Spring 5-25-2021

Fault Modeling and Test Vector Generation for ASIC Devices Exposed to Space Single Event Environment

Ahmed Mohamed
ahmed-ibraahim@aucegypt.edu

The American University in Cairo

School of Science and Engineering

Fault Modeling and Test Vector Generation for ASIC Devices Exposed to Space Single Event Environment

A Thesis Submitted to
Electronics and Communications Engineering Department

In partial fulfillment of the requirements for the degree of Master of Science

by Ahmed Ibrahim Abdelhafeez Mohamed

Under the Supervision of Prof. Ahmed Abou-Auf

May/2021

ACKNOWLEDGMENT

I would like to express my sincere gratitude and appreciation to my advisor Prof. Ahmed Abou-Auf for his continuous support, guidance, and patience throughout my thesis.

I would like to thank my examiners Prof. Amr Wassal and Prof. Mohab Anis for devoting time to review the thesis and providing valuable comments.

Also, I would like to thank all my colleagues that offered help and support throughout the research work, especially Eng. Ayman Mohmed, Eng. Mohamed Wael and Eng. Mostafa Abdelaziz.

ABSTRACT

This work aims at providing a concise automated flow to predict the effect of Single Event Transients (SETs) on ASIC chips by developing a method to characterize the circuit susceptibility to SET pulses propagation and then generation of the required input vectors that sensitize the victim paths.

A new enhanced method for SET electrical propagation modeling is proposed and compared to a previously published analytical model. The method was applied on different standard cells libraries built over XFAB Xh018 technology and verified for accuracy against simulations. The new method showed enhancement in accuracy compared with previous work in literature.

Industrial ATPG tool for SET test vectors generation was used with some modifications to its native flow to fit the different nature of SETs. The proposed steps were tested using ISCAS ’85 benchmarks synthesized with the XFAB standard cells.

Table of Contents

Table of Figures

List of Tables

Chapter 1

Overview

The environment of operation of electronic devices can have an impact on the performance of the device. This work will be concerned with a certain effect of radiation environment, which is called the Single Event Transients and will be detailed in the next chapters. The motivation behind this work is that these effects can cause faulty behavior in some critical components that can affect the reliability of a complex systems. Such as navigation controls in airplanes or advanced electronic systems in satellites. This work main contributions are; 1) To present an approach to model single event transients propagation effects in the circuit and 2) to provide a methodology to determine the test vectors which helps the chip owners test and enhance their IPs against single event transients. The scope of this work is confined to ASIC flows to make use of the fact that it uses well defined standard cells that can be studied and characterized. Also, this flow studies the combinational part of the circuit, but it can still be extended to sequential circuits by partitioning the circuit between the sequential elements and treating terminals to sequential element as being primary inputs and outputs for the combinational circuit.

The rest of this thesis is organized as follows, in Chapter 2 a description of the effects that appear is a radiation environment and how they can degrade the performance of electronic circuits. Chapter 3 provides an explanation of single event transients from the perspective of its generation mechanisms and the factors affecting its behavior on the device level. Then in Chapter 4, exploration of previous efforts in modelling and simulation of SETs propagation at the circuit level. Chapter 5 presents the new proposed method for SET propagation modelling accompanying by comparison of its results with simulation and a previous analytical model. A background overview is presented in Chapter 6 for the common defects in VLSI systems and how they are handled using the widely used DFT techniques and fault models, in addition to brief description of Automatic Test Pattern Generation (ATPG) concepts. The proposed test vectors generation method is detailed in Chapter 7 with the results obtained for the used benchmarks. Finally, Chapter 8 concludes this thesis and provide the future work plans.

Chapter 2

Radiation Environment Effects

Radiation can harm electrical devices rendering them faulty or operating improperly. Many sources emit radiation, and these sources can be found on the Earth due to neutrons or alpha particles' presence, which is the leading cause of radiation [1]. Also, radiation is found in space because of the particles present in the Earth's atmosphere. These particles are produced from the sun and high-energy galactic cosmic rays. Radiation can be emitted from artificial sources such as nuclear plants, biomedical apparatus, and experimental equipment with high energy particles in physics.

As the electronic devices are subjected to an environment with most of the radiation sources such as terrestrial, space, and artificial sources, it is essential to analyze their features under this situation to let them operate without any impedance under this environment and to perform their functions effectively and efficiently.

As mentioned before, terrestrial sources of radiation are found due to neutrons' presence in the Earth's atmosphere and alpha particles produced from the damage of the integrated circuits (IC) that are commonly made of Uranium and other radioactive materials such as Thorium, Platinum, and Hafnium. Also, another source mentioned before is the space or the outer atmosphere of the Earth. Space radiation is present in three different types; particles trapped in the magnetic field of the Earth, particles that occur because of the solar flares, and Galactic Cosmic Rays (GCR), which are high energy protons and heavy ions [2]. These are located outside our solar system, and these types of radiation are called ionizing radiation as it is a type of radiation in which energy is transferred to materials objected to their source. Atmospheric Neutrons are produced due to the interaction between the GCR and the outer layer of the Earth's atmosphere. When the GCR enters the Earth's atmosphere, it interacts with oxygen and nitrogen, creating protons, muons, peons and neutrons. The concentration of these particles increases firstly and then decreases when they enter the Earth's atmosphere due to the shield's presence, which protects the Earth from most of the space radiations nearly equal

to 99%. There is a proportional relation between the altitude and the neutron flux or concentration [3], shown in Figure 2-1.

