A Noise-immune Sub-threshold Circuit Design based on Selective Use of Schmitt-trigger Logic

Marco Donato*
School of Engineering
Brown University
Providence, RI 02912

Fabio Cremona
Sapienza University of Rome
00185, Italy

Warren Jin
School of Engineering
Brown University
Providence, RI 02912

R. Iris Bahar
School of Engineering
Brown University
Providence, RI 02912

William Patterson
School of Engineering
Brown University
Providence, RI 02912

Alexander Zaslavsky
School of Engineering
Brown University
Providence, RI 02912

Joseph Mundy
School of Engineering
Brown University
Providence, RI 02912

ABSTRACT

Nanoscale circuits operating at sub-threshold voltages are affected by growing impact of random telegraph signal (RTS) and thermal noise. Given the low operational voltages and subsequently lower noise margins, these noise phenomena are capable of changing the value of some of the nodes in the circuit, compromising the reliability of the computation. We propose a method for improving noise-tolerance by selectively applying feed-forward reinforcement to circuits based on use of existing invariant relationships. As reinforcement mechanism, we used a modification of the standard CMOS gates based on the Schmitt trigger circuit. SPICE simulations show our solution offers better noise immunity than both standard CMOS and fully reinforced circuits, with limited area and power overhead.

Categories and Subject Descriptors

B.2.3 [**Arithmetic and Logic Structures**]: Reliability, Testing, and Fault-Tolerance—*Redundant design*

General Terms

Design, Reliability

Keywords

Schmitt Trigger, Noise immunity, Subthreshold operation.

*Should you need further information, please contact Marco Donato (email: marco_donato@brown.edu).

GLSVLSI'12, May 3–4, 2012, Salt Lake City, Utah, USA.

1. INTRODUCTION

Modern CMOS integrated circuits have benefited from technology scaling in the nanoscale regime, which has allowed for faster circuits and increased levels of integration. However, miniaturization has also brought many challenges. First, as node capacitances shrink with smaller device dimensions, the RMS value of thermal noise also rises. Moreover, having a lower number of electrons in the channel amplifies the effect of fluctuations of the carriers due to random telegraph signal (RTS) noise. While the effects of these phenomena could be negligible for standard designs, they become critical for ultra low-power applications (ULP). One of the most appealing approaches for ULP design is operating the circuits in the sub-threshold regime. However, given the reduced noise margins of transistors operated in the sub-threshold regime, the magnitude and the frequency of soft errors could drastically reduce the reliability of the system. Therefore, it is critical to find viable solutions to improve the noise robustness of nanoscale sub-threshold circuits.

A number of techniques to improve noise-immunity of nanoscale circuits have been proposed, many of them based on adding redundant logic to the circuit (e.g., [17], [12], [6]). However, one of the main drawbacks of these solutions is the relatively high transistor count overhead required to achieve reasonable levels of noise immunity. Increasing the number of transistors leads to higher power dissipation and virtually eliminates any benefit obtained by operating the circuit in the sub-threshold regime. Therefore, new approaches to improve noise-immunity of nanoscale circuits are required to make sub-threshold operation worthwhile.

In this paper, we propose to selectively introduce redundant logic to the circuit, based on the use of existing invariant relationships (or *logic implications*) within the circuit itself. These invariant relationships represent expected logical behavior of the circuit and must be satisfied for the circuit to be operating correctly. For instance, the invariant relationship $a = 1 \Rightarrow b = 1$ implies that b (the *implicand*) must have a logic value of 1 whenever a (the *implicant*) has a logic value of 1.

Invariant relationships may be expressed through feed-forward reinforcements in the circuit. The basic idea of using implications to reinforce correct logical behavior is itself rather general, and therefore could be implemented at the circuit level in a number of

ways. One possible approach is to prevent the gate that produces the implicand from chenging its output value if this contradicts the invariant relationship. One way to accomplish this is to control the switching threshold of the gate using the gate that produces the implicant. A well known circuit that shows a similar behavior is the Schmitt trigger circuit [15].

A major advantage of using the Schmitt trigger circuits is their relatively low transistor count overhead. 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. Our SPICE simulations show that our approach has the lowest failure rate for minimum power and area overhead, proving that it is a viable solution for designing noise-immune circuits when rigid power and area constraints exist.

