Analysis and Mitigation of Multiple Radiation Induced Errors in Modern Circuits

by

Adam Watkins

M.Sc., Electrical & Computer Engineering, Southern Illinois University, 2012

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the
Doctor of Philosophy Degree

Department of Electrical and Computer Engineering
in the Graduate School
Southern Illinois University Carbondale
October, 2016

DISSERTATION APPROVAL

ANALYSIS AND MITIGATION OF MULTIPLE RADIATION INDUCED ERRORS IN MODERN CIRCUITS

By

Adam Watkins

A Dissertation Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in the field of Electrical and Computer Engineering

Approved by:

Dr. Spyros Tragoudas, Chair

Dr. Heather Quinn

Dr. Themistoklis Haniotakis

Dr. Haibo Wang

Dr. Chao Lu

Graduate School
Southern Illinois University Carbondale
(Date of Approval)

AN ABSTRACT OF THE DISSERTATION OF

Adam Watkins, for the Doctor of Philosophy degree in Electrical and Computer Engineering, presented on November 2nd, at Southern Illinois University Carbondale. (Do not use abbreviations.)

TITLE: ANALYSIS AND MITIGATION OF MULTIPLE RADIATION INDUCED ERRORS IN MODERN CIRCUITS

MAJOR PROFESSOR: Dr. S. Tragoudas

Due to technology scaling, the probability of a high energy radiation particle striking multiple transistors has continued to increase. This, in turn has created a need for new circuit designs that are can tolerate multiple simultaneous errors. A common type of error in memory elements is the double node upset (DNU) which has continued to become more common. All existing DNU tolerant designs either suffer from high area and performance overhead, may lose the data stored in the element or are vulnerable to an error after a DNU occurs which makes the devices unsuitable for clock gating. In this dissertation, a novel latch design is proposed in which all nodes are capable for fully recovering their correct value after a single or double node upset which is referred to as DNU robust. The proposed latch offers lower delay, power consumption and area requirements compared to existing DNU robust designs.

Multiple simultaneous radiation induced errors are a current problem that must be studied in combinational logic. Typically, simulators are used early in the design phase which use a netlist and rudimentary information of the process parameters to determine the error rate of a circuit. Existing simulators are able to accurately determine the effects when the problem space is limited to one simultaneous error. However, existing methods do not provide accurate information when multiple concurrent errors occur due to inaccurate

approximation of the glitch shape when multiple errors meet at a gate. To improve existing error simulation, a novel analytical methodology to accurate determine the pulse shape when two simultaneous errors occur is proposed. Through extensive simulations, it was shown that the proposed methodology matches closely with HSPICE while providing a speedup of 15X.

The analysis of the soft error rate of a circuit has continued to be a difficult problem due to the calculation of the logical effect on a pulse generated by a radiation particle. The most common existing methods to determine logical effects use either exhaustive input pattern simulation or binary decision diagrams. The problem with both approaches is that simulation of the circuit can intractably time consuming. To solve this issue, a simulation tool is proposed which employs an adaptive partitioning algorithm to reduce the simulation time and space overheads of binary decision diagram based simulation. Compared to existing simulation tools, the proposed tool can simulate larger circuits faster.

DEDICATION

This dissertation is dedicated to my parents Daniel and Debra Burris and my wife Yijing Zhang Watkins for their support during the creation of this work.

ACKNOWLEDGMENTS

(NOT REQUIRED IN RESEARCH PAPER)

I would like to thank Dr. Jones for his invaluable assistance and insights leading to the writing of this paper. My sincere thanks also goes to the seventeen members of my graduate committee for their patience and understanding during the nine years of effort that went into the production of this paper.

A special thanks also to Howard Anton, from whose book many of the examples used in this sample research paper have been quoted. Another special thanks to Prof. Ronald Grimmer who provided the previous thesis template upon which much of this is based and for help with graphics packages.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The continued reduction of the transistor feature size has led to the increase in the vulnerability of modern circuits to error. This effect, in turn, has made modern circuits more susceptible to radiation induced errors in space and terrestrial environments. A radiation induced error, commonly referred to as a soft error, occurs when a high energy particle from space or packaging strikes a transistor. Shown in Fig. 1, the particle deposits energy in the active volume generating electron-hole pairs. This creates a new diffusion region that could allow a non-conducting device to temporarily conduct current. The mechanism causes a temporary voltage pulse, referred to as a single event transient (SET) to occur at the device.

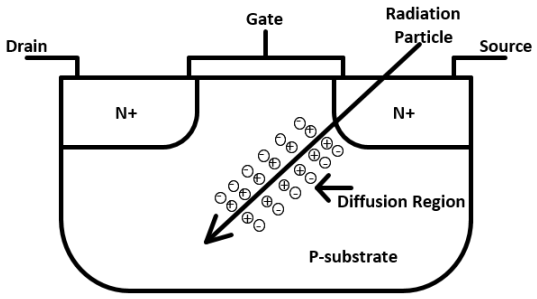


Figure 1: A energetic particle creating electron-hole pairs in a transistor.

In the case of a radiation particle striking a memory storage element, the data stored in the device can be changed. If the device loses its value due to a strike by a single particle, this is referred to as a single event upset (SEU). Additionally, due to the constant scaling down of the feature size, a single particle can also strike two transistors simultaneously which is referred to as a double node upset (DNU). Fig. 2 gives a diagram of a SEU tolerant latch to demonstrate the DNU phenomenon. In the diagram, radiation is denoted as a high energy particle which passes through two transistors on a DICE latch [3]. In the case of a SEU, the DICE latch would normally be able to tolerate the error. As can be observed, the particle passes through two transistors causing the respective node to switch

from "1" on the first node to "0" and from "0" to "1" on the second node. The upset of both nodes drives the remaining two nodes to an erroneous value. This observance is alarming since it signals that current designs are not sufficient for future processes. In turn, this provides a need for new memory element designs that can tolerate DNUs.

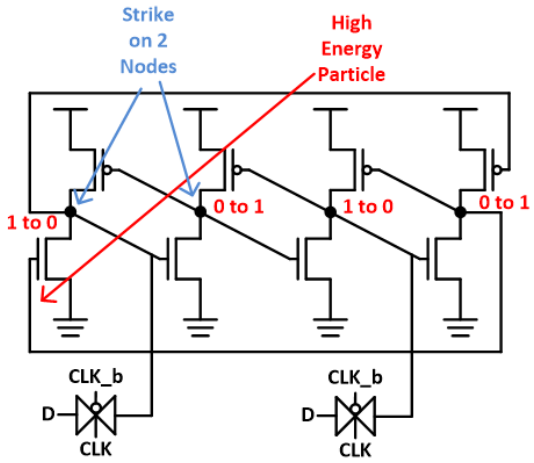


Figure 2: A particle striking two node concurrently causing a double node upset.

Additionally, modern circuit designs employ a technique referred to as clock gating. Clock gating is defined as setting the clock to constant value thus reducing the power consumed. Use of clock gating may lead to cases where the current state held by the memory element must be held for many clock cycles. This need leads to a much longer than normal vulnerable window in which an improperly hardened device may lose the stored value. While there are existing DNU tolerant designs [15, 16, 34], these designs move to a high impedance state after a DNU. If a DNU occurs while the latch is gated, the voltages within the latch may degrade. In Chapter 1, we aim to alleviate this problem by providing an efficient design that is capable of recovering all nodes after a DNU occurs. This, in effect, guarantees that the data will be held even if the latch is struck by a DNU while gated.

In addition to the effects on memory elements, accurate approximation of the error rate of a circuit is also important. The goal of circuit simulation is the accurate estimation of the soft error rate (SER). In this work, the focus is on the evaluation of combinational

circuits. Typically, this type of simulation entails the estimation of the resulting pulse shape when a particle strike a transistor, the determination of the boolean functions that allow the pulse to propagate and the estimation of the latching characteristics at an output flip-flop. Using these parameters, the probability of the pulse reaching an output flip-flop is determined. Assume that the probability of an error reaching and being latched in a flip-flop at output i of the circuit is represented as $P(O_i)$, the particle hit rate in a particular area is given as R_{PH}, the fraction of particle hits resulting in charge generation as R_{eff} and A_{cir} gives the area of the circuit. The equation for the SER at output O_i is given below [21].

$$SER_{O_i} = P(O_i) * R_{eff} * R_{PH} * A_{cir} \tag{1}$$

The focus of all SER estimation simulators is the calculation of the term $P(O_i)$. This is a difficult problem in combination circuits due to the presence of the following three masking factors: electrical masking, logical masking and temporal masking. Electrical masking deals with the calculation of the pulse shape as it propagates through the circuit. Logical masking is the estimation of the logical 1's and 0's and how they may mask the pulse. For example, if a NAND gate has a value of "0" on an input, the pulse will not propagate since the output will be held to a "1". Lastly, temporal masking involves the latching characteristics, specifically the set-up and hold times, of the output flip flop. Accurate consideration of all three masking effects is crucial to accurate SER estimation.

Accurate consideration of the electrical masking effect is an important but often simplified component of SER estimation. The most common method to calculate the pulse shape, proposed in [26] uses a linear line to approximate the rising and falling transitions giving a trapezoidal shape. While this method executes quickly and is easy to implement, it does not provide an accurate estimation of the pulse shape. As can be observed in Fig. [ref], a transient pulse does not take a trapezoidal shape in a real case. For this reason,

accurate consideration of the pulse shape must consider the non-linear aspects of the pulse. In [32], the authors proposed an enhanced pulse approximation method which is accurate within 5% of HSPICE. However, their method has the drawback that it can only consider a single pulse arriving at a gate. In the case of multiple transient pulses injected into a circuit, referred to as a multiple event transient (MET), the likelihood of two or more pulses arriving at a gate simultaneously as shown in Fig. 4 is substantially increased. To accommodate this trend, future soft error simulators must accurately consider this effect.

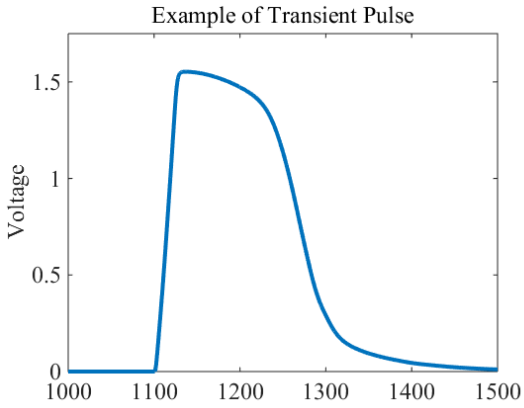


Figure 3: An example of a radiation induced transient pulse.

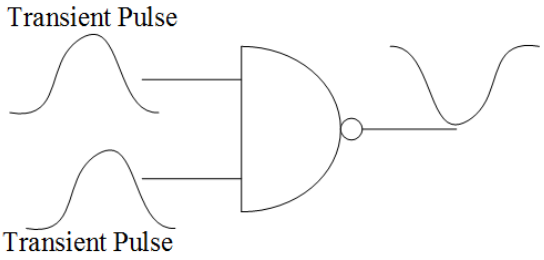


Figure 4: Two pulses arriving at a gate input.

In addition to the electrical masking effect, the logical masking effect must also be accurately considered. Many existing methods use Monte Carlo based simulation which consists of either exhaustively apply all patterns or randomly applying a subset of patterns. [32, 30, 23, 37, 29] While these methods are able to simulate with a very low memory overhead, they have an intractable execution time for circuits with more than 30 inputs. This problem is further compounded by the fact that due to the nearly infinite number of possi-

ble pulse durations and strike locations. To alleviate this issue, the authors in [38] propose the use of probabilistic transfer matricies [PTM]. While the matricies do provide exact calculation of the logical masking effect, their memory usage blows up even for relatively small circuits. As an improvement to the use of PTMs, the use of decision diagrams have become more common. [35, 20] Decision diagrams offer an improved approach since they balance the cost of execution time and memory consumption. However, decision diagrams are not a perfect solutions since, like PTMs, blow up if the circuit is sufficiently large. To solve this issue, the authors in [35] propose the use of partitioning to limit the size of the decision diagrams. The drawback with this approach is that each time the circuit is partitioned, additional error is added to the result. To ensure that the effect on the result is low, the number of partitions should be minimized based on the system used and amount of time allowed.

In this dissertation, novel approaches to the aforementioned problems are proposed. In Chapter 1, a latch design is proposed which is capable of recovering all nodes after a DNU. This design has specific applications in circuits that employ clock gating. Chapter 2 provides an enhanced analytical pulse approximation algorithm which can determine the output pulse shape when two pulses arrive at a gate input. The method is implemented and compared to existing algorithms and HSPICE. Chapter 3 discusses a soft error simulator which adaptively partitions the circuit based on the size of the BDD functions. Results are provided which compare the effect of the partitioning algorithm on the calculation of the SER.

CHAPTER 1

ROBUST LATCH DESIGNS FOR MULTIPLE NODE UPSETS

As the transistor feature side continuously scales down to improve performance, modern circuitry continues to become more susceptible to radiation induced errors commonly referred to as a soft error. Soft errors can manifest from either neutron particles originating from space or alpha particles from packaging. A soft error occurs when an energetic particle hits the diffusion region of a reverse bias transistor. This, in turn, allows an "off" transistor to temporarily conduct current which can cause a voltage change in a node connected to the affected transistor. If the error occurs in combinational logic, the resulting voltage pulse may be stored in a connected flip flop thus causing an error. On the other hand, if the error occurs in memory or a latch during the hold phase, the stored data may change. To mitigate this effect, there is a need for design methodologies that reduce the vulnerability of circuitry to radiation effects.

In this chapter, the focus is on the reliability of latches. There has been extensive research in the field of hardening latches against single even upsets (SEU). The simplest and most common design in safety critical applications is the triple modular redundancy (TMR) latch. This design consists of 3 standard latches connected to a 3-input majority voting circuit. While this design is tolerant against errors, it has the drawback of high area, delay and power consumption. For this reason there have been many other designs proposed that offer high SEU reliability with lower area, delay and power consumption. The first and most common cell is the DICE cell proposed in [3]. The design in [3] consists of eight cross-coupled PMOS and NMOS transistors connected in series which forms four nodes. Due to the relatively high delay and power consumption of the DICE latch, there have been many other SEU tolerant latch designs proposed that provide reliability using blocking Muller C-elements, redundancy or delay in the feedback path [27, 10, 14, 28, 18, 36, 24].

In more recent times, the further reduction of the transistor feature size has increase

the likelihood of a single event causing a transient on multiple nodes simultaneously, commonly referred to as a single event multiple upset (SEMU). This trend necessitates the development of new latch designs that are tolerant to multiple node strikes to guarantee reliability in current and future technologies. As in the SEU case, the goal of these designs are to minimize the power, delay and area overheads. However, contrary to the SEU case, the latches are designed to tolerate two simultaneous errors, commonly referred to as a double node upset (DNU). Currently there are many existing latch designs that are tolerant to DNUs which are discussed in Section 1.1.

Many modern circuit designs employ a technique commonly referred to as clock gating to further reduce the power consumption. Clock gating consists of setting the clock to a stable value or "gating" the clock. If clock gating is used with a latch, it may need to hold the current state for many clock cycles. In the presence of DNUs, this increases the likelihood of multiple errors occurring during the hold phase. In many existing DNU tolerant designs, a DNU puts the latch to a high impedance state in which the correct value could be lost if the latch experiences a further SEU or DNU before the transparent mode. Additionally, in many of these designs, a DNU moves the output to a high impedance state which implies that the data could discharge if the latch is gated for a sufficient number of cycles. For this reason, there is a need for new designs that are capable of holding the correct output value after a DNU for any number of clock cycles. In the chapter, we classify all DNU tolerant designs as either DNU robust or DNU non-robust. A DNU robust design is defined as being capable of resisting further errors and by not allowing any high impedance states after a DNU occurs. A DNU non-robust design is a latch that does not meet the all of stated criteria. In this chapter, a novel latch design referred to as the HRDNUT (Highly Robust Double Node Upset Tolerant) is proposed.

1.1 BACKGROUND ON EXISTING DNU TOLERANT LATCHES

Currently, there are a few existing DNU tolerant designs. The first proposed design found in [16], referred to as the DNCS latch, consists of two DICE cells connected to an output Muller C-element. This design tolerates DNU's since each DICE element requires a DNU to flip its state. Since the assumption is that only two errors can occur at once, in the worst case only one DICE element flips its state. Due to the C-element, the latch output does not change value. This design has been shown to be very resilient to DNUs at a very high cost of area, delay and power. The authors in [15] propose an enhanced design compared to [16]. Their latch design consists of six 2 input C-elements connected in series which are then fed into a 3 input C-element. Like the DNCS latch, this design offers high resiliency to DNUs, however the power consumption and area overheads are still very high.