The graph in Figure 2-1 shows that the neutrons increase with the altitude till they reach a constant high value at an altitude of 15 Km, this is the altitude where the avionics electronic devices work, so this environment which has many neutrons interact destructively with these devices and let them work improperly. Also, there is another source of radiation rather than the neutrons, the alpha particles produced from the decay of radioactive materials such as Uranium, Thorium, Platinum, and Hafnium. These materials emit radiation (alpha particles) which causes errors and defects in the electronic

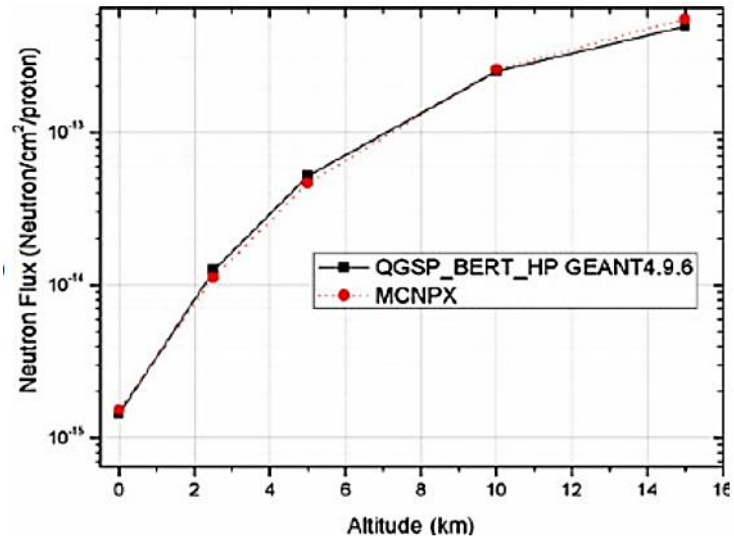


Figure 2-1 Relation between altitude and neutron flux [3]

devices. Alpha particles are causing defects and errors more than atmospheric neutrons, especially in electronic devices of small sizes. There are particles such as muons that cause defects and errors in Complementary Metallic Oxide Semiconductor (CMOS) as the device's dimensions decrease.

As mentioned above, there are three different types of radiation in space different from those found on Earth. These three types of radiation are:

1. particles trapped in the magnetic field of the Earth.
2. particles occur because of the solar flares (Solar Particle events).

3. Galactic Cosmic Rays (High energy Proton which moves with speed nears to the speed of light).

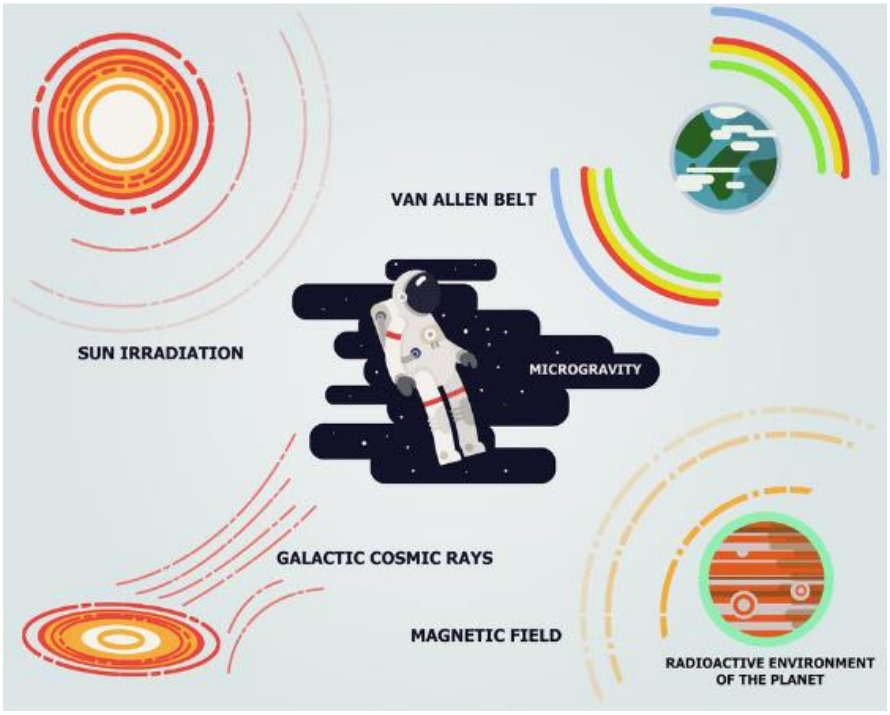


Figure 2-2 Different sources of external radiation [4]

Space is known as the harshest environment that badly affects the electronic devices as there is no defensive shield that could overcome or conquer these types of radiations, also as we get higher from the Earth's atmosphere, defensive shields decrease till it vanishes [4]. For the GCR, they are high energy protons with their surrounding electrons stripped away. Also, it is known as a dominant source of radiation. They traveled with a speed nearly equal to the speed of light. They come from outside the solar system, but their source is still unknown. GCR energy is very high as they can reach $10^{11} GeV$ which makes them quickly penetrating and ionizing any atom that intrudes its destination. Then the sun comes in the second level after the GCR in radiation as it is one of the sources of ionizing particles which can reach energies with a value greater than $10\ MeV$. This energy is produced because of the solar cycle's presence [5], which is the sun's magnetic field cycle that goes through approximately every 11 years. Every 11 years, the solar magnetic field flips. This means that the north pole and south pole switch

their positions, and after another 11 years, the magnetic field flips back again. Solar spots are produced because of the Solar cycle.

At the beginning of the cycle, the solar minimum occurs (the sun has few sunspots), and it starts to increase till it reaches the middle, where the solar maximum occurs (the sun has the most sunspots), then the sunspots decrease again to reach to the Solar minimum to begin another cycle. There is a giant eruption happening during the solar cycle, such as solar flares and coronal mass ejections. These eruptions produce a considerable amount of energy in space. This activity affects the Earth also will affect electronic devices and cause fatal damages to them.

