BROWN UNIVERSITY

Modeling Optimal Schmitt Trigger Based Implication Chains for Noise Immune Sub-threshold Circuits

Lauren R. Moser
Author

Professor Ruth Iris Bahar
Advisor

Professor William R. Patterson
Reader

4/26/2013

Submitted in partial fulfillment of the requirements of the degree of Bachelor of Science with Honors in the Division of Engineering at Brown University.

Table of Contents

Abstract

In CMOS technology, scaling contributes to decreases in node capacitances, decreases in supply voltage, and increases in clock frequencies. While the aforementioned features present advantages such as faster results, increased integration, greater density, and reduced power, they also present new opportunities for soft errors to arise. In addition, when a circuit's supply voltage remains below its threshold voltage, internal signals become more susceptible to signal disturbances caused by both thermal noise and random telegraph noise (RTS). Error correction techniques must be low power and ideally small in size overhead if they are to be installed on ultra low power (ULP) circuits. In this paper we investigate the use of implications, or invariant relationships, to correct signal errors induced by noise. We introduce a method called Fault Observability, and combine it with past error correction methods involving implications to correct soft errors in a space-efficient manner. Our results show significant noise suppression in a 38-gate 22[nm] FDSOI test circuit.

1. Introduction

For each new CMOS technology that is developed, scaling contributes to about a 30% decrease in node capacitances, a 30% decrease in supply voltage, and a 100% increase in clock frequencies [1]. While the aforementioned features present benefits such as faster results, increased integration, greater density, and reduced power, they also present new opportunities for soft errors to arise. And although all these benefits may be possible, they are not necessarily realized, in part because of reliability concerns.

Scaled circuits become particularly susceptible to soft errors when they operate in the sub-threshold region, or are classifiable as "ultra-low power" (ULP) technologies. When a circuit's supply voltage remains below its threshold voltage, internal signals become more susceptible to signal disturbances caused by both thermal noise and random telegraph noise (RTS). Error correction techniques must be low power and ideally small in size if they are to be installed on ULP circuits.

A method called "implication chain building" was developed by Marco Donato et al. specifically for correcting online errors in noisy ULP systems [2][3]. The technique relied on uncovering invariant relationships between nodes in a circuit, and using said relationships to correct errors. Say, for instance, the logic of a circuit dictated that a logical zero at a node "A" always implies that node "B" would have a logical value of 1 (A(0)→B(1)). Assuming the signal at "A," the implicant, remains at a steady state of 0, we can reinforce node "B," the implicand with the correct value of 1. "Implication chain building" is a process that ensures that each

primary output of a circuit is reinforced by a primary input. Between the input and output, signals at intermediary nodes are reinforced using a series of implications.

The focus of this thesis is twofold. Firstly, we focus on optimizing a preexisting algorithm for implication chain building, paying special attention to the minimization of area overhead and power dissipation. The method had previously only been used on single-output circuits, and we seek to optimize the method for circuits with multiple outputs, and by proxy, multiple chains. Secondly, we propose a novel method of reinforcing primary outputs using unchained implications. We propose scattering implications throughout the circuit to reinforce nodes that have a high probability of generating faults (or signal disturbances) that would propagate to the output of a circuit. Implicands are to be chosen for their high fault rate, and implciants are to be chosen because of a high probability that they remain at the appropriate logic value required for the implication. Thus, we strive to find steady implicants to reinforce nodes that are known to propagate faults to a circuit output. Ultimately we attempt to combine implication chain building and fault observability to find the most effective error correction technique.

The remainder of the thesis is as follows. Section 1 presents background information regarding previous online error detection strategies. Section 2 presents the physical implementation of an implication. Section 3 introduces implication chain building methods. Section 4 discusses results of implication chain optimization testing. Section 5 introduces fault correction methods. Section 6 analyzes results of fault correction testing. Finally, section 7 will present conclusions and future work.

2. Background

2.1 Invariant Relationships

Invariant relationships, or implications, are logical certainties between two nodes in a circuit. A notation for implications can be expressed as:

$$A(0|1) \rightarrow B(0|1)$$

In the above notation, node A is the imlicant, and B is the implicand. In a practical sense, what we define as a node is a wire in a circuit. An excerpt of the rd53 circuit can be seen in figure 1 below. A select few of its "useful" implications can be seen in table 1, below.

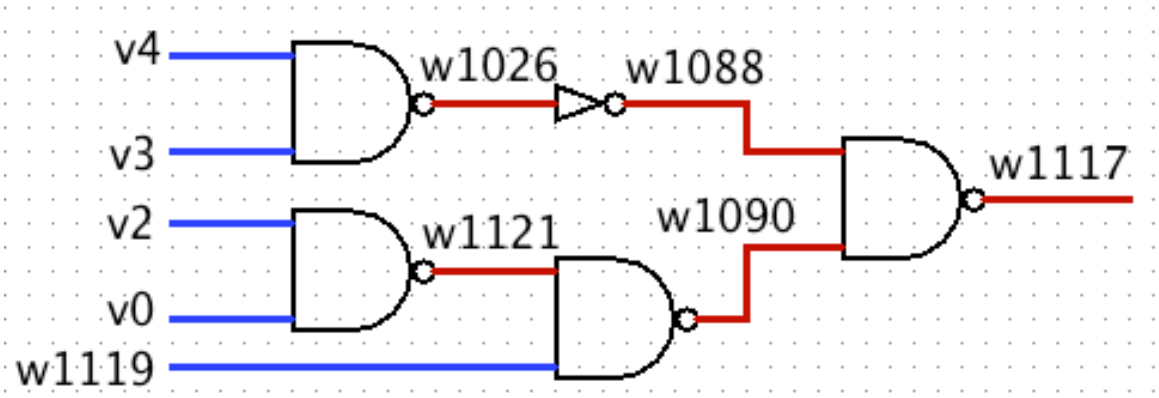

Figure 1: A Sub-circuit of the RD53 Circuit

Table 1: Implications for Rd53 Subcircuit

w1090 (0) → w1121 (1)	v3 (0) → w1117(1)
w1121 (0) → w1090(1)	w1117 (0) → w1026(0)
w1090(1)→w1083(1)	w1026(1)→w1117(1)
w1090 (0) → w1119(1)	w1117 (0) → w1088(1)
w1119 (0) → w1090(1)	w1088 (0) → w1117(1)
w1117 (0) → v4(1)	w1117 (0) → w1090(1)
v4 (0) → w1117(1)	w1090 (0) → w1117(1)
w1117 (0) → v3(1)	w1026 (0) → v4(1)

The implications in table 1 are known as "useful" implications because the implicant exists in the fan-in cone of the implicand. Validating implications has been used previously in implication detection [3]. For instance, in the implication bolded above, v4, a primary input, reinforces w1117. A non-useful implication, thus, is an implication where the implicant does not lie within the fan-in cone of the implicand.

In addition, each implicant-implicand pair has two valid implications for it. For instance, w1088 and v4 share:

$$w1088(1) \rightarrow v4(1)$$

$$v4(0) \rightarrow w1088(0)$$

And the pair including w1117 and w1088 share:

$$w1117(0) \rightarrow w1088(1)$$

$$w1088(0) \rightarrow w1117(1)$$

It makes sense that there are two possibilities for each valid implication, because according to Boolean logic, if w1117(0)→w1088(1) is true, the converse w1088(0)→w1117(1) must also be true. However, only implications where the implicant is in the fan-in cone of the implicand may be used. If we considered the opposite scenario in the error correction process, a feedback loop could be created and errors could be reinforced rather than forced to a correct value.

2.2 Schmitt Triggers

Once implications have been identified and chosen, it becomes necessary to ensure that the invariant relationship is reinforced in the circuit through feed-forward reinforcement. If the relationship outlined by an implication is violated, it

must be changed back. To this end, the Schmitt trigger is an effective tool, as it has been used in various contexts to extract signals from noisy conditions.

Gates constructed from Schmitt triggers have higher noise margins than those constructed from regular CMOS technology, and can be used effectively in sub-threshold circuits [4]. Figure 2, below, shows an inverter composed of Schmitt triggers.

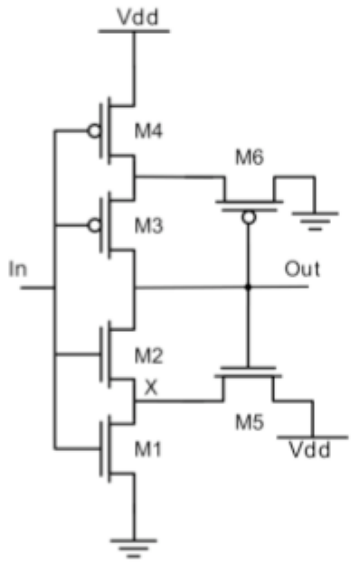

Figure 2: A Schmitt Trigger Inverter taken from [2]

It is important to note the role of transistors M5 and M6 in this design—they provide feedback from the output to adjust the switching threshold of the gate, thus providing the gate with a higher noise margin.

NAND and NOR gates can also be successfully built using Schmitt triggers [5], but the transistor count in traditionally built Schmitt gates is quite high. A regular inverter has a transistor count of two, and figure 2 displays six. Remembering that area overhead and power consumption are at stake, so we modify the NAND and

NOR gates in a more space efficient way. A Schmitt trigger NAND and a modified Schmitt trigger NAND are visible in figure 3.

