An Automated Synthesis Tool for Generating Noise-Immune Sub-Threshold Circuits

Amrita Mazumdar
am3210@columbia.edu

August 27, 2013

Abstract

Nanoscale circuits operating at sub-threshold voltages are increasingly susceptible to the impact of random telegraph signal (RTS) and thermal noise, resulting in soft errors that compromise a circuit's reliability. This work presents a low-power, area-efficient error correction technique and an automated tool to synthesize noise-immune circuits. The tool selectively applies reinforcement using invariant relationships to correct noise-induced signal errors. Simulations demonstrate our synthesized circuits provide better noise immunity than standard CMOS technology in tests with limited area and power overhead.

1 Introduction

As CMOS technology shrinks in accordance with Moore's Law, nanoscale circuits are required to functionally operate at sub-threshold voltages [1]. At such low power, circuits become much more susceptible to the impact of random telegraph signal (RTS) and thermal noise as node capacitances and voltage supplies decrease, and produce single-event upsets (SEUs) that compromise a circuit's reliability [2, 3].

Because these SEUs are, by nature, ephemeral and variable in frequency, their effect may be negligible in standard circuit design. Ultra low-power (ULP) applications, which primarily operate in the sub-threshold voltage region, become increasingly susceptible to critical errors as a result of soft errors [3]. Techniques to combat the effect of soft errors must also be low power and area- efficient, so as to not exceed the constraints of nanoscale circuit design [2].

This work presents a methodology for improving the noise tolerance of sub-threshold circuits using Schmitt trigger logic and invariant relationships, and the results of varied implementations of this methodology on circuits from the MCNC benchmark netlists. This work builds off of prior results from Nepal et al. [4], Alves et al. [5], Donato et al. [2], and Moser [6].

2 Background

The strategy presented in this work for suppressing noise and reducing errors is fundamentally built on the concepts of invariant relationships and Schmitt trigger logic. This section provides some background on these topics, as well as a review of the circuits used for simulation and testing.

2.1 Invariant Relationships

The invariant relationships, or implications, used in this work are formal assertions of gate-level logic values in a circuit. For clarity, we express implications in the following form:

$$siteA = x \rightarrow siteB = y, \{x, y\} \epsilon \{0, 1\}$$

where $siteA$ is referred to as the *implicant* and $siteB$ is referred to as the *implicand*. In digital circuits, these implications occur naturally, and logical constraints ensure several implications exist between nodes in a circuit. Implications are an effective choice for error correction in digital logic because they are naturally occurring and no knowledge of high-level behavioral constraints is required to identify them. [5]. For example, in Figure 1, we show the basic implication $N3 = 0 \rightarrow N24 = 0$, meaning if N3 = 0, then N24 = 0 also at all times, regardless of other node values.

While implications are a basic and efficient means of error detection and correction in a circuit, they do have some limitations [5]. Firstly, a single implication can only suppress errors of a single polarity

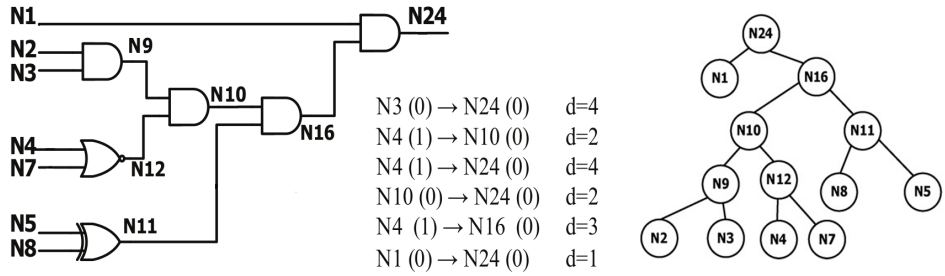

Figure 1: Example circuit with implications and graph representation, from [8]

for a given implicand, which motivates an informed selection of the best implication chosen to reinforce a node. Additionally, circuits optimized to minimize area and/or delay can constrain the number of usable two-site implications, further motivating a strategy for implication selection that can reinforce both optimized and unoptimized logic. If, however, valuable invariant relationships can be specifically identified and reinforced, they can become a powerful tool for real-time error detection and correction [4]. However, reinforcing all possible implications at runtime becomes costly in terms of area and power overhead, and determining some means of selectively choosing optimal implication sets is required [6]. This reasoning motivates our exploration of various methods for selective implication reinforcement.