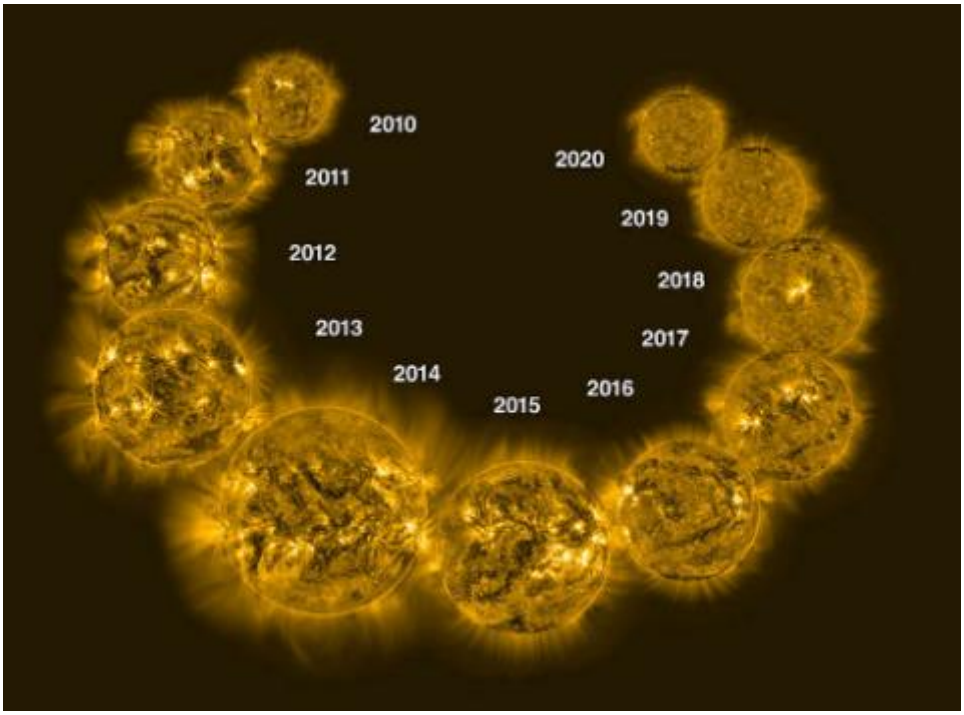


Figure 2-3 Evolution of the Sun in extreme ultraviolet light from 2010 through 2020 [5]

There are particles trapped in the Earth's atmosphere due to the interaction between Solar flares and Earth's magnetic field, also because of the high energy GCR, which penetrate and ionize atoms, producing particles that are trapped in the magnetic field of the Earth. These particles have energy with values up to $5MeV$ and $800MeV$, as they have been trapped, they move in a spiral way forming a radiation belt surrounding the Earth called the Van Allen Radiation belt. Van Allen radiation belt consists of two belts, outer belt which contains mostly electrons with energy up to $10MeV$ and the inner

building which contains both electrons and protons, photons have energy equal to $100MeV$ while the electrons have energy with the value of hundreds KeV.

The third and final source of radiation is the human-made artificial radiations such as biomedical devices and high energy particles in physics equipment, for example, the Large Hadron Collider (LHC) found at CERN in Switzerland, so the devices in this equipment as designed and manufactured in a way that overcomes and beat these amount of radiations. Nuclear reactor stations produce high concentrations of neutrons with an energy value up to $14MeV$ which will hit electronic devices found in this environment.

For analyzing and designing electronic devices in environments full of radiation, one must know the effect of the radiation mechanism on the target material. From this

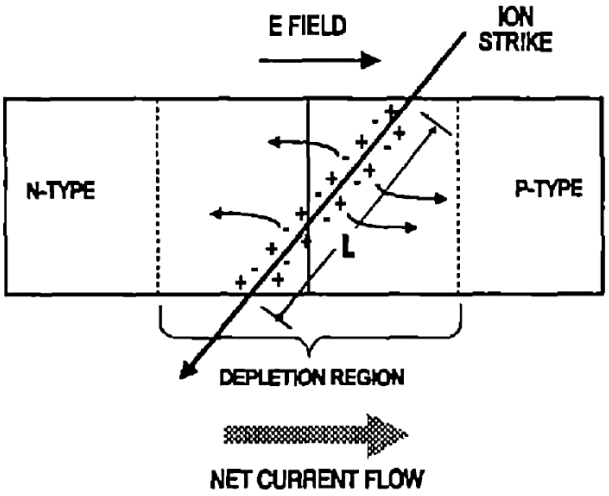


Figure 2-4 SET charge collection mechanism [6]

radiation mechanism, when a charged particle hits an atom, it produces an electron-hole pair in this atom according to Coulomb's force, but in metal, due to the free mobility of electrons, electron-hole pairs disappear immediately. The effect of radiation did not occur, but if the target material is semiconductor material, it depends on the applied electric field, an accumulation of holes can occur inside or on the surface of the dielectric material, causing collection of charges [6] and bet current flow which results in a single event transient pulse (SET) as shown in Figure 2-4. Another radiation effect mechanism is that the lattice shape of the target material changes due to the collision of the neutron and photons to the target material. These displacement or position changes can cause

dislocation loops which will finally cause a change in the features and characteristics of the target material [7], as shown in Figure 2-5.

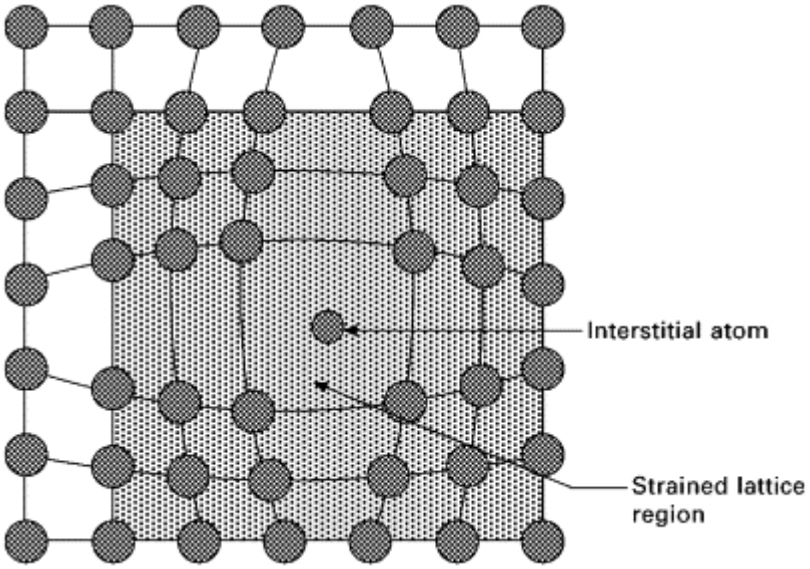


Figure 2-5 Lattice dislocations due to particle strike [7]

Chapter 3

SET Description and Modelling

3.1 Introduction

Integrated circuit performance is affected by the working environment of the chip. The circuit's sensitivity to external factors issue has become more prevalent due to the downscaling trend, especially in the downscaling of supply voltage which made the effect of SETs a critical concern for modern ICs [8]. This phenomenon is not confined to only CMOS circuits but can also be manifested in other technologies, like memristors, which are mainly used in the new memories technologies [9].

A Single Event Transient (SET) is generated on the impact of a heavy charged particle to an active area inside the silicon die. The particle can be an alpha particle, ion, proton, muon, neutron, or particle with considerable mass and charge. These types of particles usually exist in cosmic radiations or human-made radiations. Spontaneous emissions from the material used to fabricate the chip packages can result in heavy alpha particles and neutrons.