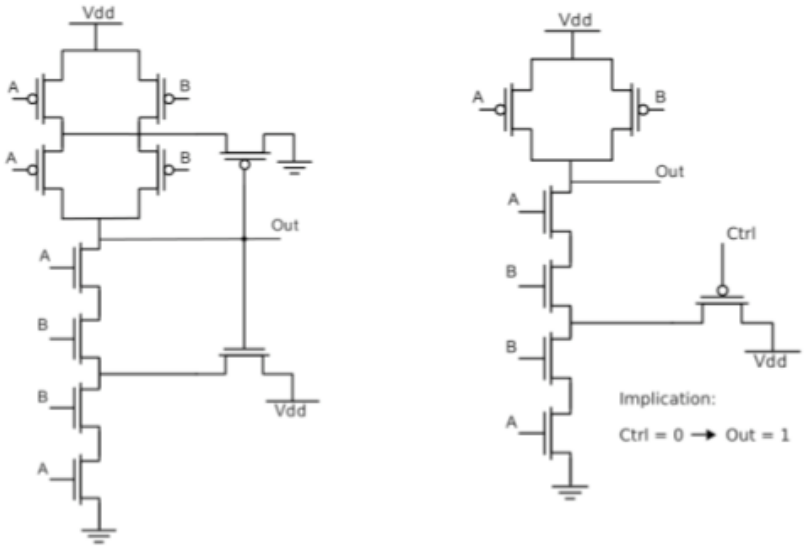

Figure 3: A Schmitt Trigger NAND Gate (Left) and a Modified Schmitt Trigger NAND Gate (Right) taken from [2]

The original Schmitt trigger constructed NAND is visible to the left, and a single permutation of a modified pull-down Schmitt trigger NAND is visible to the right. The pull-down version breaks the feedback loop originally attached to the output and creates a new input, Ctrl. Because of Ctrl's positioning at the output, it serves as an implicant. In the version above, the implication is Ctrl(0)→ out(1). Note that there is a different permutation of this gate for each type of implication. Said permutations can be seen in table 2, below:

Table 2: Ctrl Transistors and their Implications in Modified NAND Gates

PMOS on pull-up network	Ctrl(0)→Out(0)
NMOS on pull-up network	Ctrl(1)→Out(0)
PMOS on pull-down network	Ctrl(0)→Out(1)
NMOS on pull-down network	Ctrl(1)→Out(1)

2.3. Finding Implications

In this section, we present the workflow that can automatically be used to identify and validate implications in a circuit, given a verilog netlist of said circuit. It was originally developed for use in the implication chain building algorithm, but it is also used in the fault observability algorithm [3]. The workflow can be seen below in figure 4.

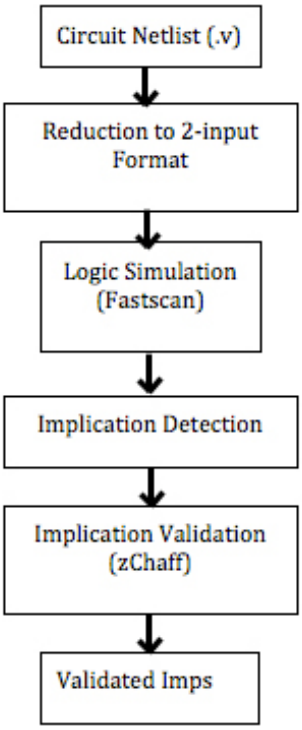

Figure 4: Implication Detection Workflow taken from [3]

We begin with a verilog netlist. We first reduce said netlist to a 2-input format so that Boolean logic can be more easily parsed into implications. The resulting netlist is then run through Fastscan (2009.1.10), a Mentor Graphics tool that specializes in generating test patterns, among other things.

We invoke a process called Automatic Test Pattern Generation (ATPG) to generate a series of input and output patterns for the circuit, and then perform a functional simulation on those patterns. During the functional simulation, we record each logic value at every node in the circuit. After these values are recorded, we run a custom application that checks whether an implication exists at each node. This application iterates through the ATPG runs and checks whether a pair of logic conditions is always satisfied. Thus if the script determines that a logical relationship is true for every tested input vector, the logical relationship is a potential implication.

However, the ATPG function of Fastscan does not simulate over every possible input pattern, so at this point, implications found using ATPG simulations may not be true for the circuit as a whole. Thus, in order to verify that an implication is true for all input patterns, we use the zChaff SAT solver to attempt to invalidate the implication with previously untested logical scenarios. Thus, if we cannot invalidate the logic pair using the SAT solver, the logic pair is a potential implication.

Also note that this method of testing a fraction of input vectors using ATPG and filling in the gaps with zChaff SAT solver was meant for large circuits, where it was impractical to test every input vector. For small circuits, implications can be

found simply by running functional simulations exhaustively for every possible input vector.

The result is a list of implications. That is to say, the implications are valid, but are not necessarily sequential or able to be physically reinforced. Thus, we complete the implication searching process by running the list through a custom application that checks that implications meet the aforementioned “useful” criteria.

2.4. The RD53 Circuit

The methods to be tested in this thesis, including implication chain building and fault observability, will be tested on the rd53 circuit, seen below in figure 5.

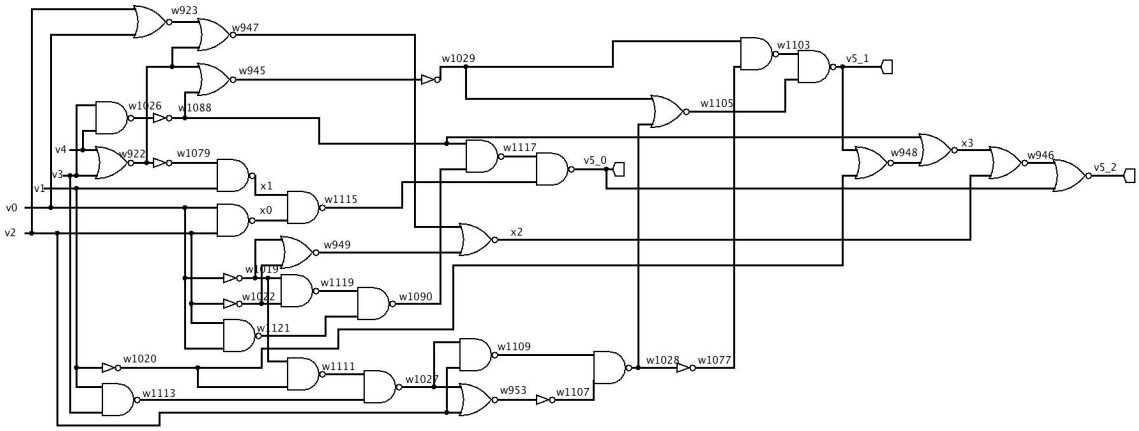

Figure 5: The RD53 Circuit

The rd53 circuit is made of 16 two-input NAND gates, 11 two-input NOR gates, and 8 inverters. RD53 was chosen partially because it had various combinations of logic within its relatively small 22nm gate size, and partially because it had both multiple inputs and multiple outputs. Implication chains must reach from a single primary output to a single primary input, and had only ever

been tested on single-output circuits in [2]. With limits on power dissipation and implication area overhead, we seek to test rd53 for the effects of chain sharing. At the same time, however, correcting errors via chains restricts error correction efforts to a few isolated paths. This is why we also investigate Fault Observability.

Fault Observability relies on correcting the node that has a high probability of generating a fault that gets propagated to a circuit output. It is important to test this method on a circuit wide and deep enough to deduce when the method becomes successful, or what fraction of high-fault nodes must be corrected before the method produces glitch-free outputs.

Figure 6 below shows the rd53 circuit after 80mV noise has been injected without any error correcting techniques.

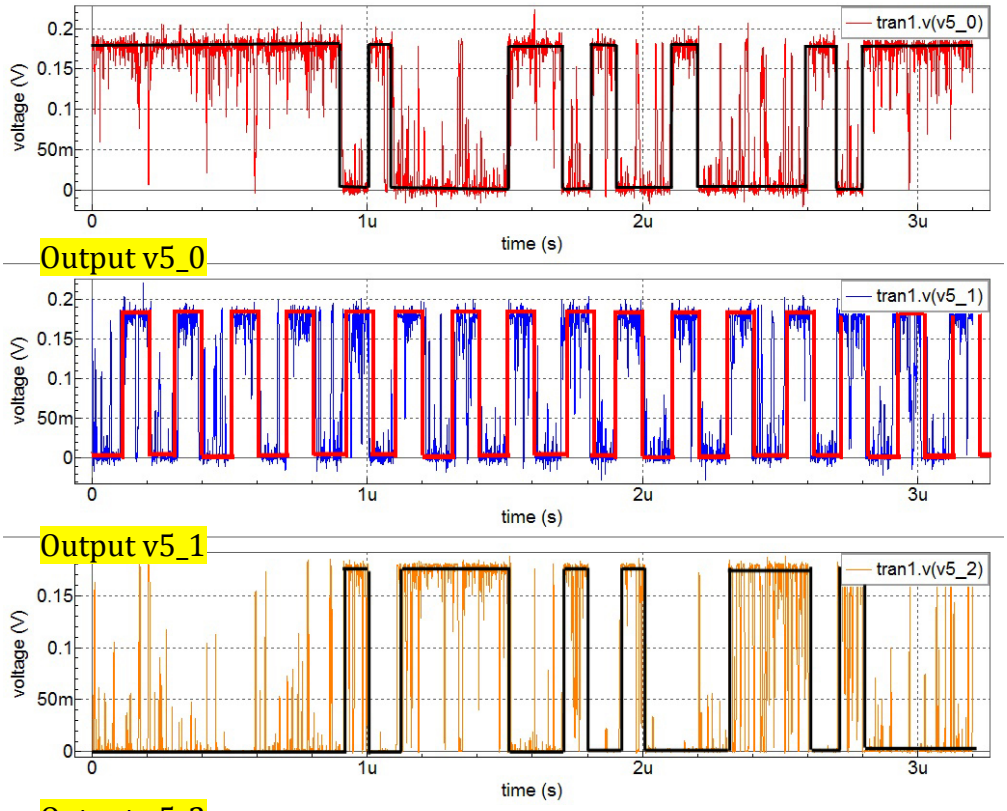

Figure 6: Outputs v5_0, v5_1, and v5_2 (from Top to Bottom) with 80mv Noise Injected

Note that the noise in figure 6 is meant to model thermal and RTS noise as explained earlier. Noise levels are meant to be high because the RD53 is simulated on a 22nm FDSOI process and is operating with a supply voltage of 180[mV] and a threshold of 0.3[V]. Signals at the outputs show several visible errors that we aim to correct throughout this paper.

We take the following sections to explain the two algorithms used to correct noise through implications. The first algorithm is based on previous work, and the subsequent algorithm is new work based on fault observability.

