole with a single fault versus two faults. Producing results this way is more akin to simulating two simultaneous particle strikes, rather than merely one affecting multiple, adjacent nodes. While it may give some information on the possible behavior that multiple faults may produce, it is an inaccurate way to predict precisely what that behavior will be.

Another method is to use information from the circuit organization and induce upsets at nodes that are fan-in/fan-out neighbors [22, 24]. This method is decidedly more accurate, as these nodes are likely to be placed next to each other. However, physical adjacency is not always coupled with logical adjacency. If nodes are chosen this way, then the level of reconvergence will be much higher than that seen in an actual manufactured design. The only true way to simulate faults at adjacent nodes is to utilize layout information to determine which nodes have actually been placed next to each other on a chip. Some multiple-fault events will affect cells that are logically connected (i.e., within the same logic cone), while others will affect cells that are logically separated. When performed on several different designs, these simulations will provide an accurate look at the effects of radiation-induced multiple faults in combinational logic in general.

Studies thus far on the effects of multiple faults in combinational logic have therefore been found lacking in accuracy – a full circuit must be simulated as a whole, rather than as individual gates, and the correct mechanism must be chosen in order to not over- or under-estimate the effects of logical reconverging fan-out. At the same time, simulation methodology must be carefully chosen so that the tests are doable in a reasonable amount of time, while the circuits must still be large enough to be practically useful.

CHAPTER III

LAYOUT-BASED ANALYSIS OF COMBINATIONAL LOGIC

Design Choices

The contribution of this work is to conduct multiple-fault analysis coupled with actual physical layouts, as opposed to previous studies that use assumed dependencies or randomized multiple-fault analysis. This work focuses on using layout information from an automated place-and-route tool. This method of circuit characterization enhances awareness of all information available during the design of a combinational logic module. By comparing the adjacent cells in these layouts, reconvergence is also observed and studied within combinational logic.

Many different methods exist to create a circuit layout from design synthesis. Therefore, the most general steps are taken for this research in order to provide a snapshot of a typical manufactured circuit and its behavior with multiple induced faults. Multiple circuits were chosen as well, both to provide different sets of results and to enable general conclusions about combinational circuits as a whole.

In future analyses, this work can be expanded to more specific synthesis methods or larger circuits. One aim would be to see if area-minimized synthesis was more reliable versus power-minimized synthesis. Or, larger circuits should show the greater or smaller effects that a strike in a single location may cause. It is anticipated that larger circuits would see smaller effects overall due to a comparably smaller area that is directly affected by a particle strike.

However, the scope of this work requires some limitations; therefore, standard synthesis using tools by a single manufacturer (Synopsys) was selected. All default options are chosen when possible, and standard benchmark circuits are sent through the process. Larger circuits are often composed of these smaller benchmark circuits, so they are valuable pieces to analyze. Understanding the reliability characteristics of different pieces of an overall design allows us to determine the reliability of the circuit as a whole.

Since sequential circuits are not analyzed in this work, circuit timing was not considered. Each circuit is studied as a stand-alone combinational logic model, where a given set of inputs and a particle strike at a specified physical sector of the circuit will produce a definite set of outputs. Given the speed of microprocessor designs of today, it is easily possible for the effects of a particle strike to last multiple clock cycles [3], so this is well within the boundaries of normal circuit operation.

Once simulation data are collected for varying numbers of adjacent faults per circuit, these data can then be analyzed to show precisely how much of an impact particle strikes will have on combinational logic. It is expected that there will be a wide range of behavior, but this study will show how multiple-fault characteristics differ from the single-fault reliability approach, as well as show what limitations there may be, through the effects of reconverging fan-out and the behavior of different gates or circuits.

Synthesis Review

One of the key background elements of this work is how it relates to circuit design and manufacturing. Figure 1 illustrates the synthesis process of a typical design. In this process, an idea is first given specifications, then is written down with a hardware description language in code (e.g., VHDL [25], Verilog [26]). At this point, the design is sent through automated synthesis, which performs a series of operations on the code: it is first paired with a standard library of gates to produce a gate-level netlist, then a physical layout of the gates as arranged on a die is generated, then ultimately a portfolio of information is generated that can be used to automatically manufacture the circuit on a chip. This thesis describes how the gate-level netlist can be used in conjunction with the physical layout in order to understand the effects of single events that induce multiple transients in combinational logic. In order to better understand the relation between these two pieces of information, it is worthwhile to explore the synthesis process.

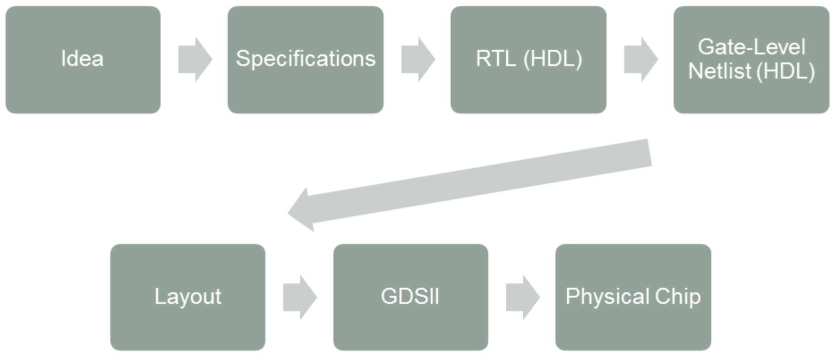

Figure 1: Synthesis flow for circuit design.

This work uses the Synopsys Design Compiler and IC Compiler electronic design automation (EDA) tools to perform all of the required processing on the circuits before simulation. Initially, Design Compiler reads the user-supplied code and links it with a selected cell library. Next, the compilation step will rewrite the given code to ensure that its format matches that of the library and is optimized for the logic gates available within that library. This result is called the “mapped” code, which is used directly in this thesis to run simulations using a logic simulator. The mapped code will likely differ in appearance from the original written code, but functionally they are the same. The compilation process can be modified based on the preferences of the user – certain gates or blocks can be preferred over others, or the design can be synthesized to prioritize timing, area, or power constraints. The flexibility of the code without any physical information can be exploited to pursue these design choices.

Next, IC Compiler transforms the final code into a chip layout. The synthesized code is again linked with the cell library, and then a basic floorplan is initialized. The user has the option to prioritize different characteristics or include specific features in the layout. IC Compiler will then optimize several items. Typically, area is minimized by placing the cells in an efficient grid. Congestion can be minimized by rearranging these cells according to how they are used and how they interact with other cells. The automated place-and-route manages all of these concerns as well as others, such as minimizing power, and design for test.

EDA tools such as these are used for all modern chip designs. Manual place-and-route to produce layouts is an overly complex and lengthy process, so tools such as the Synopsys suite allows the different synthesis choices to be made quickly and automatically. Given the large number of different layouts that can be generated for a given design, this thesis chooses to utilize EDA tools in a very general matter, producing a “normal” layout for each chosen circuit. There have been studies that examine the effects of different synthesis constraints on reliability for single fault propagation [27], and this may be extended for multiple transients in future work, but this thesis focuses only on “default” synthesis. The effect of using automated place-and-route is that adjacent cells may or may not be logically related, which produces the interesting results found in this work.

CHAPTER IV

METHODOLOGY

Following an industrial design flow from circuit selection through synthesis and layout development, this research requires a thorough methodology with information collected from each step. First, circuits are selected to test. Second, layout information is generated for each. Third, faults are injected to collect data for analysis.

Test Circuits

The first step was the selection of appropriate circuits to analyze. Thoroughness in simulation was the primary intent, so smaller circuits were chosen out of the 74XXX suite, enabling us to perform exhaustive fault injection. The use of the 74XXX suite ensures that these logic structures are commonly used and worthy of examination. Several of these circuits were used in Verilog form, taken from the University of Michigan suite [28]. The 74182 is a 4-bit carry look-ahead generator, the 74283 is a 4-bit adder, the 74L85 is a 4-bit magnitude comparator, and the 74181 is a 4-bit ALU. This selection gives a fairly good cross-section of common functions that are also at a smaller, more manageable size for simulations. Table 1 provides other pertinent information about these circuits and their relative sizes. Table 2 gives further information about the purpose of each circuit's output for better insight into analyzing the results of the simulations later on. The number of gates is defined via the original code in the University of Michigan suite; when linked to a library in the synthesis process, this count is reduced to the number of nodes listed.

Table 1: Circuits selected for simulation and analysis

Circuit	Description	Gates	Inputs	Outputs	Nodes
74182	4-bit carry look-ahead generator	19	9	5	14
74283	4-bit adder	36	9	5	12
74L85	4-bit magnitude comparator	33	11	3	28
74181	4-bit ALU	61	14	8	39

Table 2: Circuit output information

Circuit	Output	Description
4-bit carry look-ahead generator	PBo	Propagate signal to cascade to another block
	GBo	Generate signal to cascade to another block
	CNX/Y/Z	Three carry out signals based on inputs
4-bit adder	S[3..0]	4-bit sum of 4-bit inputs A and B
	C4	Carry-out associated with sum
4-bit magnitude comparator	ALBo	Signal high if A < B
	AGBo	Signal high if A > B
	AEBo	Signal high if A = B
4-bit ALU	F[3..0]	4-bit computation based on 4-bit inputs A and B
	X	Propagate signal (look-ahead carry)
	Y	Generate signal (look-ahead carry)
	CN4b	Ripple carry output associated with F
	AEB	Indicates if A = B

For this thesis, all four circuits were examined and tested through all possible simultaneous faults. The 4-bit carry look-ahead generator, the 4-bit adder, and the 4-bit magnitude comparator are small enough that every possible input combination was used in each simulation run. The 4-bit ALU circuit required additional premeditative analysis using the Synopsys TetraMAX tool for automated test pattern generation [29] because of its larger size; covering all input vectors would require a significant amount of simulation time.

Ideally, analyses of the effects of multiple adjacent upsets would include much larger circuits, such as those from the standard ISCAS'85 benchmark suite. However, the primary goal of this work is thoroughness in all of the possible fault combinations. Testing circuits with a very large number of nodes is beyond the scope of this work; rather, smaller "building block" circuits can be analyzed and the effects will contribute similarly to when these smaller circuits are used in larger designs.

Layout Generation

Each circuit was taken in Verilog form and sent through a series of Synopsys software suites to emulate the typical design process taken by a circuit designer. These EDA tools produced a place-and-route layout of the circuit. The aim is to ultimately test these circuits for single-event multiple upsets when the circuit is in the form seen by most designers.

Using the standard 90-nm Synopsys design library [30], the Synopsys Design Compiler synthesizes the circuits and produces gate-level netlists. In order to allow for fault insertion at the output of each actual gate, only low-level gates (i.e., no multiplexers, adders, or other complex blocks) were permitted in the synthesis process. More information about the logic gates used and the library chosen is available in Appendix A. Synopsys IC Compiler then creates a layout to show which gates are adjacent. These tools were used in the most general manner possible to produce these circuits in "common" forms. Since only combinational circuits were selected, clock and timing information was not described or constrained, area was minimized, and all other settings remained unconstrained or at their default positions.

At this point in the methodology, two important pieces of information are available for each circuit: (1) a Verilog file describing the gate-level netlist, which can be used to run simulations inserting faults at the desired nodes, and (2) a layout of this same netlist, showing which gates would be physically adjacent to each other if the circuit were to be synthesized at default settings and manufactured.

Fault Injection Models

There are multiple options for inserting a fault at a node into a circuit. Simulations examining single event upsets typically include an additional input line paired with XOR, OR, or INV-AND gates to set a logic signal high, low, or flip its value. Previous work conducted on a related study showed that set-high and set-low faults generally occur at similar rates [31], so it was chosen in this work to use a glitch model for faults. Although the XOR model is also used by several other similar studies in the field of multiple transient effects, it is more precise to run simulations with faults modeled as setting values high (for particle strikes in PFET transistors) and again with faults modeled as setting values low (for particle strikes in NFET transistors) and to use these data to understand the circuit's behavior. Of course, doing this will result in twice as much information necessary to characterize a circuit. Each circuit in this work was simulated with the XOR model for fault injection, which provides a moderate response as compared to running complementary simulations for set-high and set-low models. For an added depth to this work, some circuits were also simulated using the set-high and set-low models to compare both methodologies.

The most common point at which a fault is produced is at the output of a gate [22]. Since this work is primarily concerned with the logical behavior of a circuit design rather than physical effects, faults within a cell are not considered. Inserting faults at the output of each cell presents the effects of faults between any two gates when a node is upset. The synthesized Verilog code was modified to allow the introduction of a fault at each of these locations, by inserting an XOR gate, OR gate, or INV and AND gates with an additional input line. This input line can be triggered high in order to flip that bit, set it high, or set it low, respectively, to simulate a fault. The last column of Table 1 indicates how many of these nodes were available for each circuit. See Figure 2 for an example of inserting logic gates to allow fault triggering. More information on fault insertion, including a code example, is available in Appendix B.

Previous research simulated multiple faults by injecting faults at 2 or 3 random points in the circuit [22] or gates that are fan-in/fan-out neighbors [24]; in real operation, these assumptions may not accurately represent physical adjacency. A single particle strike will cause one or more errors in a single localized region, affecting gates that are physically next to each other (and therefore may or may not be fan-in/fan-out neighbors). Therefore, the layout information produced by the IC Compiler is used to identify adjacent nodes. Once these nodes are identified, their fault inputs can be set high accordingly in order to simulate faults at multiple nodes during a test.

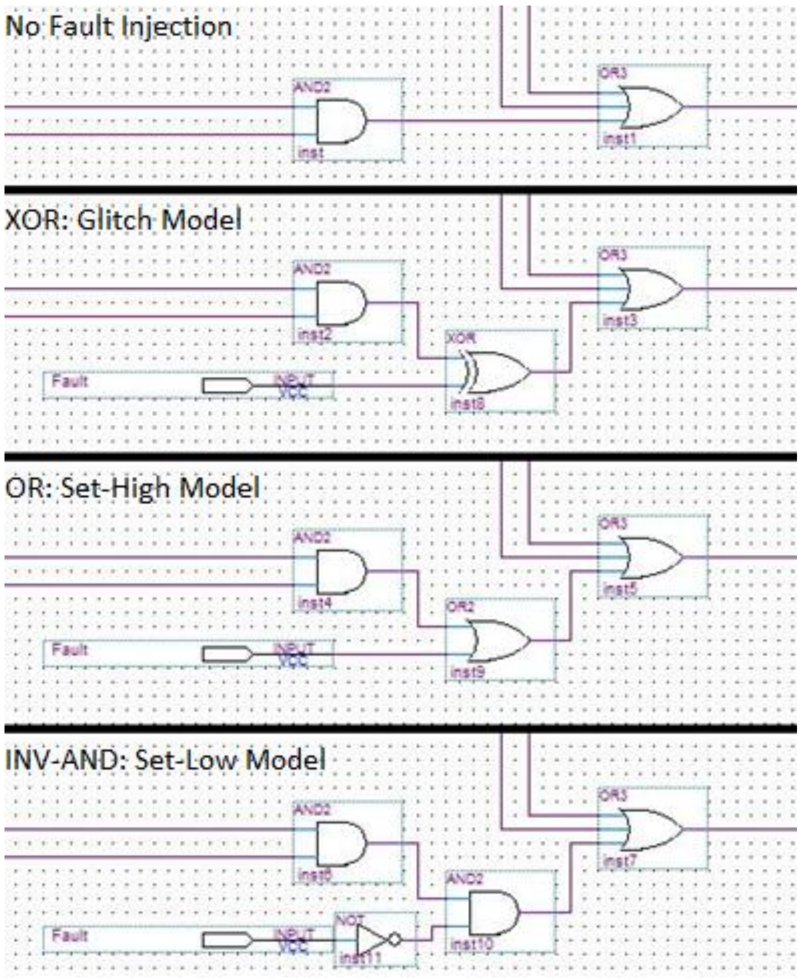

Figure 2: Example of fault insertion. The top figure represents an original circuit. The second is of a glitch XOR model, the third is of a set-high OR model, and the fourth is of a set-low INV-AND model. Setting Fault = 1 upsets the output of the AND gate, simulating a glitch, stuck-high fault, or stuck-low fault at that node.