A SET appears electrically as a current pulse that propagates through the circuit, which can change the voltage levels of several circuit nodes. Usually, the SET does not cause physical damage to the circuit, and the circuit voltage levels are restored to normal operating points, given that the incident happened within a combinational part of the circuit. However, if that voltage perturbation is latched into a memory element erroneously, it causes a soft error, and the incorrect value saved into the sequential element is retained in the circuit, which causes what is called a Single Event Upset (SEU). An SEU is a type of data corruption that may result in unpredicted faulty behavior.

Single event effects can cause physical damage, as in the case of single-event gate rupture or latch-up in which the induced voltage is high enough to do permanent damage. This problem is simpler to be observed than SEU because it will render a device in a permanent faulty state in contrast to SEU, which is unpredictable.

3.2 Mechanisms of SETs

To be able to accurately model the effect of SET on integrated circuits, a good understanding of the physical mechanisms of SET incidents is required.

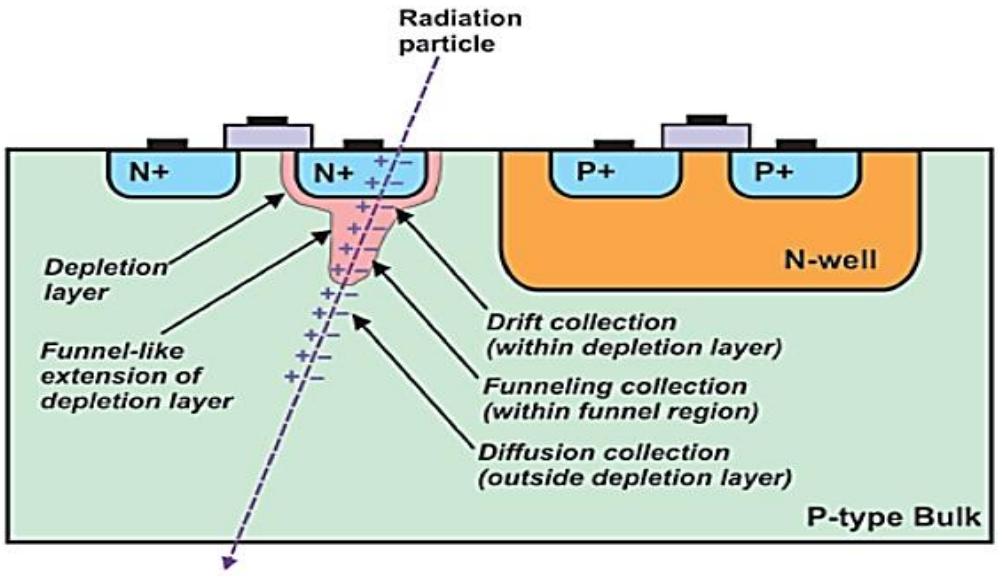


Figure 3-1 Particle propagation through active device and charge collection mechanisms [10]

The diffusion areas within p-MOS or n-MOS transistors are the most prone to ionization events. The flow of energized particles through the victim silicon area will deposit extra charges to the depletion areas around the active drain/source. Induced charge collection can be manifested through different mechanisms, which are shown in Figure 3-1. These different mechanisms affect the shape of the generated induced pulse. The induced current pulse width ranges between picoseconds orders to nanoseconds, while the amplitude can be from microamperes to milliamperes [10]. The shape of the generated pulse depends on the charge collection process. If the process is dominated by the drift or funneling process, the pulses will be narrower because the drift process mobility of charge carriers is high. On the other hand, the diffusion process has a much slower collection rate because the carriers are less mobile, which results in wider pulse width [11]. These mechanisms are defined by numerous parameters related to technology, circuit, and radiation parameters. The technology parameters include semiconductor doping concentration and profiles, carriers' mobility, and lifetime. In contrast, the circuit operation factors are such as temperature and supply voltage. Moreover, the radiation-

dependent parameters are the particle energy, its position, and direction of impact with the device [12].

3.3 SET Propagation

Considering a simple CMOS inverter circuit illustrated in Figure 3-2, if the particle strikes an n-MOS transistor while it is in OFF state, the charge deposited by the particle will open the channel between the source and drain so the transistor will draw current (I_{SET}) acting as if the device was ON temporarily and tried to pull the output node to 0 voltage instead of the correct output of 1 volt. Depending on the current drive of the p-MOS and load capacitance which acts as the restoration element, the duration of the erroneous output voltage level is determined. This duration is translated to a voltage pulse that can propagate to the next gate in the circuit. The amplitude of the voltage pulse may reach the gate threshold voltage of the devices, so it can flip a node from logic 0 to logic 1 or vice versa. The shape of the fault pulse can change during the propagation through the circuit, and it can be attenuated or amplified depending on the pulse characteristics and the gate. If the pulse gets attenuated while it is logically allowed to propagate, i.e., the Boolean logic of the gates allows for the pulse to be observable to the output, then this is called electrical masking. Hence, accurate modeling and characterization are needed to predict the effect of SETs on a circuit is necessary for fault-tolerant systems

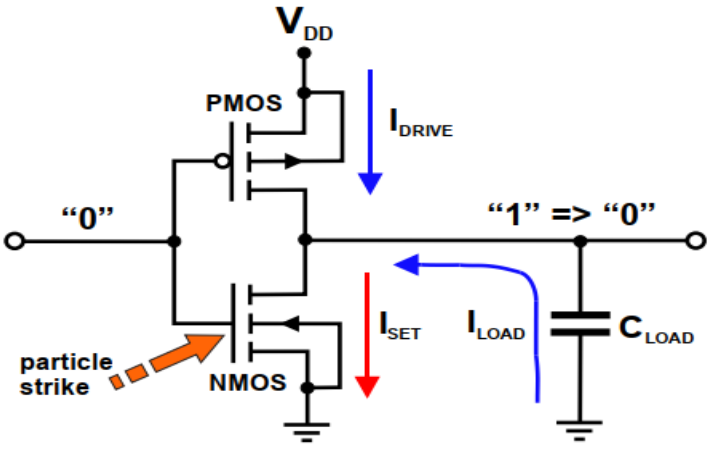


Figure 3-2 Effect of particle strike as current pulse generation [10]

and to provide insight for designers on how to harden their chips against radiation effects.