The rest of this paper is organized as follows. In Section 2 we will describe the motivation for our work and give some background on the building blocks of our approach. In Section 3 we first give an example explaining the importance of taking into account the effect of different noise sources in nanoscale circuits. We then present an approach for implementing the reinforcement using Schmitt-trigger based gates. In Section 4 we will describe in detail our time-domain noise simulations and give a quantitative analysis of the noise immunity of the various solutions, as well as an estimate of the introduced power and area overhead. In Section 5 we will introduce a design flow that will allow to automatically find implication paths that can be used to feed-forward reinforcement. Finally, we present conclusions and future work in Section 6.

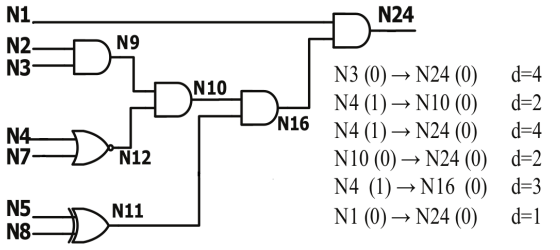

Figure 1: Example circuit with implications, taken from [1].

2. BACKGROUND

2.1 Noise-Immune Circuit Design

The basic concept of error-immune design is far from new. One of the earliest examples is the Modular Redundancy approach introduced by Von Neumann in [17] where the logic is replicated a certain number of times (e.g., three times in the case of Triple Modular Redundancy (TMR)) and all the outputs are then sent to a majority gate in order to get the final output. This technique introduces two main problems: first, the area overhead is always greater than 200% depending on how many times the main logic is replicated and the size of the majority gate; in addition, the error immunity is still determined by the majority gate that could be still affected by errors.

Other techniques use gate resizing for improving the immunity of selected gates to single event upsets (SEU) (see, for example [13]). These techniques could still be insufficient if the noise in the stages preceding the resized gate is strong enough to generate a signal flip, since the voltage at the input node of the resized gate would be already different from the correct value and there would be no means for correcting the signal. Therefore, there is a need for reinforcing techniques that can be effective across multiple gates.

A different approach has been considered in [12], where a probabilistic framework based on Markov Random Fields (MRF) is proposed for designing noise-immune circuits. In particular, the MRF framework used feedback of the satisfiability constraints of a gate's function to reinforce correct behavior. More recently, Turtle Logic (TL) has been proposed to achieve noise immunity by exploiting port redundancy and coherence analysis of the redundant data [6]. Both these approaches show a greatly improved immunity to noise compared to standard implementations; however, they carry significant area overhead which can have a substantial negative impact on power dissipation and delay. These factors may significantly limit their suitability for sub-threshold circuit design, where performance is already traded off for reduced power dissipation.

2.2 Implications

Exploiting knowledge of invariant relationships (or implications) is a simple yet useful concept that has already found application in error detection and logic testing methodologies [1] [2] [11]. Implications are logic relationships between nodes in the circuit that hold for any input vector. An example is illustrated in Figure 1, where six different implications are enumerated.

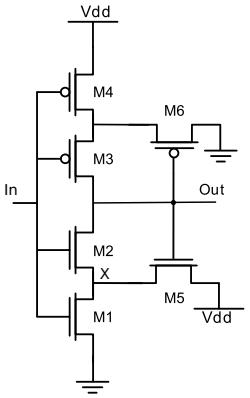

Figure 2: Schmitt trigger circuit implementing an inverter function.

If we consider for instance the first line in the implications list, it states that for any input vector for which $N3$ (the *implicant*) is equal to 0, the output $N24$ (the *implicand*) will have to be at 0 as well. The same reasoning applies to all the other implication relationships shown in Figure 1. In the works of [1] [2] [11], the authors used implications for online error detection by adding redundant circuitry (i.e., checker logic) to verify if selected implication relationships were valid. If there were an inconsistency in the checker logic, an error would be flagged.

Taking the ideas from [1], we could imagine using implication logic not just to detect errors but also to reinforce correct logic behavior. As a simple application of these implications, we could imagine using them to monitor the errors on certain paths to the primary output and, if an error is detected, flip the output signal. However, this solution would require adding not only the logic for checking the implications but also a multiplexer for selecting between the direct or inverted output. Also, the main problem would be that the multiplexing operation at the last stage would have the same issues as the voting system for the modular redundancy design. As we will show shortly, we can obtain solutions with less overhead and better noise immunity than this simple approach.