2.2 Schmitt Triggers

The Schmitt trigger circuit has been classically used for applications requiring hysteresis, or lag, in the voltage transfer characteristic [7]. Hysteresis can quite useful in suppressing wide-range noise, and the Schmitt trigger has been effectively used to extract signals from noisy circuits [6]. Also notable in the Schmitt trigger's voltage transfer function is the slow input response coupled with fast output transition. These characteristics make the Schmitt trigger ideal for soft error correction in ULP logic, as its transfer function effectively "ignores" slight voltage deviations. A comprehensive evaluation of CMOS Schmitt trigger gates can be found in Dokic, and Sasaki et al. presented general soft error and noise masking using classical Schmitt trigger logic with good results, indicating a selective usage of Schmitt trigger gates may provide similar success [7, 3].

A major advantage of using the Schmitt trigger circuits is their relatively low transistor count overhead; as noted in Dokic, a CMOS . Nevertheless, replacing every gate in the circuit with a Schmitt trigger implementation would still lead to an unacceptable transistor count. Instead, we propose selectively adding Schmitt gates to reinforce signals that are most likely to produce bit flips at primary outputs.

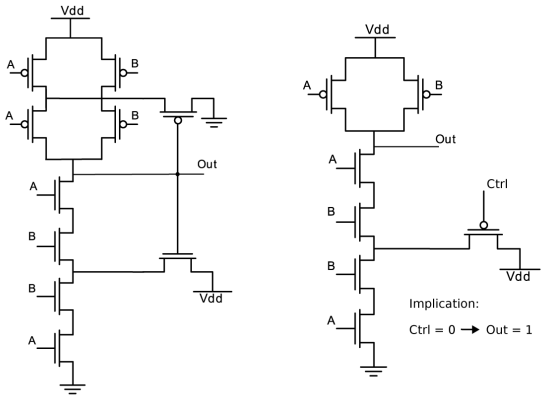

Figure 2: Basic Schmitt trigger NAND (left) and modified Schmitt trigger NAND (right) for implication reinforcement, from [2]

We use a modified Schmitt trigger gate, first presented in Donato et al., with feed forward capabilities to implement implication reinforcement in a circuit [2]. Referring again to the implication set in Figure 1, we can use the implicant $N4$ in the implication $N4 = 1 \rightarrow N10 = 0$ to reinforce, or correct, the value at $N10$ through a feed-forward mechanism. Because Schmitt triggers typically reinforce an output using a feed-back mechanism, we must modify the conventional gate to add a control input for the implicant and to maintain proper operation when the implication is not activated. A conventional

Schmitt trigger NAND and its modified counterpart are shown in whatever figure for reference. One notable consequence of the modified Schmitt trigger is that four possible modified Schmitt gates are possible for each gate, one for each combination of pull and pass transistor used to reinforce the implication [2]. Also notable is that while the total number of transistors used in a conventional m-input Schmitt gate is $2(2m+1)$, our modified Schmitt gate uses $3m+1$ transistors, which for standard 2- and 3-input gate is smaller and more convenient for ULP circuits which typically have area and power constraints.

2.3 MCNC Benchmark Netlists

The Microelectronics Center of North Carolina (MCNC) benchmark suite is a collection of standardized libraries and circuits. The suite is widely used in academic research for testing, and has a number of circuits ranging from simple to complex. For synthesis and testing in this work, we use a range of circuits from the MCNC benchmark suite implemented using a 22nm FDSOI model card, specifically the circuits RD53, RD73, MISEX1, T481, and 5XP1.

3 Methodology

In hopes of experimentally determining an effective scheme for selecting an implication set with optimal noise suppression and SEU correction, we present two methodologies for selecting implication sets: scattered high-fault node reinforcement and implication chain-building. Both methodologies require preliminary use of implication detection techniques developed by Alves et al. and Nepal et al., and are presented first for completeness [8, 5, 4].

3.1 Implication Detection

In this section, we present the workflow used to automatically identify and validate available implications in a circuit. Originally presented in Alves et al., the workflow is implemented as a custom script which generates implications from a circuit's Verilog netlist [5].

The workflow can be seen in Figure 3 and is implemented in a custom script. The custom script invokes Mentor Graphics FastScan (2009.1.10) to run Automatic Test Pattern Generation and functional simulations. The results of these simulations are used to populate a set of input-output vectors. To reduce the heavy computational load of running ATPG and functional simulations for larger circuits, we only use ATPG to generate a subset of possible input patterns, and then use the zChaff SAT solver to invalidate the implications.