More recently, a highly area and power efficient design has been proposed in [34] and is referred to as the HSMUF latch. Fig. 1.1 provides the design. The HSMUF uses the TP-DICE [1] structure which consists of 6 cross-coupled elements. In the case of a DNU, if the error is on an adjacent node (such as a strike on n1 and n2), the TP-DICE element will fully recover the previous state. However, if the strike occurs on two non-adjacent nodes, the TP-DICE will not fully recover leaving one output node with an erroneous value, one node at high impedance and the remaining output node held at the error-free value. To provide reliability, the three nodes are connected to a C-element, as in Fig. 1.1, which allows the correct value to be held at the latch output.

While all of the previously discussed designs do provide high DNU reliability, none of them are classified as DNU robust since a DNU will result in high impedance states on the internal and output nodes. If an error occurs after a DNU, these latch designs will flip their held value. A popular remedy to this issue is to place a weak keeper on the latch output as in Fig. 1.1. However, adding a weak keeper greatly increases the power, area and delay overheads since the output C-element must be re-sized so that the C-element's driving strength exceeds that of the keeper. According to our simulations in Section 1.3,

the addition of the keeper to the HSMUF latch nearly triples the power consumption and delay. Additionally, the latch is still vulnerable to error after a DNU since the TP-DICE is in a high impedance state.

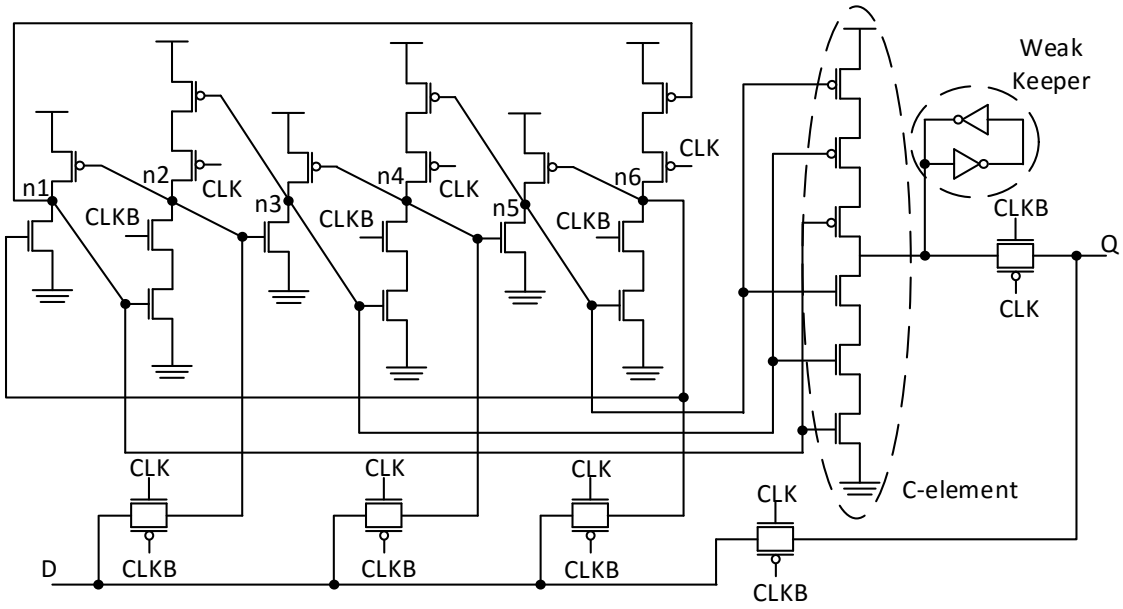


Figure 1.1: HSMUF latch [34] with a weak keeper on the output.

To the author's knowledge, the existing most efficient DNU robust design capable of recovering all nodes after a DNU is the DONUT latch [6] in Fig 1.2. The design, as proposed in their paper, uses only 36 transistors but has a much higher power consumption compared to the HSMUF (See Section 1.3). The reason for the high power consumption is due to contention on the input lines during the transparent mode. For example, if we observe node *n2* in Fig. 1.2 during the transparent mode, the node is driven by three cross-coupled elements. This contention will increase the amount of time required to change the node thus drastically increasing the dynamic power consumption. To optimize their design, we create the 48 transistor DONUT-M latch in which each component connected to an input node is modified, as shown in Fig. 1.3 so that the line is at high impedance for the whole duration of the transparent mode. This, in effect, removes the data contention problem

thus reducing the overall dynamic power and delay.

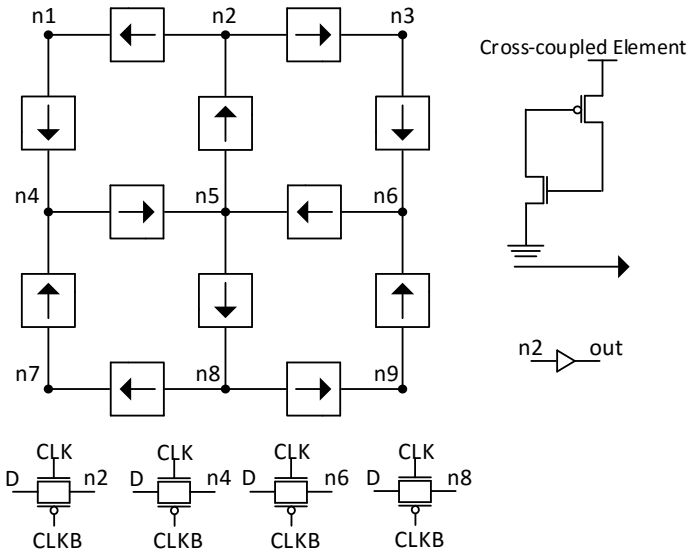


Figure 1.2: DONUT latch as proposed in [6].

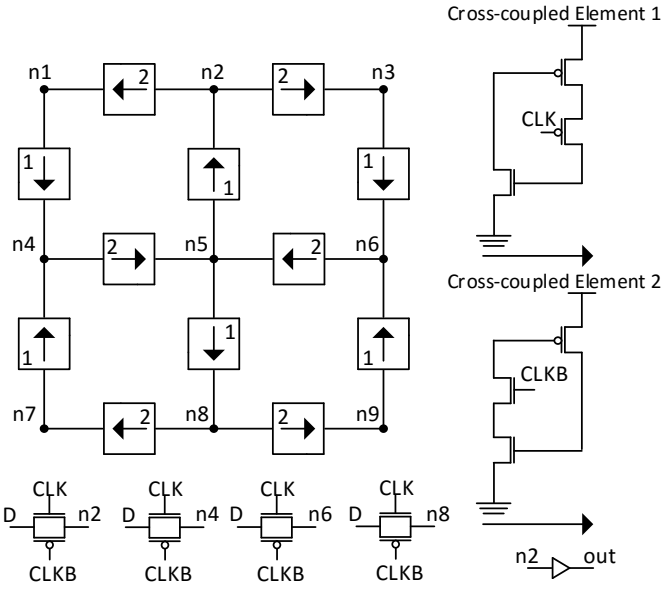


Figure 1.3: Modified low-power DONUT latch.

1.2 PROPOSED HRDNUT LATCH DESIGN

In this section we discuss the proposed DNU robust latch. The latch implementation is based on three cross connected storage loops connected to three C-elements. The basic design of the storage loop is given in Fig. 1.4. The data loop is based on the standard latch design with a 3-input C-element inserted to replace one of the inverters. The purpose of the C-element is to separate the feedback loop so that an error will not be held. Additionally, a PMOS is connected to the positive clock signal (CLK) and a NMOS is connected to the negative clock signal (CLKB) to remove contention when data is loaded to the latch. As in the modified DONUT latch, the addition of these transistors drastically reduces the delay and power consumption.

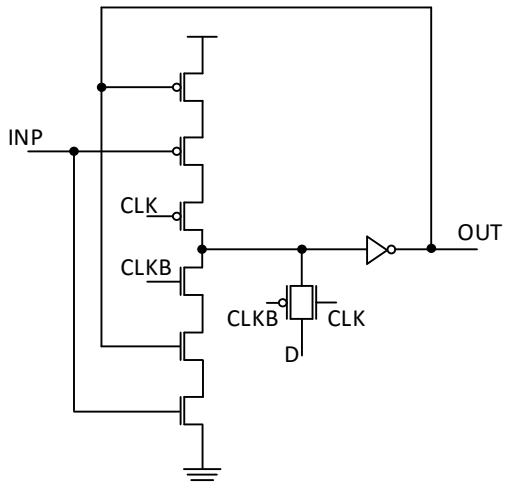


Figure 1.4: Basic data storage loop block.

Using the basic storage block we construct the block based latch as in Fig. 1.5. The latch was designed with the goal of ensuring that none of the nodes directly drove itself. For example, it can see that in Fig. 1.4 the node *out* is fed into the input of the 3-input C-element. If an error strikes node *out*, the cell will never be able to recover its previous state since one of the C-element inputs will be held to an erroneous value by its output. To prevent this issue, our design is based on cross-connecting three of the storage loop blocks

so that the C-element is driven by three separate block outputs. In Fig. 1.5 we provide a basic latch design using this idea. If a single error occurs on any node in this design, the circuit is capable fully recovering the previous data.

To demonstrate this, consider a strike on node *n2*. When the strike occurs, the erroneous value will be propagated to the C-elements driving nodes *n1* and *n3*. However, since there is no change on *n1* or *n3*, the C-elements *C1* and *C3* will hold their previous value thus preventing the error from propagating to the output. Additionally, since node *n2* is driven by nodes *n1* and *n3*, *n2* will completely recover the correct state.

A problem, however, with the latch in Fig. 1.5 is that it is not capable of tolerating DNUs. For example, if an error occurs on nodes *n1* and *n2* the erroneous values will propagate to the inputs of C-element *C3* and flip the value of *n3* thus changing the output value. However, since the latch has recovery capability for SEUs, we modify it so it can tolerate DNUs and recover all nodes to the previous state. In Fig. 1.7 we provide the schematic of the proposed HRDNUT latch. The design uses the block-based latch in Fig. 1.5 as a base and adds additional C-elements to prevent errors from being held by the data loop.

Initially, we will evaluate the HRDNUT latch during normal operation. When the positive clock signal (CLK) has a high value and the negative clock signal (CLKB) has a low value, the latch is in transparent mode. At this stage, the transistors connected to the clock signal in C-element *C1* deactivates the PMOS and NMOS stacks thus causing the node *n1* to be in a high impedance state. This, in effect, reduces data contention thus reducing delay and dynamic power consumption. Next, the data is loaded through the pass gates connected to nodes *n1*, *n22* and *out*. Since the output node *out* is loaded directly, the data to out delay is minimized and all nodes are set to their respective error free values. When CLK changes to a low value and CLKB to a high value, the latch moves into the hold mode. In this stage, the pass gates are deactivated and the state of the latch is held since each node is driven to the correct value using a C-element. Fig. 1.6 provides

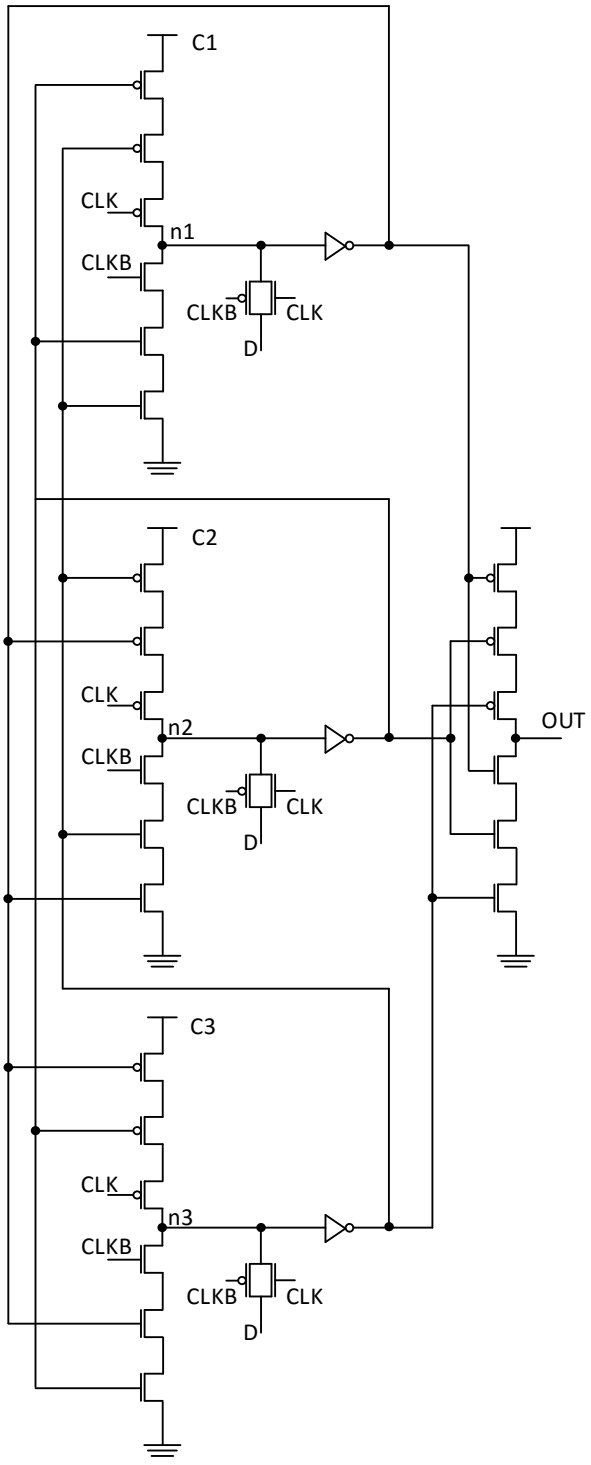


Figure 1.5: Schematic of the block-based latch.

the waveforms of the CLK, D and OUT nodes for both the transparent and hold modes of operation.

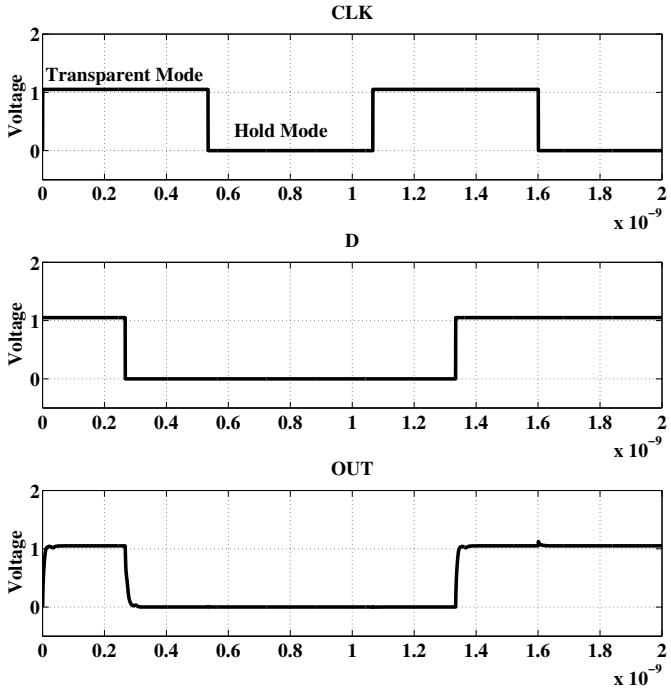


Figure 1.6: Waveforms of the HRDNUT latch during normal operation.

In the case of an SEU, the HRDNUT retains the excellent resiliency of the block based latch and the ability to recover every node after an error. In the case of any internal node being struck by an error, the latch will not change value due to all internal C-elements requiring at least 2 identical input values to change values. In the case of an error hitting the output node *out*, the latch fully recovers since *out* does not directly drive C-element *C7*.