3. Implication Chain Building Methodology

3.1 Given Workflow

In this section we present a workflow that was developed with the implication chain method that can automatically identify implications and insert them into a circuit, visible in figure 7. Said workflow begins by feeding a list of useful implications to a custom application that performs implication selection. Said application determines the optimal implication chain(s) for a circuit that theoretically yield the lowest power consumption and area overhead. That is to say, this application searches for chains with the least amount of implications that satisfy conditions outlined in the following section titled “Implication Selection.” We then run a simulation in SPICE where RTS and thermal noise are injected into the circuit.

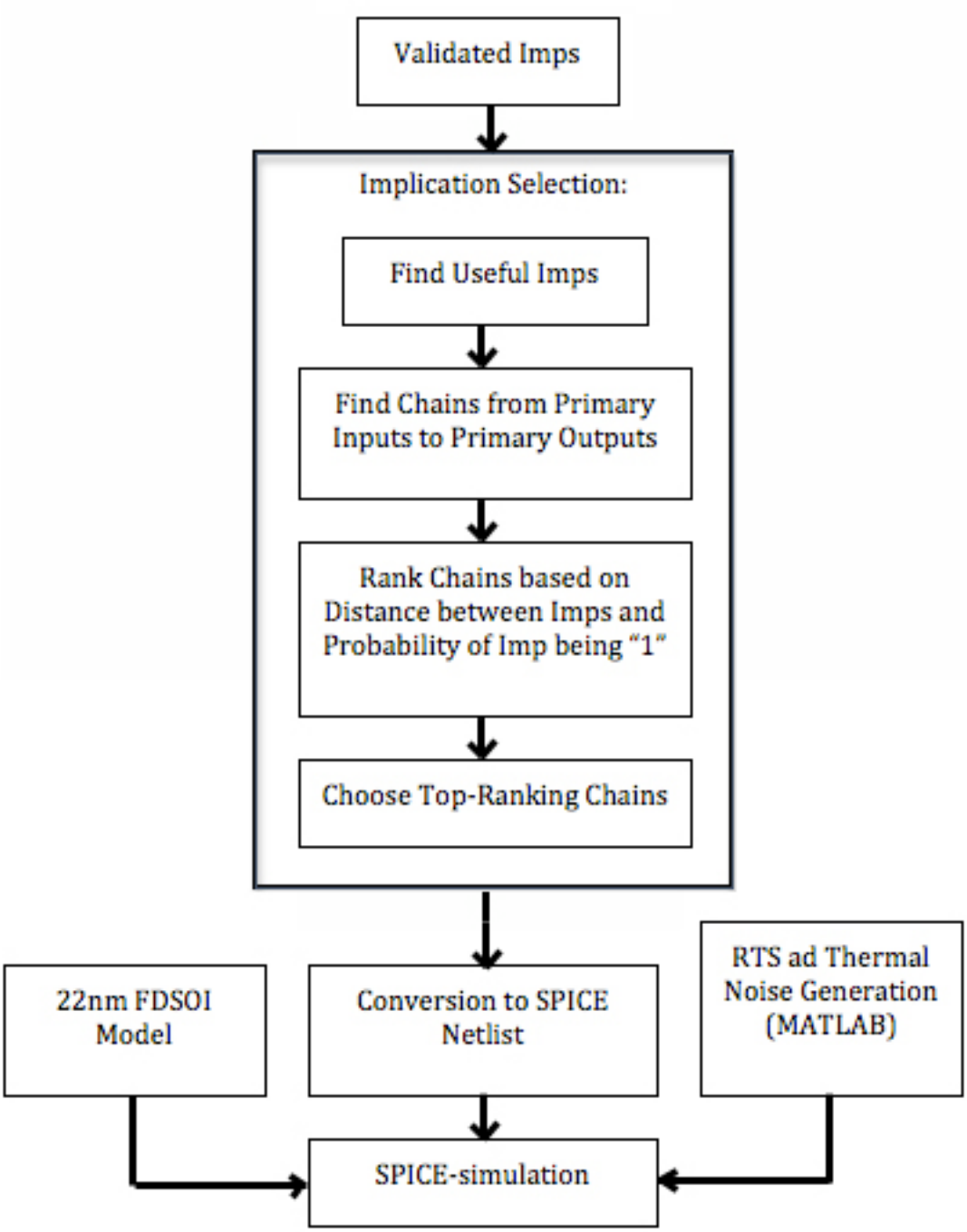

Figure 7: Flow for Selecting and Simulating Implication Chains taken and modified from [3]

3.2 Implication Selection

In the custom application that was inherited from the original implication chain building workflow, implication chains were selected with regard to the following criteria [2]:

- Probability that an implication will be activated by a "1"
- Distance between Implications

If given an output node, the application sought to build chains from said node to any of the circuit's input nodes.

The probability that the implicant will be "1" was taken into account because it was originally believed that an implication activated by a "1" would be less prone to errors [2]. Note that the probability of an implicant being 1 is deduced from the original functional simulation of the circuit during the implication finding process.

In addition, the distance between implications was taken into account because (a) we aim to use the smallest number of implications in a chain while keeping signal integrity, and (b) we wish to eliminate "self reinforcing implications." Thus, distances higher than one and less than four are favored when selecting implication chains. This stems from the fact that implications with a distance of 1, or "self reinforcing implications" do little to correct errors, because they rely heavily on their own logic. Furthermore, when implications are too far away, the signal sent from the implicant risks being degraded or worse, overrun by noise.

The application functions by parsing a verilog netlist of the circuit, and placing all available nodes into a graph. It then parses the validated list of implications and places implications on the graph. When given an output, the application traverses

the implications as many ways as possible, building an exhaustive list of implication chains. When each chain is made, it is given a score based on the probability of its implications' implcants being 1, and the distance between its implications being between two and four nodes apart. The chain with the highest score is predicted to have the most error correcting potential [6].

3.3 Implication Chain Optimizations

The inherited workflow did not include a mechanism for deducing chain overlap. Because we are attempting to enforce error correction on a sub-threshold, nanoscale circuit, it is imperative that we minimize the number of implications used while still correcting as many errors as possible. But, if a given circuit has x outputs, we run the risk of generating "x" chains and overshooting our power and area thresholds. Thus, we created a custom application to identify overlapping chains, and weigh them with regard to error correcting potential. A modified workflow can be seen on the following page in figure 8.

The problem with this method is, however, when chains are formed, several implications with a distance of 1 are chosen. Without these short, "self-implications," chains would not reach from a primary input to primary output. However, self-implications do not actively assist error correction, because the activation node and the output node involve the same gate. No chain for the RD53 circuit could be formed without self-implications. Three scenarios including these chains can be seen in the following section.

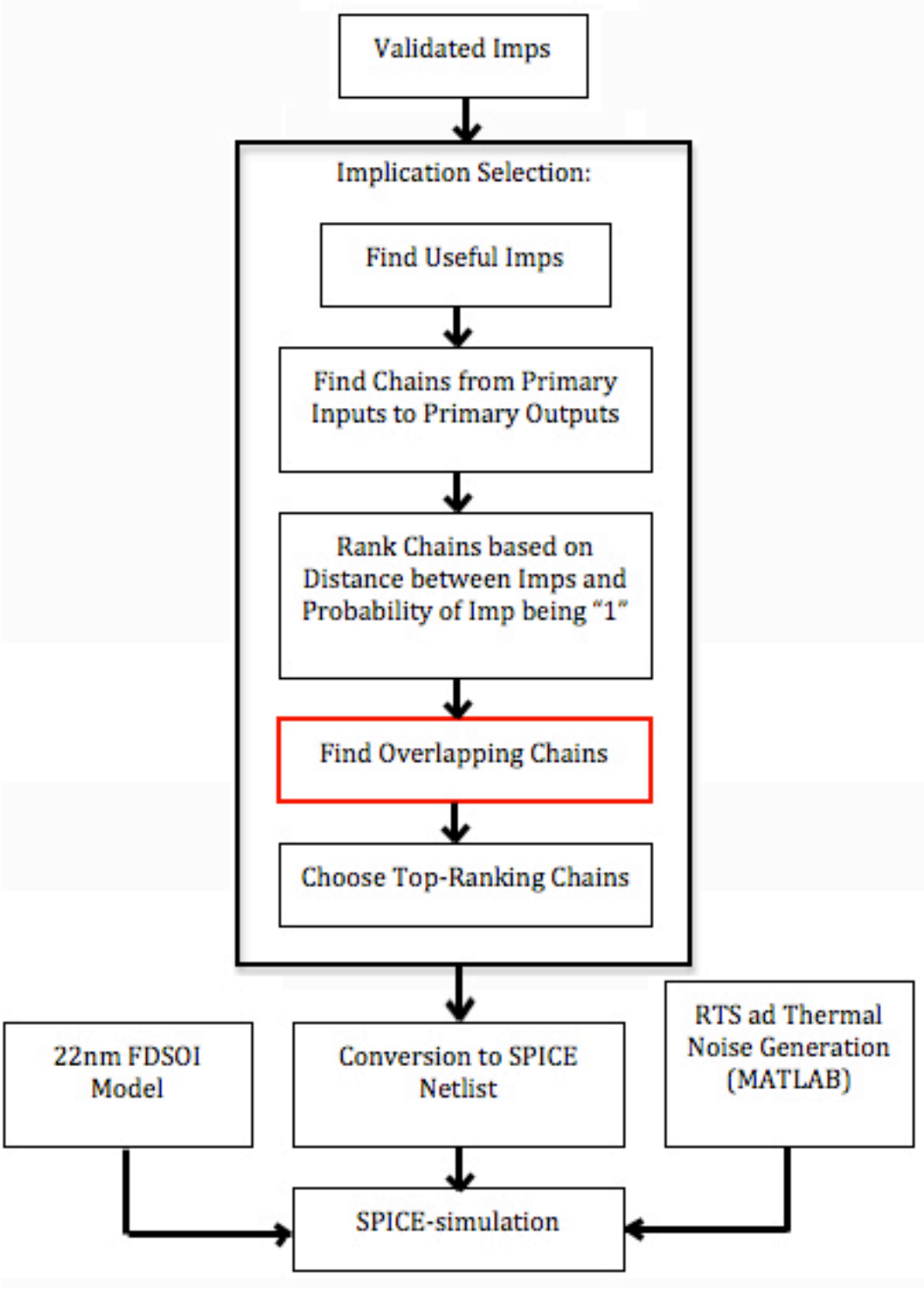

Figure 8: Implication Simulation Workflow taken and modified from [3]

The presence of self-implications is one shortcoming that we seek to remedy using the fault correction method.