Another aspect of SET propagation through logic circuits is logical masking. Logical masking is the stopping of the pulse from the propagation through a gate because the input, on which the pulse is arriving, does not influence the output (non-observable) due to the logic function of the gate and the value of the other inputs. Figure 3-3 shows different situations where a pulse gets masked or not [13]. In case (a), the SET pulse is masked as the output of the AND gate will always be zero regardless of the input terminal that has the SET pulse, while the pulse can propagate through the gate to the output in case (b). Similar is for the OR gate in the (c) and (d).

Since the logic state is proven to affect the propagation of the pulse, then logical analysis or simulation is needed to identify the logic paths that can allow a SET propagation from a given input to a given output. This explains why the response of a circuit to SETs will be affected by the value of the input excitation. The process of identifying the SET-prone paths is called path sensitization and is a critical step during the process of test vector generation that will be detailed later in this work.

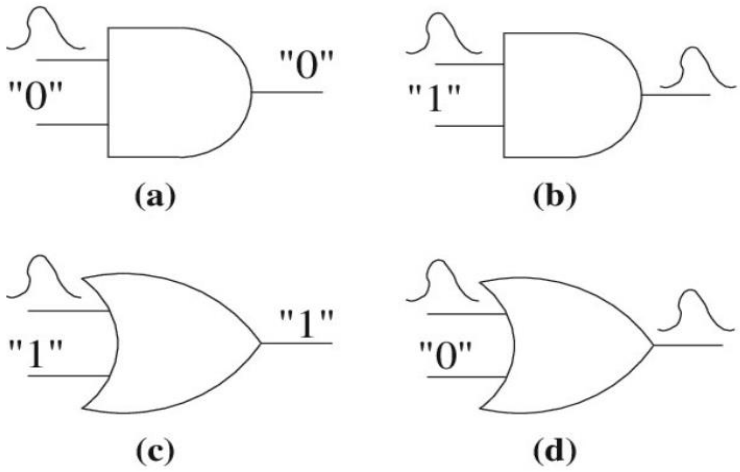


Figure 3-3 Illustration of logical masking in AND and OR gates [13]

Chapter 4

Methods of Modelling and Simulation of SETs

Analysis and characterization of a circuit response to SETs can be done by controlled irradiation experiments in which the device under test is subject to a dose of radiation then its electrical behavior is observed. However, this suffers from two major drawbacks. The first is that this cannot be done until the design is fabricated and packaged, and it will be difficult to introduce any design enhancements to improve the sensitivity of the circuit. The second is that experimental radiation setups are usually very expensive and need a lot of precautionary procedures. Therefore, computer-aided models and simulations are used extensively.

There are several approaches to tackle the modeling and simulation of the generation of SETs and the circuit response to them. Device-level modeling through TCAD simulations, which considers most of the physical details and mechanisms of SETs within the body of the transistor, are known for being the most accurate, but it cannot be used in gate level modelling or circuit level modelling as it will be computationally infeasible. So TCAD simulation are mostly used for modelling the particle impact and charge collection mechanisms to produce a model for pulse generation as a voltage or current pulse that can be incorporated into SPICE simulations for large number of devices.

Studying the effects of SETs on a circuit begins with injecting a current pulse into the starting node of interest. The shape of the current pulse has several models some of which will be discussed shortly.

There are two main levels by which the problem can be tackled: -

1. Macro-modelling (Gate Level):

   The injected current pulse is connected in the circuit between different gates, no modification is introduced to the gate structure nor the internal devices, as shown in Figure 4-1.

2. Micro-modelling (Transistor Level):

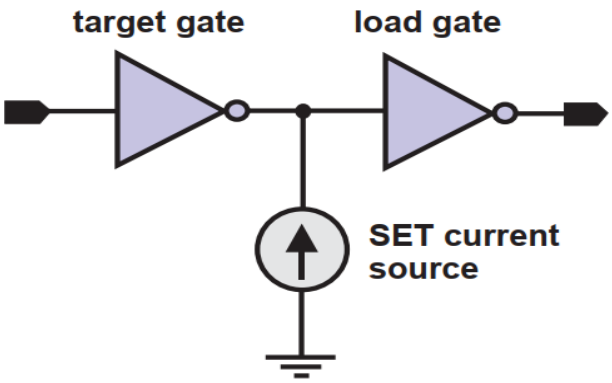


Figure 4-1 Macro model for SET current pulse [10]

The injected current source is embedded into the transistor, as shown in Figure 4-2, which implies the need for editing the transistor model. This process is difficult as the access to transistor model in most of the commercial technologies is restricted to the fabrication foundry.

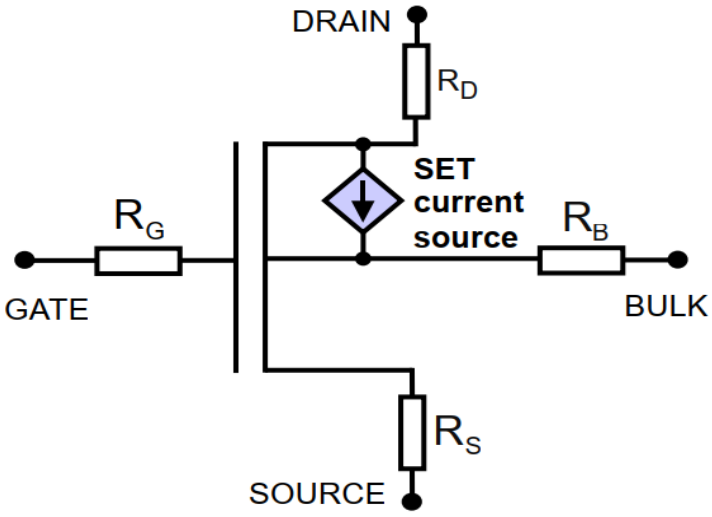


Figure 4-2 Micro model for SET current pulse [10]

4.1 SET pulse models

4.1.1 Double-Exponential Current Model

The most widely SET pulse source model developed by [14]

$$I_{SET}(t) = I_0 \left(e^{\alpha t} - e^{-\beta t}\right) \qquad (4\text{-}1)$$

where I_0 is the pulse height, $1/\alpha$ is the collection time constant of the junction, which corresponds to the fall time of the pulses as shown in Figure 5-2, and $1/\beta$ is the time constant for establishing the ion track. The junction time constant is shown to be

$$\frac{k\ \varepsilon_0 \varepsilon_r}{q\ \mu\ N_D} \qquad (4\text{-}2)$$

Where k in the Boltzmann universal constant, $\varepsilon_0 \varepsilon_r$ are the permittivity of the junction, q is the single-electron charge, N_D is the donor dopant concentration and μ is the electron mobility.