Lastly, the latch will be evaluated in the case of a DNU. Note that unless otherwise stated, it is assumed that the analysis applies to both when D=0 and D=1. For the analysis, the possible DNU strike combinations are categorized into 9 distinct cases based on their effect in the HRDNUT latch. The categories are discussed in detail below:

1. Consider strikes at nodes *n1* and *n2*. In this case, the error at *n1* will propagate

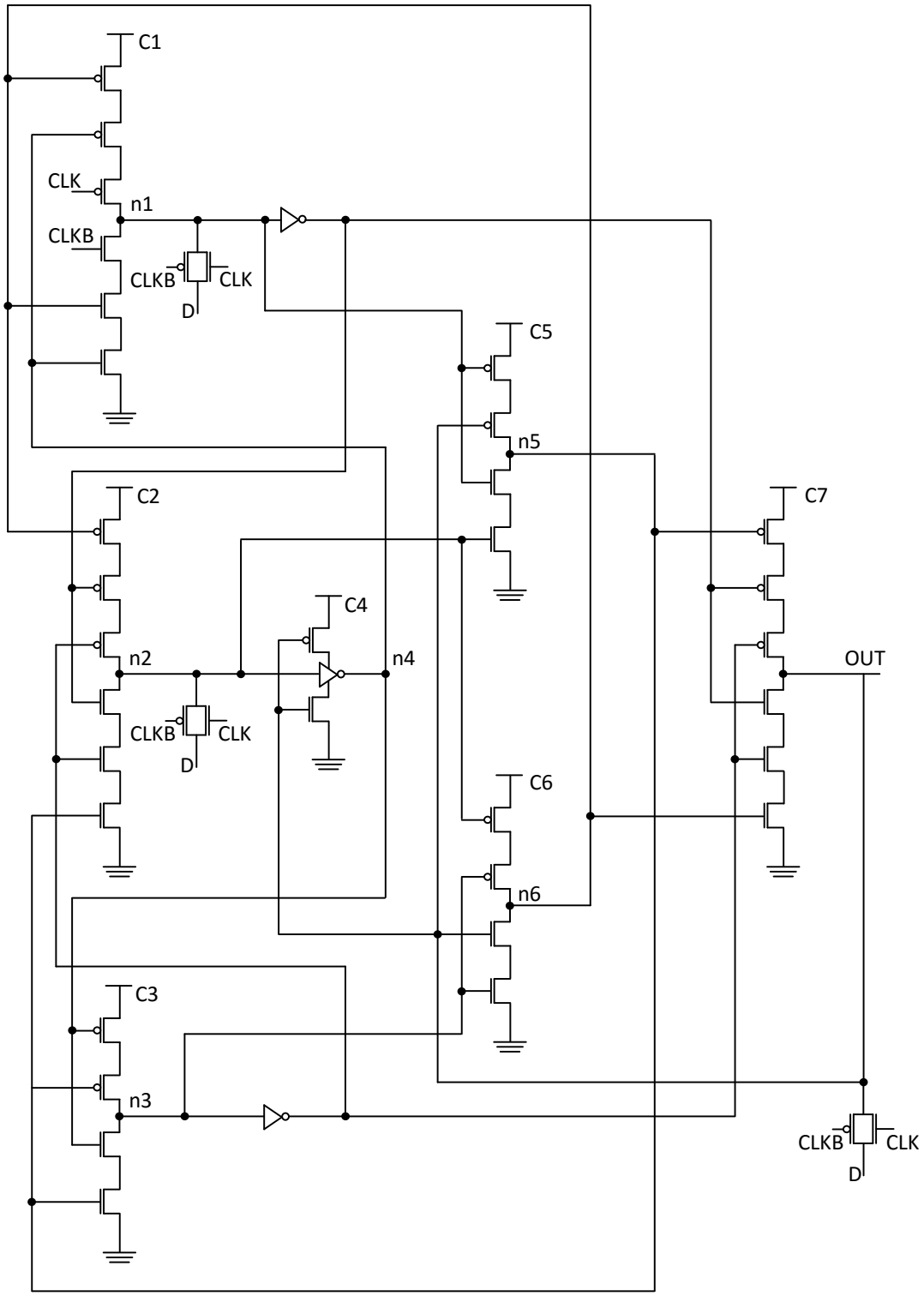


Figure 1.7: Schematic of the HRDNUT latch.

to C-elements *C5* and *C7* but will not cause a flip since the error at *n2* will be blocked by C-element *C4*. Additionally, since the inputs of C-elements *C1* and *C2* are unchanged, the nodes will recover their initial values. This analysis can be applied to node combinations containing node *n2* except for the combination with node *out* since the error will be blocked by C-element *C4*.

2. In the case of a DNU upsetting nodes *n2* and *out*, the error at *n2* will propagate through C-element *C4*. However, C-elements *C1* and *C3* will block the error and nodes *n1*, *n3*, *n5* and *n6* will hold their values thus driving node *out* to the correct state.

3. Consider when a DNU strikes nodes *n1* and *n5*. In this case, the error at *n1* hits the output of C-element *C1* which is propagated to *C7*. The error on *n5* is also propagated to C-element *C7*. Since node *n3* and the inputs of C-elements *C1* and *C5* are unaffected by an error, the output retains the error-free value and the latch fully recovers the previous state. The above analysis also applies to the node combination (*n3*, *n6*).

4. In the case of a DNU hitting nodes *n3* and *n4*, the error at *n4* is propagated to C-element *C3* and the error at *n3* is propagated to *C7* and *C6*. After the error on *n3* subsides, *C4* will drive node *n4* and, due to the connection at *C3*, node *n3* back to the error-free value. The node combination (*n1*, *n1*) can be analyzed similarly. For the node combinations of (*n4*, *n5*) and (*n4*, *n6*), the latch will also recover the previous result since the inputs to *C4* are unchanged. This implies that after the error occurs at *n4*, the node will be driven back to the correct value thus also driving the nodes *n5* or *n6* back to the correct value.

5. When a DNU upsets the combination of *n4* and *out*, the error at *out* is propagated to *C4*, *C5* and *C6* and the error at *n4* to *C1* and *C3*. Since none of the inputs to

C7 are changed by the error, *out* is flipped back to its error-free value which drives *n4* through *C4* back to its previous state.

6. Consider when a DNU strikes nodes *n1* and *n3* being struck. In this case, the errors are propagated to C-elements *C2*, *C5*, *C6* and *C7*. However, since the errors do not manifest into an error on any other node, the latch fully recovers from the error.

7. When a DNU strikes the nodes *n1* and *n6*. The error at node *n6* propagates to *C1* and *C7* while the error at *n1* also propagates to *C7*. Due to the error-free node *n3* driving *C7*, the previous value is held at the output by *C7*. Additionally, *n3* will drive *C6* back to its previous value thus driving *C1* back to the error free state. This analysis can be applied similarly to the node combination of (*n3*, *n5*).

8. In the case where a DNU strikes nodes *n5* and *out* the error at *n5* propagates to *C7*, *C2* and *C3* and the error at *out* goes to *C4*, a PMOS in *C5* and a NMOS in *C6*. When the error-free value at *out* is 1, the value at *n5* is 0. The error at the nodes change the values to 0 and 1 respectively and the erroneous value at *out* is propagated to the PMOS at *C5* and the NMOS at *C6*. This, in effect, causes the PMOS at *C5* to be activated and the NMOS at *C6* to be deactivated. However, since nodes *n1* and *n2* remain error-free, the NMOS stack of *C5* will drive *n5* back to the correct value. This, in turn, forces *C7* to also drive *out* back to the error-free value. In the case where *out* has an ideal value of 0, the error will be fully recovered since the NMOS stack will be entirely driven by fault-free nodes. The above analysis can be applied to the node combination of (*n6*, *out*).

9. Lastly, we analyze the node combinations (*n1*, *out*), (*n3*, *out*) and (*n5*, *n6*). In these cases the errors do not cause a change on the inputs of any C-elements driving the node thus the previous value will always be recovered.

1.3 SIMULATION RESULTS

The proposed HRDNUT latch has been implement using the 1.05V 32nm PTM library [38] and simulated in HSPICE. All transistors were set to the minimum size with the PMOS widths set to W=80nm and the NMOS widths set to W=40nm. To evaluate the DNU reliability of the design, current pulses were injected for every possible error combination. The injection current was calculated using the equation found in [39]. The equation is given below with τ as the technology dependent constant, Q_o as the injection current value and t as the variable for time.

$$I(t) = \frac{2Q_o}{\tau\sqrt{\pi}}\sqrt{\frac{t}{\tau}}e^{\frac{-t}{\tau}} \tag{1.1}$$

Using equation (1.1) τ was set to 32×10^{-12} and Q_o was set to 5fC. In all simulations, the latch was operated at a frequency of 1Ghz. In Appendix I Figs. 3.7-3.16, the waveforms for each case discussed in Section 1.2 are presented and show that the HRDNUT is fully capable of recovering all nodes in the presence of a DNU.

Next, we compare the HRDNUT to existing SEU and DNU tolerant methods. As in the HRDNUT latch, all latches were designed using the 32nm PTM library and operated at 1Ghz. For the analysis, we compare to the following SEU tolerant latches: DICE [3], FERST [10] and HIPER [27]. Additionally, we also compare to the following DNU tolerant designs: DNCS [16], Interception [15], HSMUF [34] and DONUT [6]. All transistors for the implemented latches were set to minimum width and length except for the designs that use a C-element with a weak keeper. In these designs the C-element's PMOS width was set to W=320nm and the NMOS width was set to W=160nm and the weak keeper was sized to be at minimum width. The C-element was sized so that the output driving strength did not allow the keeper to drive an erroneous value in the event of an error.

To provide a fair comparison, we measure the propagation delay, average power consumption and area of all designs and categorize them base on whether they can tolerate a

DNU and if they are robust from error after a DNU occurs. The delay was measured as the time between when a transition occurs on input D to when a transition was observed on the output. The average power was computed using the error-free operation for each latch for a duration of 200ns. To compare the area overhead, we adopt the unit size transistor (UST) metric as in [16] which represents the number of unit sized (minimum width is W=40nm in this case) transistors required for the design. Table 1.1 provides the results of these simulations.

Table 1.1: SPICE Simulations of Existing Latches using the 1.05V 32nm PTM library

Latch	DNU Immune	DNU Robust	Power (μW)	Delay (ps)	Area (UST)
DICE	No	No	1.332	8.145	16
FERST	No	No	3.178	31.648	60
HIPER	No	No	1.292	2.221	27
DNCS	Yes	No	4.948	22.486	61
[15]	Yes	No	5.606	79.168	89
HSMUF	Yes	No	1.871	1.0626	51
HSMUF (Keeper)	Yes	No	3.787	3.945	78
DONUT [6]	Yes	Yes	4.021	14.722	54
DONUT-M (Section 1.1)	Yes	Yes	2.760	8.421	72
HRDNUT (Proposed)	Yes	Yes	2.450	2.310	66

According to Table 1.1 the only DNU robust designs are the two DONUT latch implementations and the HRDNUT. Compared to the modified DONUT latch, the HRDNUT provides DNU robustness while reducing the power consumption and the number of transistors by 11.3% and 8.33% respectively while also reducing the delay by 72.5%. For the above reasons, the HRDNUT is the best design for clock gating applications due to its high robustness, even after a DNU occurs, and lower power, delay and area overheads.

1.4 FILTERING OF TRANSIENTS ARRIVING ON THE LATCH INPUT

In addition to radiation particles hitting the internal nodes in a latch, transient voltage pulses may arrive on the inputs. In a typical digital system design, combinational logic is placed between two sets of flip-flops, one set driving the input and the other being driven by the output of the combinational circuit. While there are masking factors in these types of circuits, it may still be likely that a pulse will arrive at the output of the circuit. A method to filter incoming pulses was proposed in [10]. Their design consisted of routing a single combinational output through two paths, one with no delay and one will sufficient delay that the two pulses will not overlap. The general design is given below.

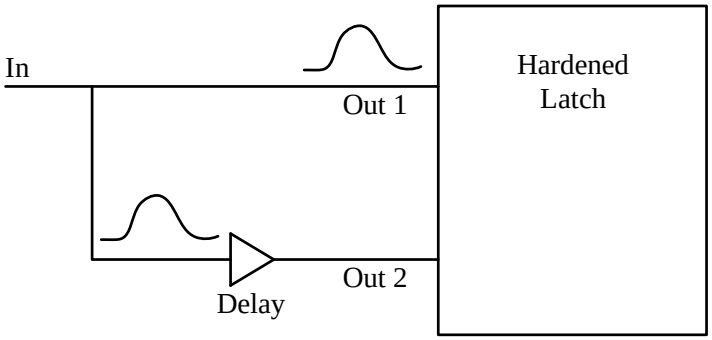


Figure 1.8: The pulse filtering circuit provided in [10].

An issue with the approach in Fig. 1.8 is that in a high radiation environment, a high energy particle can hit the delay element causing a voltage pulse on the node. If this occurs while a pulse being propagated through the circuit, it could delay the pulse at the delay element causing the erroneous value to be latched in the flip flop. A simple way to mitigate this effect is to add an additional delay element with twice the delay. This, in effect, will allow the device to function properly even if an error occurs during the filtering stage. The design of the device is given in Fig. [ref]. Due to the use of three outputs, the device can be used with the HRDNUT latch without modification.

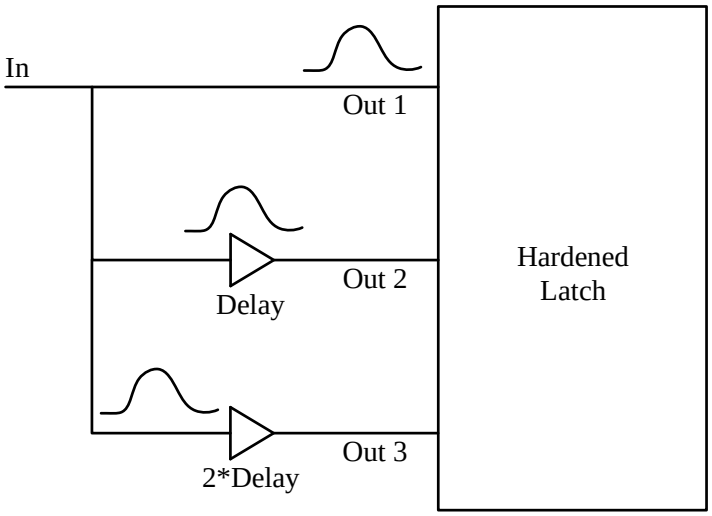


Figure 1.9: The enhanced pulse filtering circuit that can tolerate an error.

1.5 ANALYSIS OF A TRIPLE NODE UPSET TOLERANT LATCH

An inevitable issue with the development of more complex latches to tolerate multiple errors is the increase of the number of transistors and area. This issue leads to the possibility of new latch designs being vulnerable to more simultaneous error such as a triple node upset (TNU). The goal of this section is to estimate the area and device overhead and to evaluate the feasibility of a TNU tolerant latch. To begin the analysis, the design methodology used in the development of the HRDNUT is applied. The basic element used in this design is the block based latch given in Fig. 1.5. In the case of the HRDNUT, 3 blocks were needed to harden the latch against 2 errors. Based on this observance, it is estimated that hardening against TNUs will require 4 blocks. Additionally, each c-element will have and additional transistor added to accompany the extra block. The block based latch for TNUs is given in Fig. 1.10 and shows that TNU hardening requires a 35% transistor overhead.

It can be extrapolated from the overhead of the block based latch that the overhead of the TNU tolerant latch will be similar. Due to the high overhead, it can be concluded that creating latches that tolerate more than the DNU will be intractably large since a higher area increases the number of multi-node upsets. While it is likely that the TNU

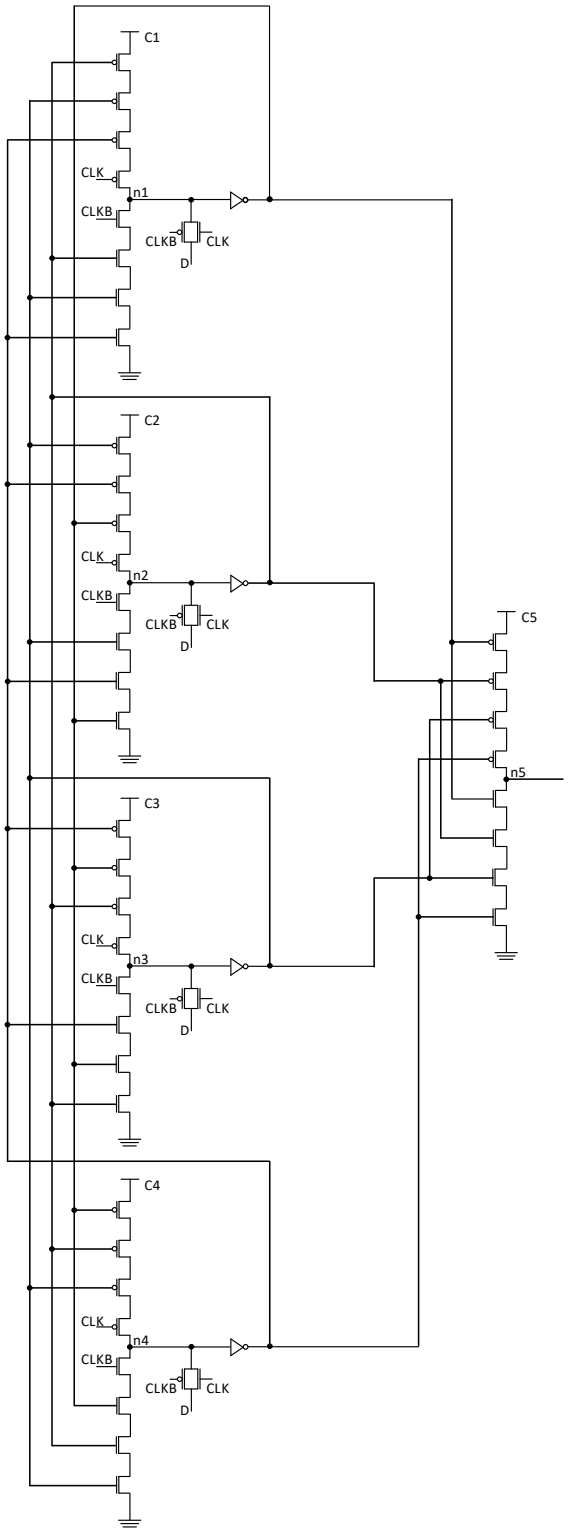


Figure 1.10: Block based latch that forms the basis for a TNU tolerant design.

tolerant latch will offer more reliability since a TNU should be a more rare case compared to a DNU, the power and area overhead will not offset the increased reliability.