In this paper, we propose to use implicants as control signals for

the gates that produce the implicands. If we consider again the circuit in Figure 1 with its implication $N3 = 0 \Rightarrow N24 = 0$, we can use the value at node $N3$ to force the value at node $N24$ through a feed-forward mechanism. Results from [1] show that thousands of these relationships exist in standard benchmark logic circuits. Therefore, we can choose a subset of these implications and use them to selectively reinforce the circuit. Depending on the architecture that is used for implementing the feed-forward system, this approach can lead to notable error-rate reduction with very modest area, delay and power overhead.

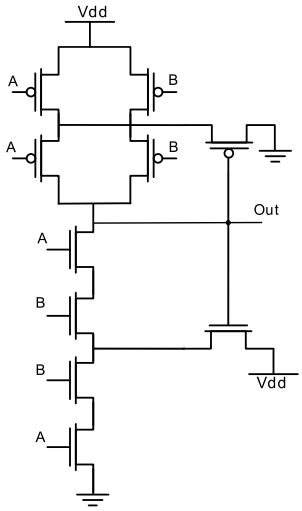

Figure 3: Schmitt trigger based NAND gate.

2.3 Schmitt Trigger Circuits

We introduced the use of implications for increasing the reliability of a circuit as a general approach. However, we need to find an architecture for implementing the feed-forwarding reinforcement. The Schmitt trigger circuit [15] is generally used to extract signals from noisy environments and can serve our purpose. In particular, we refer to the implementation shown in Figure 2. We can describe its operation as follows: let us consider the case in which the input of the trigger is at logic value 0 and the output is at 1. The NMOS pass transistor $M5$ is on, increasing the potential at node x; if the gate voltage moves from its initial value, it will have to overcome the regular switching threshold voltage of the NMOS transistors in the pull-down network (PDN) by the value at x. The same behavior can be seen for the pull-up network (PUN) and the two structures combined produce the well known hysteresis characteristic of the Schmitt trigger. This circuit has been demonstrated to be effective also in subthreshold design [9]. Moreover, Dokic has shown in [5] the possibility of extending the same design structure of the Schmitt trigger to more general logic gates such as NAND and NOR. The Schmitt implementation of a NAND function is shown in Figure 3. In all these cases the reinforcement is introduced using a feed-back mechanism. Our first modification to the circuit consists in connecting the gates of the pass transistors to a node different from the gate's output. This modification provides the means for implementing the implication reinforcement. Let us consider the example in which the implication relationship is $Ctrl = 0 \Rightarrow Out = 1$, where $Ctrl$ could be any node in the circuit and Out is the output of a NAND and an implicand related to the implicant $Ctrl$. Note that this kind of implication is unidirectional and valid only for one of the two logic levels. This requires a further modification of the Schmitt gate structure; in this particular case we will connect the gate of the pass-transistor (PMOS) to the node $Ctrl$, and the drain of the pass-transistor will be connected to the middle node of the PDN. In this way, whenever the signal $Ctrl$ is 0, the gate will be prevented from performing the transition 1 to 0. The resulting circuit is shown in Figure 4. In the general case, the implicant will determine the type of pass-transistor (NMOS if the implicant is 1 and PMOS otherwise) while the implicand will determine to which network (pull-up or pull-down) the reinforcement will be applied, leading to four possible Schmitt gates in total.

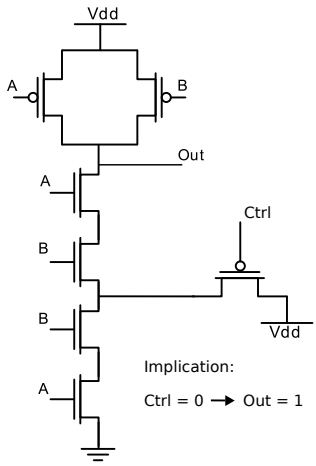

Figure 4: A modified Schmitt NAND for implication reinforcement.

3. METHODOLOGY

In this section we will show an example of how the performance of a circuit operating at sub-threshold supply voltage can be dramatically affected by noise. We choose to simulate our circuit using a 22nm FDSOI model card [3]. The importance of both thermal noise and random telegraph signal (RTS) noise for nanoscale circuits has been widely discussed [14] [8] [4] [16]. We therefore consider both sources of noise in our simulations to evaluation signal integrity.