Circuit Simulation

ModelSim [32] is used to simulate the circuits for all possible input vectors for several runs of each circuit: (1) no faults injected, (2) one fault injected at each internal node at a time, and (3) each possible combination of 2, 3, 4, and up to a maximum of 5 adjacent faults at a time. Figure 3 shows an example of gate adjacency where multiple upsets may occur.

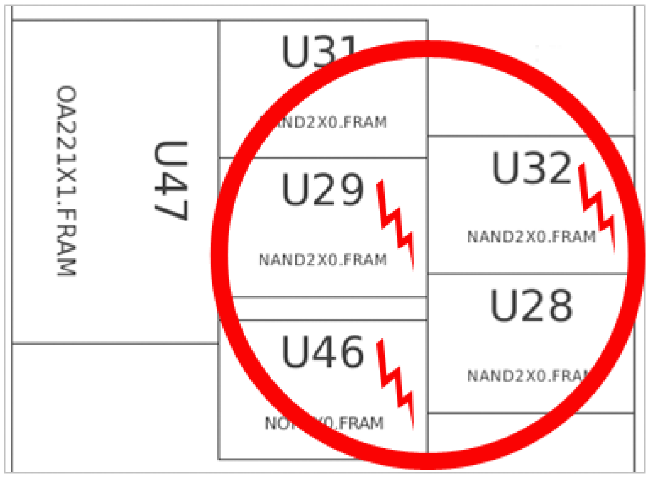

Figure 3: Example of gate adjacency in layout; cell U29 has five other cells in close proximity, allowing for simulation of several upset nodes centered at the U29 location. For example, with a particle strike affecting the circuit area as shown (red circle), ceel U29, U32, and U46 could simultaneously experience upsets.

For small circuits, all of the specified multiple-fault testing can be done in a matter of minutes. With increasing numbers of inputs and nodes to be tested, testing time increases exponentially. For each test, 2 or more physically adjacent gates are selected. The fault line inputs for these gates are set high, inducing a glitch at the output of the gates. ModelSim then runs through all of the given input vectors and records the output for each set of inputs. This information is identified by which faults are active during the test, and each test can later be compared to a “base run” with no faults during analysis to determine which parts of the circuit are most affected when strikes occur.

As an example pertaining to Figure 3, single-fault testing would run through all input combinations with a fault inserted at the output of gate U29. Two-fault testing would repeat this process with faults inserted at each pair of gates (e.g., U29 and U28, U29 and U32) for all 5 possibilities. Three-fault and four-fault testing each have 10 possible combinations that include gate U29. And lastly, there are five different possible fault combinations when testing five faults at a time centered at gate U29. Each of these tests is run, and data are collected at the outputs of the circuit.

With small circuits, it is possible to run through all possible input combinations for each test in ModelSim. Beginning with the 4-bit ALU and any further work in larger circuits, this method becomes intractable with the large number of inputs. Therefore, Synopsys' TetraMAX tool was used to generate an automatic test pattern – a series of input combinations that provide 100% coverage of fault detection for single stuck-at faults. These test vectors can also provide multiple-fault detection [33]. This set of vectors is used instead, enabling much faster testing while still retaining a good demonstration of the variety of input possibilities.

Since this research is concerned only with combinational logic, not storage elements, no clock signals are considered, and therefore timing and glitch duration are ignored. The goal is to insert each fault statically; for each simulation, all given input combinations will be run while a specific fault configuration is set, and information about the output values are collected.

The tests can all be scripted and run automatically, and the results are then imported into a numerical analysis program for processing. For each circuit, data can be grouped according to: (1) the node at which the inserted fault(s) were centered, (2) the output at which an error was detected, (3) the type of gate at the center of the fault(s), or (4) whether the faults affected logically connected or logically separated gates.

With a 100% coverage of all possible fault combinations (up to 5 faults simulated at a time) and all possible input vectors, the amount of data for even the small circuits selected for this work is significant. The data are separated according to the number of simultaneous faults and where they are located, then imported into a numerical analysis program. Herein it can be analyzed for trends and to observe the behavior of these circuits under the stress of single-event multiple-upset incidents.

CHAPTER V

MULTIPLE-FAULT RESULTS

Individual Benchmark Circuits – Output Error Analysis

Several different methods of analysis are possible based upon the collected data. The initial approach identifies each node according to its physical location and gate type and views its effect on the circuit outputs when one or more faults are inserted centered at that node. Each fault simulation is compared to a simulation with no faults, and the data are aggregated to state what percentage of input vectors produces an error at each output. Unless otherwise stated, the data shown are from simulations using a glitch model for fault injection.

Figure 4 shows a collection of data for the 4-bit adder. The percentage of tests that result in errors seen at the outputs for each combination of faults and input vectors are averaged together to show which outputs of the circuit are most likely to have an error with single or multiple faults. Traditionally, circuit designers would expect that the likelihood of an error seen at an output would increase progressively with an increasing number of faults. However, due to logical reconvergence, this is not always the case. Three of the five outputs are shown to actually decrease in the number of errors at some point during this testing. For example, the probability of an error present at output **S[1]** decreases by 2.4% going from three adjacent faults to four, and again from four to five.

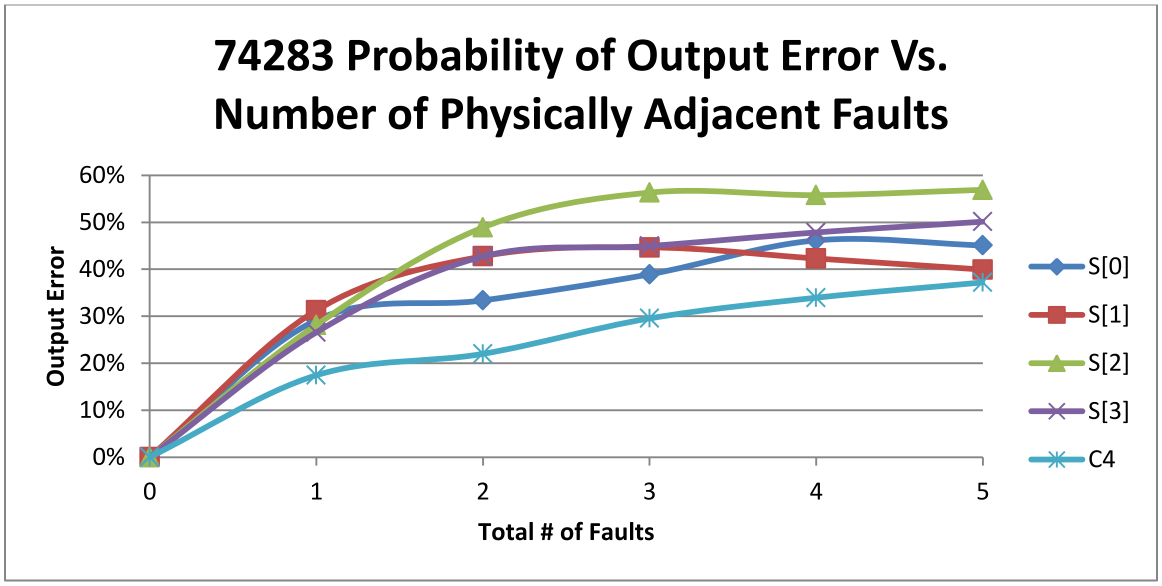

Figure 4: Percentage of input vectors that result in errors seen at an output (**S[3-0]** and **C4**) vs. the number of physically adjacent faults inserted into the circuit at any gate in the circuit. The circuit is a 4-bit fast adder; **S[3-0]** represents a sum and **C4** is the carry-out.

In general, with an increasing number of faults in a circuit, the probability that an error is seen at an output tends to rise most quickly from no faults to a single fault, then asymptotically for two or more faults injected. When this probability progressively increases (due to a low level of logical reconvergence), its behavior is logarithmic. When the probability decreases, such as in the case of output **S[1]** mentioned earlier (i.e., a higher level of logical reconvergence), then its behavior is more unique to its case and prone to change based on the circuit and the layout.

The other circuits produced show very similar results from the layout-based fault injection. Traditional models cannot be applied to multiple-fault analysis; rather, some methods of circuit design or some logical structures may be more or less vulnerable to multiple-fault effects.

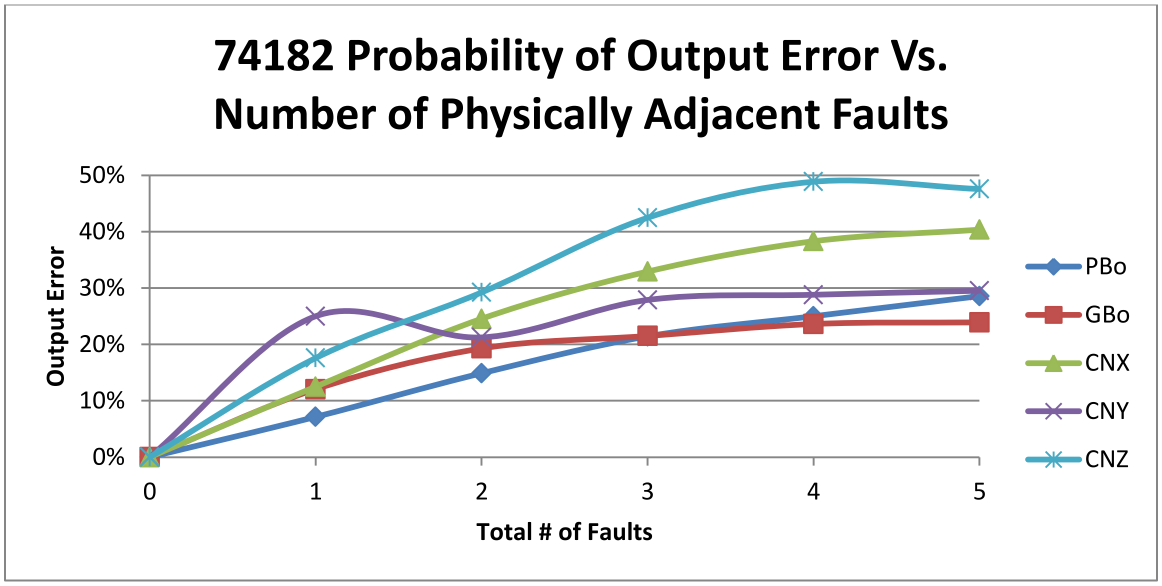

Figure 5: Percentage of input vectors that result in errors seen at an output (**PBo**, **GBo**, **CNX**, **CNY**, and **CNZ**) vs. the number of physically adjacent faults inserted into the circuit at any gate in the circuit. This circuit is a 4-bit carry look-ahead generator; **PBo** and **GBo** are propagate and generate cascade signals, and the others are carry-out signals.

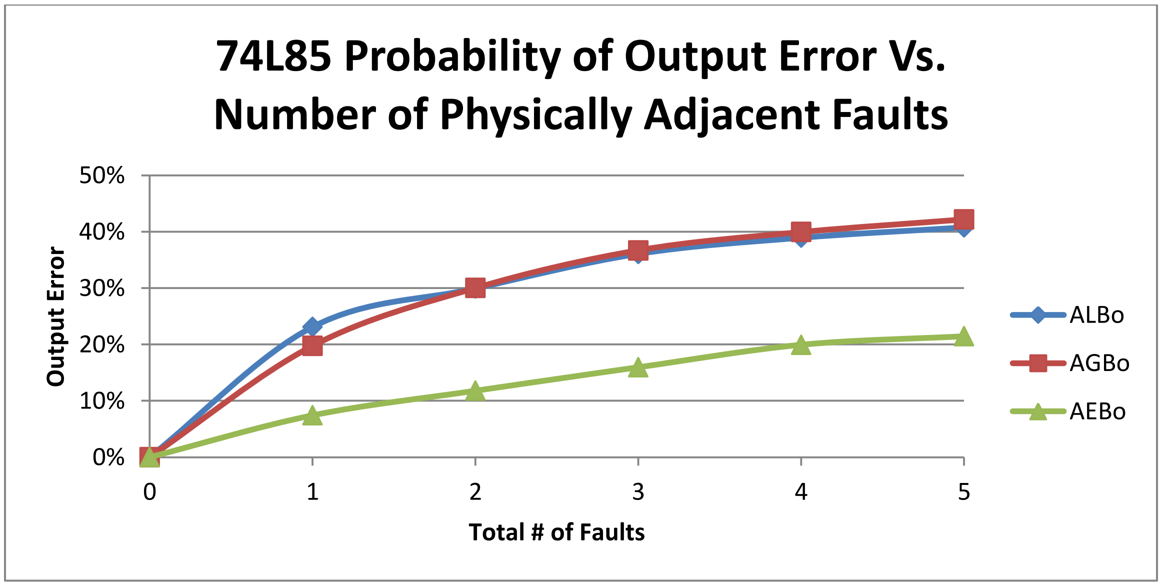

Figure 6: Percentage of input vectors that result in errors seen at an output (**ALBo**, **AGBo**, and **AEBo**) vs. the number of physically adjacent faults inserted into the circuit at any gate in the circuit. This circuit is a 4-bit magnitude comparator; the outputs indicate if A is less than, greater than, or equal to B.

Figures 5 and 6 show similar data for two larger circuits: a 4-bit carry look-ahead generator and a 4-bit magnitude comparator. Figure 5 again demonstrates a high level of logical reconvergence in output **CNY**. If a dip is present in a curve on these graphs, it indicates that regardless of where a multiple-fault-inducing strike occurs in a circuit, it is likely that these faults will reconverge in the logical path towards the indicated output. Here, for example, injecting a single fault into the circuit will on average present a 25% probability of seeing an error at the **CNY** output. However, when two physically adjacent faults are injected into the circuit, many of these faults will reconverge and cancel each other out, with the result that the **CNY** output will only have a 21% probability of seeing an error. Data will be presented further on to show how this occurs on a gate-by-gate basis to present a little more detail.

Conclusions can begin to be drawn based on this data on the reliability of specific circuit structures. Figure 5 shows that the carry-out signals of this standard carry look-ahead circuit are most fragile when it comes to single- and multiple- faults. However, the generate and propagate cascade signals are more reliable. Note that, given the design of the circuit, there are several logic gates shared between the logic paths leading to the **CNX**, **CNY**, and **CNZ** outputs. Specifically, much of the logic in the **CNX** logic path is contained within **CNY**, and much of that is contained within **CNZ**. Although there are therefore more gates and vulnerabilities in the **CNZ** path over **CNY** and **CNY** over **CNX,** there are also more opportunities for reconverging fan-out and logical masking to have effects. This is why the results do not show errors increasing in order between **CNX**, **CNY**, and **CNZ**. Errors at outputs depend upon the logical masking of a circuit, and under multiple-fault analysis, also the physical layout (i.e., gates that are adjacent and experience related upsets).