1.6 CONCLUSION

In this section the HRDNUT latch was proposed which is suited for clock gating schemes. Since clock gating may require the latch to remain in a hold state for many clock cycles, the susceptibility of error increases. In many existing designs, a DNU may either change the state of the latch or push the latch into a state were the output may discharge over time due to a high impedance state. A common method to solve this problem is the addition of a weak keeper on the output. As shown in this paper, the addition of the keeper causes much higher power consumption. Since the HRDNUT latch does not stay in a high impedance state after a DNU, the HRDNUT provides high reliability during the whole duration of the hold mode while providing the lowest delay, power and area compared to other latches suitable for clock gating. Simulation results show that the HRDNUT is 11.3% more power efficient while requiring 8% less transistors and 72.5% less delay compared to the highly robust DONUT latch.

Additionally, an enhanced circuit to filter incoming transient pulses was proposed. The new circuit is able to tolerate a single error during the filtering phase. The area additional area of the new filter circuit is dependent on the size of the expected transient but overall is low. Lastly, an analysis of the TNU was conducted and the block based circuit was given. It was determined that a TNU tolerant circuit would have a 35% greater overhead compared to the HRDNUT. Based off this finding, it is recommended that a TNU tolerant latch is used in extreme radiation environments where system stability is crucial.

CHAPTER 2

AN ANALYTICAL PULSE APPROXIMATION ALGORITHM FOR MULTIPLE EVENT TRANSIENTS

As the transistor size has continued to decrease, the probability of a SEU occurring has continued to greatly increase. This, in effect, has created a need for accurate and efficient SEU simulators. The SEU problem has been investigated deeply. [20, 37, 35] However, due to the close proximity of transistors and the increase in transistor error sensitivity, the probability of a single or multiple radiation strikes producing multiple voltage pulses has become a concern. The authors in [13] investigated a 65 nm 3x3 inverter matrix and found that 40% of all single radiation strikes resulted in the generation multiple significant voltage pulses. Considering the possibility that a modern circuit is also vulnerable to multiple simultaneous strike, there is a significant concern for multiple transients. This creates the need for either the enhancement of existing SEU tools or the development of new tools which can consider the single event multiple transient (SEMT) and multiple event multiple transient (MEMT) effect.

Previous work in SEUs have shown that combinational circuits are vulnerable to 3 types of error masking effects: logical masking, electrical masking and temporal masking. Accurate estimation of the soft error rate requires consideration of all masking factors for not only SEUs but also the SEMT and MEMT cases. In this paper, the focus is on the consideration of the electrical masking effect. There have been many approaches to the electrical masking problem for the SEU case. [26, 32, 33] However, a major drawback with these approaches is that they do not accurately consider the effect of multiple pulses occurring on the gate input (referred to as convergent pulses). Since, in the SEMT and MEMT cases, there are many pulses generated, the occurrence of convergent pulses is much higher. To model these cases, an accurate electrical masking model must consider convergent pulses.

There has been a few attempts at modeling the effect multiple transient pules, however all approaches suffer from inaccurate estimation of electrical masking. The authors in [21, 8] both proposed a simulation tool to determine the soft error rate (SER) of a combinational circuit in the presence of multiple transients, however, they calculate the output transient of a gate with a convergent input through the simple superposition of the pulse widths. As stated in [32], this is insufficient since accurate electrical masking modeling requires the inclusion of the pulse amplitude and non-linear portions of the glitch.

An additional possible solution to the pulse convergence problem is the simple extension of [32] for convergent pulses. In [32] an accurate iterative model for the single pulse case is proposed. Their model can easily be extended by generating larger look-up tables. However, since they characterize stacked transistors using look-up tables, gates with more than 2-inputs require tables with 125,000 entries for 3-input gates and over 7 million for 4-input gates. Thus our model provides an enhancement to [32] by allowing the characterization of stacked transistors using a look-up tables for the transistor and the Miller capacitance between the gate and drain terminals.

Based on the above discussion, there is a lack of accurate and fast methods to consider convergent pulses. This provides the motivation of this work. To the authors knowledge, the proposed model is the first masking model that accurately and efficiently approximates the entire output pulse shape for convergent input pulses. Through extensive experimentation, the proposed model has been shown to be robust for various input shapes and process technologies while providing a 15X speed-up over HSPICE.

2.1 PULSE APPROXIMATION MODEL

As in [32], an important aspect of the proposed model is the accurate approximation of the transistor drain to source current (IDS). To consider the I_{DS} current, a look-up table for each transistor is created by sweeping both the gate-source voltage (VGS) and the drain-source voltage (V_{DS}).

Additionally, as in [32], the effects of pulse overshoot and undershoot are considered through the use of a Miller capacitor. In contrast to [32], however, we consider the MOS parasitic effects on each individual transistor through the inclusion of a Miller capacitance between the gate and drain terminals. Fig. 2.1 demonstrates the proposed model for a NMOS and a PMOS transistor.

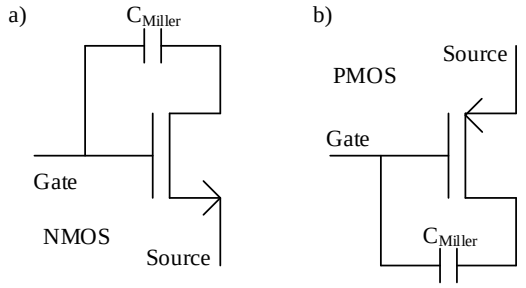


Figure 2.1: (a) The transistor model used for a NMOS. (b) The transistor model for a PMOS.

As stated in [32], it was observed that the value of the Miller capacitance varied over the range of drain-source (V_{DS}) and gate-source (V_{GS}) voltages. To consider this effect, it is proposed to generate a look-up table with each entry pertaining to the Miller capacitance at each specific V_{GS} and V_{GS} bias. The Miller capacitance look-up tables can be characterized using HSPICE by sweeping the V_{GS} and V_{GS} voltages and measuring the current flowing through the gate node of the transistor (i_m). Using the current i_m, the derivative of the difference between the gate and drain voltages and equation 2.1, the capacitance C_{Miller} can be calculated.

$$C_{Miller} = \frac{i_m}{\frac{d(V_{DS}-V_{GS})}{dt}} \tag{2.1}$$

As stated before, the IDS current and the Miller capacitance for both the PMOS and NMOS transistors are characterized. Using this data, each transistor is modeled as a current source and its Miller capacitor. Any given gate can then be represented by replacing

each transistor with a current source and a Miller capacitor connected between the drain and gate terminals. At the output of the gate, a load capacitance is added to consider the gate loading effects and to calibrate the model. Fig. 2.2 demonstrates the conversion to the proposed model on a 2-input NAND gate.

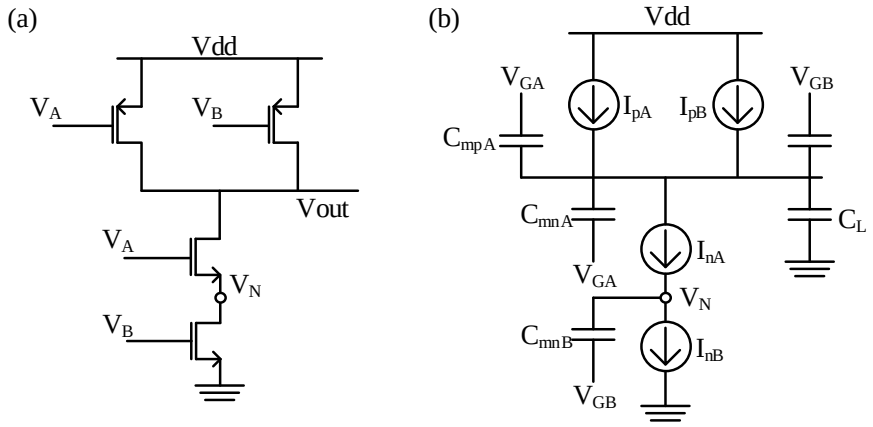


Figure 2.2: (a) The transistor model used for a NMOS. (b) The transistor model for a PMOS.

Before beginning the discussion on the equation derivation it is important to note that this model is general for all types of gates but for the sake of this discussion, the derivation will focus on CMOS NAND and NOR gates.

The proposed gate modeling for a given gate, as in Fig 2.2b, can be generalized for an i number of inputs as in Fig. 2.3 where a single current source represents the current for the pull-up network and a second current source represents the pull-down network. This is based on the assumption that the gate is a CMOS gate in which one network consists of parallel current sources and the equivalent current is equal to the sum of each parallel transistor current. The other current source represents the current flowing through series connected transistors which ideally have a single constant current. However, in practice, due to parasitic effects, the current in each stacked transistor varies. It was found through many simulations that the best approximation for this current was by taking the average

between each transistor current in the stack. For example, if we have the circuit in Fig. 2.2b, the current on the stacked transistors can be calculated using the following equation:

$$I_{pd} = \frac{I_{nA} + I_{nB}}{2} \tag{2.2}$$

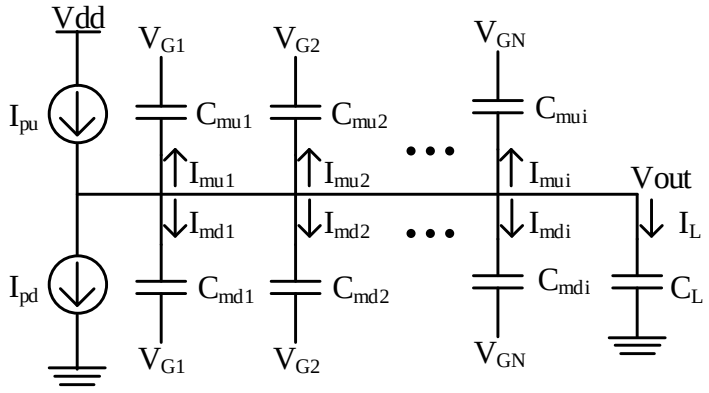


Figure 2.3: A generalized representation of the gate model.

To derive the output equation for the node V_{out} in Fig. 2.3, we first use Kirchoffs Current Law (KCL) at the node V_{out}. Assuming that I_{mu_i} and I_{md_i} are the currents through the i-th Miller capacitor, I_L is the current through the load capacitor and I_{pu} and I_{pd} represent the pull-up and pull-down current respectively, KCL will result in the following:

$$I_{pu} = I_{pd} + I_L + \sum_{i=1}^{N} I_{mu_i} + \sum_{i=1}^{N} I_{md_i} \tag{2.3}$$

For our model, we discretize the time into time steps with the variable t_step. Additionally, we represent each variable in the current time instant with the subscript i and each variable in the previous time instant with the subscript $i-1$. Lastly, we represent the change in a value between two time instances using Δ (i.e. $V_i - V_{i-1} = \Delta V$).

According to the charge conservation law, we get equation 2.4 for each Miller capacitor

C_{m_i} connected between V_{out} and the input voltage V_{G_i}.

$$I_{m_i} = \frac{[\Delta V_{out} - \Delta V_{G_i}]C_{m_i}}{t_{step}} \tag{2.4}$$

Next, consideration of the load capacitance C_L gives the following equation:

$$I_L = \frac{(\Delta V_{out} C_L)}{t_{step}} \tag{2.5}$$

By substituting into equation 2.3 the current I_{m_i} from 2.4 for the currents I_{mu_i} and I_{md_i} and replacing the current I_L with equation 2.5, we get the following for a N_u number of Miller capacitors connected between V_{out} and the i-th input in the pull-up network (denoted by C_{mu_i} and a N_d number of Miller capacitors connected to V_{out} and the i-th input on the pull-down network (denoted by C_{md_i}):

$$I_{pu} = I_{pd} + \frac{(\Delta V_{out} C_L)}{t_{step}} + \sum_{i=1}^{N_u} \frac{[\Delta V_{out} - \Delta V_{G_i}]C_{mu_i}}{t_{step}} + \sum_{i=1}^{N_d} \frac{[\Delta V_{out} - \Delta V_{G_i}]C_{md_i}}{t_{step}} \tag{2.6}$$

Rearranging 2.6 and solving for ΔV_{out} gives the following:

$$\Delta V_{out} = \frac{(I_{pu} - I_{pd})t_{step} + \sum_{i-1}^{N_u} \Delta V_{G_i} C_{mu_i} + \sum_{i=1}^{N_d} \Delta V_{G_i} C_{md_i}}{C_L + \sum_{i=1}^{N_u} C_{mu_i} + \sum_{i=1}^{N_d} C_{md_i}} \tag{2.7}$$

Assuming that ΔV_{out_i} equals $V_{out_i} - V_{out_{(i-1)}}$ results in:

$$V_{out_i} = \Delta V_{out} + V_{out_{i-1}} \tag{2.8}$$

As can be observed in equations 2.7 and 2.8 an important aspect of the proposed model is the accurate consideration of both the pull-up and pull-down currents. As stated before, cases where the pull-up or pull-down transistors are composed of parallel connected transistors the current can simply be found through the sum of the drain to source currents.

However, in cases where the transistors are in series, the determination of the equivalent current is more complicated.

In Fig. 2.2a the pull-down branch consists of series connected NMOS transistors which will be used to demonstrate the derivation. Note that using the same method, the equations for stacked PMOS devices can be easily derived. Within the stack there is an intermediate node voltage V_n which is crucial for the determination of the drain to source currents in the transistors. Through many simulations using the transistor drain to source current look-up tables, it was found that accurate approximation of the node voltages provided accurate current values for each transistor in the stack.

To calculate the intermediate voltage, it is proposed to first convert each transistor to the representations in Fig. 2.1a and 2.1b. which results in the circuit in Fig. 2.2b. In Fig. 2.2b. the intermediate node voltage V_n is connected to a static capacitor representing the loading effect of the subsequent transistor stack. Next, the node voltage equation will be derived.

Similar to equation 2.7, the node voltage V_n is modeled using discrete time steps. Using KCL at V_n in Fig. 2.2b we get the following equation:

$$I_{D1} = I_{D2} + I_m \tag{2.9}$$

As in equation 2.4, the current through the static capacitor can be determined using the charge conservation law resulting in the following for the current I_m.

$$I_m = \frac{C_{m1}(V_{n_i} - V_{n_{i-1}})}{t_{step}} \tag{2.10}$$

By substituting 2.10 into 2.9 and solving for V_{n_i}, the equation for the node voltage V_n at the time instant i is derived.

$$V_{n_i} = \frac{(I_{D1} - I_{D2})t_{step}}{C_{m1}} + V_{n_{i-1}} \tag{2.11}$$

In cases where there are more than 2 stacked transistors, the described model is directly extendable for each intermediate node. Fig. 2.4a and b provides the conversion for 3 stacked NMOS transistors to our model. The node voltages V_{N1} and V_{N2} can be found in any order using equation 2.11 and by considering the currents flowing into the node. For example, to solve for the node voltage V_{N2} the currents I_{D2} and I_{D3} would be used in equation 2.11 since they are directly connected to the node. Additionally, the order for solving the node equations does not matter since the node voltages from the previous step $i-1$ are used.

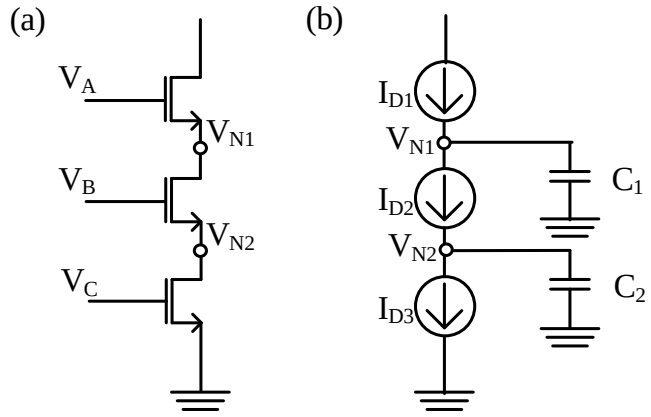


Figure 2.4: (a) Three stacked NMOS transistors with two intermediate nodes V_{N1} and V_{N2}. (b) The transistors converted to the proposed circuit model.