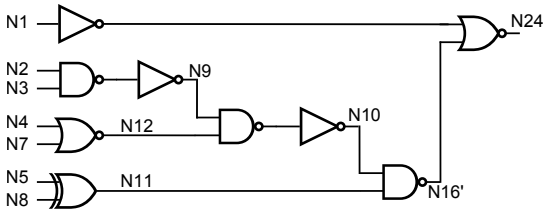

Figure 5: Implemented version of the example circuit shown in Figure 1.

3.1 Motivational Example

Let us consider again the circuit in Figure 1. The slightly modified version used for our SPICE simulations is shown in Figure 5. In order to add noise to our simulations we considered the methods described in [10] and [19]. In particular, we generated additive white Gaussian noise (AWGN) with 0 mean and $10mV$ standard deviation (a reasonable value for a 22nm technology node), RTS

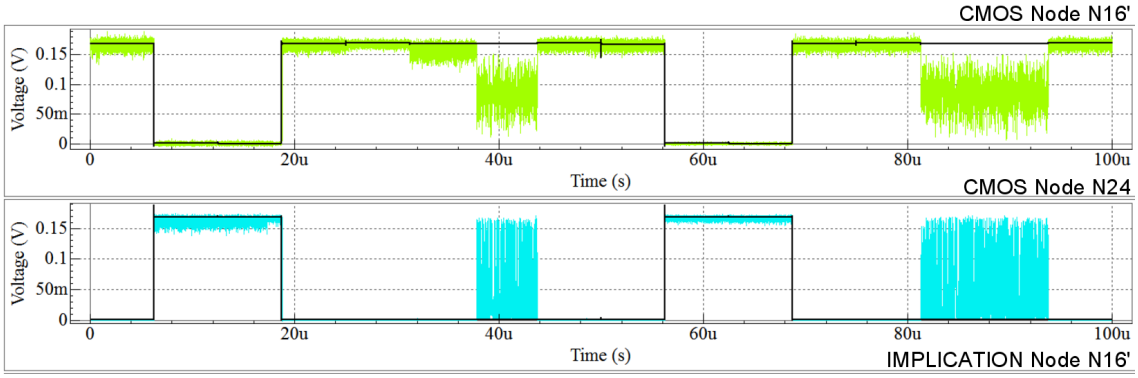

Figure 6: Example circuit's output and input to the last stage for standard CMOS implementation

noise with values of the amplitude equal to $50mV$, and we considered the Fast Slow corner library that represents the worst-case scenario. For nanoscale technologies it is very hard to control the process variations and produce a circuit following the nominal behavior, therefore we think that considering the corner libraries is the most realistic condition. The circuit was simulated using a supply voltage of $V_{DD} = 170mV$, i.e., below the threshold voltage of the devices. The signal traces from the SPICE simulations are shown in Figure 6. It is clear from this figure that the circuit is unable to perform a correct computation under the imposed noisy conditions.

In order to explain how our solution works we will empirically describe the approach. We will use the implication set listed in Figure 1 for this purpose. Since we are interested in producing the correct signal at the output, we will first consider all the implications to the output node $N24$ and then try to build a path of implications through the entire circuit. Therefore, we have the following set of implications to start with: $N3 = 0 \Rightarrow N24 = 0$, $N4 = 1 \Rightarrow N24 = 0$, $N10 = 0 \Rightarrow N24 = 0$, $N1 = 0 \Rightarrow N24 = 0$. We can immediately discard the implication with $N1$ since it is a primary input and therefore it does not provide any means for reinforcing internal nodes. The same reasoning applies to $N3$ and $N4$, therefore our choice is the implication $N10 = 0 \Rightarrow N24 = 0$. At this point we need to find an implication that has $N10$ as its implicand and a node closer to the primary inputs as implicant. In our example we will pick $N4 = 1 \Rightarrow N10 = 0$. In this way we have created a reinforcement path to the primary output where all the nodes involved in the process are reinforced along the way. The next step is modifying the circuit according to the chosen implications. The gate that produces $N24$ will have a reinforced PUN controlled by a PMOS (since the implicant is 0), while the gate that produces $N10$ will have a reinforced PUN controlled this time by an NMOS (implicant 1). The total overhead required by this solution is 6 additional transistors.