3.4 Implication Chain Scenarios

The rd53 circuit was run through the above workflow and code, and various implication chains were generated. A presentation and explanation of the top three resulting chains and their scores can be read below.

The paths that the implication chains of the three scenarios follow are illustrated below in figures 9, 10 and 11 below. Each exhibits six implications total.

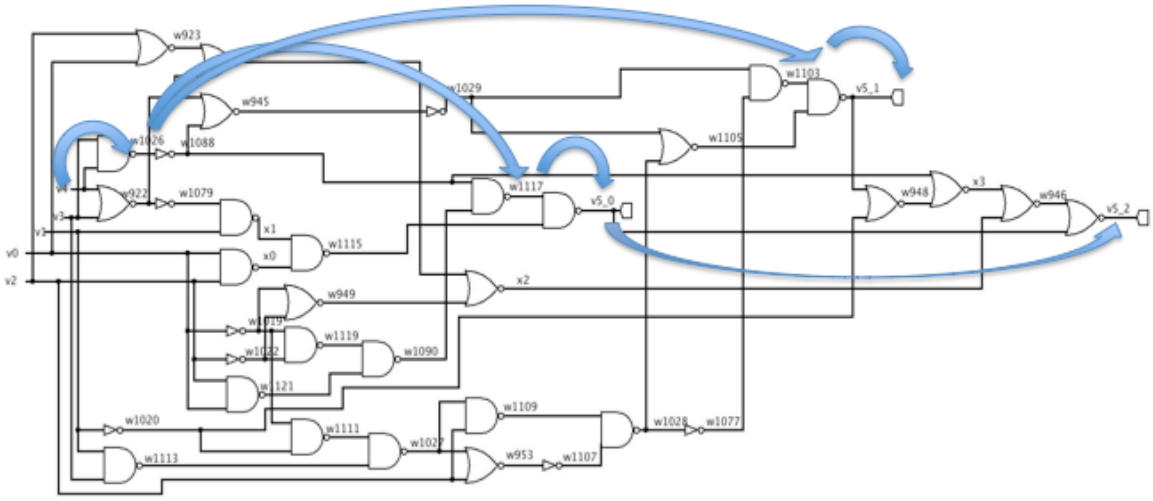

Figure 9: Scenario 1 Represented Graphically

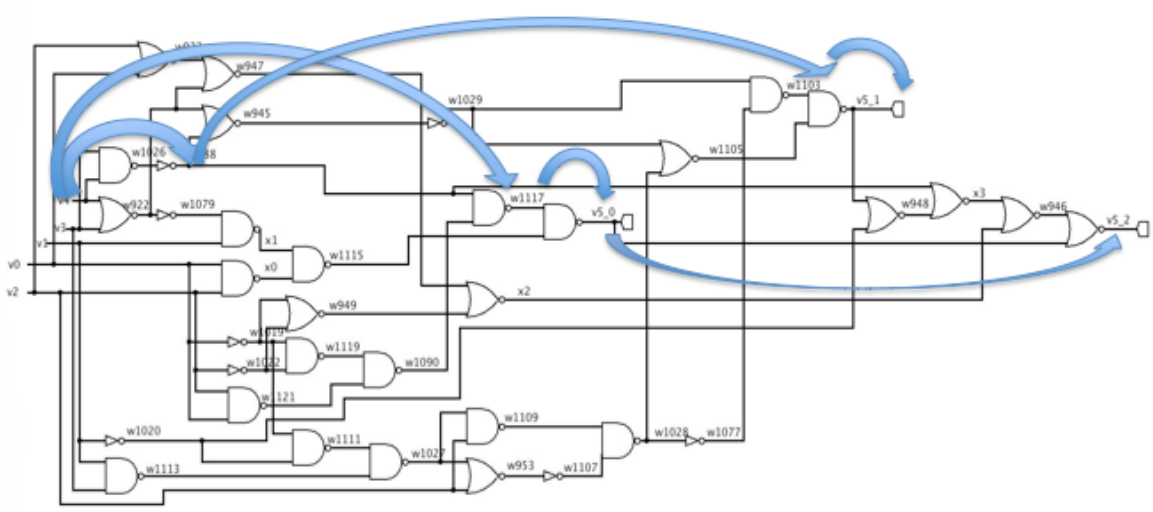

Figure 10: Scenario 2 Represented Graphically

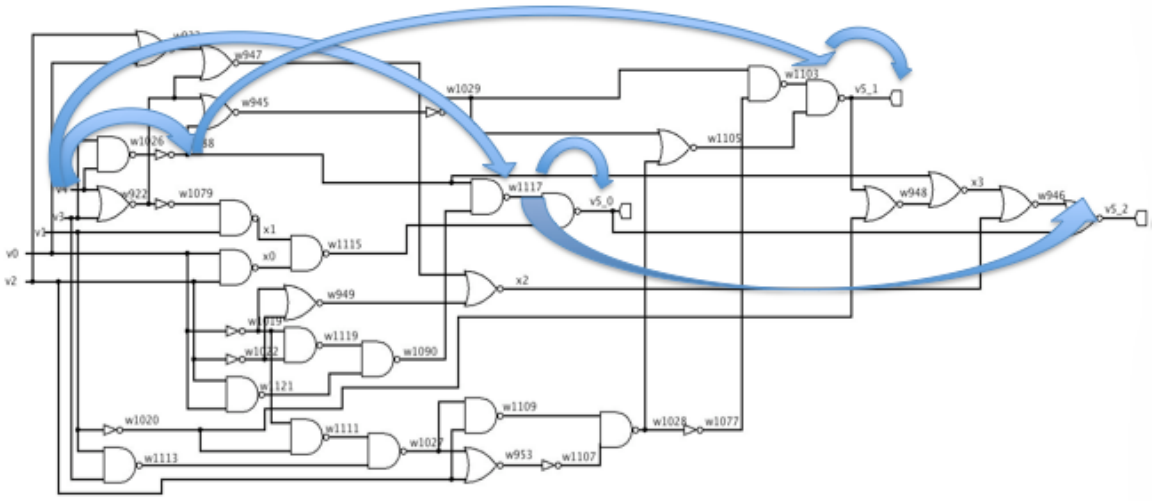

Figure 11: Scenario 3 Represented Graphically

The scenarios are classified as follows. The first scenario illustrates the highest scored overlapping chains where the v5_0 output reinforces the v5_2 output. The second scenario illustrates the highest scored separate chains where outputs reinforce each other. Lastly, the third scenario illustrates highest scored chains where outputs do not directly reinforce each other.

Notice how all three scenarios contain self-implications. We are thus not using these implications to their full benefit. Again, we seek to remedy this using fault observability.

The simulations for Scenarios 1-3 can be seen in the following section.

3.5 Implication Chain Simulations—Results and Analysis

We performed simulations on scenarios 1-3 on the rd53 circuit, which we presented in section 3.4. Each output (v5_0, v5_1, and v5_2) was probed for about

3.2 [µs]. The voltage vs. time plot for the v5_0 output can be seen in figure 12, below. 80 [mV] noise, meant to simulate thermal noise and RTS noise, is injected at each node in the simulation.