In Figure 6 looking at the output error probabilities, we see that a magnitude-comparator runs a high risk of giving an unreliable answer when the two input numbers are different. The **ALBo** and **AGBo** output signals have a 30-40% chance of registering an error when multiple faults occur in the circuit, while the **AEBo** output signal only ranges from 10-20%. Observations like these can help circuit designers understand the weak points in their designs and therefore which circuit functions should be avoided or selectively hardened against radiation effects.

Individual Benchmark Circuits – Alternative Fault Injection Techniques

As mentioned earlier, using an XOR glitch model is typical in fault injection studies, and also allows for succinct data reporting. However, to be more precise, a particle strike in a circuit will most likely affect either NFETs or PFETs, with only a rare occurrence of both. Although the focus of this work is on understanding multiple-fault effects and reconverging fan-out with layout information, two of the test circuits were also simulated using set-high and set-low models, to further enhance the understanding of fault behavior in combinational logic. These tests can be examined separately to understand PFET and NFET strikes individually, or averaged for the entire circuit's response, assuming particle strike effects are equally probable in either location.

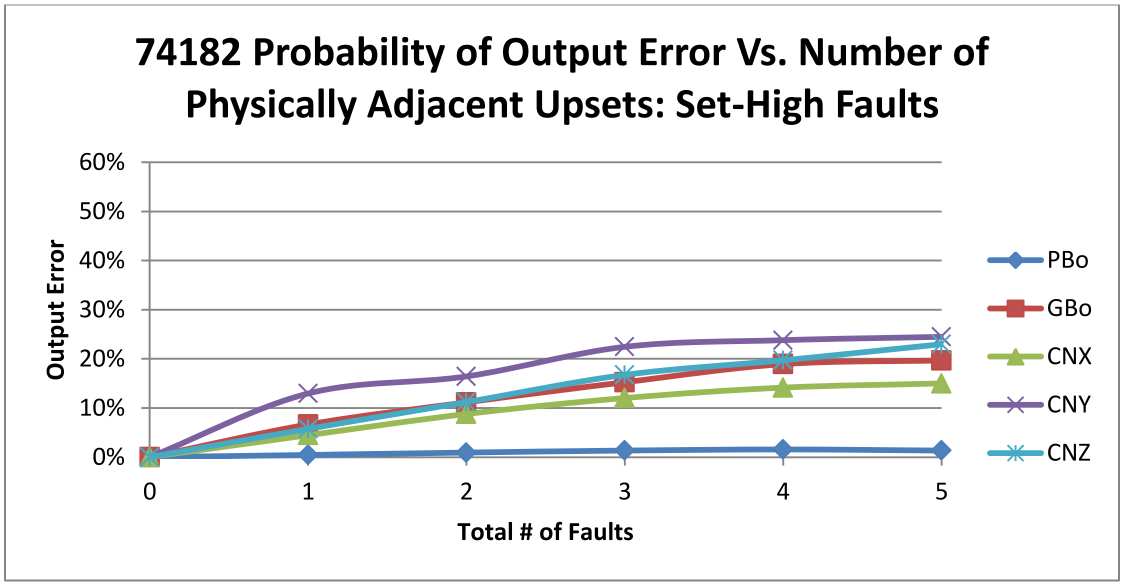

Figure 7: Percentage of input vectors that result in errors seen at an output (**PBo**, **GBo**, **CNX**, **CNY**, and **CNZ**) vs. the number of physically adjacent set-high faults inserted into the 4-bit carry look-ahead generator circuit at any gate in the circuit. Compare to Figure 5's XOR model.

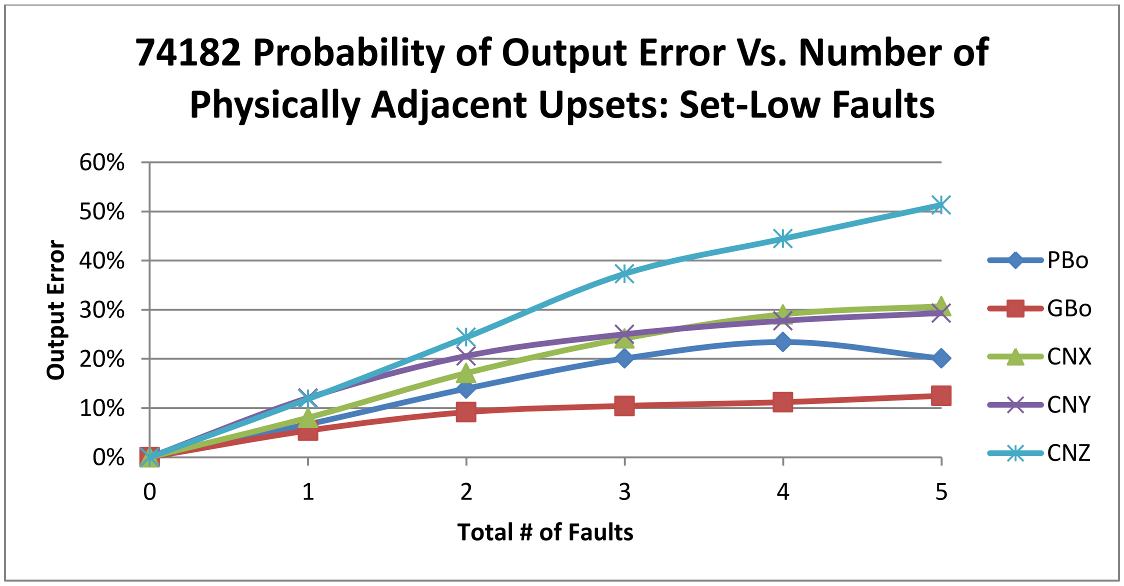

Figure 8: Percentage of input vectors that result in errors seen at an output (**PBo**, **GBo**, **CNX**, **CNY**, and **CNZ**) vs. the number of physically adjacent set-low faults inserted into the 4-bit carry look-ahead generator circuit at any gate in the circuit. Compare to Figure 5's XOR model.

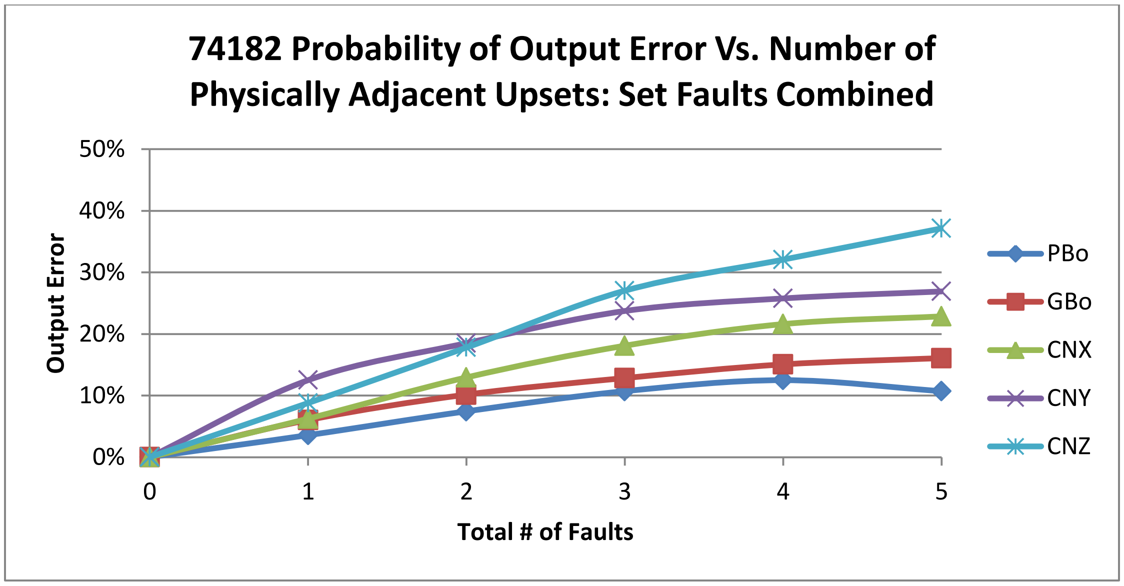

Figure 9: Percentage of input vectors that result in errors seen at an output (**PBo**, **GBo**, **CNX**, **CNY**, and **CNZ**) vs. the number of physically adjacent faults inserted into the 4-bit carry look-ahead generator circuit at any gate in the circuit. Figures 7 and 8 averaged.

Figures 7 and 8 demonstrate the results of these alternative methods for the 4-bit carry look-ahead generator, where faults are inserted via set-high and set-low models, respectively. These results can be compared to each other and with the XOR model (shown previously in Figure 5) to gain further precision in fault characterization. Notice that the set-high model presents a much lower chance of error and that the set-low model presents a higher chance of error, indicating that signals within the circuit tend to be high rather than low. The XOR glitch model used before gives a moderate response that is in between both of these cases. While it gives a succinct, reasonable result, generating data with these set-high and set-low models provide greater precision.

The data of Figures 7 and 8 can be averaged to produce Figure 9, under the assumption that the likelihood of a particle strike in an NFET region of a circuit is equal to that in a PFET region. Comparing Figure 9 to Figure 5, the circuit shows similar behavior. The three carry outputs are still the least reliable. **CNY** no longer has a dramatic decrease in error between single- and dual-fault analyses, but in both figures, it shows the worst response in single-fault analysis and is surpassed by **CNZ** for multiple-fault analysis. In this analysis as well, note that **CNX**, **CNY**, and **CNZ** experience increasing error rates in that order for multiple-fault analysis, consistent with the fact that **CNX** is a subset of **CNY** and **CNY** is a subset of **CNZ**.

Overall, Figure 9 shows lower probabilities of output error given the combined set-high and set-low models versus the glitch model of Figure 5. But this sort of behavior depends on both the specific circuit as well as the typical input vectors. Behavior is similar between the two; just overall levels differ somewhat. To offer another look at this concept, the 4-bit adder was run through the same analysis.

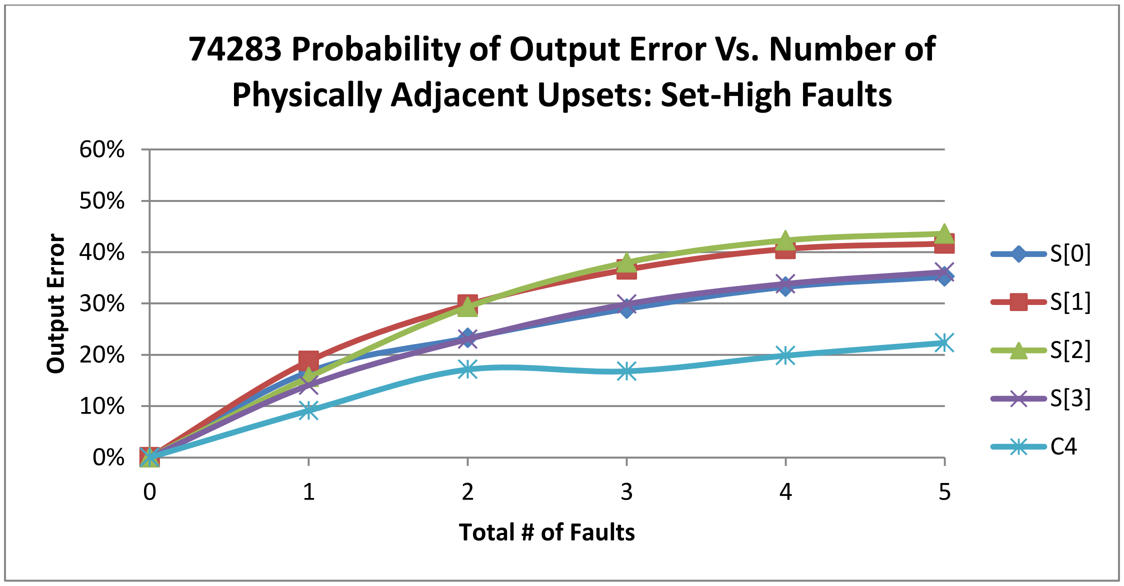

Figure 10: Percentage of input vectors that result in errors seen at an output (**S[3-0]** and **C4**) vs. the number of physically adjacent set-high faults inserted into the 4-bit adder circuit at any gate in the circuit. Compare to Figure 4's XOR model.

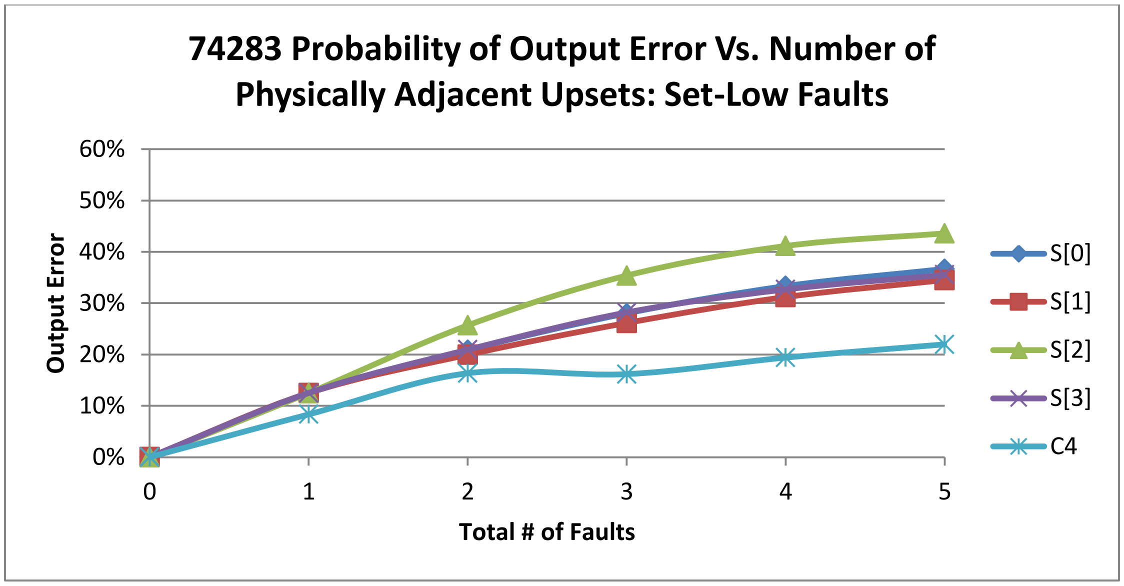

Figure 11: Percentage of input vectors that result in errors seen at an output (**S[3-0]** and **C4**) vs. the number of physically adjacent set-low faults inserted into the 4-bit adder circuit at any gate in the circuit. Compare to Figure 4's XOR model.