As mentioned earlier, we built a library based on a 22nm FDSOI model of standard CMOS gates and modified Schmitt gates. Considering that all devices are operating in the subthreshold regime and that the Schmitt design requires doubling the number of transistors in a stack, we decided to limit our library to only two-input gates. With regard to the implementation of a 2-input XOR gate, the authors of [18] have shown that degradation of the output signal can arise in a transmission-gate based XOR circuit due to the imbalance between the on current and the leakage current. For this reason, we decided to use the solution proposed in [7] which does not require additional inverters for generating complementary inputs and has also shown a better noise immunity when compared to other design solutions. Note however, that nothing prevents from

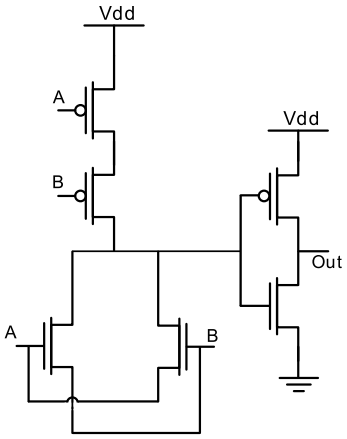

Figure 7: XOR gate adopted for the library

using any other XOR design with our approach. The adopted XOR circuit is shown in Figure 7.

4. SIMULATIONS

In this section we show the results and describe the details of the SPICE noise simulations. Let us compare the simulations for the example circuit of Figure 5 implemented in standard CMOS as well as with the implication reinforcement discussed in the previous section. The results are shown in Figure 8. The superimposed black trace is the result for the noiseless simulation of the CMOS circuit. Notice that, when simulated with additional noise, the standard CMOS implementation produces unacceptably noisy signals, whereas our implication reinforced circuit reduces noise on the output signal significantly. It is also worth noticing that in both cases the noise at node $N16'$ is the same meaning that the Schmitt gate is actually correcting the errors on the signal.

Next, we also need to compare against other reinforcement approaches. Several works on circuit reliability and noise immunity have focused their attention on strengthening just the memory elements in the circuit. One way to investigate how this approach would work would be to reinforce just the final stage of the circuit (i.e., the stage feeding directly into the memory element) with a *standard* Schmitt gate (i.e., the one shown in Figure 2). Another approach would be to fully reinforce the circuit by substituting every CMOS gate with an equivalent Schmitt gate. The results for these two solutions are shown in Figure 9. As seen from the waveforms in Figure 9, the first approach of reinforcing only the final output is not robust enough. While the second approach of fully re-

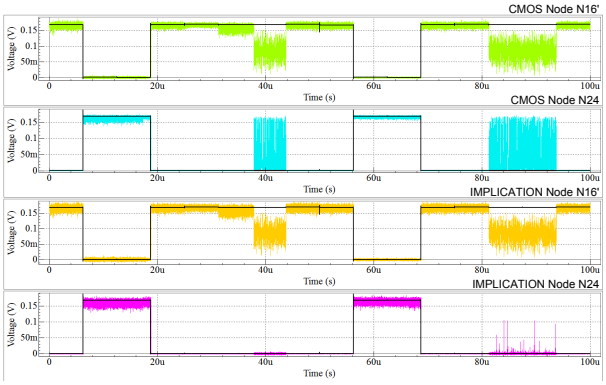

Figure 8: Comparison between standard CMOS implementation and implication/reinforced implementation.

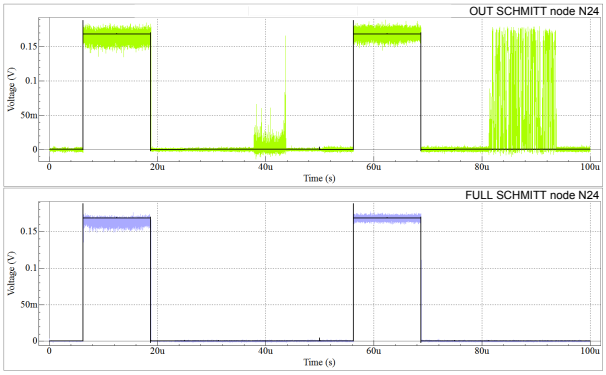

Figure 9: Example circuit's output for full Schmitt and standard CMOS with Schmitt gate on the last stage

inforcing every gate in the circuit produces a clean output, its cost in terms of transistor count is more than twice that of the initial CMOS design. Hence, while our implication approach still leaves some noise on the output, it provides a much better tradeoff between noise immunity and area overhead.