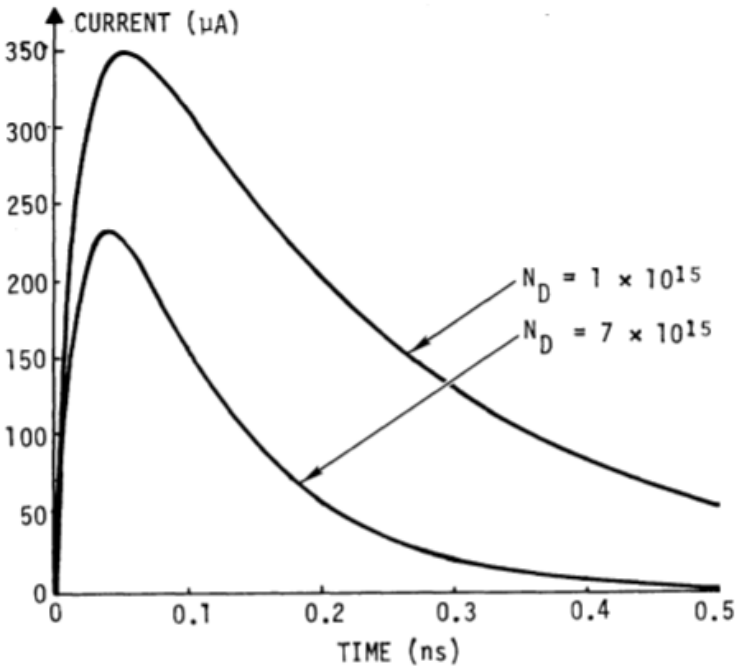


Figure 4-3 Double exponential current pulse model [14]

Other works also proposed several exponential current sources with different parameters [15]–[17].

4.1.2 Double Sinusoidal Voltage Pulse

A voltage model for the SET pulse propagating through a CMOS circuit that represents an imperfect square pulse was proposed by [18], described by equation (4-3) and shown in Figure 4-4

$$V(t) = \begin{cases} 0 & t \le t_0 \\ \frac{A}{2}\left(sin\left(\omega\,(t-t_0) - \frac{\pi}{2}\right) + 1\right) & t_0 \le t \le t_1 \\ A & t_2 \le t \le t_3 \\ \frac{A}{2}\left(sin\left(\omega\,(t-t_2) + \frac{\pi}{2}\right) + 1\right) & t_2 \le t \le t_3 \\ 0 & t_3 \le t \end{cases} \qquad (4\text{-}3)$$

Where $V(t)$ is the instantaneous voltage value of the pulse, *A*, is the pulse height, t_0, t_1, t_2, t_3 and ω are parameters that control the pulse shape. Pulse width, which is the duration between the two moments having $V(t) = A/2$, can be concluded from them.

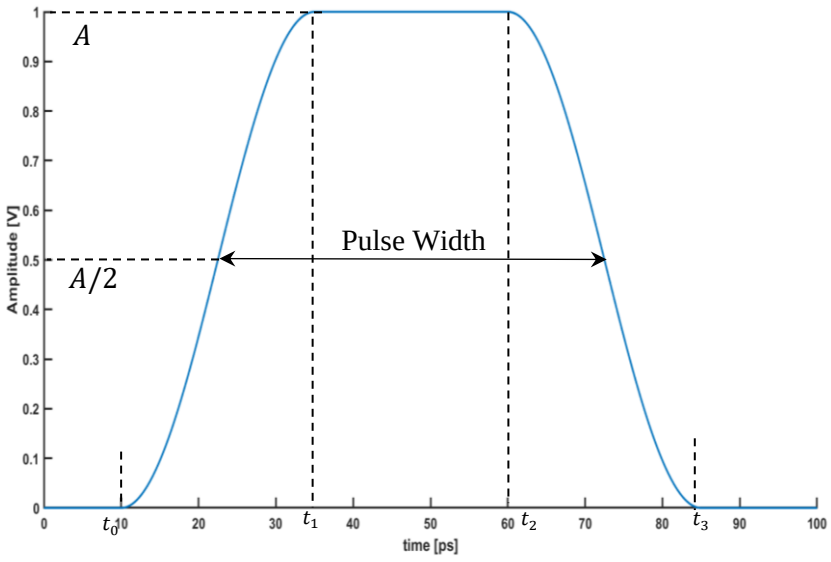


Figure 4-4 Double Sinusoidal Current pulse

These two models discussed above are implemented in Verilog-A to enable the modeling SET pulses throughout this work. The codes can be found in Appendix A.

.

4.2 SET Propagation Modelling

4.2.1 Mixed Mode Simulation

The work in [19], [20] employed a mixed-level device (TCAD) and circuit simulator (SPICE) called Davinci to study the production and propagation of SETs in a chain of ten CMOS inverters in the setup shown in Figure 4-6.

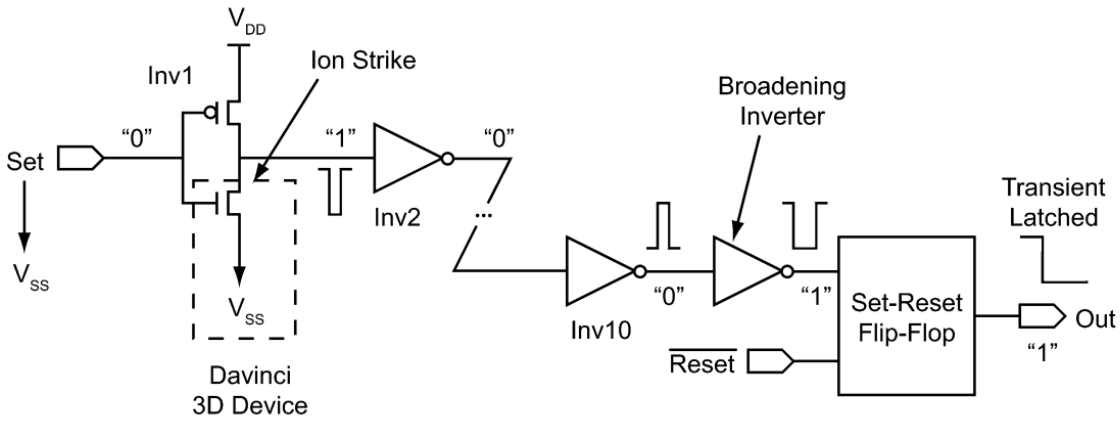


Figure 4-6 Mixed level modelling of cascade composed of 10 inverters [19]