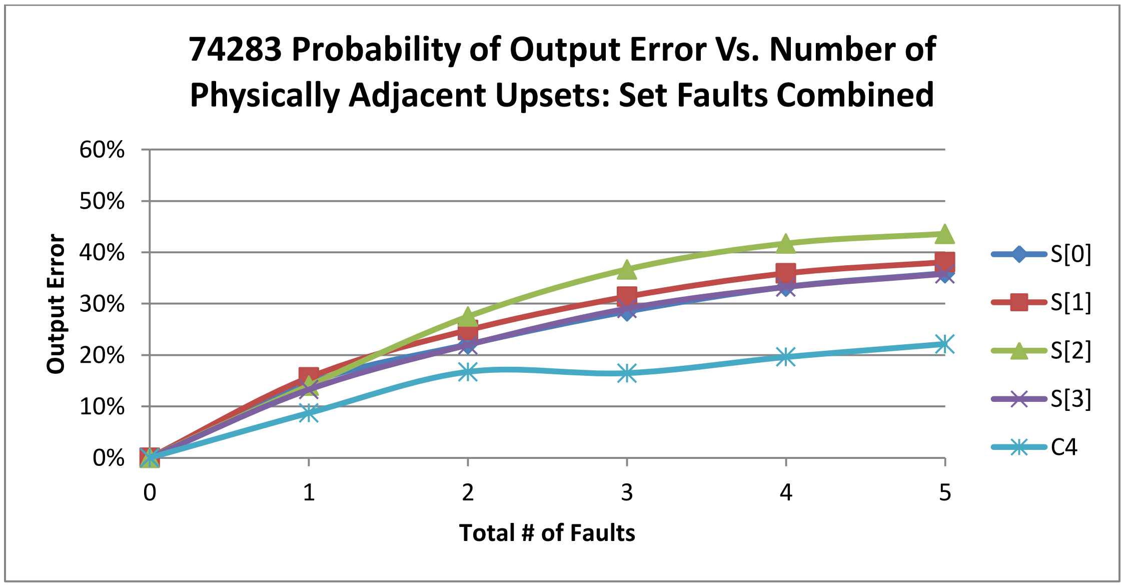

Figure 12: Percentage of input vectors that result in errors seen at an output (**S[3-0]** and **C4**) vs. the number of physically adjacent faults inserted into the circuit at any gate in the 4-bit adder circuit. Figures 10 and 11 averaged.

Figures 10 and 11 show the set-high and set-low simulation data for the 4-bit adder circuit. Since this circuit has a very standard design, where outputs depend entirely on the inputs and do not have a limited range, the signals within the circuit are not particularly weighted towards being high or low, and the set-high and set-low fault models correspondingly provide similar results. When averaged together, there are very similar results between Figure 12 and the earlier Figure 4 that used glitch model data. The carry output is still most reliable, and **S[2]** is still least reliable. Error rates are slightly lower with the set-high/set-low data, but besides that, there are still similar behaviors shown. Note that high levels of reconvergence are still displayed in this analysis; the **C4** output sees a slight decrease in errors seen at that output going from when 2 faults are injected in the circuit to when 3 faults are injected in the circuit.

In summary, using set-high and set-low fault models provide a more precise look at fault generation and propagation in a circuit. However, classic glitch models still present similar views of circuit behavior, with the added benefit of being able to characterize the circuit in half the simulation time and model it with half the information. Hence, simulation data using the glitch fault model will be used for the rest of the analyses in this thesis.

Individual Benchmark Circuits – Gate-Specific Analysis

All of the figures shown thus far show the change in errors seen at circuit outputs as it relates to the number of physically adjacent faults injected into the circuit, regardless of location. This gives circuit designers good insight into the reliability of specific functions of these circuits. But another interesting angle would be to observe where the injected faults originated from, and therefore which logic gates and which sectors of the circuit contribute most to reliability concerns.

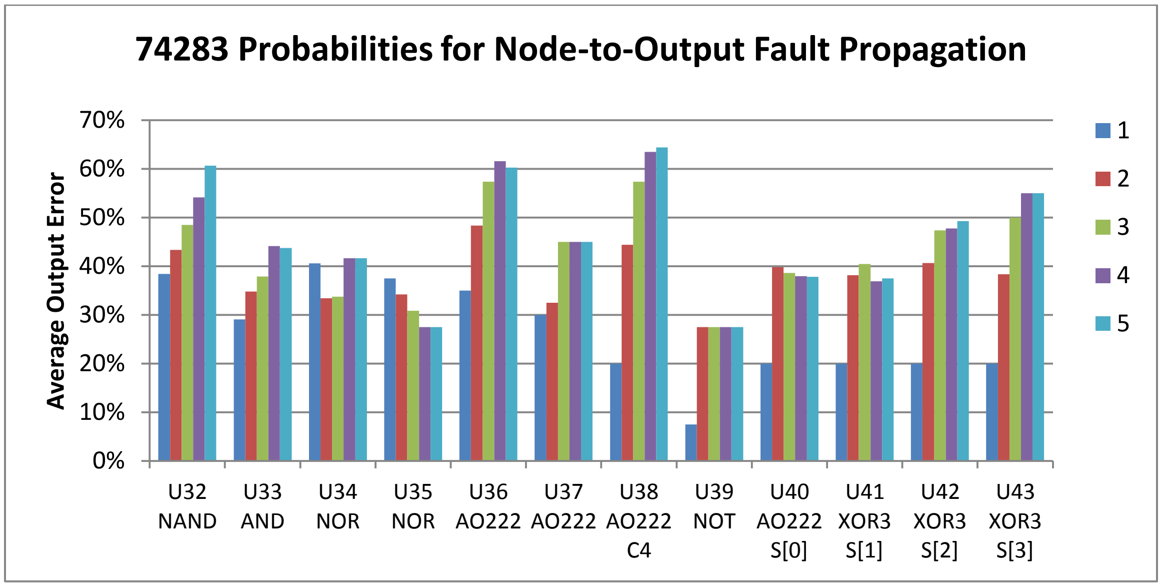

Figure 13: Percentage of input vectors that result in errors seen at any output of the 4-bit adder circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output. U32-U43 are individual gates in the circuit, some internal and some directly connected to circuit outputs as noted.

Figure 13 shows the same data that from earlier in this discussion (Figure 4) for the 4-bit adder, arranged according to the node where single and multiple-fault testing was centered. The percentage of tests that result in errors seen at any output for each node are all collected and averaged, according to how many faults are simulated at a time (1-5). This collection allows a look at specific sectors of the circuit layout to see how individual nodes or areas affect reliability. Much of the data show an increasing probability of output errors with an increasing number of injected faults, but again, this is not always the case. Nodes can experience little change (e.g., U34) or even show decreasing fault propagation probabilities with an increasing number of injected faults (e.g., U35), which demonstrates a high level of logical reconvergence.

When a cell is surrounded by logically related cells (e.g., fan-in/fan-out neighbors), then logical reconvergence may occur under the effect of multiple faults, presenting more reliable performance than if just one fault was simulated. Alternately, if a cell is surrounded by unrelated cells, then unrelated faults may propagate through the entire circuit and not be mitigated at all, worsening the effect on the overall circuit. It is important to allow for both of these possibilities in a unified analysis such as that presented here, in order to provide a reasonable picture of how this circuit would function in an actual design.

One notable feature of the chosen methodology is the way nodes on the boundary of a design contribute data. Node U39, for example, is located in the corner of the layout and there is only one adjacent gate. Tests can be run for a single fault or 2 faults including U39, but not for 3, 4, or 5 faults. In a larger circuit design which uses the 4-bit adder structure alongside other sub-designs, a particle strike in this location of the chip would affect the 4-bit adder sector as shown, and the other surrounding structures would exhibit their own behavior. If these sub-designs interact later in the circuit, this behavior can be predicted simply by summing the effects of this individual sub-designs. For the purposes of this testing, the test results for 2 faults at U39 are copied over to 3, 4, and 5 faults to fill out the diagram; this technique should also represent how the circuit would actually operate as part of a larger design.

The data shown in Figure 13 averages together the results of each output equally. However, of course, injecting faults at gates will result in more errors at some outputs than at others. To see this behavior more precisely, this graph can be split up to examine the effects of faults injected centered at each node on each individual output of the circuit. The results will show that for single-fault testing, only gates within the logic path of an output contribute to potential errors at that output. Typically, the closer to an output a logic gate is, the stronger its effect is, as there is less room for logical masking. With multiple-fault testing, overlapping particle strikes at physically adjacent cells will contribute to output errors as well. The following graphs demonstrate these ideas for the 4-bit adder circuit.

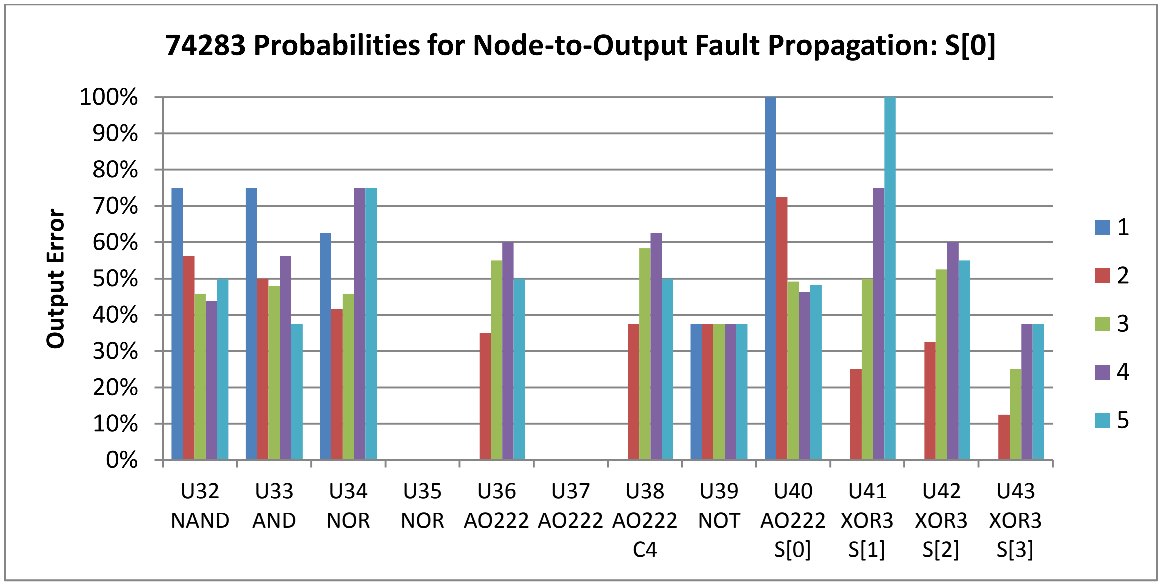

Figure 14: Percentage of input vectors that result in errors seen at the **S[0]** output of the 4-bit adder circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

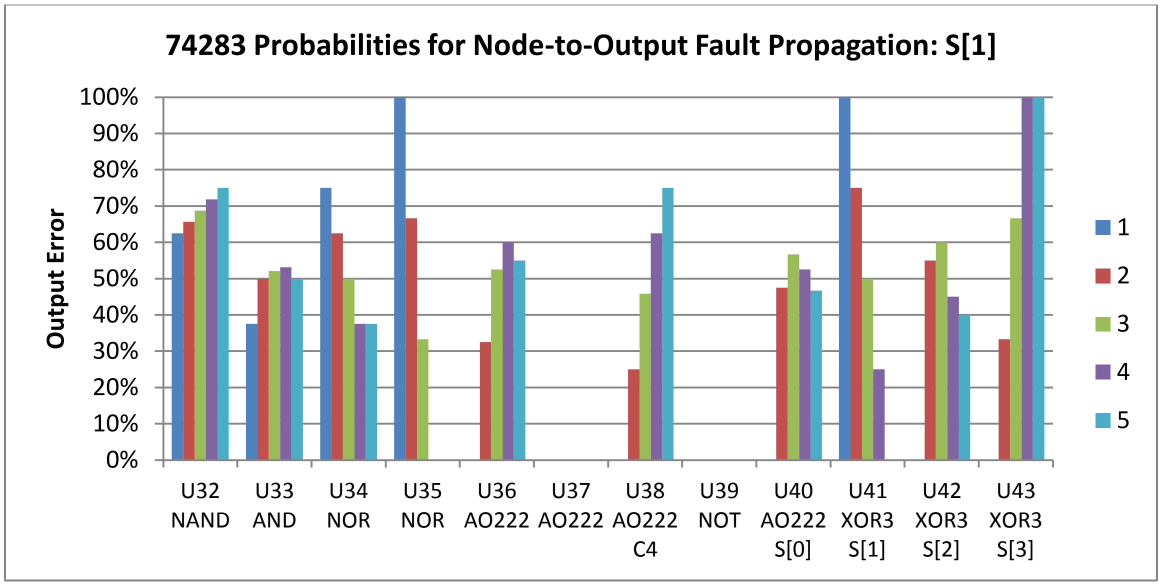

Figure 15: Percentage of input vectors that result in errors seen at the **S[1]** output of the 4-bit adder circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

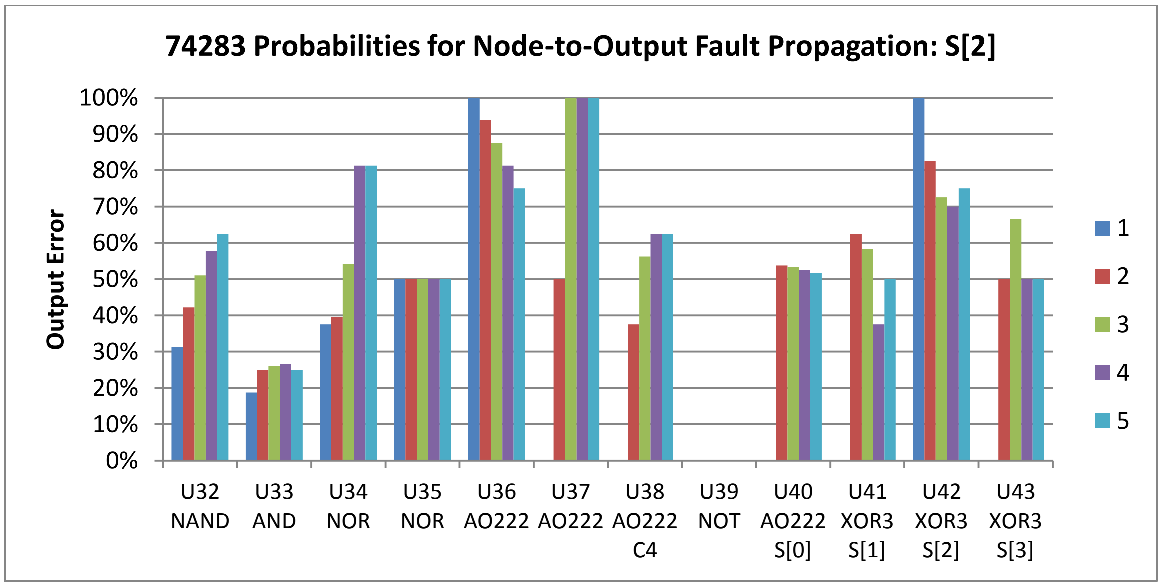

Figure 16: Percentage of input vectors that result in errors seen at the **S[2]** output of the 4-bit adder circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