The main challenge in carrying out time-domain noise simulations is the computation time. Considering the slow variability of RTS noise, with RTS glitches occuring on the ms time scale, direct time domain simulation is prohibitively long. Therefore, to better evaluate the performance of our solution, we decided to generate several samples of the injected RTS noise and perform 100 distinct simulations on a time of 100 μsec, thereby artificially increasing the frequency of the RTS noise. On each of the 100 simulations we made the inputs loop over a predefined test vector. We then considered each iteration of the loop as the time window for the error rate: if the signal crosses the error threshold for more than 10% of the time, we record an upset; this helps filtering out small glitches and transitions.

For the tested circuit we used 2 implications, producing an overhead of 18.75% in number of transistors and 24% in power consumption. With this cost we were able to reduce the failure rate from 6.5% to 0%. It is important to mention that in the case of many upsets producing failure at the primary output we count just a single error. Therefore, the failure rate does not represent the number of total errors registered in the simulations. While the CMOS implementation shows an unacceptable failure rate, the implication

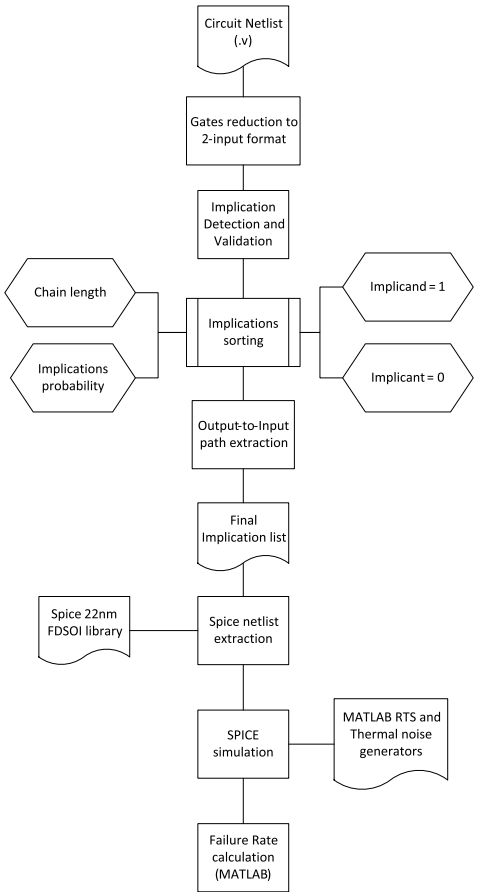

Figure 10: Detailed design flow chart

design offers a faultless output, with reasonable area and power overhead compared to full logic replication.

5. DESIGN FLOW

Following the idea described in Section 3, we will now present a potential work flow that can be used for automatically inserting the implication reinforcement into a standard CMOS design. First, we need a method for identifying the implications given a circuit netlist. For this purpose we can partially re-use the flow shown in [1]. It is important to clarify that at this stage we are interested only in the implication discovery process, while the process of trimming the list trying to get rid of *weak* implications might actually hinder our purpose. In particular, the authors of [1] consider as *weak*, those implications that can be included in other implications and implications where the implicant and the implicand can be derived from a common node; while these implications can be discarded for online error detection, they might be critical for finding a path between the primary outputs and the primary inputs. Therefore, the design flow we envision is illustrated in Figure 10 and can be described as follows:

- **Implication discovery and verification:** in this initial part, a logic simulation of the circuit over a limited number of input vectors is carried out. The logic values at each node in the circuit are then used as input to a script that verifies the presence of possible implications. While the previous step produces a list of possible implications, these values have

been tested only on a limited number of vectors. Therefore, we need to to check for the presence of input patterns that would violate possible implications values. This can be done implicitly using, for instance, a SAT solver.

- **Implication sorting:** once we have a verified list of implications we need to understand which implications will be more suitable for reinforcement. The best implication for our purpose is the one that holds for most of the input patterns. Once again, a logic simulation can be used for extrapolating the implications' frequency.

- **Output-to-input path extraction:** Once we have a score assigned to any implication in the list we, can build our reinforcing paths starting from those implications that have primary outputs as implicands, and then move backwards to the primary inputs considering the implications with higher score first. In this step, a threshold for the chain length could be used to set the maximum area overhead.

- **SPICE netlist extraction and simulation:** The final implication list is used in combination with the verilog netlist to extract a SPICE netlist in which the Schmitt gates are automatically inserted. This netlist is then used for circuit simulations.