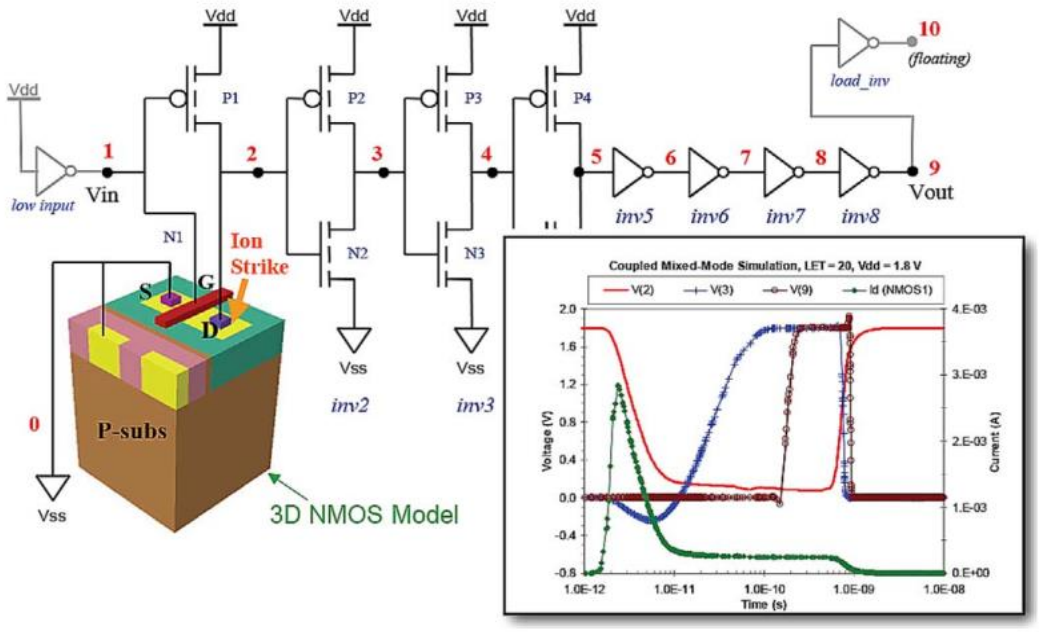


Figure 4-5 Mixed level modelling combines the 3D NMOS model and the electrical SPICE simulation [19]

The ion strike to the n-MOS device in the first inverter is modeled at the device level to obtain the SET pulse characteristics, and the rest of the inverters are modeled at the circuit level as a SPICE simulation. Figure 4-7 shows the SET voltage pulse from the device level simulation at the drain (output) of the device that was struck by an ion with linear energy transfer (LET) from 1 to 30 $MeV\text{-}cm^2/mg$ for two different fabrication wafers, SOI (Silicon on Insulator) and bulk wafers for comparison. This figure manifests

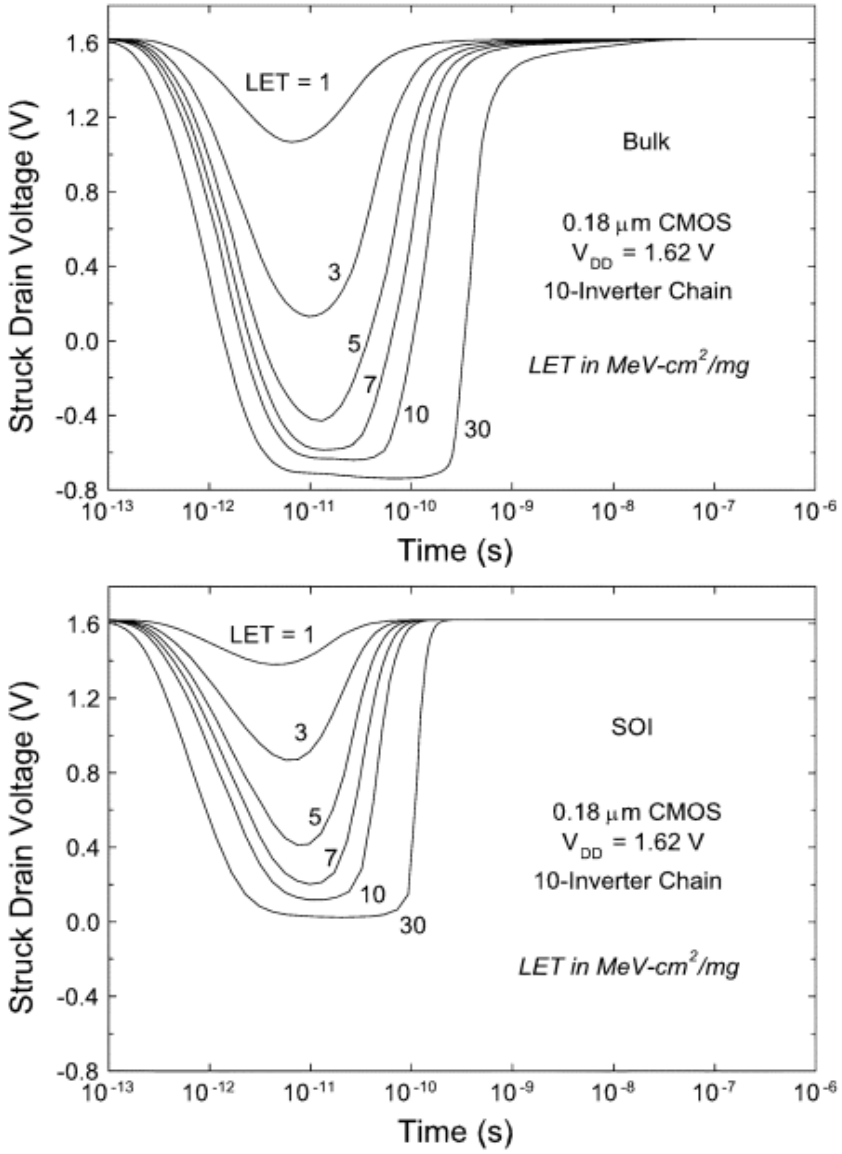


Figure 4-7 Effect of particle strike on drain voltage as function of LET [19]

the usefulness of the voltage pulse model proposed by [18] and discussed in Chapter 4, which has a similar shape to the pulses calculated by the device level simulation.