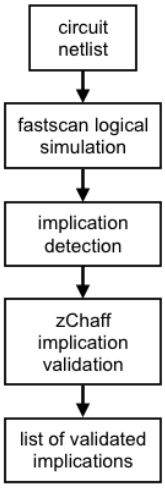

Figure 3: Design workflow for implication detection, adapted from [5]

The result of this process is an exhaustive list of valid implications present in a given circuit. This list is typically much larger than would be useful in using implications to reinforce nodes in a circuit, and includes many self-reinforcing, "backwards", or otherwise ineffective implications. A self-reinforcing implication, one where the implicant is the input to a gate and the implicand is the output of the same gate, is not considered ideal because it does not provide any new reinforcement to the node and can potentially strengthen a glitchy output. A backwards implication, where the implicant is closer to an output than the implicand, is inherently ineffective at the single-cycle reinforcement we are working to promote. Other criteria for a clearly ineffective implication include a wide distance, which could induce more errors, and implications that share an implicand, as our Schmitt trigger-based reinforcement method only has one control input for an implicant. Activation probability, or the probability that an implication will be invoked, can be extrapolated from the functional simulations conducted during implication detection, and is also a useful metric for gauging the efficacy of an implication. These criteria can all constrain the final set of implications an automatic tool selects, but de-

termining which constraints are most critical, how to weigh and prioritize the different options, and how to distribute implications across a circuit all complicate the selection process and require a bit of exploration to determine best practices for implication set selection.

To that end, we present two different algorithms for implication selection: scattered reinforcement of high-fault nodes, and implication chain building. Custom scripts were developed for each methodology and simulations on their results were run to examine error correction and overall noise suppression.

3.2 Scattered Reinforcement of High Fault-probability Nodes

The primary goal of scattered reinforcement of high fault-probability (HF) nodes is to use simulation tools to generate a list of potentially error-prone nodes in a circuit, and use implications to specifically target these nodes. While this fault simulation technique cannot irrefutably determine soft errors, which by nature are transient and non-repeating, it can serve as a good indicator of nodes with high fault observability. To ensure that all outputs are reinforced, we run this selection process within the fan-in cone (FIC) of each outputs, rather than blanket coverage of the circuit. After a lower bound of nodes are reinforced for each output, if area and power constraints have not been met, we then select more implications that reinforce the remaining "high fault probability" nodes in the circuit.

Fault observability appears a good selection criteria because of increased breadth in coverage, targeted selection of demonstrated "important" implications, and more flexibility in implicant selection [6]. Reinforcements can be placed across the most likely points of failure in a circuit, rather than constrained to a certain path, which provides both more flexibility in specific selection and in distribution of coverage. If HF nodes are concentrated in a specific area, a likely scenario when dealing with soft errors, they can be target without having to parse out the ones that fit into some more rigid arrangement scheme. Additionally, we can choose the strongest implicants to reinforce a node without focusing on a given path or nodes originating from one input, increasing the likelihood of propagating a correct signal rather than a corrupted one.

The workflow, shown in Figure 4, essentially implements the following algorithm. It describes the process for producing a list of implications that reinforce high-fault nodes within given power and area constraints.

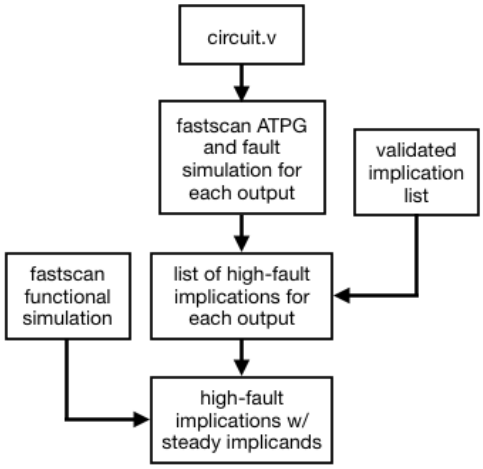

Figure 4: Design workflow for HF-node reinforcement

This process equal weights the activation probability of an implication, the "high fault-probability", or percentage of faults at an output caused by a given implicand, and the "low-fault probability", or steadiness (low fault count) of an implicant in a fault simulation of an output. Additional trials weighted different combinations of these criteria to evaluate which characteristics were most influential on an output's error count.

3.3 Building Implication Chains

An implication chain is a set of implications which reinforce nodes along a continuous path from a circuit's input to an output. The motivation for building implication chains stems from our assumption that input vectors are pure signals with few glitches, and the glitches arise organically during circuit operation. By reinforcing nodes along the path from a stable input to an output, the output node can be significantly reinforced.