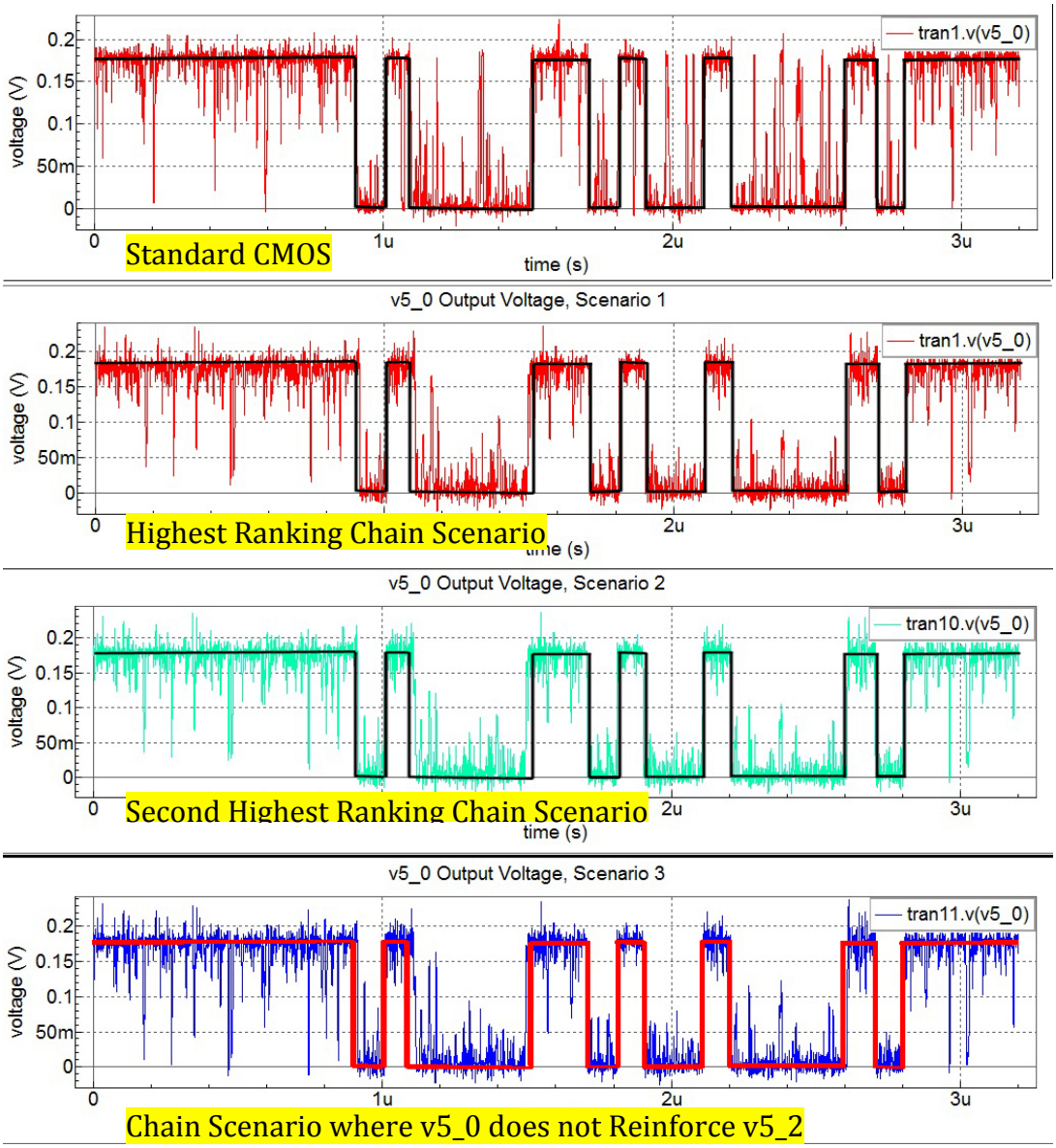

Figure 12: v5_0 Output Voltage for Standard CMOS and Scenarios 1-3

Scenarios 1-3 suppress noise on the low level most effectively. Although all three scenarios appear to correct errors with approximately equal effectiveness, scenario two appears to provide the best results. This, however, appears to be a random occurrence, because scenarios 2 and 3 use the same implications to reinforce v5_0.

The voltage vs. time plot for the v5_1 output can be seen in figure 13, below.

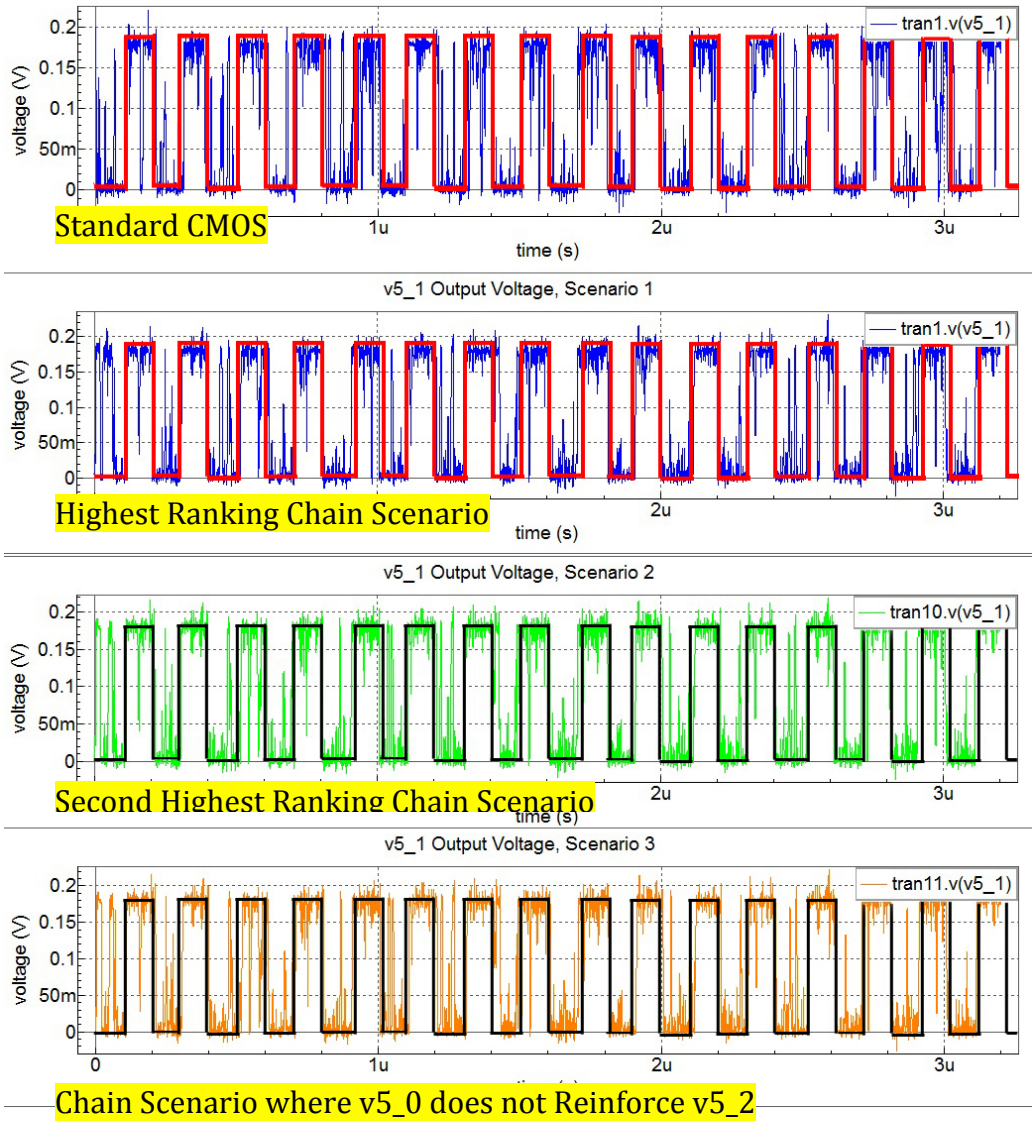

Figure 13: v5_1 Output Voltage for Standard CMOS and Scenarios 1-3

Error correction, again, looks very similar for all three scenarios for the v5_1 output voltage. Noise suppression appears to be more effective on the high level in all cases. However, scenarios two and three, which use the same implication chain, actually create errors on the low level. Thus, the implication chain used in scenario 1 appears to be the best choice.

Lastly, the voltage vs. time plot for the v5_2 output can be seen in figure 14, below.

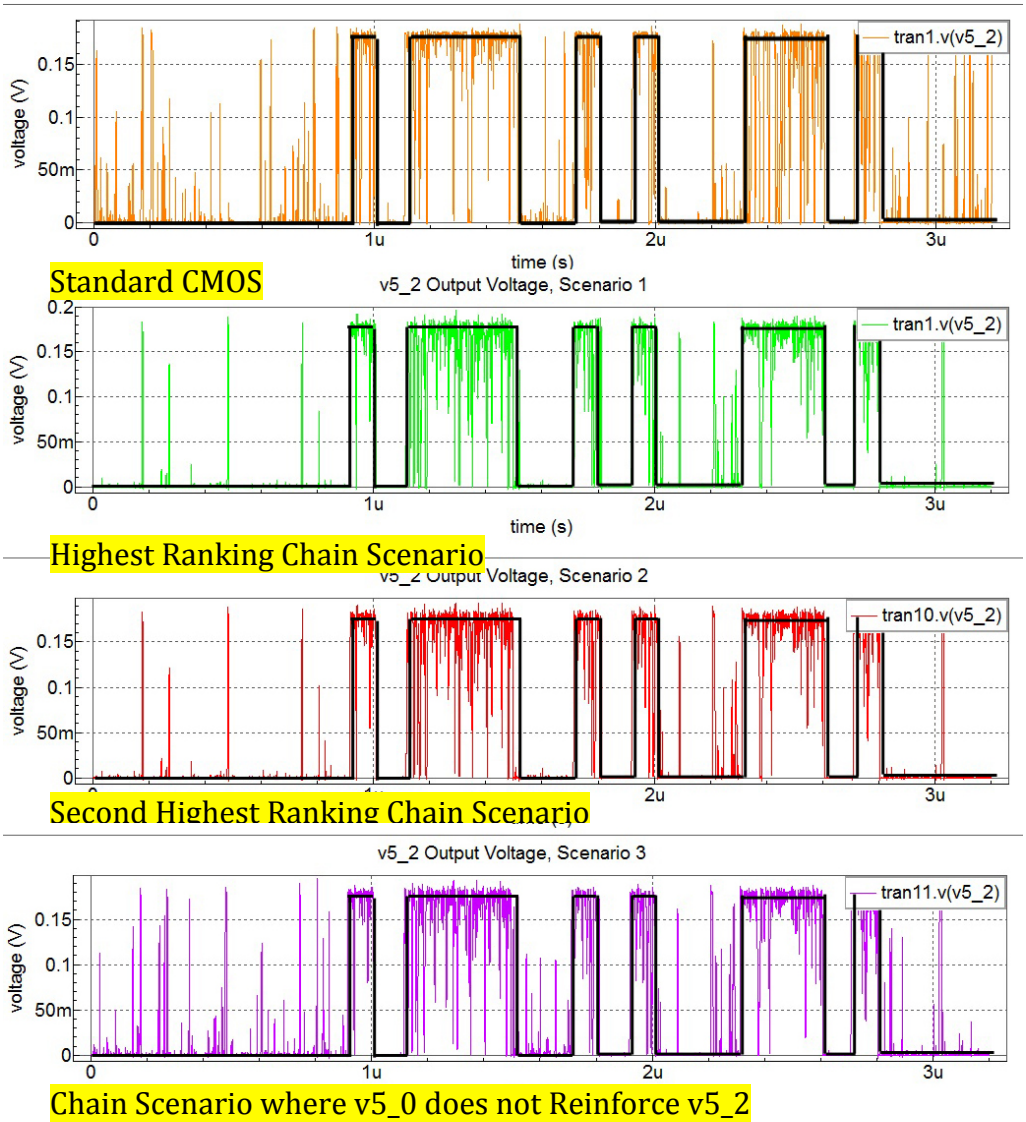

Figure 14: v5_1 Output Voltage for Standard CMOS and Scenarios 1-3

Scenarios 1 and 2 are much more effective at suppressing noise on the low level than scenario 3. In scenarios 1 and 2, v5_2 is corrected directly from the v5_0 output (a distance 1 away), and in scenario 3, v5_2 is corrected via a single node chain stemming from an input node (v4).