2.2 EXPERIMENTAL RESULTS

The proposed algorithm was implemented in Matlab and characterized using the 32 nm and 45 nm PTM library [38] and the 65 nm IBM library. To test the proposed model, 3 benchmark circuits for each process library were created to test two convergent pulses on a 2 and 3 input NAND gate and a 2 input NOR gate. The first two benchmark circuits denoted by NAND2R and NAND2F in column 1 of Table 2.1, consists of two NAND gate chains with each off-input held to a non-controlling value. The chains contain two gates and

the R and F in the benchmark names represent either a rising (R) or falling (F) pulse on the convergent gates inputs. Each chain is then connected to an input of the NAND gate. The benchmark circuits NOR2R and NOR2F are constructed similarly but the chains consist of NOR gates and are connected to the input of a 2 input NOR gate. The last benchmark circuits constructed, NAND3R and NAND3F, were created with 2 2 gate chains connected to the first and second input of the 3-input NAND gate. The remaining off-input was held to a non-controlling value.

Next, to accurately model the injected pulse shape, the injection current is represented as a current source located at the output of the first gate in the chain. We then used the function for the pulse shape given in [39] to relate the strike current to a given charge. Equation 1.1 was used to model the current generated by a pulse strike.

Using equation 1.1, Q_o was varied over a range of charge values that results in an observable error. For our test cases, we found that a charge ranging from 3-16 fC for a of 90x10-12 provided an observable output shape for all processes tested. We then propagated the pulses through the gate chains to gate of interest and compared the generated outputs using the proposed model and the model in [32] to HSPICE. Table 2.1 details the average error for 30 cases per gate and transition type for the 32, 45 and 65 nm processes using 2400 data points over a range of 0 to 1.2 pico-seconds. The first column gives the benchmark name. All subsequent columns provide the MSE error for each test case for the proposed method and the method in [32]. Assuming that N represents the number of calculated points, V_i^H represents the pulse voltage at point i on the signal from HSPICE and V_i^C is the voltage calculated by the proposed model, the MSE was calculated using the following equation:

$$MSE = \frac{\sum_{i=0}^{N}(V_i^H - V_i^C)^2}{N} \tag{2.12}$$

As can be seen from Table 2.1, the proposed model is much more accurate over the

model proposed in [32]. In all tested cases, the method in [32] predicted there would be an output pulse. However, they only propagate the pulse with the largest width which fails to consider the effect of multiple pulses and provides a pulse with a much smaller width. For this reason, their method tends to underestimate the severity of the pulse in the presence of convergent pulses. According to Table 2.1, our method closely predicts the output pulse shape.

Next, we run a simulation to observe how the number of points selected relates to the speed and accuracy. In this simulation, we used the NAND2F and NAND3F benchmark and applied a pulse to each input of the terminating NAND gate. We then varied the number of output data points for the terminating NAND gate in both HSPICE and the proposed model. Table 2.2 provides the results of this simulation.

The first column provides the number of simulation points used to calculate the result. Columns 2 and 3 provide the error of the proposed model and [32] for the given number of points. Columns 4, 5 and 6 gives the execution time of the proposed method, the method in [32] and HSPICE respectively. As can be seen from Table 2.2, the reduction in simulation time scales linearly to the reduction in simulation points.

In Fig. 5, we provide the output waveform of the simulation in Table 2 for 900 and 300 points respectively. As can be observed in Fig. 5a, the result for the proposed model closely matches the result obtained in HSPICE when 900 simulation points are used. If a higher accuracy is desired, the proposed model provides a near perfect match with 2400 points. For both the 900 and 300 point cases, it can be observed that the method in [32] greatly underestimates the resulting pulse thus providing a high MSE. The proposed model provides an accurate result with an MSE of 7.40x10-3 while still providing a speed-up of 15X compared to HSPICE for this benchmark. In general we have observed a speed-up of 15X for any 2-input gate using 900 points while maintaining a high accuracy.

In Fig. 6a and 6b, we provide the output waveforms for the NAND3F simulation for 900 and 300 points respectively. Similar to the NAND2F benchmark, a simulation

with 900 points provides a close waveform approximation. However, if a lower accuracy is tolerable, the number of points can be reduced further providing a faster simulation time as demonstrated in Table 2.2 and Fig. 6b. As can be observed in Table 2, proposed model provides a speed-up of 12X for the 900 point case. In general for any 3-input gate, we have observed a speed-up of 12X using 900 points while maintaining a high accuracy

Table 2.1: Proposed and [32] compared to HSPICE for 2400 points.

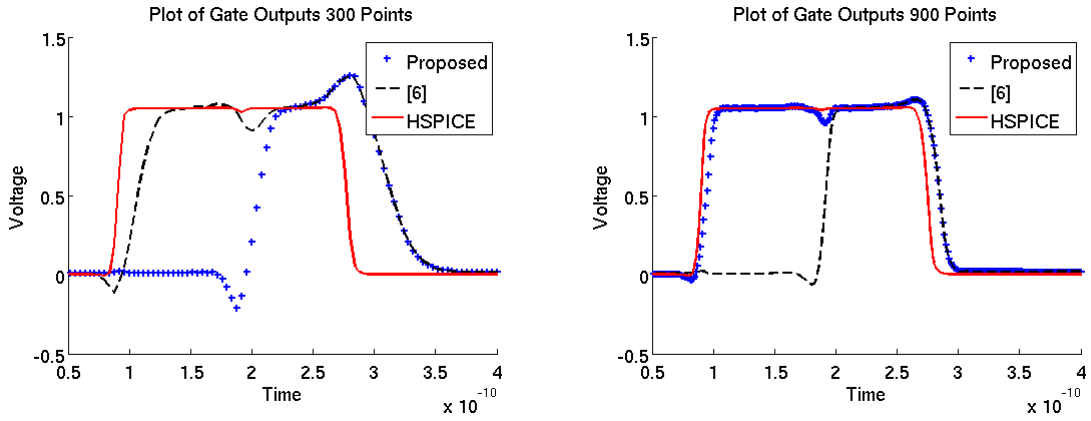
Circuit	MSE [65nm] ($x10^{-3}$)		MSE [45nm] ($x10^{-3}$)		MSE [32nm] ($x10^{-3}$)	
	Proposed	[32]	Proposed	[32]	Proposed	[32]
NAND2R	0.760	108	0.453	35.6	0.190	45.6
NAND2F	1.000	110	0.550	67.8	0.467	50.9
NOR2R	0.220	23.0	0.345	133.0	0.405	146.0
NOR2F	0.986	103	0.678	42.5	0.493	33.9
NAND3R	0.887	115	0.785	55.6	0.352	39.2
NAND3F	0.998	98.9	0.467	39.8	0.465	30.8

Table 2.2: Execution Time vs Accuracy ($x10^{-3}$)

Points	Error Proposed	Error [32]	Execution Time Proposed	Execution Time [32]	Execution Time HSPICE
NAND2F					
2400	0.486	85.7	8.39	8.03	100
1200	3.90	92.0	4.34	4.01	60
900	7.40	97.8	3.43	3.05	50
300	41.1	115.9	1.23	1.18	20
NAND3F					
2400	0.529	37.5	12.1	11.4	120
1200	6.40	38.7	6.40	5.90	70
900	13.0	42.8	4.90	4.50	60
300	60.9	85.0	1.91	1.50	30

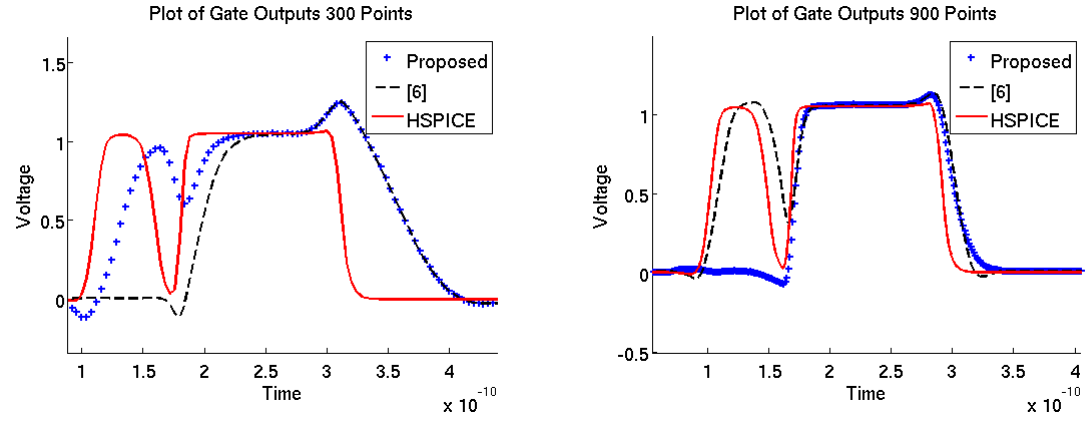
2.3 CONCLUSION

In this section, a model was proposed that improves SEMT and MEMT error simulation by calculating the output pulse shape in the presence of convergence with an MSE of



(a) (b)

Figure 2.5: (a) Output of NAND2F with 300 points. (b) Output of NAND2F with 900 points



(a) (b)

Figure 2.6: (a) Output of NAND3F with 300 points. (b) Output of NAND3F with 900 points

less than 7.40x10-3 for 2-input gates and 13x10-3 for 3-input gates using 900 points and a speed-up of 15X and 12X respectively compared to HSPICE. Future work with our model includes the study of the number of points effect the simulation result on large benchmarks and the study of SEMT and MEMT on large standard benchmark circuits using the proposed model.

CHAPTER 3

ACCURATE SOFT ERROR SIMULATION USING ADAPTIVE PARTITIONING

As technology continues to the scale down, the likelihood of a radiation induced error increases. This trend provides a need for accurate and efficient methods to calculate the soft error rate of a given circuit. Since it is expensive and time consuming to design and circuit and test at a later time, efficient tools that can accurately determine the error rate for a given netlist before fabrication can significantly reduce the design time. However, it has proven to be very difficult to characterize combinational circuits due to three pulse masking factors. The masking factors are as follows:

1. Logical Masking: The pulse is removed due to the boolean inputs on the gate during pulse generation or a controlling value on an off-input during pulse propagation.

2. Electrical Masking: The first order RLC effects within the gate cause the pulse to degrade and fully attenuate.

3. Temporal Masking: The pulse arrives at a flip-flop but does not meet the set-up and hold time parameters.

Considering the above factors, all current soft error simulators consist of modeling a particle at a specific energy which is used in an equation to model the error pulse shape [39]. Once the pulse is generated, it is propagated through the combinational logic to the output. It is then assumed that the output is connected to a flip-flop thus the set-up and hold times are considered. It has been shown in [20, 21] that concurrent estimation of all masking factors is required in order to ensure that the soft error rate is calculated accurately. However, due to limitations on simulation time and available memory, efficient and accurate consideration of all masking factors has proven to be a difficult and ongoing problem.

There have been many approaches to improve the estimation of the three factors. First the existing approaches to the estimation of logical masking will be discussed. The most common method to calculate the logical masking effects is the use of a Monte Carlo based simulation which consists of randomly applying patterns as in [32, 37, 23, 30, 29]. While this approach is very easy to implement, it suffers from accuracy and run time issues when the circuit has more than 30 inputs. In the case of large circuits, the simulator must resort to generating random input patterns which cover only a small subspace of the possible inputs. This implies that the error in logical masking estimation increases with the circuit size.

To enhance on Monte Carlo based simulation, the authors in [38] propose the use of probabilistic transfer matrices (PTM). This approach consists of representing the Boolean functions within a matrix in which Boolean operations can be applied. While being very accurate, the use of PTMs are non-ideal since they have high memory and simulation time costs that explode on small circuits. The authors in [12] propose an enhanced alternative to the PTM method which uses stochastic logic to calculate the signal probabilities for logical masking estimation. While their method is fast, it relies on the use of random input patterns that can limit the accuracy of the calculation.

Another common approach found in [4, 17] uses the correlation coefficient method (CCM) proposed in [7]. This method uses basic gate signal probability functions to determine the output. For example, the probability of an AND gate is given as $P(O) = P(A)P(B)$ where $P(O)$ is the output probability, $P(A)$ is the probability of input A being "1" and $P(B)$ is the probability of inputs B being "1". A drawback with using the basic probability functions is that they are unable to consider the correlations between signals. CCM improves on this by add a correlation factor to estimate the signal probability. While CCM does provide quick estimation of the logical masking effects with virtually no memory overhead, it can only estimate first order correlation. This implies that as the circuit gets larger, the accuracy of the method substantially decreases.

The last approach discussed in this section, which is of most concern in this paper, is the use of binary decision diagrams (BDDs). BDDs provide a very accurate way to determine the logical masking effects since they are a condensed canonical form of a Boolean function. The main problem with BDDs however, is that they scale exponentially with circuit size thus leading to the possibility of blow up on medium to large circuits. Despite this problem, there have been many proposed simulators that the use BDDs [35, 20, 21]. The main advantage to the use of BDDs is that all input patterns can be considered concurrently thus allowing for exact calculation of logical masking probabilities.

To alleviate the inherent blow up problem, the authors in [35] propose the use of partitioning. In their work, they show that the use of partitioning allows for much faster calculation of the soft error with no threat of blow up at a cost of accuracy. The authors in [21] also investigate partitioning but do not give an in depth study on how partitioning effects the output error. In both approaches, the circuit is partitioned before simulation with each partition size being set to a predetermined value. This is problematic since the size may be much smaller or larger compared to the given time, processing power and memory. For this reason, this section proposes the use of an adaptive partitioning algorithm that scales the partition size based on the number of nodes. It is shown that the adaptive method allows for faster and more accurate estimation of the logical masking effect compared to the non adaptive approach.

In addition to the adaptive partitioning approach, the proposed simulator has the capability of considering multiple concurrent pulse strikes commonly referred to as multiple event transients (METs). The consideration of multiple strikes has become of greater concern due to the small feature size, leading to a lower critical charge for upset, and the closer placement of transistors [31]. These type of events can occur in two forms: single event multiple transient (SEMT) and multiple event multiple transient (MEMT). A SEMT is defined as the case where a single radiation particle upsets multiple transistors causing multiple transients pulses. A MEMT is defined as when multiple particles hit different

circuit components simultaneously. While the probability of the SEMT vs the MEMT depends strongly on the radiation environment, methods developed for one type of error can easily be modified to consider the other.

There are a few existing methods that include the consideration of multiple event errors. The authors in [21] consider the correlation of METs however their method uses BDDs which tend to blow up. While they do investigate the use of partitioning, their method is only tested for small circuits using only a few concurrent pulses. An additional simulation tool used for MET analysis found in [9] uses probabilistic functions. This approach has shown to be fast and efficient but is not suited for the use of complicated electrical masking methods. To improve on the existing methods, the simulator FAST_MET (Fast and Accurate Simulation Tool for Multiple Event Transients) which uses BDDs to calculate the logical maksing effect in conjunction with the accurate pulse approximation tool proposed in Chapter 2. FAST_SET improves on the method in [21] by allowing for the integration of accurate electrical masking models and through the extensive use and evaluation of partitioning to avoid BDD blow up.

3.1 PRELIMINARIES

A crucial aspect in the estimation of the logical masking effects is accurate creation of the Boolean functions. To ensure full consideration of the logical masking effects, Boolean functions must be created to evaluate the circuit inputs for pulse generation and the off-inputs for pulse propagation. First the method to generate the functions for error pulse generation will be discussed.

When a high energy particle strikes a transistor, the magnitude and polarity of the pulse will depend on the configuration of the transistor. In the case of CMOS logic, a transient pulse is generated when a particle hits a blocking transistor. This will, in turn cause the transistor to temporarily conduct current allowing for the generation of the voltage pulse. In Fig. 3.1 a 2 input NAND gate is given to demonstrate this mechanism.

In the NAND gate, a particle can hit a single PMOS or a NMOS transistor to cause an error. However, since the transistor must be blocking, the input pattern has a large effect on where the error will occur and the polarity of the pulse.

For example, in the 2 input NAND gate in Fig. 3.1, if both inputs have a value of "1", a strike on either of the PMOS transistors will likely cause a transient pulse. In the case of one input having a "0" value and the other a "1" value, a strike on the blocking NMOS transistor will cause an error. In the case of both inputs having a "1" value, the gate will only generate a transient pulse if both NMOS transistors are struck concurrently. Addition, the polarity of the pulse also requires consideration. If the output is has a "1" value, the pulse will start at a high value, go low and back to a high value. When the output has a "0" the pulse will start at a low value, go high and back to a low value. Based on this observation, the function for the pulse generation for the AND, NAND, OR and NOR are given in Table 3.1 for a N number of inputs with each input denoted as I_i.