The algorithm and selection criteria developed for implication chain-building were comprehensively explored in Moser, and we present a summarized overview of the process for completeness [6]. The inherited workflow and script have been adapted and optimized based on results from the high-fault node reinforcement trials, and an updated workflow and

Input: circuit.v, implication list
Generate graph representation of circuit;
Import all valid implications into graph;
foreach *output in the circuit* **do**
  Make verilog from output's FIC;
  Run ATPG, fault simulation;
  Make sorted HF node list from fault report;
  foreach *HF node at each output* **do**
    Get list of implications with node as implicand;
    Make verilog from HF node's FIC;
    Run ATPG, fault simulation;
    Parse fault report in reverse;
    Make sorted low-fault node list;
    Create list of implications sorted by
    - activation probability
    - HF implicand probability
    - low-fault implicant probability
  end
end
repeat populate best implications list
  Add top implication at each output to final list
until *power and area constraints are met*;
Result: List of best implications

Algorithm 1: Scattered High-fault node selection process

algorithm are presented.

The advantage of using an implication chain is clear: a clean signal is pushed through to an output via reinforcements, and glitchy nodes not along the chain's path are suppressed. The drawback, however, is that the number of implications used is limited by the shortest path from an input to an output. We seek the shortest path with implications as adding nodes to extend the chain may not be as area- or power-efficient, but chains with a small number of implications from an input to output may not cover the other paths of greater fault probability. The script output all chain sets for all outputs, and chains were chosen by inspection based on total chain length, activation probability, and overall score. Overlapping chains, chains with different primary outputs which contained overlapping implications, were discarded, to ensure effective and well-distributed Schmitt trigger placement while constrained with power and area limitations. Additionally, primary outputs that reinforced other primary outputs were prioritized, to help minimize the number of nodes needed.

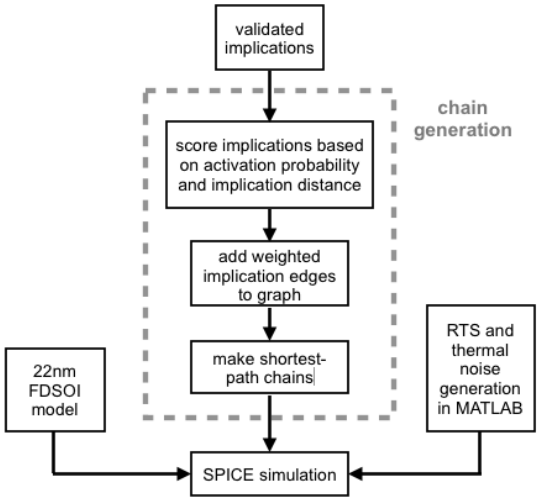

Figure 5: Design workflow for implication chain-building

The original workflow for chain-building traverses the graph from all outputs to all inputs, finding all implication chains from a primary output to a primary input. It compiles an exhaustive list, and scores each chain based on an earlier preferred criteria of implication activation on a high signal and small implication coverage distance ($1 < distance < 4$).

An evaluation of the previous script demonstrated average activation probability and distance were strong indicators of a chain's success rate, and some metrics used by the previous scoring metric (prioritizing high-valued activation, for instance) could be eliminated. Additionally, we modify the workflow so implications are added as edges on the circuit's graph representation, and a traversal from primary output to primary input could contain normal nodes or implications in a search for the shortest input-to-output path. The updated workflow is shown in Figure 5.

We express the "distance" of an implication's graph edge as a negative-valued distance, to indicate it is a higher-priority and unconventional graph edge. The distance's weight is determined by its actual distance and activation probability of the implication. Figure 6 shows a visualization of this implication edge insertion.

Because the workflow now seeks only the shortest path and the implication's graph distances are a function of an implication's calculated efficacy, the shortest path is almost always implication-reinforced, en-

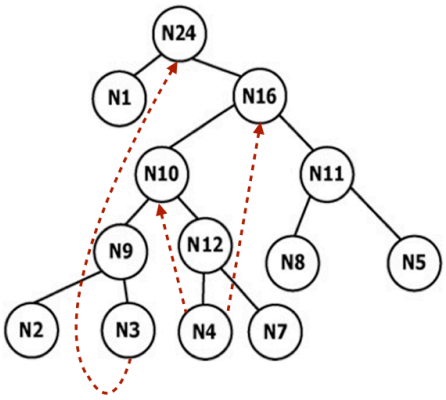

Figure 6: Updated graph of circuit with select implication edges added

Input: circuit.v, implication list
begin preliminary work
  Generate graph representation of circuit;
  Import all valid implications;
end
Add valid implications as graph nodes with weighted edges;
foreach *output in the circuit* **do**
  Build shortest-path chain from output to a primary input
end
Result: Input-output chains, implication set
Algorithm 2: Implication chain-building process

suring that the workflow works to reduce errors while working under power and area constraints.