The implication chain-building algorithm is reasonably successful at error correction and noise suppression at the output. Scenario 1 appears to be the most effective, but at each output, at least one level remains very noisy. We seek to correct this with the fault observability method.

4. Fault Observability

4.1 Fault Observability Methodology

The fault correction method seeks to correct errors at the output of a circuit by placing implications on targeted nodes rather than paths. Said nodes are determined to be the area in the circuit that propagates the most errors, or faults, to the output. An area and power threshold caps the number of implications placed on a circuit.

To ensure that all outputs are reinforced, we search for the nodes with the maximum number of faults within the fan-in cone for each output.

In circuits with multiple outputs, targeted fault observability theoretically presents many benefits over implication chain building. First, it provides more breadth. Rather than all outputs relying on a single input node for error correction, outputs rely on several scattered implications. Second, fault correction aims to correct many of the redundancies of chain building. We can do away with

intermediary implications that are classifiable as “self-implications” and simply correct the node that we know is causing problems. In both methods, we can target individual outputs.

An obvious drawback however, to fault correction, is that the signal activating an implication may contain noise, or be erroneous. In an implication chain, this is not the case. However, because with fault observability, we are not attempting to build chains, we can eliminate self-implications from our selection process. This means we can theoretically use each implication to its fullest potential.

4.2 Workflow

In this section, we present the workflow that, when given a verilog netlist and a list of validated implications, can automatically be used to identify implications in a circuit that correspond to high-fault nodes in every output’s fan-in cone. It can be seen on the following page in figure 15. A custom script reduces the fan-in cone’s netlist, and each netlist is run through the Fastscan regimen automatically but separately.

4.3 Fastscan

Mentor Graphic’s Fascan is used in two capacities in the fault correction workflow. First, it is used to generate a set of exhaustive input and output patterns using the ATPG function. ATPG produces patterns by forcing all input nodes to a pattern, running a functional simulation on the circuit, and recording the output nodes’ ideal values.

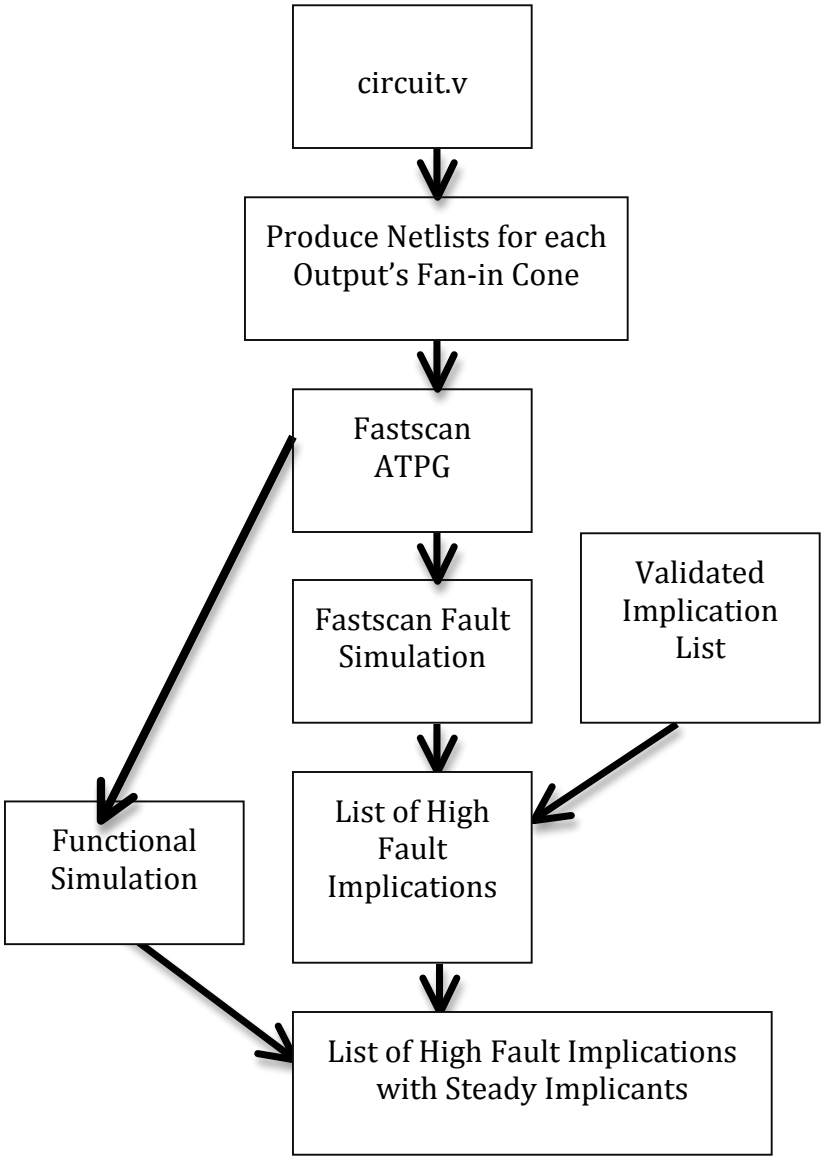

Figure 15: Fault Identification Workflow

During said functional simulations, the value of the signal at each node is recorded. The average value of a signal at each node over these simulations is taken, and is eventually used to determine the optimal implicant.

Fastscan is then used as a Fault Simulator. Fastscan fault simulation aims to predict faults in a given circuit with regard to the most common defects that arise due to a circuit's manufacturing process. While various types of faults can be simulated for, including slow transistors, circuitry shorting, etc, we are searching for faults known as "stuck at" faults.

Stuck at faults are aptly named—a "stuck at 1" fault occurs when a device fails to produce a logical zero, and a "stuck at 0" fault occurs when a device fails to produce a logical one. In order to detect these types of faults, we run a "functional test" on the circuit via fastscan. This means that fascscan checks for the existence of static defects (open, short, stuck-on, and stuck-open conditions), by comparing a given pattern to a simulated pattern. If their outputs differ, a fault has occurred.

The exhaustive ATPG runs produces as many patters as there are input value combinations. RD53, for example, has 5 inputs. Thus, 32 patterns are produced for each fan-in cone. Each pattern is then run through a functional test, and returns a list of nodes and the number of faults that have occurred at that node. The nodes are tallied over all simulations, and the nodes that produce the highest number of faults are determined to be nodes that need to be corrected. Thus, these high fault nodes are slated to be implicands.

4.4 Selecting an Implicant

As stated during the fastscan overview, during the initial ATPG functional simulations, the value of the signal at each node is recorded. The average value of a signal at each node over these simulations is taken. If the average value is close to 1,

we know that the node tends to be high, and if the average value is close to 0, we know the average value tends to be low.

Thus, we have a high-fault implicand, a list of validated implications, and average signal values at every ciruit node. The process of choosing implications from here is straightforward, and handled by a custom application. First, implications are selected from the validated list such that the implicand is a high fault node. Then, implications where the implicant is not in the fan-in cone of the implication are eliminated. Lastly, the list of implications is sorted with regard to how extreme a probability is. That is to say, implicant nodes with probabilities close to 0 or 1 are favored.

4.5 Fault Observability Scenarios

The following implications were generated for each fan-in cone corresponding to an output of the RD53 circuit. Within each cone, the implications chosen reinforce the node that was found to cause the most faults at the single output. Thus, if we are looking at the v5_2 scenario, we are reinforcing the node that is known to cause the most faults at v5_2. We are not reinforcing v5_2 itself. The three scenarios, each corresponding to an output, can be seen below.

Fan-in Cone of v5_2:

The v5_0 node was found to be the highest fault-producing node for the v5_2 fan-in cone. It produced 10 faults at the v5_2 output. The implications that were

found to reinforce v5_0 can be seen below in figure 16. The green triangle denotes the fan-in cone boundaries, and the blue arrows denote implications.

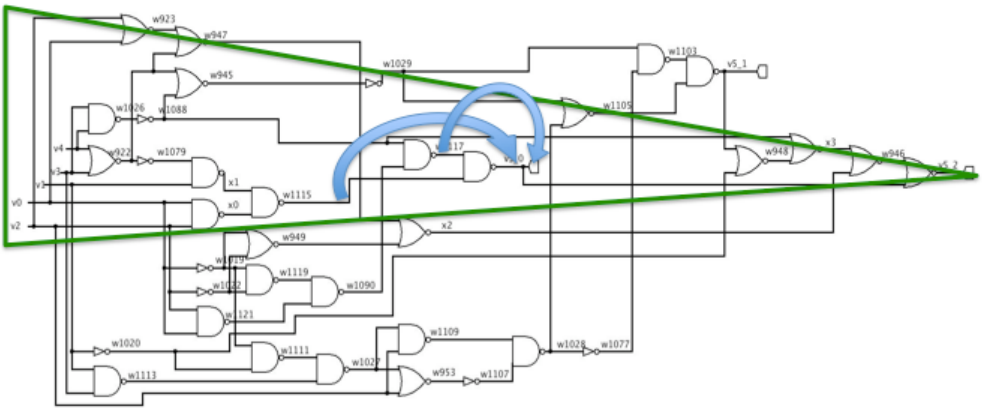

Figure 16: Implications for v5_2 fan-in cone

Fan-in Cone of v5_0:

The w1088 node was found to be the highest fault-producing node for the v5_0 fan-in cone. It produced 13 faults at the v5_0 output. The implications that were found to reinforce v5_0 can be seen below in figure 17. Again, the green triangle denotes the fan-in cone boundaries, and the blue arrows denote implications.