6. CONCLUSIONS

In this work we have shown a very cost-effective solution for improving the reliability of sub-threshold nanoscale circuits. The implication methodology represents a general framework for selective reinforcement. We decided to use Schmitt trigger based circuits for implementing the implication reinforcement, and we have demonstrated that this solution offers good noise immunity with a limited cost in terms of area and power overhead. Future work will focus on implementing the design flow described in Section 5 for automating the selection of implications and the insertion of the reinforced gates. We will also expand the simulations to a wider range of circuits as well as to other noise-immune design solutions.

7. ACKNOWLEDGMENTS

This work was supported in part by DTRA under Grant HDTRA 1-10-1-0013.

8. REFERENCES

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

[2] N. Alves, Y. Shi, J. Dworak, R. I. Bahar, and K. Nepal. Enhancing online error detection through area-efficient multi-site implications. In *29th VLSI Test Symposium*, pages 241–246. IEEE, May 2011.

[3] D. Bol, S. Bernard, and D. Flandre. Pre-silicon 22/20 nm compact MOSFET models for bulk vs. FD SOI low-power circuit benchmarks. In *IEEE 2011 International SOI Conference*, pages 1–2. IEEE, Oct. 2011.

[4] L. Brusamarello, G. I. Wirth, and R. da Silva. Statistical RTS model for digital circuits. *Microelectronics Reliability*, 49(9-11):1064–1069, Sept. 2009.

[5] B. Dokic. CMOS NAND and NOR Schmitt circuits. *Microelectronics Journal*, 27(8):757–765, Nov. 1996.

[6] L. García-Leyva, D. Andrade, S. Gómez, A. Calomarde, F. Moll, and A. Rubio. New redundant logic design concept for high noise and low voltage scenarios. *Microelectronics Journal*, 42(12):1359–1369, Dec. 2011.

[7] W. Jyh-Ming, F. Sung-Chuan, and F. Wu-Shiung. New efficient designs for XOR and XNOR functions on the transistor level. *IEEE Journal of Solid-State Circuits*, 29(7):780–786, July 1994.

[8] L. B. Kish. End of Moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305(3-4):144–149, Dec. 2002.

[9] J. P. Kulkarni, K. Kim, and K. Roy. A 160 mV, fully differential, robust schmitt trigger based sub-threshold SRAM. In *Proceedings of the 2007 international symposium on Low power electronics and design - ISLPED '07*, pages 171–176, New York, New York, USA, 2007. ACM Press.

[10] T. Lee and G. Cho. Monte Carlo based time-domain Hspice noise simulation for CSA-CRRC circuit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 505(1-2):328–333, June 2003.

[11] K. Nepal, N. Alves, J. Dworak, and R. Bahar. Using Implications for Online Error Detection. In *2008 IEEE International Test Conference*, pages 1–10. IEEE, Oct. 2008.

[12] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, and A. Zaslavsky. Designing Nanoscale Logic Circuits Based on Markov Random Fields. *Journal of Electronic Testing*, 23(2-3):255–266, Mar. 2007.

[13] R. Rao, D. Blaauw, and D. Sylvester. Soft Error Reduction in Combinational Logic Using Gate Resizing and Flipflop Selection. In *2006 IEEE/ACM International Conference on Computer Aided Design*, pages 502–509. IEEE, Nov. 2006.

[14] R. Sarpeshkar, T. Delbruck, and C. Mead. White noise in MOS transistors and resistors. *IEEE Circuits and Devices Magazine*, 9(6):23–29, 1993.

[15] O. H. Schmitt. A thermionic trigger. *Journal of Scientific Instruments*, 15(1):24–26, Jan. 1938.

[16] N. Tega, H. Miki, Z. Ren, C. P. D'Emic, Y. Zhu, D. J. Frank, J. Cai, M. A. Guillorn, D.-G. Park, W. Haensch, and K. Torii. Reduction of random telegraph noise in High-Đž / metal-gate stacks for 22 nm generation FETs. In *2009 IEEE International Electron Devices Meeting (IEDM)*, pages 1–4. IEEE, Dec. 2009.

[17] J. Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components, 1956.

[18] A. Wang and A. Chandrakasan. A 180-mV subthreshold FFT processor using a minimum energy design methodology. *IEEE Journal of Solid-State Circuits*, 40(1):310–319, Jan. 2005.

[19] Y. Ye, C.-C. Wang, and Y. Cao. Simulation of random telegraph Noise with 2-stage equivalent circuit. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 709–713. IEEE, Nov. 2010.