3.4 Simulation

Simulations were conducted in Silvaco SmartSpice for 3.2μs, a simulation time chosen to accommodate the restrictive computation limits for longer simulations. To artificially accelerate the frequency of RTS glitches, which typically appear on the ms time scale, we conduct several simulations on different RTS noise samples [2]. Across the set of RTS noise simulations, we inject 150mV RTS noise and 80mV thermal noise while the inputs loop over a predefined Grey code input vector.

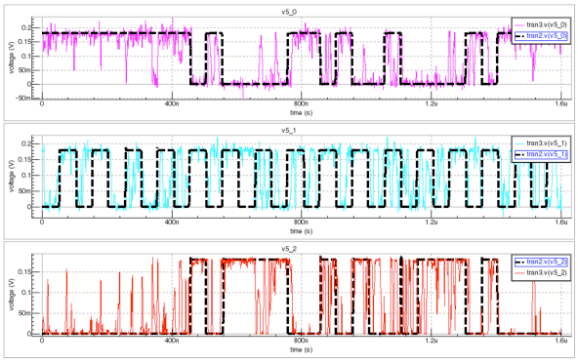

Figure 7: SPICE simulations of RD53, no implications added

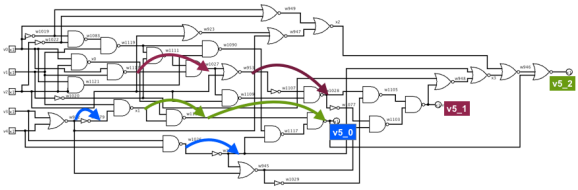

Figure 8: RD53 circuit with scattered HF-node reinforcement

4 Results

This section presents graphical implication sets and simulation results for the RD53 circuit, one of the simplest circuits in the MCNC benchmark. Simulations were also conducted on RD73, MISEX1, 5XP1, and T481, and, while not presented due to their complexity, results were fairly similar across all circuits.

We first present a simulation of the RD53 circuit with no implications and injected noise, (shown in Figure 7, as a demonstration of typical circuit output. The outputs are referred to sequentially, as $v5_0$, $v5_1$, and $v5_2$. Noise corruption can be visually noted in the signals overall, and can $v5_1$ especially demonstrates significant degradation. In implication-reinforced simulations, we gauge successful noise suppression and error reduction based on an increased adherence to the unadulterated reference output (shown in black at each output).