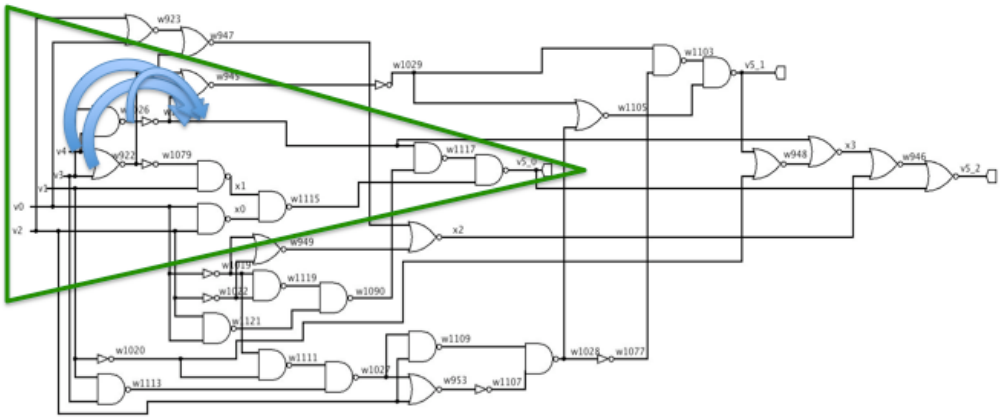

Figure 17: Implications for v5_0 fan-in cone

Fan-in Cone of v5_1:

The w1028 and 1027 nodes were found to be the highest fault-producing nodes for the v5_0 fan-in cone. They each produced 16 faults at the v5_0 output. The implications that were found to reinforce v5_1 can be seen below in figure 18.

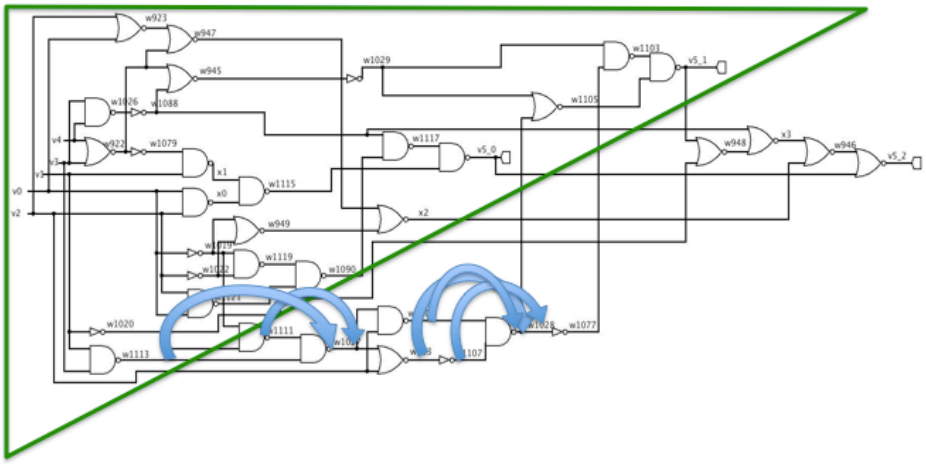

Figure 18: Implications for v5_1 fan-in cone

Interestingly, the majority of the implications that were discovered as valid reinforcements had a distance of 1 and are "self-reinforcing implications."

4.6 Fault Reinforcement Simulations – Results and Analysis

Fault Observability trials in which implications functioned correctly were found to be much more effective at reinforcing outputs than chains. For instance, the following implications (visible in figure 19) were reinforced in a separate trial, titled trial B.

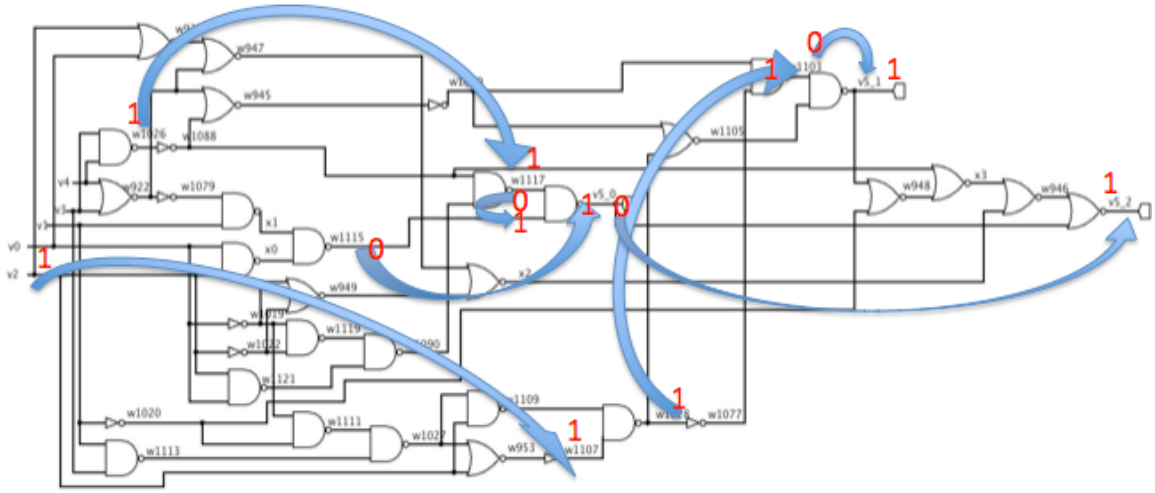

Figure 19: Trial B Implications

Simulation results can be seen for trial B in figure 20.

In figure 20, output v5_0 appears to be a cleaner signal than it was in the implication chain scenarios, although a high signal was wrongly suppressed in trial B. Noise suppression at the v5_0 output was mainly due to an implication that reinforced the w1115 node. In addition, the v5_1 output voltage is cleaner than the implication chain scenarios for the first 1.5[μs], after which it degrades. The v5_2 output remains noisy.

The discrepancy in the v5_1 output voltage can be explained by the implication that reinforces the v5_1 node. Recall that the implication chain leading up to v5_1 in trial B is:

w1028(1)→w1103(1)

w1103(0)→v5_1(1)

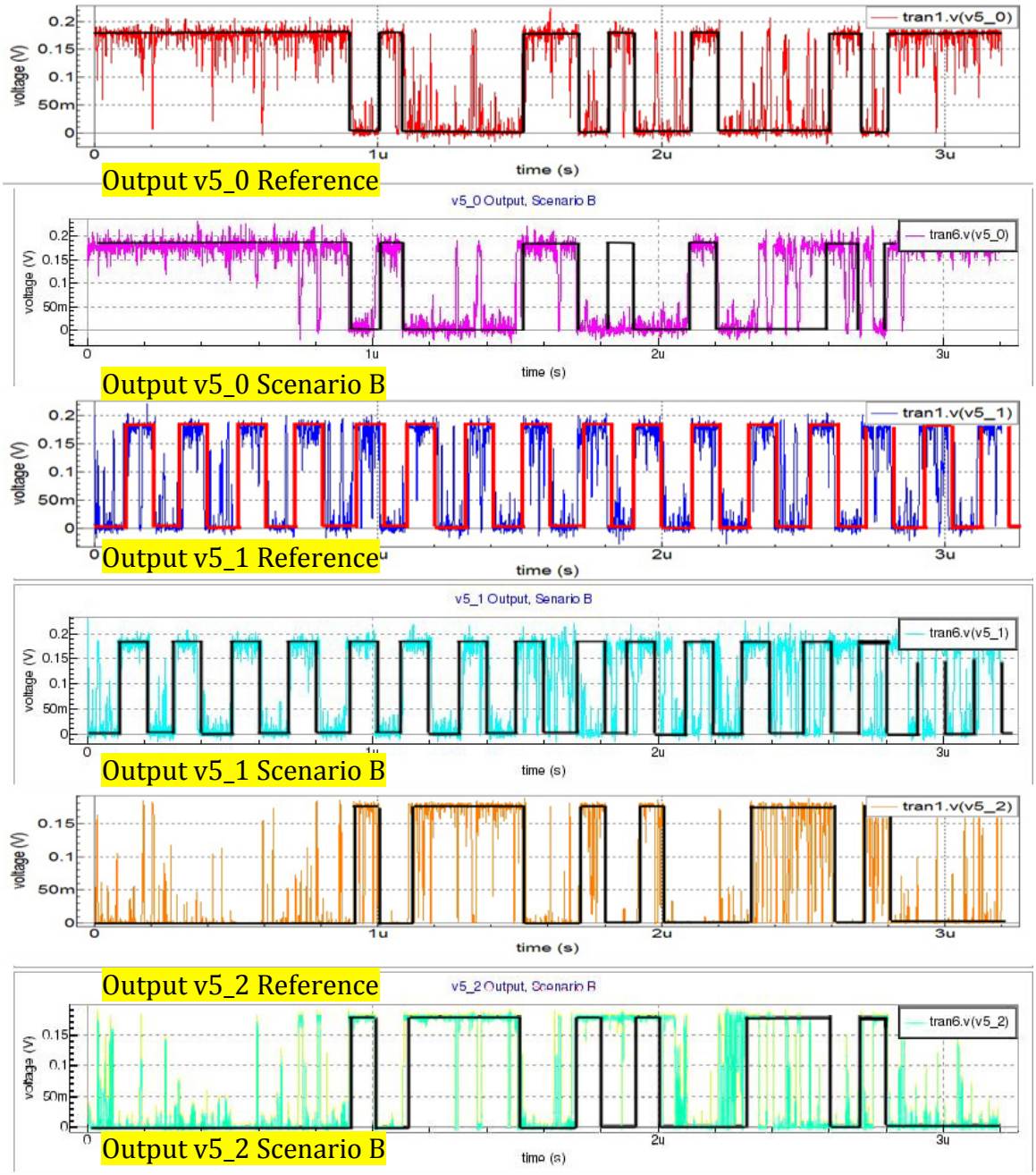

Figure 20: Trial B Output Voltages

Thus, the w1103 node corrects the v5_1 output when w1103 is low. When w1103 is high, it does not correct the output. Simulation results can be seen for the two nodes, w1103 and w1109, that produce the output at v5_1, in figure 21, below.