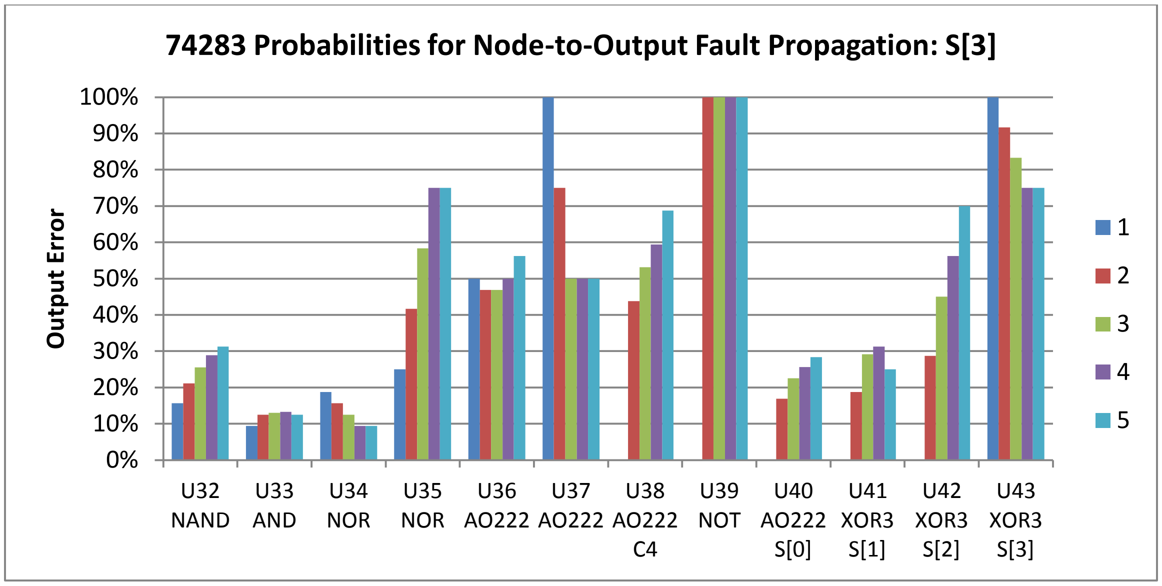

Figure 17: Percentage of input vectors that result in errors seen at the **S[3]** output of the 4-bit adder circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

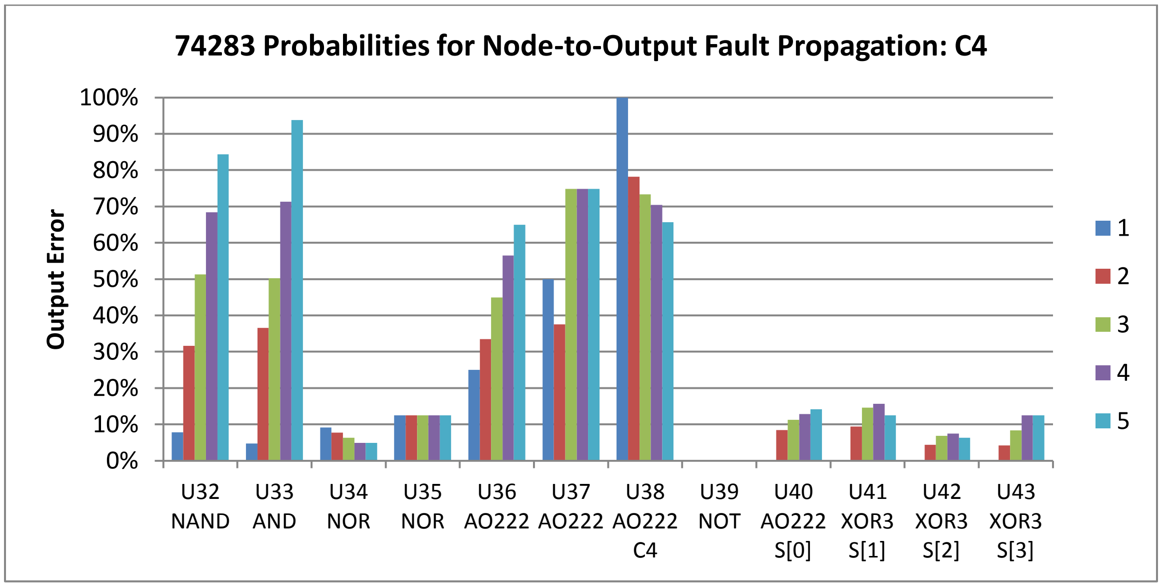

Figure 18: Percentage of input vectors that result in errors seen at the **C4** output of the 4-bit adder circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

The preceding five figures (14 through 18) demonstrate the contribution injected faults centered at each gate in the circuit have to each output. The four sum outputs of course have similar responses in general, each weighted towards gates in their own specific logic path. The carry output sees a lower number of logic gates which contribute greatly to output errors, which was reflected in the overall output error graph (Figure 4) with a lower probability of errors for **C4**.

Note how single-fault errors are typically maximized for logic gates within the logical path of a specific output, then decrease for multiple-fault errors. This indicates the effect of logical reconvergence. For output **S[2]**, a strike at the output of gates U36 or U42 will flip that value and cause that output to be incorrect for all cases. But if two or more faults are injected at this location in the circuit, neighboring upset nodes will propagate faults through the circuit, and in some cases, these multiple faults will reconverge and cancel out their effects, leading to a lower chance of error. This effect is much more pronounced when the data is separated for each circuit output as done here. Of course, in other cases, multiple faults injected into a circuit are situated such that reconvergence is unlikely, and the contribution to errors at the chosen output increases.

The next several pages contain Figures 19-24, detailing the per-gate responses for the 4-bit carry look-ahead generator for average output error and for individual output error, and Figures 25-28, which presents data organized in the same way for the 4-bit magnitude comparator. These are larger circuits analyzed with the same method, producing similar results.

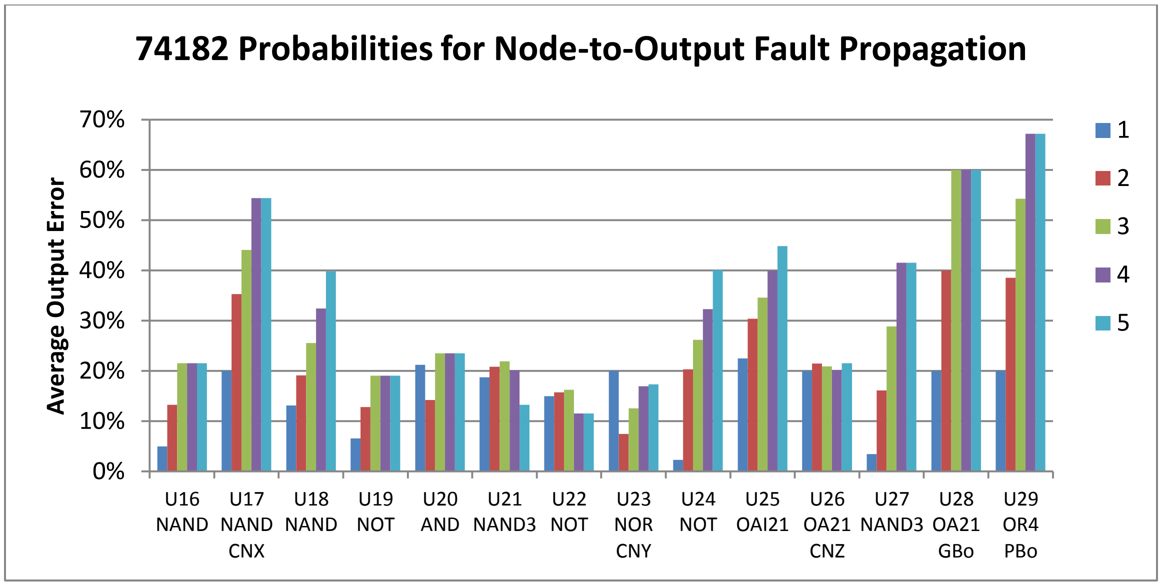

Figure 19: Percentage of input vectors that result in errors seen at any output of the 4-bit carry look-ahead generator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

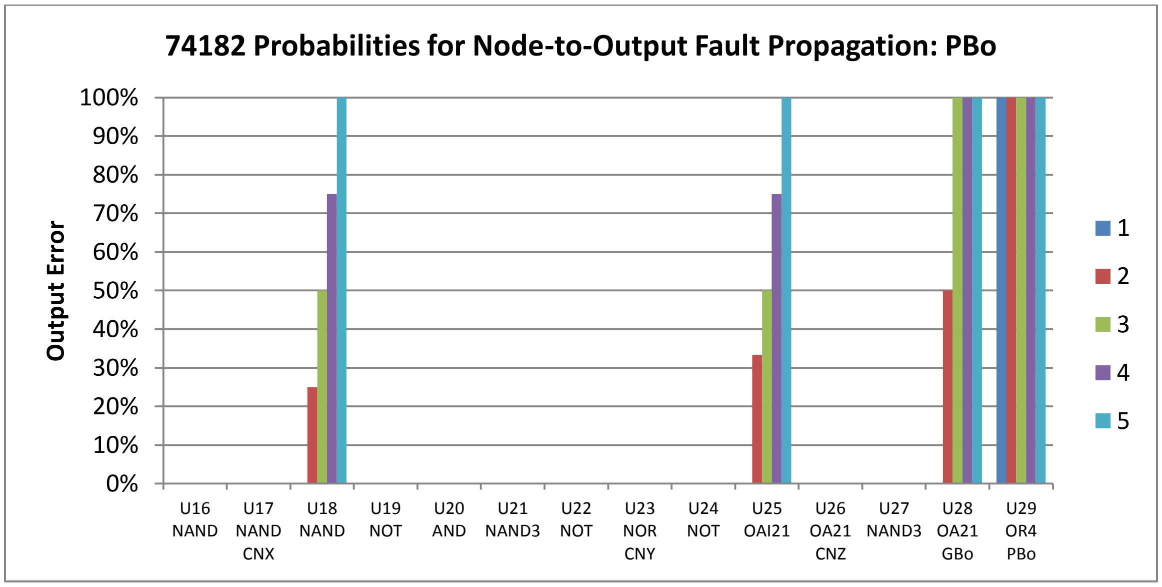

Figure 20: Percentage of input vectors that result in errors seen at the **PBo** output of the 4-bit carry look-ahead generator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

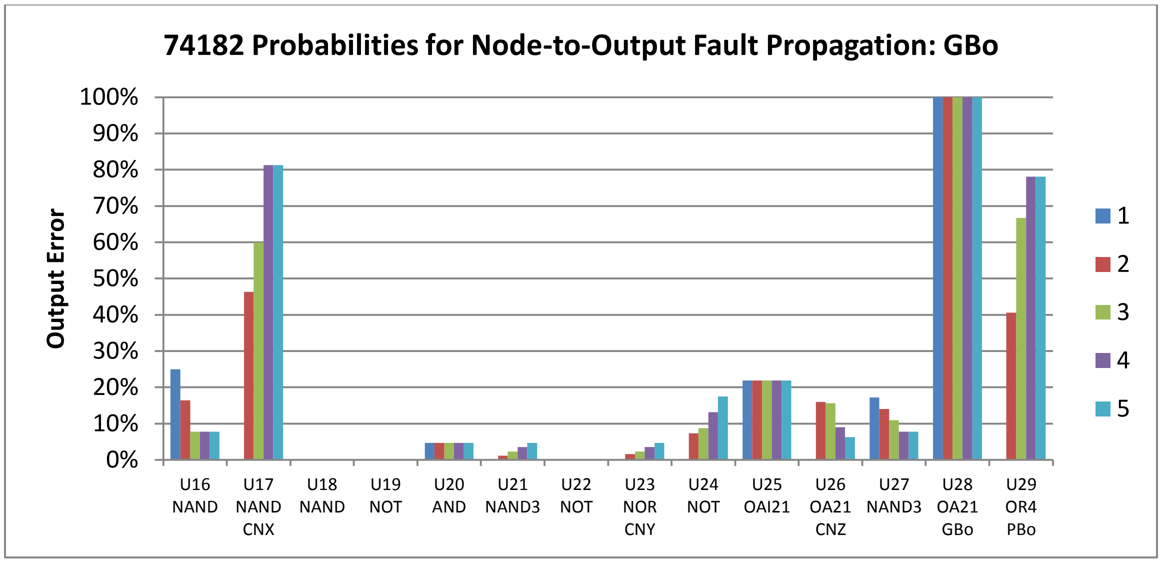

Figure 21: Percentage of input vectors that result in errors seen at the **GBo** output of the 4-bit carry look-ahead generator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

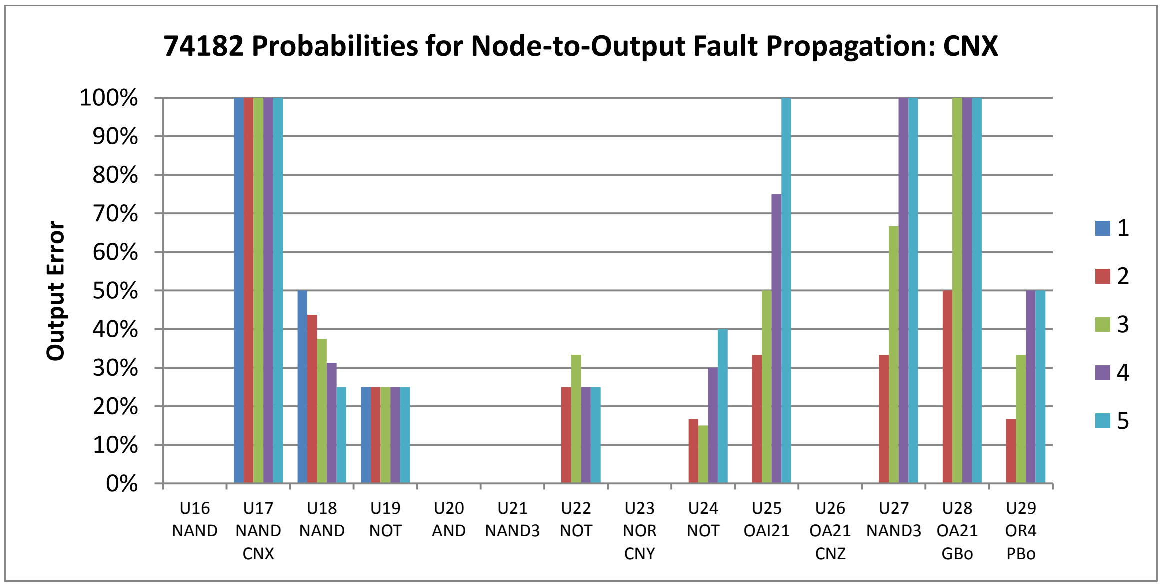

Figure 22: Percentage of input vectors that result in errors seen at the **CNX** output of the 4-bit carry look-ahead generator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

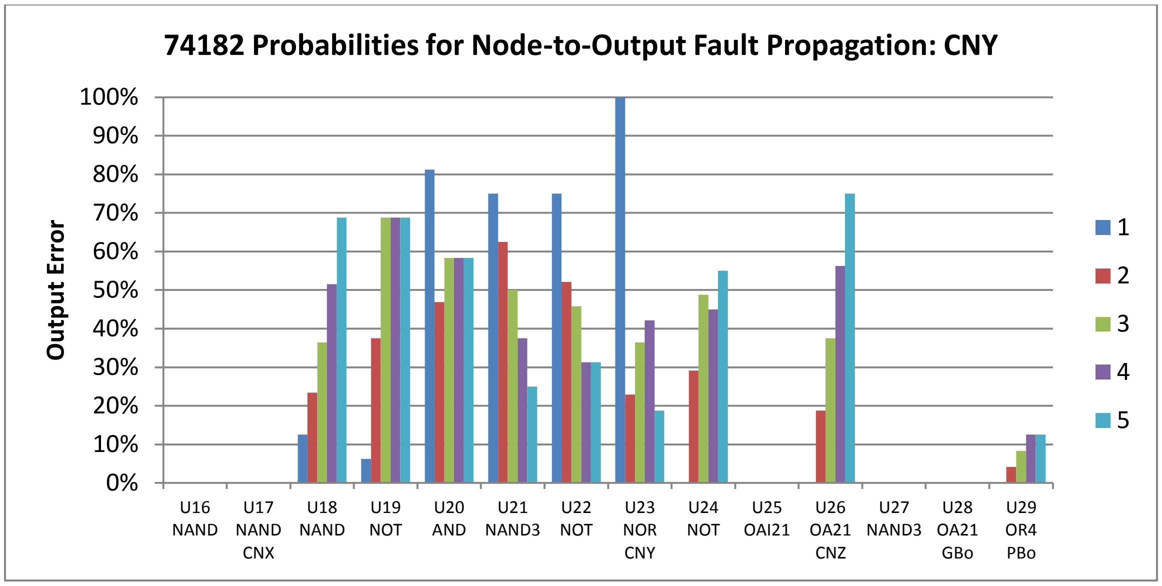

Figure 23: Percentage of input vectors that result in errors seen at the **CNY** output of the 4-bit carry look-ahead generator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