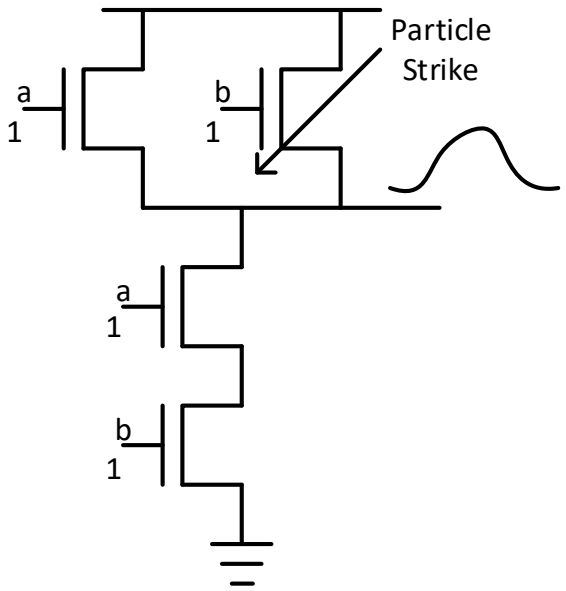


Figure 3.1: A low-high-low transient pulse being generated by a strike on the PMOS.

For the consideration of the generation of METs, the above equations must be modified

Table 3.1: Functions for Transient Pulse Generation

Gate	Polarity	Generation Function ($F(G)$)
AND	Rising	$F(G) = \neg(I_1 \wedge I_2 \wedge ...I_N)$
AND	Falling	$F(G) = I_1 \wedge I_2 \wedge ...I_N$
NAND	Rising	$F(G) = I_1 \wedge I_2 \wedge ...I_N$
NAND	Falling	$F(G) = \neg(I_1 \wedge I_2 \wedge ...I_N)$
OR	Rising	$F(G) = \bar{I_1} \wedge \bar{I_2} \wedge ...\bar{I_N}$
OR	Falling	$F(G) = \neg(\bar{I_1} \wedge \bar{I_2} \wedge ...\bar{I_N})$
NOR	Rising	$F(G) = \neg(\bar{I_1} \wedge \bar{I_2} \wedge ...\bar{I_N})$
NOR	Falling	$F(G) = \bar{I_1} \wedge \bar{I_2} \wedge ...\bar{I_N}$

such that the joint probability of the error occurring is considered. In the case of pulse generation, it is assumed that a single radiation particle will strike two or more gates causing transient pulses. As stated previously, in order for a pulse with a specific polarity to be generated, the output of the gate must have a specific value. Based on this, the probability of pulses being generated (P_{gen}) assuming strikes N gates with sufficient energy is given in equation 3.1 where $P(G_{i,p})$ is the probability that gate G_i is capable of generating a pulse of polarity p. If the pulse generated is rising, $G_{i,p}$ pertains to the probability that the output of $G_{i,p}$ is "0". If the pulse is falling, $G_{i,p}$ is the probability of the output being "1".

$$P_{gen} = \prod_{i=1}^{N} P(G_{i,p}) \tag{3.1}$$

A particle hitting two transistors simultaneously allows for multiple cases since the input may be such that the struck gate is not blocking. Based on this, the total number of possible ways a strike can produce a pulse (P_{num}) is given in equation 3.2 assuming a N number of struck gates and r as the number of gates that produce a transient pulse.

$$P_{num} = 2 * \sum_{r=1}^{N} \frac{N!}{(N-r)!} \tag{3.2}$$

Equation 3.2 is based on the fact that multiple pulses are not always generated when a particle strikes both transistors. As stated previously, the polarity and existence of the

pulse depends on the input patterns. If one assumes that the particle hits a N number of gates, this creates a P_{num} number of cases where $P(C_i)$ represents the probability of case i calculated using equation 3.1. The total probability of the inputs having the values to generate a SET or MET P_{gen} is given below:

$$P_{gen} = \sum_{i=1}^{P_{num}} P(C_i) \tag{3.3}$$

In the case of a pulse being propagated to the input of gate, the pulse will be masked if one of the off-inputs has a controlling value. Fig. 3.2 gives a NAND gate with one of the inputs having a pulse and the other input having a controlling "0" value. In this case, the output will be a logical "1" value despite the pulse on the input.

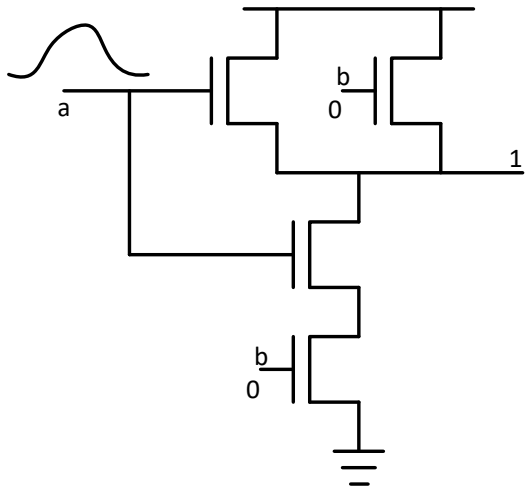


Figure 3.2: A transient pulse being masking by a controlling value on the off-input.

To calculate the logical masking factor during pulse propagation, the functions of the off-inputs are calculated using the logical AND operation. Assume that $F(OI_i)$ is the Boolean function of the off-input OI_i representing the input patterns that allow a non-controlling value and there are a N number of inputs, the function representing the case where the pulse is propagated $F(M)$ is given below:

$$F(M) = F(OI_1) \wedge F(OI_2) \wedge ...F(OI_N) \tag{3.4}$$

In equation 3.4 only $N-1$ inputs are considered since the input which contains the transient pulse is not included in the calculation.

For the consideration of METs, two or more pulses may arrive at a gate. If the two pulse case is assumed, there are 3 possible ways that the pulses can be propagated: the pulse on a is propagated and b is masked, the pulse on b is propagated and a is masked and, lastly, the both pulses are propagated simultaneously providing an output pulse. Fig. 3.3 provides an illustration of the cases.

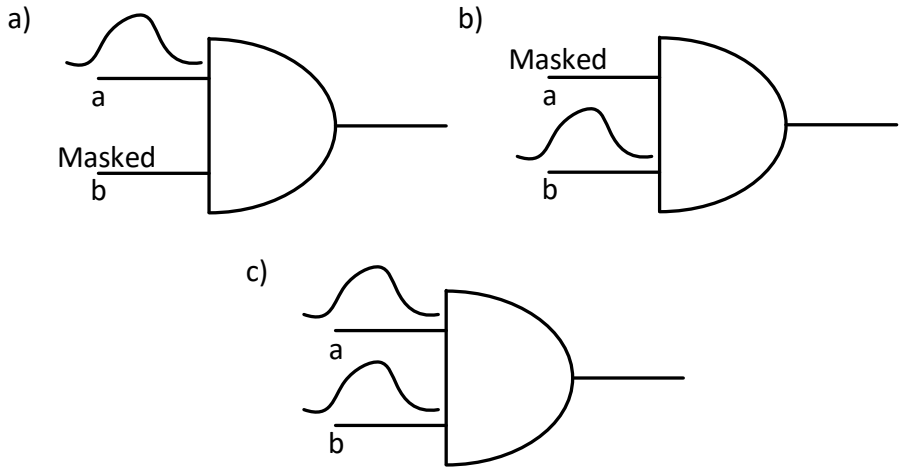


Figure 3.3: (a) A transient pulse arriving on input a and masked on b. (b) A pulse arriving on input b and masked on a. (c) Two pulses arriving on the inputs simultaneously.

Table 3.2 gives the propagation functions $F(G)$ for each case with $F(a_p)$ and $F(b_p)$ being the functions representing the input values that allow for the pulse to be propagated to inputs a and b respectively. While this example is only for a two input gate, larger gates can be considered similarly.

Based off equation 3.2, the total number of cases can be calculated as the number of combinations assuming $i = 1, 2, ...N$ simultaneous pulses. Assuming that P_{num} is the

Table 3.2: Functions for a MET Arriving on the Inputs of a NAND Gate

Case	Propagation Function ($F(G)$)
a	$F(G) = F(a_p) \wedge \neg F(b_p)$
b	$F(G) = \neg F(a_p) \wedge F(b_p)$
c	$F(G) = F(a_p) \wedge F(b_p)$

number of possible cases and r_i represents a i number of concurrent pulses, the equation for the total number of propagation cases is given below:

$$P_{num} = \sum_{i=1}^{N} \frac{N!}{(N - r_i)!} \tag{3.5}$$

Based on equation 3.5, the total probability of multiple convergent pulses at a gate G ($P(G_{tot})$) with the assumption that $P(G_i)$ is the probability of case i on gate G is given below:

$$P(G_{tot}) = \sum_{i}^{P_{num}} P(G_i) \tag{3.6}$$

When a MET is generated, it is represented as an event E_k with k being the event number. For each event, the generated transients are propagated through gates to the primary outputs. Once the pulse arrive at the gate output, the temporal masking probability can be considered. Based off the equation found in [26], let $P_{L,i}$ is the probability of the pulse being latch on gate i, W be the pulse width, t_{setutp} and t_{hold} being the setup and hold times respectively and T_{clk} be the clock period, the probability of a pulse being latch on an output flip-flop is calculated in the below equation:

$$P_{L,i} = \frac{W - (t_{setup} + t_{hold})}{T_{clk}} \tag{3.7}$$

Assuming that the pulses from event E_k propagate to a M number of primary outputs with each gate having a probability of $P_{L,i}$ calculated as in equation 3.7, the probability of

error for event E_k is calculated as the following:

$$P(E_k) = \sum_{i=1}^{M} P_{L,i} \tag{3.8}$$

To evaluate the error probability for all events, the mean error susceptibility (MES), defined in [21] is used. The MES represents the average probability of error for all events combined. Assuming that there are a n_E number of events, n_d number of input probability distributions and that K is the number of injected events, the MES is found using the following equation:

$$MES = \sum_{k=1}^{K} \frac{P(E_k)}{n_E n_d} \tag{3.9}$$

Based off the MES, the soft error rate (SER) can be calculated as in below where R_{eff} represents the effective concentration of particle in the given area, P_{eff} is the probability of a particle hitting the sensitive region of a transistor and A as the area of the sensitive volumes.

$$SER = MES * R_{eff} * P_{eff} * A \tag{3.10}$$

3.2 DESCRIPTION OF THE FAST_MET SIMULATOR

FAST_MET is a fast and accurate tool to estimate the soft error rate. The method aims to provide a good trade-off between simulation time and accurate SER calculation. To accurately determine the logical masking effects, BDDs are used. A BDD is a tree structure which can be used to represent a Boolean function. The first instance of the use of BDDs for function representation was proposed in [2] and has continued to be used regularly in circuit simulation and synthesis since. Fig. 3.4 gives the representation of a AND gate as BDD. Note that the structure consists of nodes which represent variables. For each node, there are two branches, one denoting a "true" value and the other a "false" value. The

BDD graph also consists of multiple paths to two terminating nodes, one with a "true" value denoted by "1" and one with a "false" value denoted by "0".

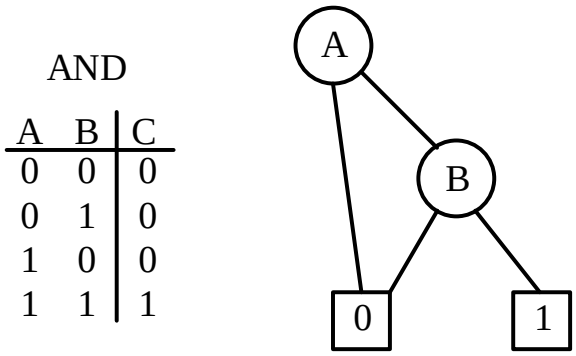


Figure 3.4: The truth table and corresponding BDD for an AND gate.

The main advantage of a BDD is that they provide an efficient canonical form to represent a Boolean function. Specifically, this means that Boolean operations (such as AND, NAND, OR ... etc) can be easily and quickly applied to the graph. In the case of the FAST_MET simulator, BDD representation of the functions are used to both calculate the signal probability and the probability for pulse propagation.

In FAST_MET, the goal is to accurately and efficiently represent a sensitization function as a BDD. A sensitization function is defined as the Boolean function that allows for a generated transient pulse to propagate through a circuit. In other words, a sensitization is the function that represents all input patterns that sensitize the pulse to the output.

The FAST_MET simulator operates in a topological order. This implies that the primary inputs will be visited first. Once a primary input is visited, it is represented as a variable for in a BDD. Once all inputs are visited, the simulator starts on the gates within the circuit. When a gate is visited, the sensitization function representing the gate output is calculated. The "1" terminal on this BDD represents the Boolean function that makes the output a "1" value. To increase efficiency, the FAST_MET simulator generates a rising and falling pulse at each gate using the method proposed in Chapter 2 for a specific energy.

When a pulse is generated, the gate input functions are considered using the input sensitization functions. If the pulse is rising (starting from a low value then going high), the pulse is only generated when the output is low. To consider this, the BDD function is inverted such that a termination to "1" represents that case at which the pulse exists. Similarly, if the pulse is falling, the output must be a high value. Since the function already denotes a high value on the output, the sensitization function can be used directly. Additionally, as discussed in Section 3.1, in the case of a gate having multiple inputs, the output function must be considered according to Table 3.1. Using the table, the correct generation function can be found by applying the function for the gate specified. Fig. 3.5 provides an example of the creation of the generation functions for a gate.

To consider the MET case, the location and correlation between the pulses must be considered. As discussed in [5], layout information is important in accurately modeling the location of the MET for a real design since gates that are near each other in a netlist may not necessarily be close in the layout. In this section, for the sake of simplicity, METs are generated using close proximity nodes from the netlist. For the sake of simple comparison and analysis, using the netlist is not problematic.

To determine the logical masking effects of a MET, the correlation between the inputs of both struck gates are found by using the logical AND operation between the generation functions derived as in Table 3.1. The logic behind this operation is that all pulses must be logically sensitized concurrently. Since the functions may share common inputs, they may have dependencies that must be considered.

In the case of pulse propagation, the sensitization functions are represented as BDDs. As discussed in Section 3.1, a pulse will propagate through a gate if all off-inputs have a non-controlling value. As can be observed in equation 3.4, the functions representing the case at which the off-input has a non-controlling value are logically AND'd together. The basis behind this idea is that all off-inputs must be non-controlling in order for the pulse to be propagated.

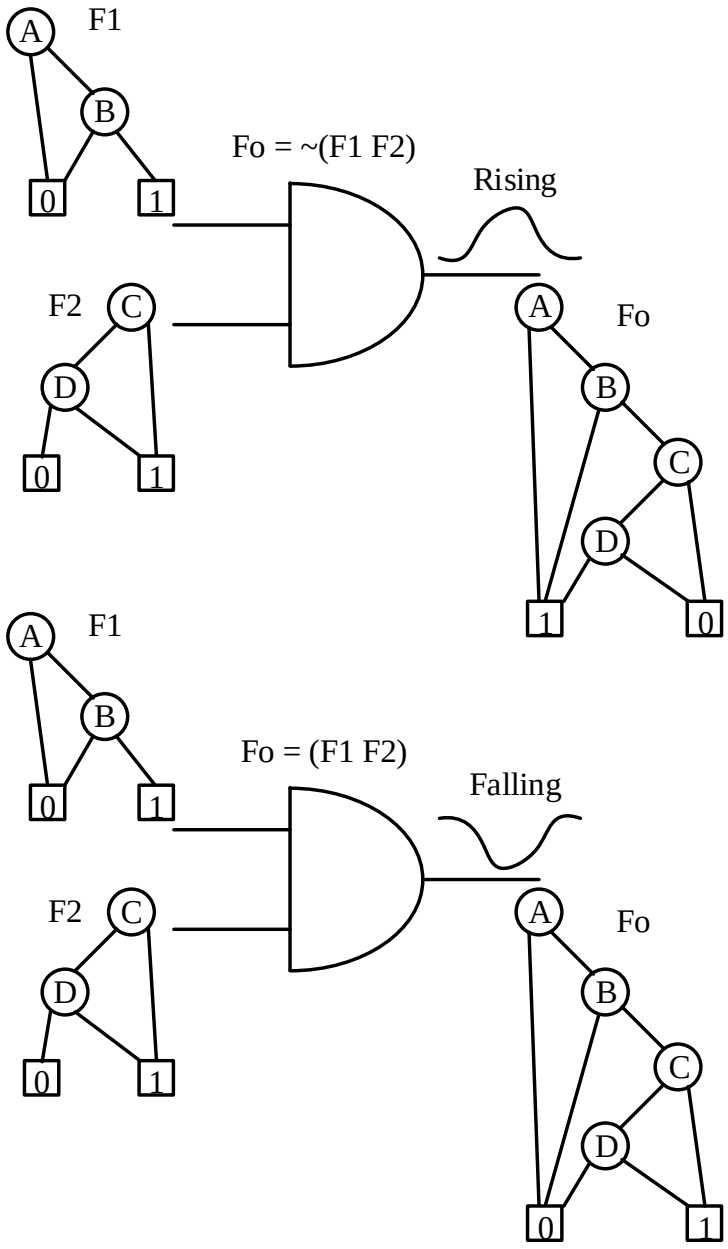


Figure 3.5: An example of the BDD creation for a rising and falling pulse.