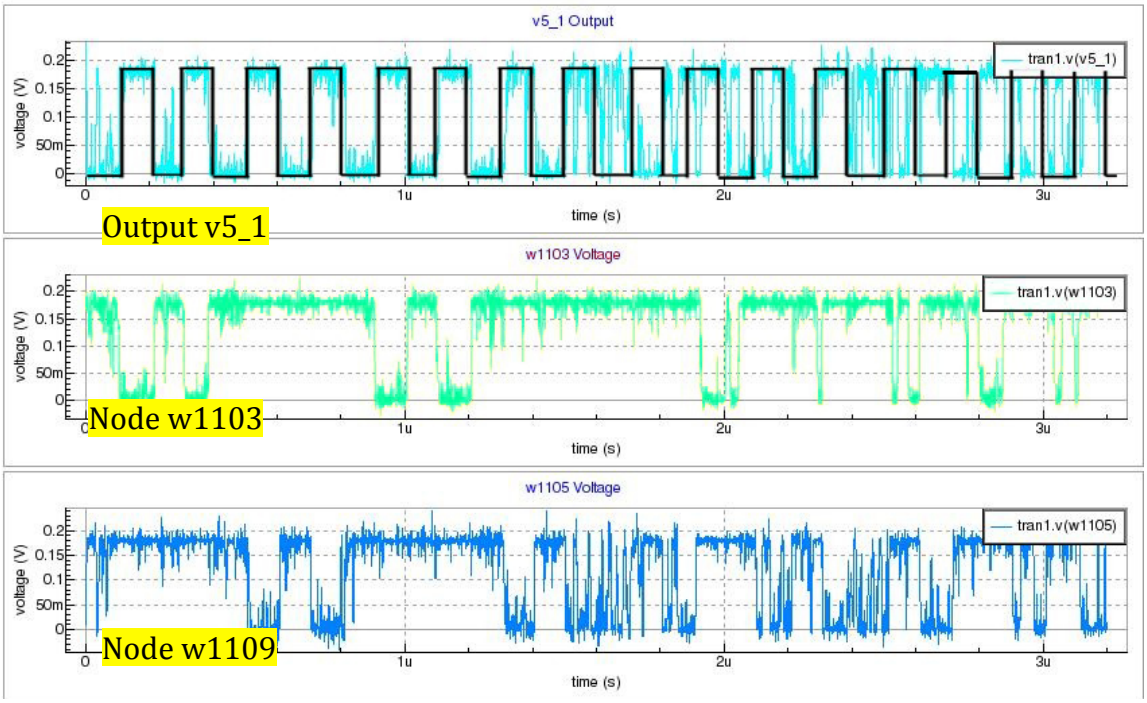

Figure 21: Trial B Node Voltages for v5_1

Notice how the node w1103 remains high, and thus is not in a position to reinforce v5_1. When w1109 is a clean signal for the first 1.5[μs], the v5_1 output is effective at suppressing noise. However, when the w1109 node is noisy, the v5_1 output is overrun by noise. This is because the probability that the w1103 node will be zero is 0.16. In many cases, implications found for correcting nodes with high fault observability exhibit this problem—that the implicant is not at the necessary value enough of the time to be an effective control signal.

It is important to note that implications should be chosen not only by their fault observability, but also their probability of activation. Implications that have a high probability of being activated should be chosen over those with a low probability of being activated.

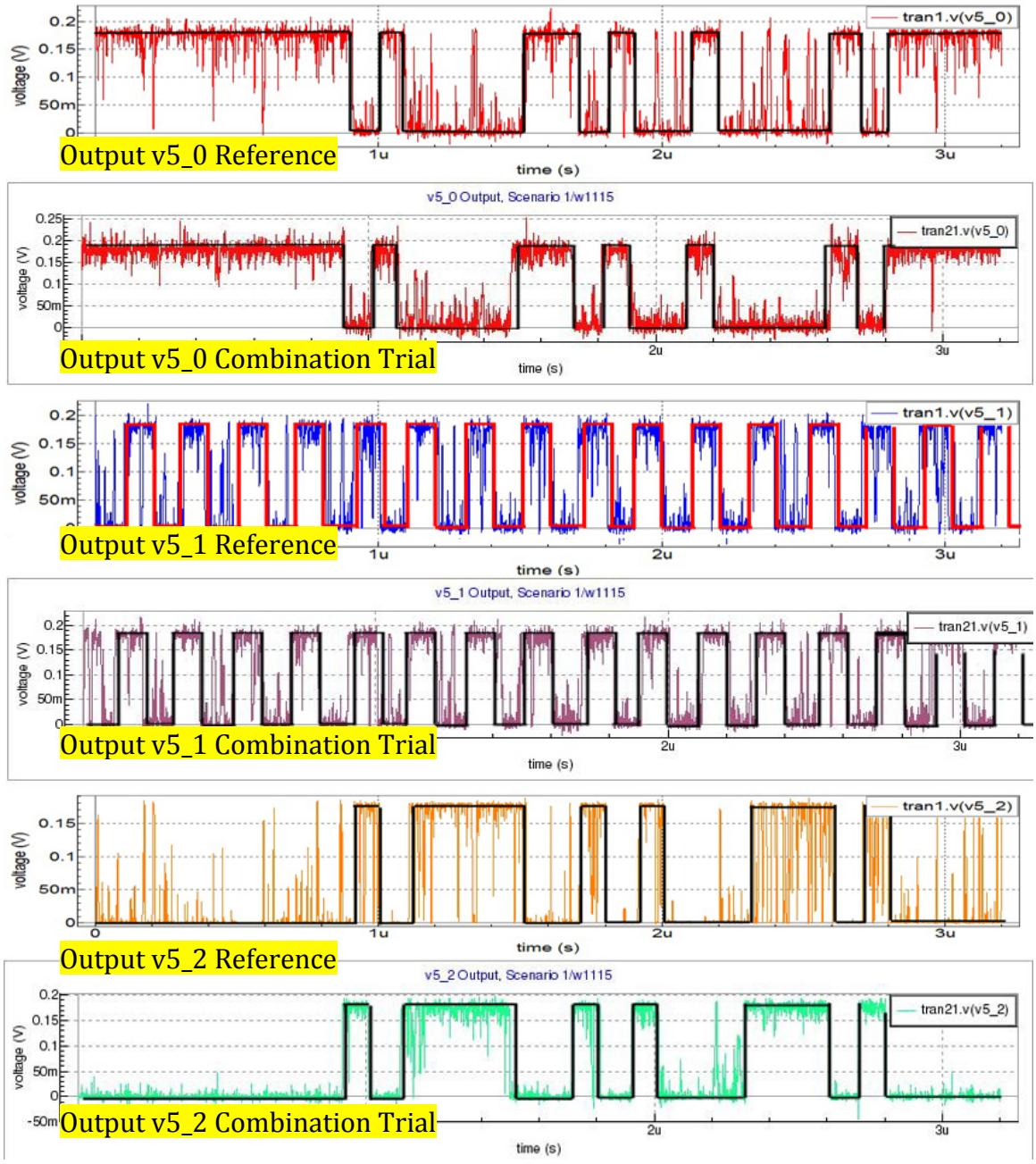

Figure 22: Combination Fault Observability and Implication Chain Trial

Thus, we propose that including implications selected by fault observability in implication chains would be an effective way to circumvent this problem. To

make the cleanest possible output signals for the rd53 circuit, we proposed the following actions:

- Reinforcing the w1115 node within the scenario 1 implication chain using a single implication
- Reinforcing the v5_2 output node with the Schmitt trigger configuration seen in figure 3 (left). This allows us to suppress noise on both the high and the low polarities, which otherwise would remain noisy.

The resulting simulation results can be seen on the previous page, in figure 22.

The v5_0 output is the cleanest we have seen it, as is the v5_2 output. The output v5_1 contains glitches, but much of the noise from the reference trial is suppressed.

The combination implication chain and fault observability trial is the most effective method for correcting errors at the output.

5. Conclusions

We have demonstrated a technique for suppressing the effects of noise in multiple output, combinational circuits. This technique uses invariant relationships to reinforce correct logic in feed-forward chains. The chains are then augmented by adding select implications that are known to reinforce nodes with high fault obeservability.

Fault observability is useful in correcting errors at the output, but it cannot be used alone. Fault observability cannot be used alone because the probability of activation for an implication on an "observable fault" node may be low. Using fault observability in concurrence with implication chain building is an efficient and effective mode of error correction.

The technique is desirable because it produces cleaner output signals than either implication chain building or fault observability alone. In addition, it has smaller area overhead and power dissipation than were we to simply reinforce every node with a Schmitt Trigger gate.

6. Future Work

So far, the combination implication chain and fault observability technique is not completely automated. Though we have a method to identify chains, and though we have a method to identify implications that reinforce nodes with high fault observability, we do not have an algorithm that combines the two.

In that vein, we wish to investigate a means of weighing fault observability into the chain selection algorithm from an earlier stage. From that standpoint, it is necessary to weigh the importance of fault observability against the importance of the implicant tending to be at the desired logic value.

In addition, we wish to test this method on circuits other than rd53. Circuits with gates with more than 2 inputs have not yet been investigated, and we hope to see how they affect fault observability and chain building.

7. Acknowledgements

This thesis was prepared with significant contributions from Marco Donato, Brown University.

In addition, thank you to advisors Professor Ruth Iris Bahar, Professor William R. Patterson, and Professor Alexander Zaslavsky.

8. References

[1] V. De and S. Borkar, “Technology and design challenges for low power and high performance,” in *Proc. ISLPED*, 1999, pp. 163–168.

[2] M. Donato, F. Cremona, W. Jin, R. I. Bahar, W. Patterson, A. Zaslavsky, J. Mundy. A Noise-immune Sub-threshold Circuit Design based on Selective Use of Schmitt-trigger Logic. In *GLSVLSI’12*, pages 39-44, IEEE, May 2012.

[3] N. Alves, Y. Shi, J. Dworak, R. I. Bahar and K. Nepal, "A Cost Effective Approach for Online Error Detection Using Invariant Relationships," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29(5), p. 788–801, May 2010.

[4] B. Dokic, "CMOS NAND and NOR Schmitt circuits," Microelectronics Journal, 27(8), p. 757–765, Nov. 1996.

[5] A. Wang and A. Chandrakasan., "A 180-mV subthreshold FFT processor using a minimum energy design methodology," IEEE Journal of Solid-State Circuits, pp. 310-319, Jan. 2005.

[6] W. Jin, “A Schmitt-Trigger-Based Approach to Noise-Immune Sub-Threshold Circuit Design,” Undergraduate Honors Thesis, Brown University School of Engineering, April 2012.