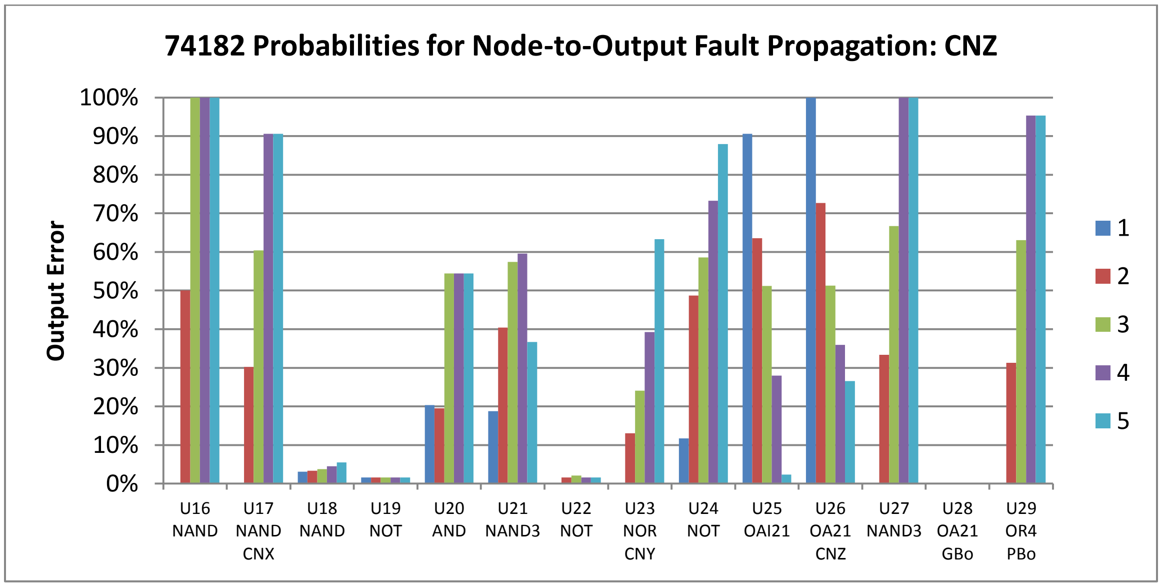

Figure 24: Percentage of input vectors that result in errors seen at the **CNZ** output of the 4-bit carry look-ahead generator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

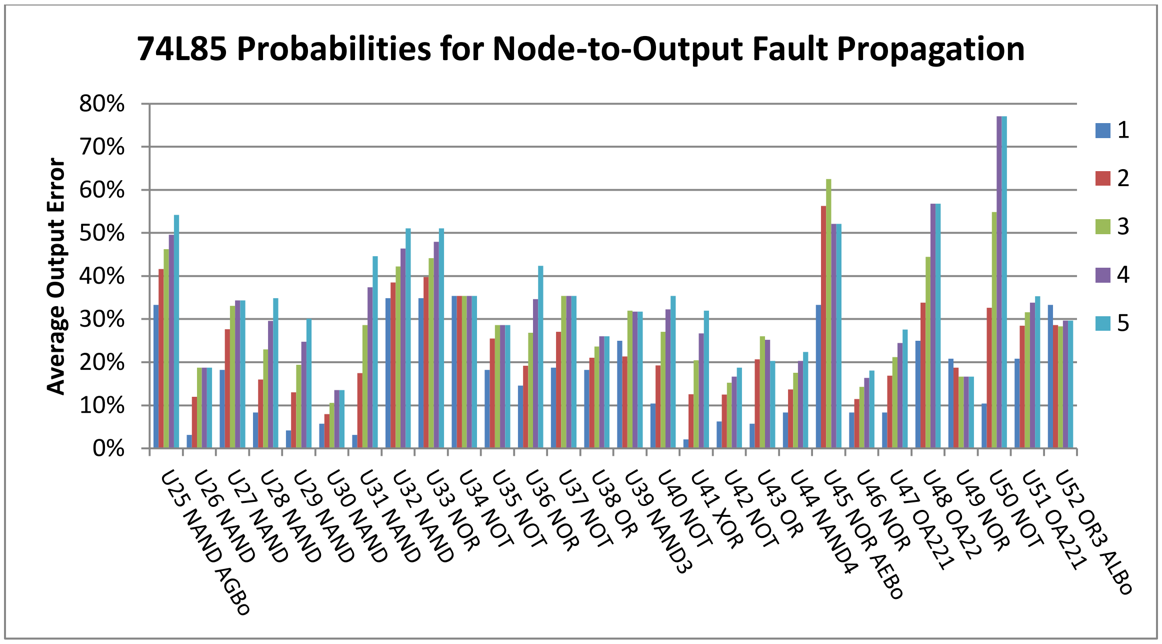

Figure 25: Percentage of input vectors that result in errors seen at any output of the 4-bit magnitude comparator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

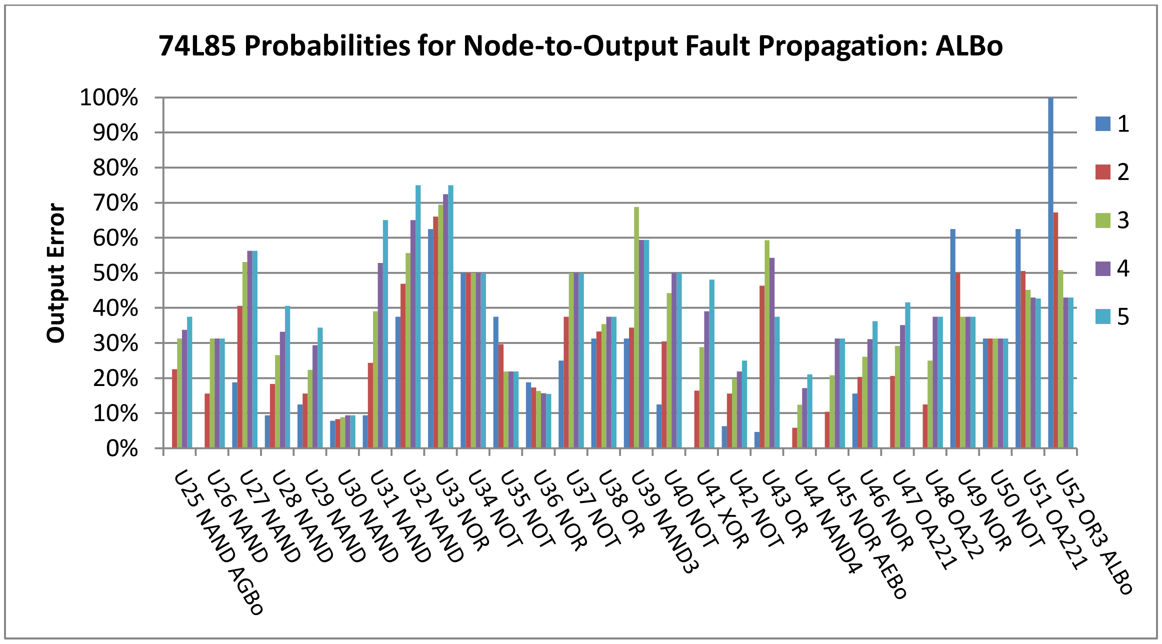

Figure 26: Percentage of input vectors that result in errors seen at the **ALBo** output of the 4-bit magnitude comparator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

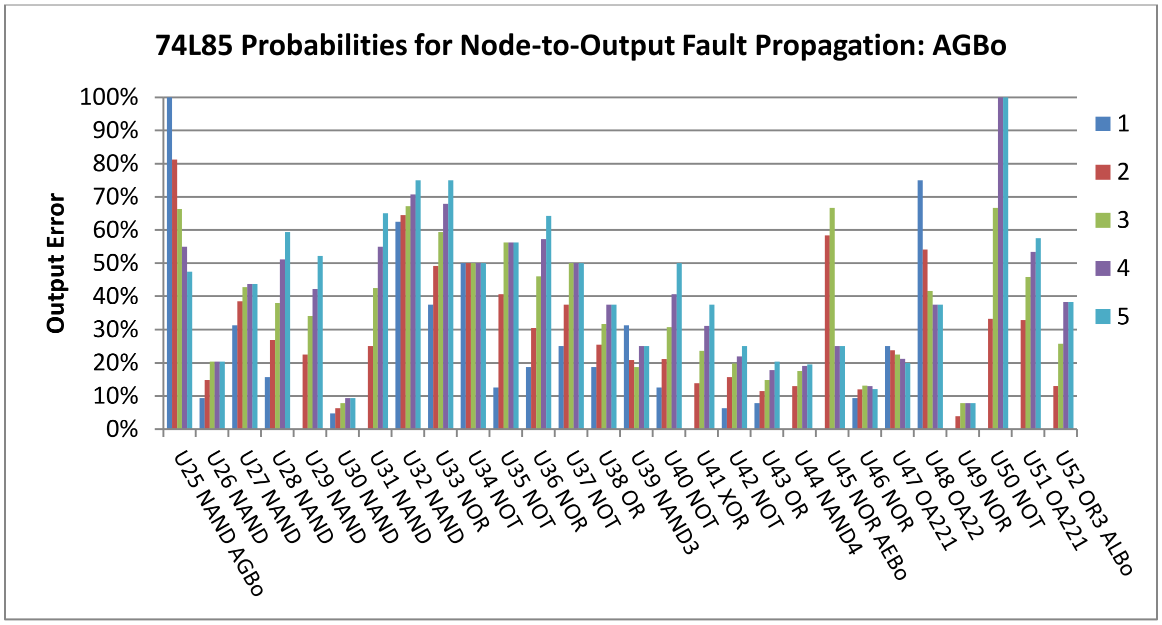

Figure 27: Percentage of input vectors that result in errors seen at the **AGBo** output of the 4-bit magnitude comparator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

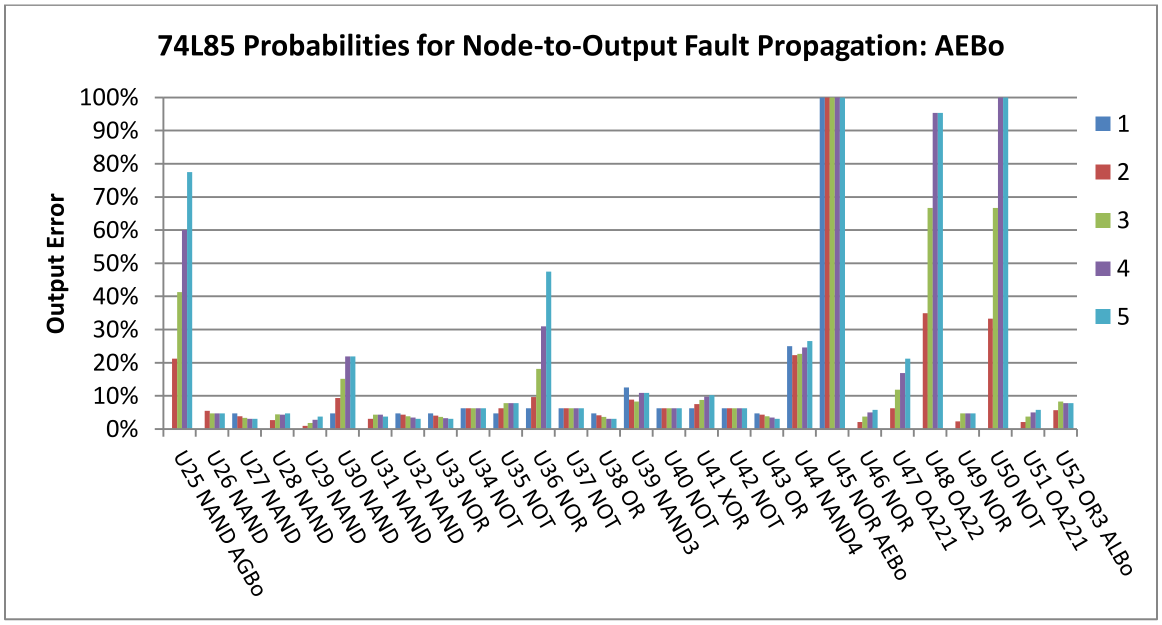

Figure 28: Percentage of input vectors that result in errors seen at the **AEBo** output of the 4-bit magnitude comparator circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

Traditional reliability analysis assumes that decreasing feature size, allowing for an increasing number of affected nodes in a local area due to a single strike, would result in an increasing number of errors. But this is not always the case – some nodes exhibit similar behavior in single and multiple faults (e.g., 74182 U20, U21, U23, U26; 74L85 U52 for average output error), or are even less volatile under multiple faults (e.g., 74182 U22, 74L85 U49 for average output error). The analysis done here to produce the graphs on the last several pages explicitly show circuit designers which gates and sections of the circuit they need to worry about most when guaranteeing the reliability of specific functions of a circuit.

Rather than hardening an entire circuit to multiple-fault effects, this analysis provides the possibility of hardening only select gates, minimizing speed and area penalties while gaining significant reliability benefits. For example, if a circuit designer wanted to harden the **AEBo** logic path for the 4-bit magnitude comparator against multiple-fault effects, Figure 28 could be generated and utilized. Selectively hardening the U25, U45, U48, and U50 gates would make the **AEBo** logic path almost invulnerable to single-event errors, at only a slight cost as compared to hardening all 28 gates in the circuit. Similar operations could be performed on the 4-bit carry look-ahead generator given Figures 20-25.

While the 4-bit carry look-ahead generator circuit is still small, the 4-bit magnitude comparator is barely tenable for observing gate-specific results. There are still only 28 nodes considered, but it is difficult to aggregate this data into a form that is useful for analysis. Regardless, it is important to look at a variety of circuits of different sizes in order to see how reliability characteristics may change according to variables such as circuit size and function. For example, here the function of these two circuits can be compared. The 4-bit magnitude comparator uses two modified 4-bit carry generators in its netlist for the **AGBo** and **ALBo** output alongside just a few more supporting logic gates for the **AEBo** output. Understandably, the **AGBo** and **ALBo** logic paths exhibit similar reliability, while the **AEBo** logic path has entirely different behavior.

The 4-bit adder circuit observed earlier (Figure 13) uses one modified 4-bit carry generator (Figure 19), but it also has a large number of other logic gates in its design. When synthesized, this produces an entirely different circuit that therefore behaves much differently when under strain of multiple-fault testing. Most of the output errors for this circuit are at approximately the 30-40% level when averaging together all output errors.