To consider pulse propagation in the FAST_MET simulator, the logic functions are first represented in a BDD. A logic function is the Boolean function which provides the variable states that allow a "1" and a "0" value. The sensitization function for the gate is then determined by using the logical AND operation on each off-input BDD. If the non-controlling value for the gate is a "0", such as in a OR gate, the BDDs are inverted so that the "1" terminal represents the case at which the pulse propagates. If the non-controlling value is a "1" the BDDs are left unchanged. The result of this operation is a single BDD in which the paths that lead to the "1" terminal represent all of the input patterns that allow the pulse to propagated while the paths that lead to the "0" terminal represent the patterns the will mask the pulse.

In the case of multiple pulses arriving at a gate simultaneously referred to a convergent pulses, the off-inputs are considered as discussed previously. As shown in Table 3.2, a MET can result in multiple output cases which must be considered. The first case is where only one pulse will propagate which is calculated as previously discussed. All remaining cases include the circumstances where multiple pulses will arrive at the gate input simultaneously. In these cases, the logic function is determined by using the AND operation on the input function which have a pulse. The resulting function from this operation is then AND'd with the non-controlling off-input logic functions. The basis behind this idea is that the pulse sensitization functions give the cases where a pulse will arrive at the gate input. If two pulses arrive simultaneously, both pulses must be sensitized concurrently. The logical AND states that both functions must evaluate to true while all off-inputs must be non-controlling for the pulse to be propagated.

3.3 FAST_MET SIMULATION FLOW

The goal of the FAST_MET simulator is to calculate the SER of a circuit. To do this, the simulator must inject pulses in each gate. The main difficulty for this type of simulation is that for even relatively small circuits, it may take an intractable amount of simulation

time and memory to consider all pulses. The FAST_MET simulator provides a good trade off compared to existing methods since it uses the pulse approximation method in Chapter 2 in conjunction with BDDs to accurately and quickly determine the SER.

The first step for the FAST_MET simulator is to partition the circuit. As shown in [35, 21], the use of BDD functions can lead to memory blow up. To alleviate this issue, the FAST_MET partitions the circuit into a user defined number of parts. It is shown in Section 3.4 that the use of partitioning can substantially reduce the simulation time at a cost of accuracy in the SER estimation. For this work, the circuits were partitioned using the Fiduccia and Mattheyses (FM) algorithm [11] which allows for a circuit to be partitioned into two parts in linear time. To achieve a k part partition, the FM algorithm can be applied recursively a k-1 number of times. This algorithm was chosen based off its linear time complexity and relatively easy integration into the FAST_MET tool. Through proper integration, one can also use other partitioning tools such as hMetis [19]. Partitioning the circuit improves the performance by reducing the size of the logic and pulse sensitization BDDs. For moderately large circuits or larger, partitioning must be used since the BDD functions are too large and complex for realistic simulation.

After partitioning, the FAST_MET simulator processes the primary inputs of the circuit. At each input node, the signal probability is set and the BDD variables are initialized such that each input is a variable. In this work, it is assumed that there are not pulses on the inputs, however, especially if the method is used for the consideration of sequential circuits such as in [22], pulses can be injected on the inputs. This is due to the inputs being connected to the output of a flip-flop which drives the circuit.

After the inputs are processed, the internal nodes are visited in a topological order. At each node, at least one pulse is injected for a given charge. The particle effect is modeled as a current pulse using equation 1.1. A current source is placed on the output of the gate to simulate the particle effect and the resulting pulse shape is determined using the method discussed in Chapter 2. Multiple pulses can be injected concurrently in a gate, however, the

simulation time and memory resources increase substantially as more pulses are injected.

Additionally, FAST_MET is able to simulate the effects of a MET. To accommodate this, each injected pulse or MET is given an event number k to create an event E_k. This is important since FAST_MET injects pulses at each gate concurrently to speed up the SER characterization. Thus, when an MET is generated, a pulse is created on two or more gates which all have the same event number.

Once the pulses are injected and the corresponding BDD function are determined, FAST_MET propagates all pulses found on the gate inputs. Each fanin gate is checked individually for pulses. If a pulse is found the propagation function is determined as in Section 3.2. If the function does not evaluate to a false value, implying that there are no input patterns that allow the pulse to be propagated, the pulse is not considered further. However, if the function is not false, the output pulse shape is calculated. This operation continues for each pulse found on the inputs.

When a node is on the edge of a partition circuit, the effects of the previous partition must be considered. To do this, each node on the output edge of a partition is evaluated to determine their probability. This value is stored with the pulse which is propagated between the partitions. Since a partition may be separated from the primary inputs that drive it, virtual inputs are created which contain all pulses and their probabilities and the probabilities from the logic functions of the gate that was connected to input on the previous partition. Using these values, the probabilities are multiplied by the evaluated probability of the function to determine the overall probability. This is demonstrated in Fig. 3.6.

An issue that was found through numerous simulations using this method is that the functions will not evaluate to a false value even if the pulse does not propagate since a single partition does not consider the whole circuit. This, in effect, substantially reduces the number of pulses that are attenuated which can offset any improvement increases gained through partitioning. To solve this issue, it was found that setting a minimum probability

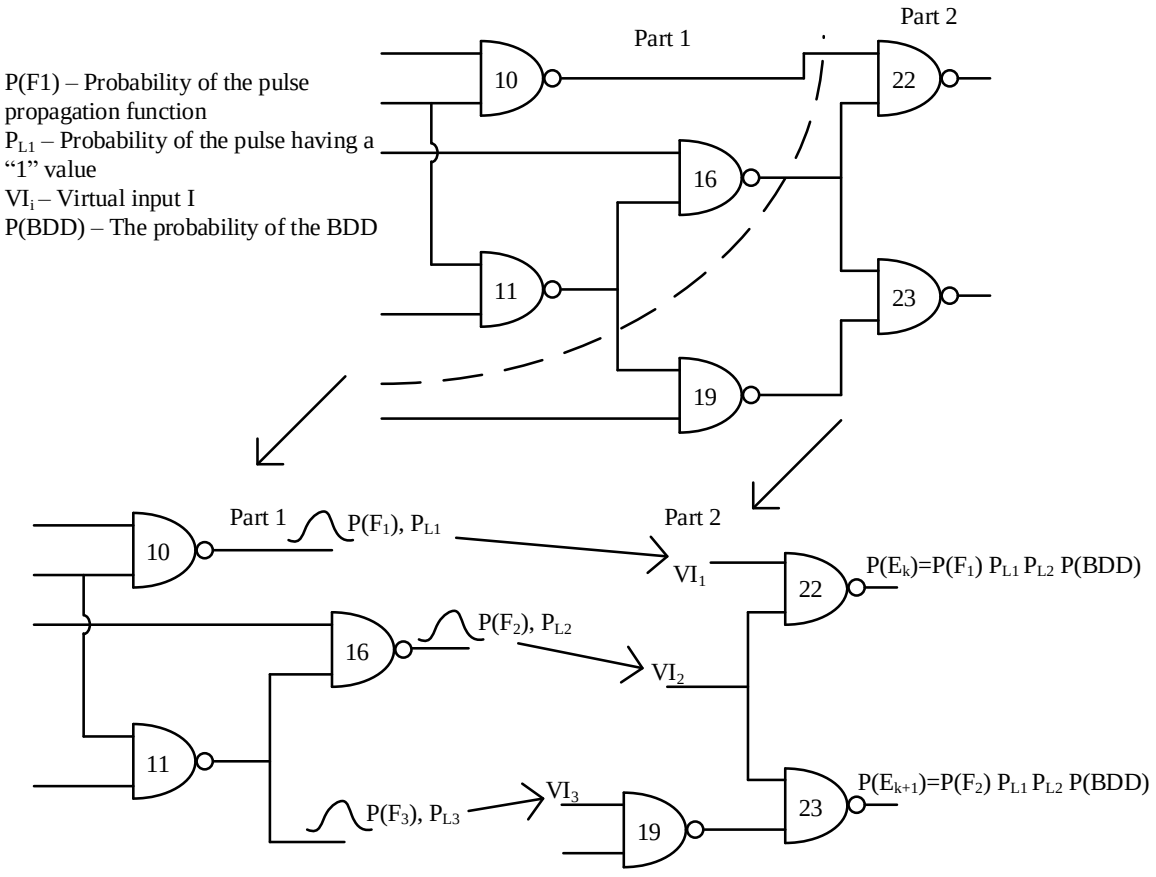


Figure 3.6: An example on c17 showing how the pulses are propagated between partitions.

threshold (P_{thr}) in which only pulses with a probability greater than this number are considered substantially saved simulation time with a slight increase in estimation error.

Once each individual pulse is propagated, a routine searches the inputs pulses and organizes them into a has table based on their event number. For each event E_k, the pulses are checked for overlap. If the pulses do overlap, they are treated as convergent pulses and are considered as previously discussed. Once issue that arises during this step is that there may be many possible cases in which the pulses overlap. For example, there may be 2, 3 or more concurrent overlapping pulses. In this work, to reduce the simulation time, only two pulses are considered. While the electrical masking model can accurately consider the effects of 3 or more convergent pulses, it is assumed that the likelihood of the pulses propagating is low.

Lastly, when the simulator processes a circuit or partition output, the BDD functions for each pulse are solved to determine the probability. Using the probability the MES is determined as discussed in 3.1. An overview of the whole simulation flow is given in Algorithm 1.

3.4 RESULTS

The FAST_MET simulator was test on the ISCAS 85 combinational benchmark circuits. FAST_MET was implemented in C++ on a machine with a quad core i7 with super-scaling and 16GB of RAM. The transistor look-up tables were characterized in HSPICE using the 32nm Predictive Technology Model (PTM) [38] where V_{dd} was set to 1.05V. The unit gate capacitance was set to a constant capacitance of 2 fF to simulate the loading effects of the gates and interconnects. To calculate the MES, a single pulse with an energy of 15 fC was applied to each gate using equation 1.1 with τ being set to $32x10^{-15}$. It was assumed that each output is connected to a flip-flop implemented in the same process library with a setup time (t_{setup}) of 22 ps and a hold time (t_{hold}) of -7 ps in accordance to [25]. For all simulations, the probability of MET was set to 10% and the error threshold

Algorithm 1 FAST_MET

```
1: Set Process Parameters
2: Parse Netlist
3: Organize Circuit Topologically
4: Parse Circuit Into k Parts
5: for Each Partition do
6:    Extract Partition
7:    Set Virtual Inputs
8:    for Each Gate do
9:       Generate Pulse
10:      Generate Sensitization Function
11:      Propagate Pulse
12:      Calculate Convergence
13:      if Primary Output then
14:         Calculate P_{L,i}
15:      else if Edge of Partition then
16:         Solve BDD Probability
17:      end if
18:   end for
19: end for
```

Algorithm 2 Generate Pulse

```
1: F = Calculate Generation Function
2: if F != false then
3:    Calculate Pulse Shape
4:    Add To Pulse List
5: end if
```

Algorithm 3 Propagate Pulse

```
1: F_sens = Sensitization Function
2: for Each Input Pulse i do
3:    F_sens = F_sens(i) ∧ F_sens
4: end for
5: if (F_sens != false) ∧ (Solve BDD(F_sens) ¿ P_thr) then
6:    Calculate Pulse Shape
7:    Add To Pulse List
8: end if
```

Algorithm 4 Calculate Convergence

```
1: Load Hash Table For Each Event
2: for Each Event do
3:    I = Check For Overlap
4:    if I == true then
5:       F = Calculate Convergence Function
6:       if (F != false) ∧ (Solve BDD(F) ¿ P_thr) then
7:          Calculate Pulse Shape
8:          Add To Pulse List
9:       end if
10:   end if
11: end for
```

for simulation P_{thr} was set to 0.025 when partitioning was used. When an MET occured, it was injected to a neighboring gate based on the netlist.

First, the accuracy of FAST_MET without partitioning is compared to Monte Carlo simulation on c17 and c880 to ensure that the probablistic functions provided here exactly compute the logical masking effect. To ensure that both simulators use the same pulses, the pulse injection routine in FAST_MET was moved to before the main simulation loop. Larger circuits were not tested since they lead to long simulations times and lead to the possibility of memory blow up for FAST_MET without the use of partitioning. As can be observed in the Table 3.3, FAST_MET provides exact estimation of the output error with an over 5x speedup.

Table 3.3: Comparision of FAST_MET vs Monte Carlo

Circuit	Simulator	MES	Run Time
c17	Monte Carlo	$3.92x10^{-4}$	0.429
c880	Monte Carlo	$4.15x10^{-4}$	9110.35
c17	FAST_MET	$3.92x10^{-4}$	0.0974
c880	FAST_MET	$4.15x10^{-4}$	1909.37

The FAST_MET simulator can provide an additional speedup through the use of partitioning. Table 3.4 provides the MES and the simulation time for various ISCAS circuits while changing the partition size. According to the results, it shows that partitioning can

provide an upward of 20x reduction in simulation time at a cost of accuracy compared to not using partitioning. Based off the results, it is shown that increasing the number of partitions follows the law of diminishing returns. For example in c880, the use of 2 partitions reduces the simulation time by 13x while the use of 4 partitions reduces the simulation time by 18x. While this is still a large decrease in simulation time, the error is increased by 4x. Additionally, it can be seen that in c17 the use of partitioning actually increases the simulation time. This is due to the time to simulate the circuit is less than the overhead incurred by the partitioning algorithm. Based on this result, it is suggested that the partition size is carefully selected to ensure that the error is minimized.

Table 3.4: FAST_MET Performance vs Number of Partitions

Circuit	Parts	MES	Run Time
c17	2	$3.83x10^{-4}$	0.1377
c880	2	$3.13x10^{-4}$	139.54
c880	4	$8.37x10^{-4}$	107.72
c1355	2	$1.96x10^{-4}$	459.00
c1355	4	$2.85x10^{-4}$	242.60
c1355	8	$3.67x10^{-4}$	167.961

3.5 CONCLUSION

In this chapter, the FAST_MET simulator is proposed. First, probabilistic equations were provided which allow for accurate consideration of all propagation effects. These equations were then represented as BDDs and used to determine the logical masking effect. In the results it was shown that the tool provides a good trade-off between simulation time and accuracy due to the use of partitioning and an accurate masking model. Additionally, it is shown that the number of partitions reduces the simulation time up 18x compared to not using partitioning and 90x compared to Monte Carlo simulation.

REFERENCES

[1] D. R. Blum and J. G. Delgado-Frias. Schemes for eliminating transient-width clock overhead from set-tolerant memory-based systems. *IEEE Transactions on Nuclear Science*, 53(3):1564–1573, June 2006.

[2] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.

[3] T. Calin, M. Nicolaidis, and R. Velazco. Upset hardened memory design for submicron cmos technology. *IEEE Transactions on Nuclear Science*, 43(6):2874–2878, Dec 1996.

[4] Liang Chen, Mojtaba Ebrahimi, and Mehdi B. Tahoori. Cep: Correlated error propagation for hierarchical soft error analysis. *Journal of Electronic Testing*, 29(2):143–158, 2013.

[5] M. Ebrahimi, H. Asadi, R. Bishnoi, and M. B. Tahoori. Layout-based modeling and mitigation of multiple event transients. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):367–379, March 2016.

[6] N. Eftaxiopoulos, N. Axelos, and K. Pekmestzi. Donut: A double node upset tolerant latch. In *2015 IEEE Computer Society Annual Symposium on VLSI*, pages 509–514, July 2015.

[7] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco. Estimate of signal probability in combinational logic networks. In *European Test Conference, 1989., Proceedings of the 1st*, pages 132–138, Apr 1989.

[8] M. Fazeli, S. N. Ahmadian, S. G. Miremadi, H. Asadi, and M. B. Tahoori. Soft error rate estimation of digital circuits in the presence of multiple event transients (mets). In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011.

[9] M. Fazeli, S. N. Ahmadian, S. G. Miremadi, H. Asadi, and M. B. Tahoori. Soft error rate estimation of digital circuits in the presence of multiple event transients (mets). In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011.

[10] M. Fazeli, S. G. Miremadi, A. Ejlali, and A. Patooghy. Low energy single event upset/single event transient-tolerant latch for deep submicron technologies. *IET Computers Digital Techniques*, 3(3):289–303, May 2009.