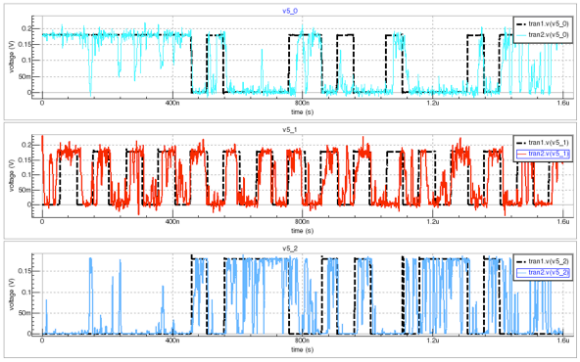

Figure 9: SPICE simulation of RD53 with HF-node reinforcement

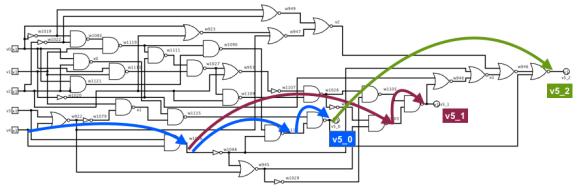

Figure 10: RD53 circuit with implication chains

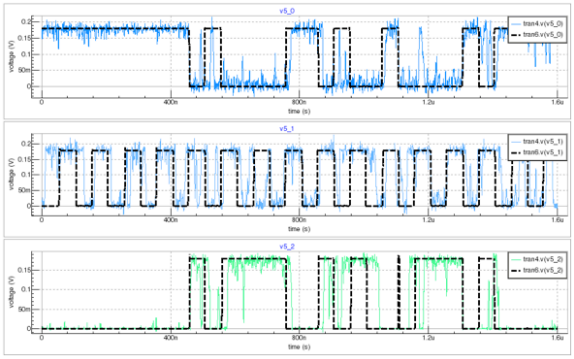

Figure 11: SPICE simulation of RD53 with implication chains

4.1 Scattered High-fault Node Reinforcement

A visual representation of scattered implication placement in the RD53 circuit is shown in Figure 8. The scattered implication set was generated under 20% node-coverage constraints and a weighted emphasis on implicant steadiness and high-fault outputs. Other trials were performed with other weight combinations for scattered reinforcement, with the best results observed under these conditions. We see from the simulation results in Figure 9 that general noise has been mitigated in $v5_0$ and $v5_1$, it appears to have increased in $v5_2$. This indicates the choice to scatter implications may not provide the best results. Additionally, examination of all the high-fault trials shows the best results were achieved under "chain-like" circumstances, where implications reinforce each other, further motivating the use of chains rather than scattering implications at points of high-failure rates.

4.2 Improved Implication Chains

A visual representation of implication chain placement in the RD53 circuit is shown in Figure 8. This set of implication chains provides significant noise suppression and error reduction across all outputs, especially outputs $v5_0$ and $v5_1$. The output $v5_2$ does demonstrate some glitching still, but the errors are significantly reduced from the original circuit and noise suppression is quite high within the first few nanoseconds. As noted earlier, these successful results are in keeping with analysis and our predictions of earlier trials.

5 Conclusions

In this work we discuss a cost-effective and successful solution for improving reliability in sub-threshold ULP circuits. Schmitt trigger circuits were chosen to implement an implication-based methodology for signal reinforcement and correction, and prior work demonstrated the effectiveness of this strategy. An automated tool synthesized reinforced circuits using two different algorithms, scattered high-fault node reinforcement and implication chain-building. Results demonstrate that implication chain-building had superior results to scattered HF-node reinforcement, and overall shows highly-effective noise suppression and error reduction characteristics.

6 Future Work

Future work will focus on extending the implication selection metric to be a bit more dynamic for circuits of varied complexities and overlap, and developing an automated metric for quantifying glitch detection in SPICE results. Though we have tested on five of the MCNC benchmarks, future work should also include

expanding simulations to a wider range of circuits as well as evaluating other noise-immune design solutions for completeness.

7 Acknowledgements

This report was compiled with guidance from Marco Donato, and would not be possible without the support of R. Iris Bahar, Brown University LEMS, and the DREU program. This work was generously funded by the DREU program, which is jointly administrated by the CRA-W and the CDC and has been supported by a grant from the NSF Broadening Participation in Computing program (NSF CNS-0540631).

References

[1] V. De and S. Borkar, "Technology and design challenges for low power and high performance [microprocessors]," in *Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on*, pp. 163–168, 1999.

[2] M. Donato, F. Cremona, W. Jin, R. I. Bahar, W. Patterson, A. Zaslavsky, and J. Mundy, "A noise-immune sub-threshold circuit design based on selective use of schmitt-trigger logic," in *Proceedings of the great lakes symposium on VLSI*, GLSVLSI '12, (New York, NY, USA), pp. 39–44, ACM, 2012.

[3] Y. Sasaki, K. Namba, and H. Ito, "Soft error masking circuit and latch using schmitt trigger circuit," in *Defect and Fault Tolerance in VLSI Systems, 2006. DFT '06. 21st IEEE International Symposium on*, pp. 327–335, 2006.

[4] K. Nepal, N. Alves, J. Dworak, and R. Bahar, "Using implications for online error detection," in *Test Conference, 2008. ITC 2008. IEEE International*, pp. 1–10, 2008.

[5] N. Alves, Y. Shi, J. Dworak, R. Bahar, and K. Nepal, "Enhancing online error detection through area-efficient multi-site implications," in *VLSI Test Symposium (VTS), 2011 IEEE 29th*, pp. 241–246, 2011.

[6] L. Moser, "Modeling optimal schmitt trigger based implication chains for noise immune sub-threshold circuits." Undergraduate honors thesis, Brown University, 2013.

[7] B. L. Dokic, "{CMOS} {NAND} and {NOR} schmitt circuits," *Microelectronics Journal*, vol. 27, no. 8, pp. 757 – 765, 1996.

[8] N. Alves, A. Buben, K. Nepal, J. Dworak, and R. Bahar, "A cost effective approach for online error detection using invariant relationships," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 5, pp. 788–801, 2010.