This work culminated with the analysis of a larger, general-purpose circuit, the 4-bit ALU. This circuit consists of 16 logic functions that operate on the 4-bit input numbers **A** and **B**. Given the economized design of this circuit as well as the further simplifying effects of synthesis, this circuit provides a good demonstration of logical interconnect among cells as well as many examples of separated logic paths.

The relevant data for this circuit are shown in Figures 29 and 30. As this is a significantly larger circuit than the others selected, there is a more reasonable, smoothed response in the final data sets. While data from Figure 30 were separated to show the response for each individual logic path, those figures are not included here since this analysis has already been shown for the other circuits. These figures serve the same usefulness as described earlier, for circuit designers to use in selectively hardening logic paths for individual circuit designs.

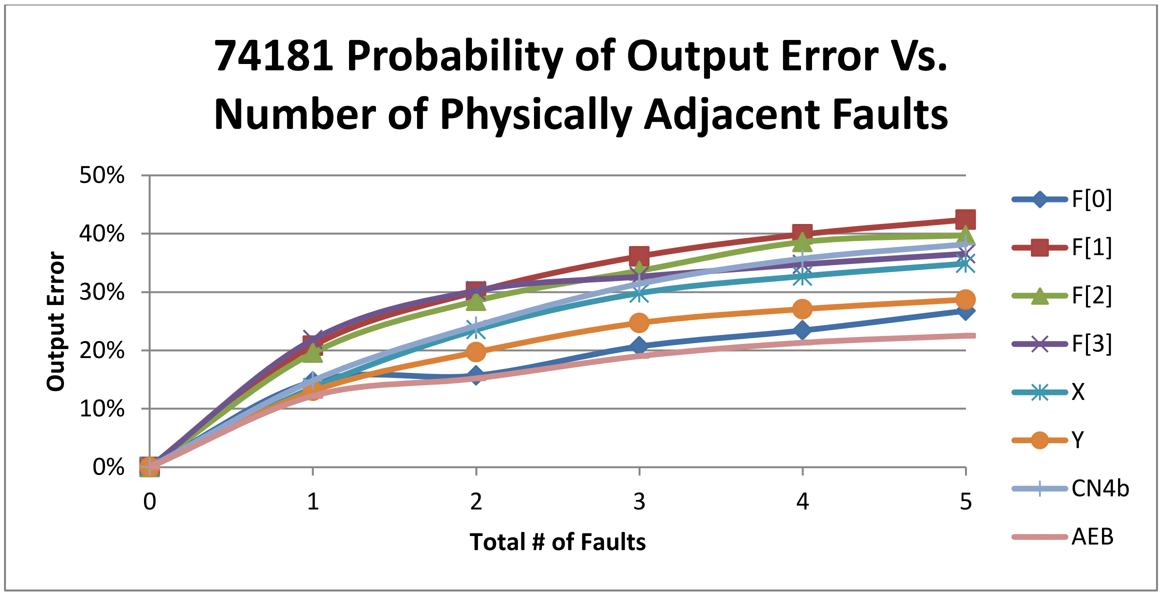

Figure 29: Percentage of input vectors that result in errors seen at an output (**F[3-0]**, **X**, **Y**, **CN4b**, and **AEB**) vs. the number of physically adjacent faults inserted into the circuit at any gate in the circuit.

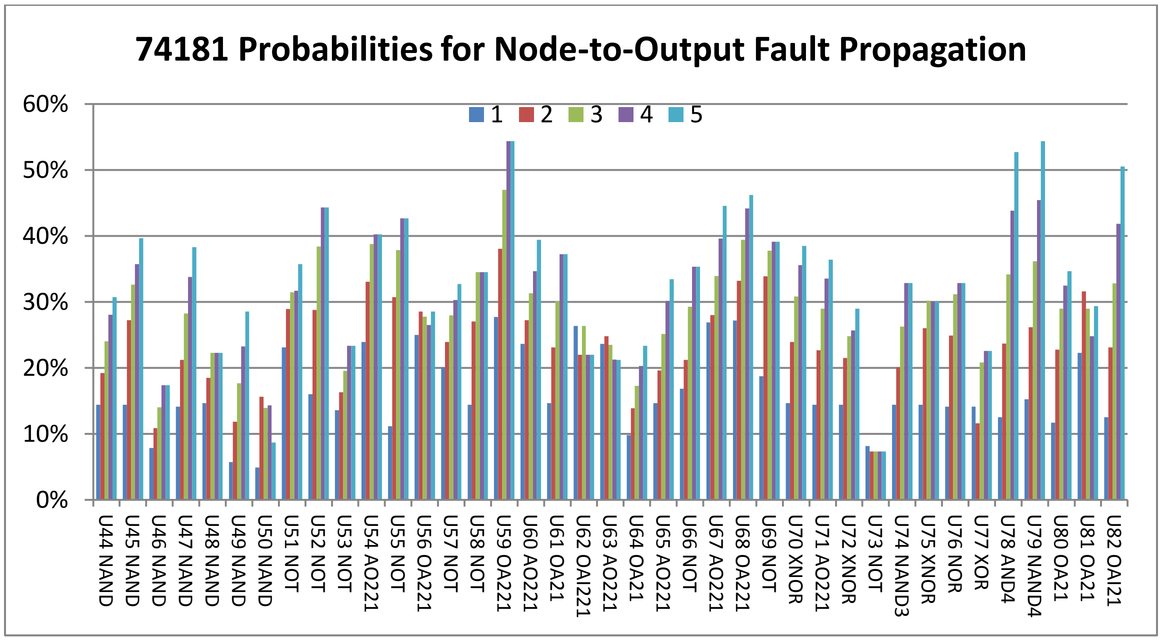

Figure 30: Percentage of input vectors that result in errors seen at any output of the 4-bit ALU circuit vs. the number of physically adjacent faults inserted into the circuit centered at each gate output.

Figure 29 shows an even response in the output error seen due to faults within the circuit. As these are averaged for a large number of nodes (39), every output continually sees an increase in error as simulations progress from single faults to 2, 3, 4, and 5 simultaneous adjacent faults. Again, these results are asymptotic; the most egregious errors are seen with single faults, while multiple faults increase the overall output error seen comparably less for each additional fault. Meanwhile, the data in Figure 30 present a more detailed look at the individual gate responses. These results continue to stress the important of analyzing each circuit on its own rather than operating simply on statements made about logic as a whole. Some gates show very low error propagation rates or rates that change very little as multiple upsets occur. Most are highly sensitive to the effects of multiple fault testing. Ideally, a tool would be created and utilized for easily examining every design sent to manufacturing, but that is beyond the scope of this work. However, there are several overall analyses that can be safely made.

Overall Analyses

This work intends to analyze several circuits and display the results, to show the general idea of what occurs under multiple upsets. But aside from this analytical aim, the possibilities of new circuit design methods become apparent. If some nodes are more resistant than others to multiple fault propagation, then this fact could theoretically be leveraged in other designs. Similar logic structures may operate similarly, giving rise to an overall circuit design that is more reliable to multiple events.

Table 2: Gate-specific data showing average output error probability change for each step in upset coverage

Gate	# Tested	Average Error Increase Per Step (%)	StDev
NAND	25	4.95	2.86
AND	3	5.10	4.47
NOT	19	4.52	4.12
OR	4	4.28	5.60
NOR	9	2.13	3.02
XOR/XNOR	8	5.20	2.22
OA/AO/OAI	24	4.85	2.99

A more specific analysis demonstrates this idea on a more fundamental level. To produce Table 2, the slope for each node in Figures 13, 19, 25, and 30 is calculated, then an average and standard deviation for each gate type present in these three circuits is calculated. More data would be necessary for higher confidence in these values, but there are some restrained suggestions to make based upon these results. For example, NOR gates seem to be more resistant to larger numbers of errors than the rest of the gates shown. However, the other gates all present averages at around the same point, and standard deviations are comparably very large. This table makes it clear that this study cannot make conclusions about reliability based purely on gates – larger logical structures such as the benchmark circuits in the preceding section of this thesis or even larger standardized circuits will have their own characteristics. The way that logic gates are used in a circuit determines their reliability behavior, rather than being inherent to the gates themselves.

Instead of looking at specific logic structures, the data can be categorized based on connection between cells. The characteristic that most seriously dictates the gravity of errors due to multiple faults is whether or not these faults originate in cells that are logically connected. If two cells are logically connected and both register a fault, then reconverging fan-out may cancel out both faults. If the two cells are in separate logic paths, then the faults will operate like separate events and cause significantly more effects in the circuit. Figure 31 demonstrates this principle by looking at all 4 circuits examined separately above.

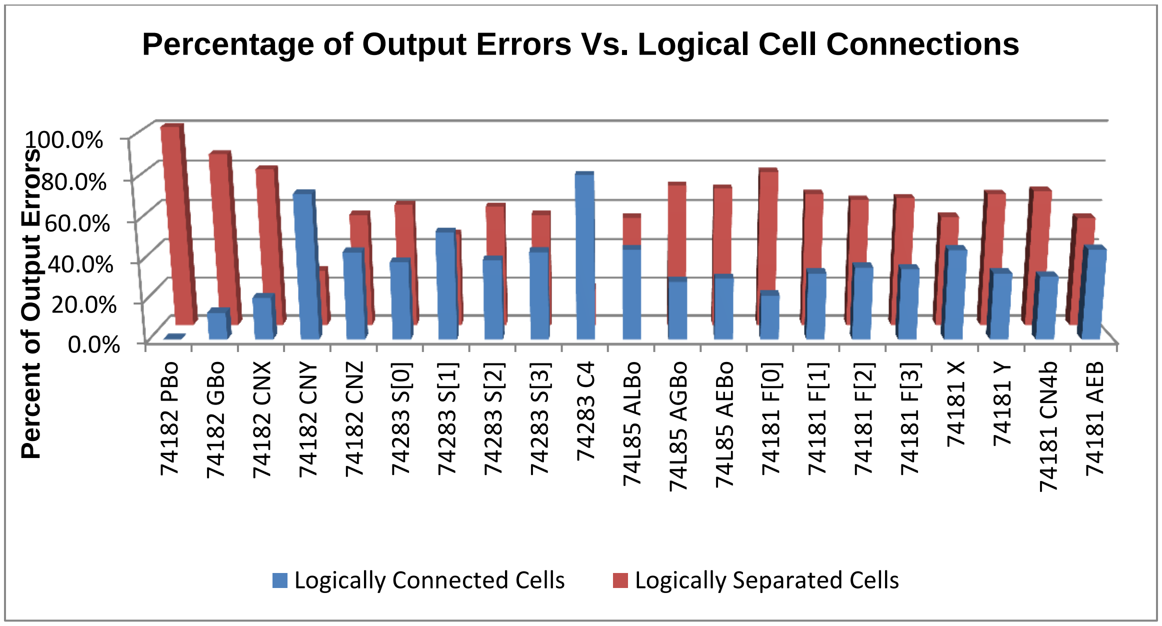

Figure 31: Aggregated analysis of logically connected and separated cells, based on data from tests simulating two simultaneous adjacent faults.

To produce this graph, the simulations run with two injected faults were reorganized based on logical connections between cells. If a simulation produced an error at an output, then the two logic gates where faults were injected were examined. If one gate propagates into the other, then the two are in the same logic path and are labeled "logically connected." If the two gates produce signals that never interact, then they are “logically separated.” Twenty-one different circuit logic paths are shown, and of these, only 3 exhibit a greater percentage of errors due to logically connected faults. In other words, 86% of the time when multiple faults occur, faults in separated logic paths contribute more to overall circuit errors than those that are logically connected. This is an important observation. If a circuit designer can guarantee that physically adjacent cells are also logically connected, then particle strikes in this region of a circuit that induce faults have the potential of canceling out each other’s effects. This will raise the level of reconvergence and therefore increase the reliability of the circuit. If this methodology is not considered and physically adjacent cells in a layout are not logically connected, then reconvergence is not possible, and output error will only increase under multiple-fault events.

CHAPTER VI

CONCLUSION

Several fundamental circuits were chosen in this work and processed with typical circuit synthesis. The layout information was used in conjunction with the circuits to perform realistic simulations of multiple transients caused by a single particle strike. The data show that, in opposition to traditional analysis, simulating single faults in a circuit is not sufficient to characterize the circuit. Typically, the probability of an error occurring increases significantly with multiple transient faults, but there are methods to mitigate or even negate this effect. Today's technology nodes require deeper analysis of these multiple errors. Specific design methods may help to absorb the impact of multiple-transient errors, or even to provide better reliability in comparison to the single fault model. Reliability-aware design will enable circuit designers to make more informed decisions and produce more dependable circuits.

It is already possible to make several conclusions about reliability under multi-fault analysis given the data from this thesis. The circuits examined herein each have their own systems of behavior, which can be used to predict their reliability contribution when referenced in larger designs overall. In general, this thesis' analysis shows that the effects of reconverging fan-out are significant under the effects of multiple upsets, and that circuits which are designed based on this principle will see better reliability behavior. Specific gates do not inherently have better characteristics than others, but certain circuit modules and design philosophies will contribute to the overall behavior of a design.

The collection of the data presented here is the formational step for analysis planned in the field of reliability. Further extensions of this work would include collecting data for larger benchmark circuits, such as those in the ISCAS85 suite. These data can be combined with what has already been collected to further support the observations reported within this thesis and offer more practical application of this analysis.

When aggregated properly, the data already collected and that from the larger circuits will be able to present more information regarding circuit design. It is anticipated that these data will allow further conclusions to be drawn about reliability based on gates as well as specific design methods. If certain settings are chosen during design synthesis (i.e., area minimization versus power minimization, manual layout placement, alternate routing selections), these characteristics could affect reliability as well.

Lastly, given that the methodology of this work is already highly systemized and follows a specific formula for every circuit, it should be possible to establish a framework for automatically analyzing circuits and producing detailed reliability information, perhaps in the form of a simple-to-use application. This aim is on the order of a full-scale application for commercial or academic uses, but is a future aspiration nonetheless.

APPENDIX A

SYNOPSYS 90-NM LIBRARY

The chosen library for this work is the Synopsys 90-nm Generic Library [30]. The choice of library is not entirely important for this thesis, except that it is compatible with the EDA tools and uses gates that are realistically sized in comparison with one another. However, the Synopsys 90-nm Generic Library still presents a full suite of information to allow its use in a variety of environments, and is particularly suited for Synopsys tools and designs optimized for low power.

Of most interest for this work is the digital standard cell library portion. This contains 340 logic cells, both combinational and sequential. A variety of gates is used in this thesis, including AND, OR, INV gates and a variety of OR-AND combinations. Higher-order designs, such as multiplexers, adders, or other complex blocks are not permitted in the Design Compiler synthesis process, since these are ultimately composed of simpler gates. When multiple upsets occur in physical operation, they will affect a small area. By inserting upsets in small gates, this work ensures that multiple upsets are reasonable to occur.