[11] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, DAC '82, pages 175–181, Piscataway, NJ, USA, 1982. IEEE Press.

[12] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi. A stochastic computational approach for accurate and efficient reliability evaluation. *IEEE Transactions on Computers*, 63(6):1336–1350, June 2014.

[13] R. Harada, Y. Mitsuyama, M. Hashimoto, and T. Onoye. Neutron induced single event multiple transients with voltage scaling and body biasing. In *Reliability Physics Symposium (IRPS), 2011 IEEE International*, pages 3C.4.1–3C.4.5, April 2011.

[14] P. Hazucha, T. Karnik, S. Walstra, B. Bloechel, J. Tschanz, J. Maiz, K. Soumyanath, G. Dermer, S. Narendra, V. De, and S. Borkar. Measurements and analysis of ser tolerant latch in a 90 nm dual-vt cmos process. In *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, pages 617–620, Sept 2003.

[15] K. Katsarou and Y. Tsiatouhas. Soft error immune latch under seu related double-node charge collection. In *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)*, pages 46–49, July 2015.

[16] K. Katsarou and Y. Tsiatouhas. Soft error interception latch: double node charge sharing seu tolerant design. *Electronics Letters*, 51(4):330–332, 2015.

[17] Ji Li and Jeffrey Draper. Accelerating soft-error-rate (ser) estimation in the presence of single event transients. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 55:1–55:6, New York, NY, USA, 2016. ACM.

[18] S. Lin, H. Yang, and R. Luo. High speed soft-error-tolerant latch and flip-flop design for multiple vdd circuit. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, pages 273–278, March 2007.

[19] Pekka Miettinen, Mikko Honkala, and Janne Roos. Using metis and hmetis algorithms in circuit partitioning.

[20] N. Miskov-Zivanov and D. Marculescu. Mars-c: modeling and reduction of soft errors in combinational circuits. In *2006 43rd ACM/IEEE Design Automation Conference*, pages 767–772, July 2006.

[21] N. Miskov-Zivanov and D. Marculescu. Multiple transient faults in combinational and sequential circuits: A systematic approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1614–1627, Oct 2010.

[22] Natasa Miskov-Zivanov and Diana Marculescu. Mars-s: Modeling and reduction of soft errors in sequential circuits. In *Proceedings of the 8th International Symposium on Quality Electronic Design*, ISQED '07, pages 893–898, Washington, DC, USA, 2007. IEEE Computer Society.

[23] P. C. Murley and G. R. Srinivasan. Soft-error monte carlo modeling program, semm. *IBM J. Res. Dev.*, 40(1):109–118, January 1996.

[24] M. Nicolaidis, R. Perez, and D. Alexandrescu. Low-cost highly-robust hardened cells using blocking feedback transistors. In *26th IEEE VLSI Test Symposium (vts 2008)*, pages 371–376, April 2008.

[25] C. Nunes, P. F. Butzen, A. I. Reis, and R. P. Ribas. A methodology to evaluate the aging impact on flip-flops performance. In *Integrated Circuits and Systems Design (SBCCI), 2013 26th Symposium on*, pages 1–6, Sept 2013.

[26] M. Omana, G. Papasso, D. Rossi, and C. Metra. A model for transient fault propagation in combinatorial logic. In *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pages 111–115, July 2003.

[27] M. Omana, D. Rossi, and C. Metra. High-performance robust latches. *IEEE Transactions on Computers*, 59(11):1455–1465, Nov 2010.

[28] Ramin Rajaei, Mahmoud Tabandeh, and Mahdi Fazeli. Single event multiple upset (semu) tolerant latch designs in presence of process and temperature variations.

Journal of Circuits, Systems and Computers, 24(01):1550007, 2015.

[29] R. Rajaraman, J. S. Kim, N. Vijaykrishnan, Y. Xie, and M. J. Irwin. Seat-la: a soft error analysis tool for combinational logic. In *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*, pages 4 pp.–, Jan 2006.

[30] R. R. Rao, K. Chopra, D. T. Blaauw, and D. M. Sylvester. Computing the soft error rate of a combinational logic circuit using parameterized descriptors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):468–479, March 2007.

[31] D. Rossi, M. Omana, F. Toma, and C. Metra. Multiple transient faults in logic: an issue for next generation ics? In *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, pages 352–360, Oct 2005.

[32] F. Wang and Y. Xie. Soft error rate analysis for combinational logic using an accurate electrical masking model. *IEEE Transactions on Dependable and Secure Computing*, 8(1):137–146, Jan 2011.

[33] A. Watkins and S. Tragoudas. Transient pulse propagation using the weibull distribution function. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 109–114, Oct 2012.

[34] A. Yan, H. Liang, Z. Huang, and C. Jiang. High-performance, low-cost, and highly reliable radiation hardened latch design. *Electronics Letters*, 52(2):139–141, 2016.

[35] Bin Zhang, Wei-Shen Wang, and M. Orshansky. Faser: fast analysis of soft error susceptibility for cell-based designs. In *7th International Symposium on Quality Electronic Design (ISQED'06)*, pages 6 pp.–760, March 2006.

[36] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel. Sequential element design with built-in soft error resilience. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(12):1368–1378, Dec 2006.

[37] M. Zhang and N. R. Shanbhag. Soft-error-rate-analysis (sera) methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2140–2155, Oct 2006.

[38] Wei Zhao and Yu Cao. Predictive technology model for nano-cmos design exploration. *J. Emerg. Technol. Comput. Syst.*, 3(1), April 2007.

[39] J. F. Ziegler. Terrestrial cosmic rays. *IBM Journal of Research and Development*, 40(1):19–39, Jan 1996.

APPENDICES

APPENDIX I

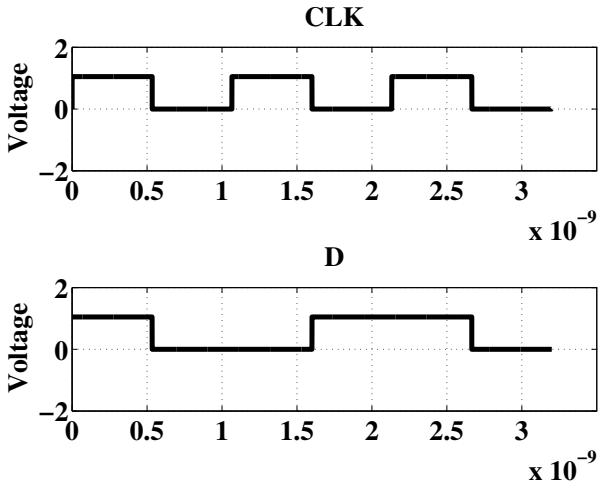


Figure 3.7: Waveforms for CLK and D.

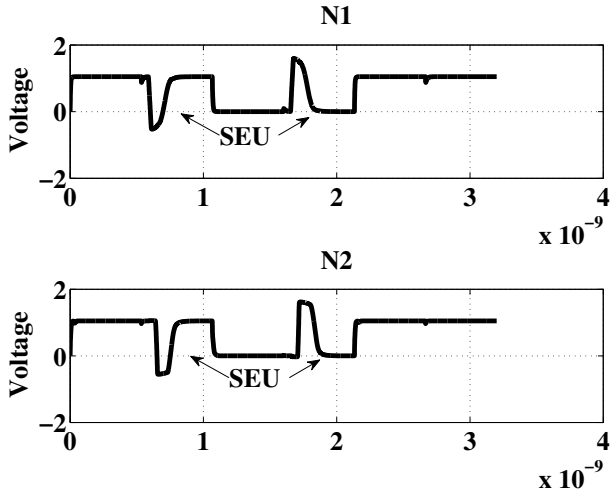


Figure 3.8: Node pair n1 and n2 upset and recovery.

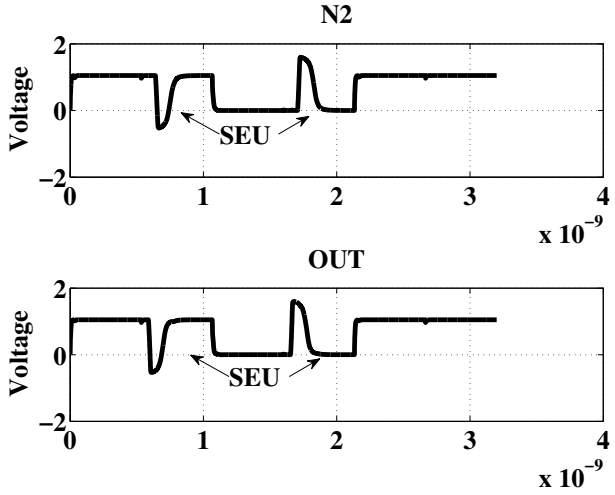


Figure 3.9: Node pair n2 and out upset and recovery.

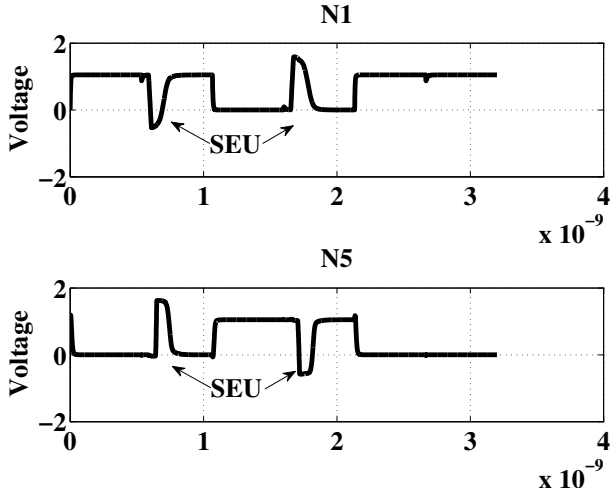


Figure 3.10: Node pair n1 and n5 upset and recovery.

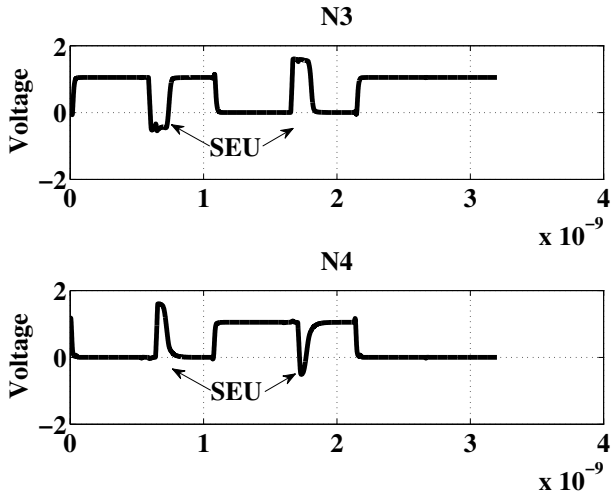


Figure 3.11: Node pair n3 and n4 upset and recovery.

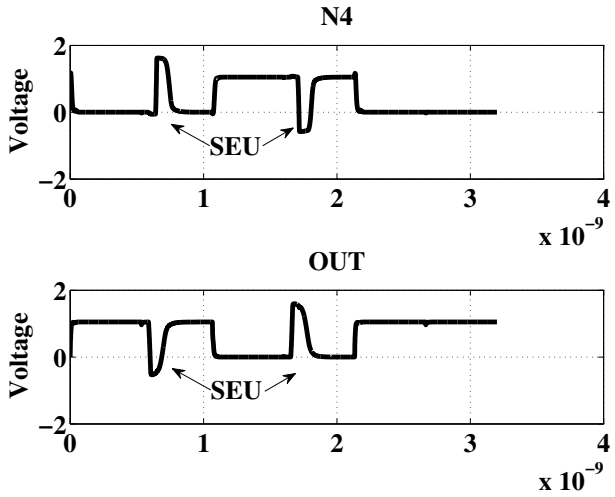


Figure 3.12: Node pair n4 and out upset and recovery.

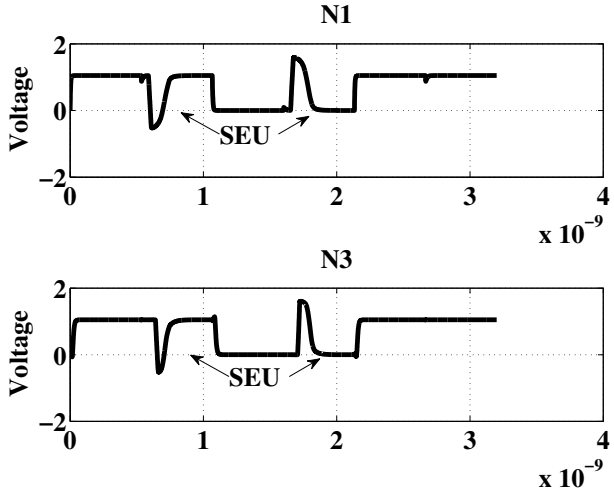


Figure 3.13: Node pair n1 and n3 upset and recovery.

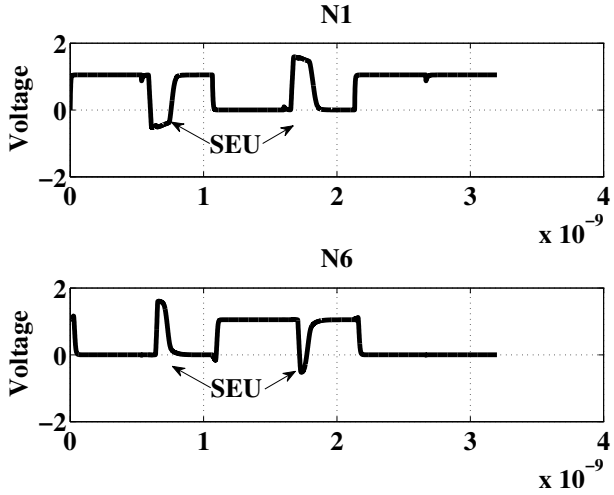


Figure 3.14: Node pair n1 and n6 upset and recovery.

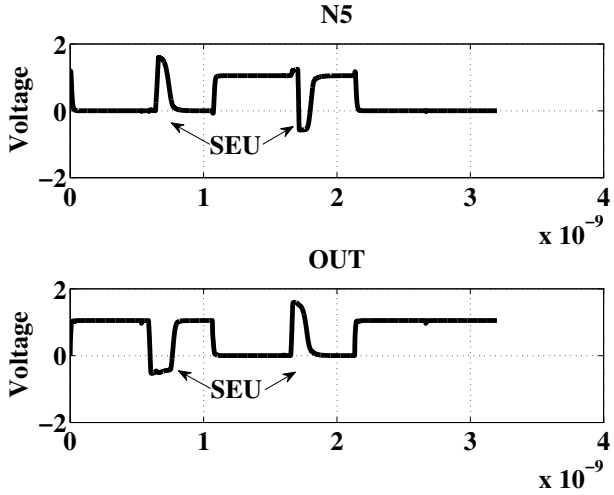


Figure 3.15: Node pair n5 and out upset and recovery.

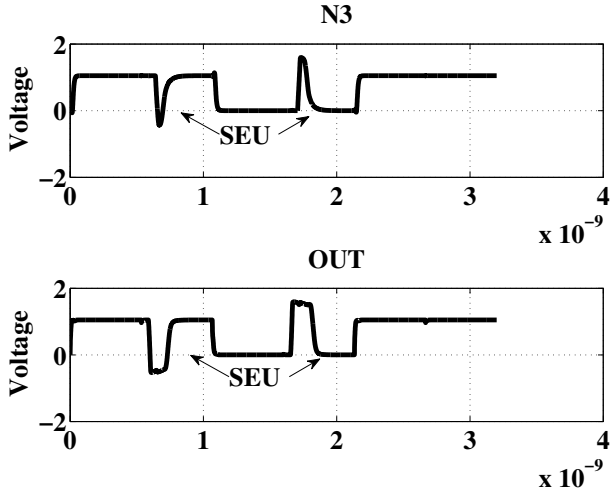


Figure 3.16: Node pair n3 and out upset and recovery.

VITA

Graduate School
Southern Illinois University

Adam Watkins Date of Birth: January 27, 1988

4158 Sycamore Unit A, Los Alamos, New Mexico, 87544

Email: acwatkins88@gmail.com

Southern Illinois University at Carbondale
Master's of Science, Electrical and Computer Engineering, May 2012

Southern Illinois University at Carbondale
Bachelor of Science, Electrical and Computer Engineering, May 2010

Special Honors and Awards: Best Student Paper Award, DFT 2016

Research Paper Title:
Analysis and Mitigation of Multiple Radiation Induced Errors in Modern Circuits

Major Professor: Dr. S. Tragoudas

Publications:

Anton, H., *Elementary Linear Algebra*, John Wiley & Sons, New York, 1977.

Huang, X. and Krantz, S.G., *On a problem of Moser*, Duke Math. J. **78** (1995), 213–228.