The original 74XXX code taken from the University of Michigan suite [28] uses generic language for logic gates. A selection of individual gates used from the Synopsys library is given in the following table, along with a general description of the operation of each gate:

Table 4: Synopsys 90-nm Generic Library standard gates

Synopsys Cell Name	**Purpose**
INVX0	Inverter
NAND2X0	2-input NAND
NAND3X0	3-input NAND
NAND4X0	4-input NAND
NOR2X0	2-input NOR
OR2X1	2-input OR
OR3X1	3-input OR
OR4X1	4-input OR
AND2X1	2-input AND
AND3X1	3-input AND
AND4X1	4-input AND
XOR2X1	2-input XOR
XOR3X1	3-input XOR
XNOR2X1	2-input XNOR
OA21X1	OR-AND 2/1
OA22X1	OR-AND 2/2
OA221X1	OR-AND 2/2/1
OAI21X1	OR-AND-Invert 2/1
OAI221X1	OR-AND-Invert 2/2/1
AO22X1	AND-OR 2/1
AO221X1	AND-OR 2/2/1
AO222X1	AND-OR 2/2/2

APPENDIX B

XOR FAULT INJECTION VERILOG MODEL

In order to inject faults into a circuit in ModelSim, the Verilog code has to be modified to include additional input lines to trigger specific nodes. Therefore, each circuit should be modified so that a fault can be artificially inserted at the output of each gate. An inserted fault could take the form of a flipped value, stuck-low, or stuck-high value; the primary simulations in this thesis use the flipped model - this would be the result of a physical occurrence called a “glitch.” The original 74XXX code is taken from the University of Michigan suite and synthesized through Synopsys Design Compiler. To prepare for fault insertion, each circuit must be modified from the resultant gate-level netlist in Verilog to insert a 2-input XOR gate at each internal node, as well as just before each output. Faults are not allowed to be inserted at input nodes in this testing, as each circuit is considered unto itself rather than as part of a larger overall design. The Verilog files are taken from the “mapped” directory (Design Compiler output) and the following changes are made:

1) Replace each internal signal *n#* with *n#i* and *n#o* and add a signal for each output
2) Add an input X# for each original internal node and each output
3) Update the inputs and outputs for gates in the code to reflect the new internal signals
4) Add XOR gates to the end of the code, tying together the new inputs and new internal signals
5) Add more XOR gates which connect the new output internal nodes with the outputs

With the size of the 74XXX benchmark circuits, these changes are made relatively easily by hand. If work were to proceed to the ISCAS85 circuits, Python will need to be used to ensure that errors in translating the circuit do not occur. An easy way to check for errors, regardless of code transformation method, is to run the regular code (with no XOR gates) and the transformed code both in ModelSim, and compare the results when all fault injection inputs are set to 0. This step was taken for each circuit to ensure that no errors had been made.

As an example of this code transformation, a small circuit, the 4-bit adder, is shown below. First, the “regular” output of Design Compiler:

```
module Circuit74283 ( C0, A, B, S, C4 );
  input [3:0] A;
  input [3:0] B;
  output [3:0] S;
  input C0;
  output C4;
  wire   \Ckt74283/C[0] , n27, n28, n29, n30, n31, n32, n33;
  assign \Ckt74283/C[0]  = C0;

  NOR2X0 U32 ( .IN1(A[0]), .IN2(B[0]), .QN(n29) );
  AND2X1 U33 ( .IN1(A[0]), .IN2(B[0]), .Q(n30) );
  NOR2X0 U34 ( .IN1(\Ckt74283/C[0] ), .IN2(n30), .QN(n27) );
  NOR2X0 U35 ( .IN1(n29), .IN2(n27), .QN(n31) );
  AO222X1 U36 ( .IN1(A[1]), .IN2(B[1]), .IN3(A[1]), .IN4(n31),
.IN5(B[1]),
        .IN6(n31), .Q(n32) );
  AO222X1 U37 ( .IN1(A[2]), .IN2(B[2]), .IN3(A[2]), .IN4(n32),
.IN5(B[2]),
        .IN6(n32), .Q(n33) );
  AO222X1 U38 ( .IN1(B[3]), .IN2(A[3]), .IN3(B[3]), .IN4(n33),
.IN5(A[3]),
        .IN6(n33), .Q(C4) );
  INVX0 U39 ( .IN(n29), .QN(n28) );
  AO222X1 U40 ( .IN1(\Ckt74283/C[0] ), .IN2(n30),
.IN3(\Ckt74283/C[0] ), .IN4(
        n29), .IN5(n28), .IN6(n27), .Q(S[0]) );
  XOR3X1 U41 ( .IN1(A[1]), .IN2(B[1]), .IN3(n31), .Q(S[1]) );
  XOR3X1 U42 ( .IN1(B[2]), .IN2(A[2]), .IN3(n32), .Q(S[2]) );
  XOR3X1 U43 ( .IN1(B[3]), .IN2(A[3]), .IN3(n33), .Q(S[3]) );
endmodule
```

Then, the code after it has been modified to allow fault injection:

```
module Circuit74283x ( C0, X, A, B, S, C4 );
  input [3:0] A;
  input [3:0] B;
  input [11:0] X;
  output [3:0] S;
  input C0;
  output C4;
  wire   \Ckt74283x/C[0] , n27i, n27o, n28i, n28o, n29i, n29o,
n30i, n30o, n31i, n31o, n32i, n32o, n33i, n33o, s0i, s1i, s2i,
s3i, c4i;
  assign \Ckt74283x/C[0]  = C0;

  NOR2X0 U32 ( .IN1(A[0]), .IN2(B[0]), .QN(n29i) );
  AND2X1 U33 ( .IN1(A[0]), .IN2(B[0]), .Q(n30i) );
  NOR2X0 U34 ( .IN1(\Ckt74283x/C[0] ), .IN2(n30o), .QN(n27i) );
  NOR2X0 U35 ( .IN1(n29o), .IN2(n27o), .QN(n31i) );
  AO222X1 U36 ( .IN1(A[1]), .IN2(B[1]), .IN3(A[1]), .IN4(n31o),
.IN5(B[1]),
        .IN6(n31o), .Q(n32i) );
  AO222X1 U37 ( .IN1(A[2]), .IN2(B[2]), .IN3(A[2]), .IN4(n32o),
.IN5(B[2]),
        .IN6(n32o), .Q(n33i) );
  AO222X1 U38 ( .IN1(B[3]), .IN2(A[3]), .IN3(B[3]), .IN4(n33o),
.IN5(A[3]),
        .IN6(n33o), .Q(c4i) );
  INVX0 U39 ( .IN(n29o), .QN(n28i) );
  AO222X1 U40 ( .IN1(\Ckt74283x/C[0] ), .IN2(n30o),
.IN3(\Ckt74283x/C[0] ), .IN4(
        n29o), .IN5(n28o), .IN6(n27o), .Q(s0i) );
  XOR3X1 U41 ( .IN1(A[1]), .IN2(B[1]), .IN3(n31o), .Q(s1i) );
  XOR3X1 U42 ( .IN1(B[2]), .IN2(A[2]), .IN3(n32o), .Q(s2i) );
  XOR3X1 U43 ( .IN1(B[3]), .IN2(A[3]), .IN3(n33o), .Q(s3i) );
  XOR2X1 U44 ( .IN1(n27i), .IN2(X[0]), .Q(n27o) );
  XOR2X1 U45 ( .IN1(n28i), .IN2(X[1]), .Q(n28o) );
  XOR2X1 U46 ( .IN1(n29i), .IN2(X[2]), .Q(n29o) );
  XOR2X1 U47 ( .IN1(n30i), .IN2(X[3]), .Q(n30o) );
  XOR2X1 U48 ( .IN1(n31i), .IN2(X[4]), .Q(n31o) );
  XOR2X1 U49 ( .IN1(n32i), .IN2(X[5]), .Q(n32o) );
  XOR2X1 U50 ( .IN1(n33i), .IN2(X[6]), .Q(n33o) );
  XOR2X1 U51 ( .IN1(s0i), .IN2(X[7]), .Q(S[0]) );
  XOR2X1 U52 ( .IN1(s1i), .IN2(X[8]), .Q(S[1]) );
  XOR2X1 U53 ( .IN1(s2i), .IN2(X[9]), .Q(S[2]) );
  XOR2X1 U54 ( .IN1(s3i), .IN2(X[10]), .Q(S[3]) );
  XOR2X1 U55 ( .IN1(c4i), .IN2(X[11]), .Q(C4) );
endmodule
```

REFERENCES

[1] J. F. Ziegler, H. W. Curtis, F. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. Freeman, P. Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O'Gorman, B. Messina, T. D. Sullivan, A. J. Sykes, H. Yourke, T. A. Enger, V. Tolat, T. S. Scott, A. H. Taber, R. J. Sussman, W. A. Klein, and C. W. Wahaus, "IBM experiments in soft fails in computer electronics (1978-1994)," *IBM Journal of Research and Development*, vol. 40, pp. 3-18, 1996.

[2] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Transactions on Nuclear Science,* vol. 50, pp. 583-602, 2003.

[3] N. N. Mahatme, S. Jagannathan, T. D. Loveless, L. W. Massengill, B. L. Bhuva, S. J. Wen, and R. Wong, "Comparison of combinational and sequential error rates for a deep submicron process," *IEEE Transactions on Nuclear Science*, vol. 58, pp. 2719-2725, 2011.

[4] G. De Micheli, *Synthesis and optimization of digital circuits*. New York: McGraw-Hill, 1994.

[5] O. A. Amusan, A. F. Witulski, L. W. Massengill, B. L. Bhuva, P. R. Fleming, M. L. Alles, A. L. Sternberg, J. D. Black, and R. D. Schrimpf, "Charge collection and charge sharing in a 130 nm CMOS technology," *IEEE Transactions on Nuclear Science*, vol. 53, pp. 3253-3258, 2006.

[6] J. D. Black, D. R. Ball II, W. H. Robinson, D. M. Fleetwood, R. D. Schrimpf, R. A. Reed, D. A. Black, K. M. Warren, A. D. Tipton, P. E. Dodd, N. F. Haddad, M. A. Xapsos, H. S. Kim, and M. Friendlich, "Characterizing SRAM single event upset in terms of single and multiple node charge collection," *IEEE Transactions on Nuclear Science*, vol. 55, pp. 2943-2947, 2008.

[7] R. Harada, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "Neutron induced single event multiple transients with voltage scaling and body biasing," *2011 International Reliability Physics Symposium*, pp. 3C.4.1-3C.4.5, April 2011.

[8] R. Velazco, S. Rezgui, and R. Ecoffet, "Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulating Upsets) injection," *IEEE Transactions on Nuclear Science,* vol. 47, pp. 2405-11, 2000.

[9] G. P. Saggese, N. J. Wang, Z. T. Kalbarczyk, S. J. Patel, and R. K. Iyer, "An experimental study of soft errors in microprocessors," *IEEE Micro,* vol. 25, pp. 30-39, 2005.

[10] C. T. Toomey, B. D. Sierawski, A. Sternberg, D. B. Limbrick, B. L. Bhuva, L. W. Massengill, W. H. Robinson, S.-J. Wen, R. Wong, and S. Martin, "Statistical fault

injection and analysis at the register transfer level using the Verilog Procedural Interface," in *36th Annual Government Microcircuit Applications and Critical Technology Conference (GOMACTech 2011)*, Orlando, FL, 2011.

[11] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, pp. 305-316, 2005.

[12] R. K. Iyer, N. M. Nakka, Z. T. Kalbarczyk, S. Mitra, "Recent advances and new avenues in hardware-level reliability support," *IEEE Micro*, vol.25, no.6, pp. 18- 29, Nov.-Dec. 2005.

[13] V. Srinivasan, A. L. Sternberg, A. R. Duncan, W. H. Robinson, B. L. Bhuva, and L. W. Massengill, "Single-event mitigation in combinational logic using targeted data path hardening," *IEEE Transactions on Nuclear Science*, vol. 52, pp. 2516-2523, 2005.

[14] D. B. Limbrick, D. A. Black, K. Dick, N. M. Atkinson, N. J. Gaspard, J. D. Black, W. H. Robinson, A. F. Witulski, "Impact of logic synthesis on soft error vulnerability using a 90-nm bulk CMOS digital cell library," *2011 Proceedings of IEEE Southeastcon*, pp.430-434, 17-20 March 2011.

[15] D. G. Mavis, P. H. Eaton, "Soft error rate mitigation techniques for modern microcircuits," *40th Annual Reliability Physics Symposium Proceedings*, pp. 216-225, 2002.

[16] H. S. Deogun, D. Sylvester, D. Blaauw, "Gate-level mitigation techniques for neutron-induced soft error rate," *6th International Symposium on Quality of Electronic Design*, pp. 175- 180, 21-23 March 2005.

[17] R. Baumann, “The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction,” in *IEDM Technical Digest*, pp. 329–332, 2002.

[18] S. Buchner, M. Baze, D. Brown, D. McMorrow, and J. Melinger., “Comparison of error rates in combinational and sequential logic.” *IEEE Transactions on Nuclear Science*, 44(6):2209–2216, December 1997.

[19] N. Miskov-Zivanov, D. Marculescu, "MARS-S: Modeling and Reduction of Soft Errors in Sequential Circuits," *8th International Symposium on Quality Electronic Design*, pp.893-898, 26-28 March 2007.

[20] G. Asadi and M. B. Tahoori, “An analytical approach for soft error rate estimation of SRAM-based FPGAs”, presented at the *2004 Military and Aerospace Programmable Logic Devices Conference (MAPLD)*, 2004.

[21] Z. Wo and I. Koren, "Technology mapping for reliability enhancement in logic synthesis," *6th International Symposium on Quality of Electronic Design*, pp. 137-142, 21-23 March 2005.

[22] N. Miskov-Zivanov and D. Marculescu, “Multiple transient faults in combinational and sequential circuits: a systematic approach,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 1614-1627, 2010.

[23] M. C. Casey, A. R. Duncan, B. L. Bhuva, W. H. Robinson, and L. W. Massengill, “Simulation study on the effect of multiple node charge collection on error cross-section in CMOS sequential logic,” *IEEE Transactions on Nuclear Science*, vol. 55, pp. 3136–3140, 2008.

[24] N. Miskov-Zivanov and D. Marculescu, “A systematic approach to modeling and analysis of transient faults in logic circuits,” in *10th International Symposium on Quality Electronic Design*, 2009, pp. 408-413.

[25] "IEEE Standard VHDL Language Reference Manual," IEEE Standard 1076-2008 (Revision of IEEE Standard 1076-2002), pp. c1-626, Jan. 26 2009.

[26] "IEEE Standard for Verilog Hardware Description Language," IEEE Standard 1364-2005 (Revision of IEEE Standard 1364-2001), pp. 0_1-560, 2006.

[27] D. B. Limbrick, S. Yue, W. H. Robinson, and B. L. Bhuva, "Impact of synthesis constraints on error propagation probability of digital circuits," in 2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Vancouver, Canada, 2011, pp. 103-111.

[28] ISCAS85 High Level Models. Available: http://www.eecs.umich.edu/~jhayes/iscas/benchmark.html

[29] Synopsys. Synopsys TetraMAX ATPG. Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/TetraMAXATPG.aspx

[30] Synopsys. Synopsys University Program. Available: http://www.synopsys.com/Community/UniversityProgram/Pages/default.aspx

[31] K. Dick, “Fault de-interleaving for reliability in high-speed circuits,” in Electrical Engineering, Master of Science, Nashville, TN: Vanderbilt University, 2010.

[32] ModelSim - Advanced Simulation and Debugging. Available: http://model.com/

[33] J. L. A. Hughes and E. J. McCluskey, "An analysis of the multiple fault detection capabilities of single stuck-at fault test sets," presented at the 1984 International Test Conference on the Three Faces of Test: Design, Characterization, Production, Philadelphia, PA, 1984.