Figure 4-8 shows the waveform of the SET pulse propagating through the inverters. The pulse is gradually attenuated as it passes through more inverters. A low energy particle with LET equals 3 $MeV\text{-}cm^2/mg$ couldn't reach the flip-flop and is fully attenuated after the fourth inverter. But a particle with higher LET equals 7

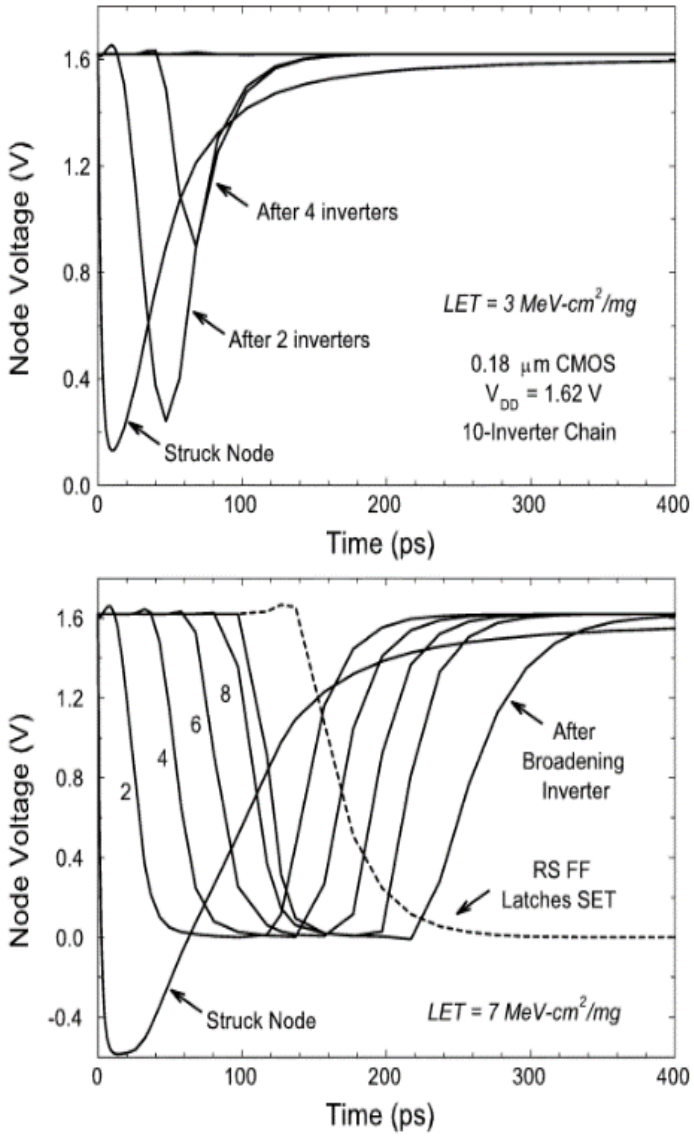


Figure 4-8 Effect of pulse propagation through gates [19]

$MeV\text{-}cm^2/mg$ could travel easily with no attenuation through the inverter chain and could reach the flip-flop hence causing an SEU.

This method is accurate and provides insight into the needed particle energy to cause an SEU, in addition, to accurately predict the phases of propagation of the pulse through the circuit. However, this will be computationally infeasible to perform device and circuit level simulations on ASIC designs that usually are comprised of an enormous number of devices. In addition, the method does not provide any regard to the logic inside the circuit, so it does not consider the logical masking effect.

4.2.2 Device Compact Model with SPICE

Work in [21] replaced the TCAD simulation step with a compact device model for the transistor drain current. The model was integrated into the device model for a 90 nm technology PDK. The model consists of four analytical equations (4-4) to (4-5) that determine the bias-dependent SET current. The schematic for the model is shown in Figure 4-10

$$I_{SRC}(t) + C_S \frac{dV(C_S)}{dt} = G_{REC}(t) + G_{SEE}(t) \quad (4\text{-}4)$$

$$G_{REC}(t) = f(V(C_S), C_S, RecombParameter) \quad (4\text{-}5)$$

$$G_{SEE}(t) = f(V(C_S), C_S)\, f\big(V(drain', body)\big) \quad (4\text{-}6)$$

$$G'_{SEE}(t) = G_{SEE}(t) \times Gain \quad (4\text{-}7)$$

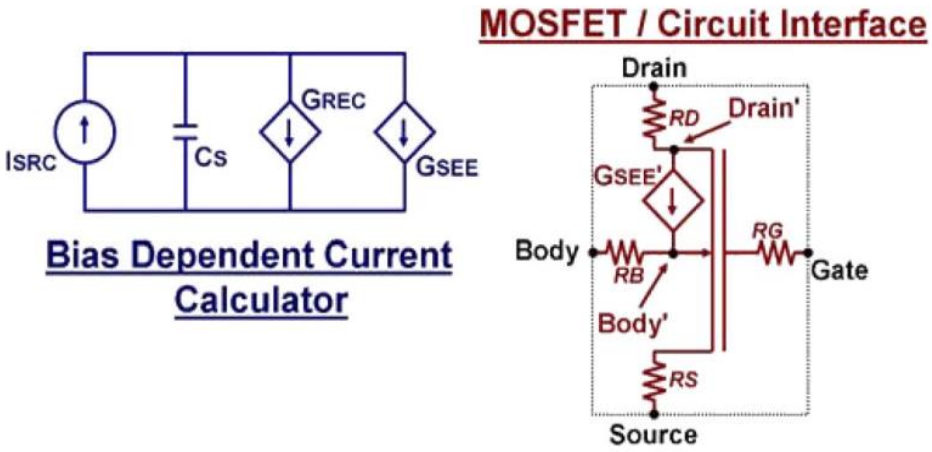


Figure 4-10 Schematic of the micro model proposed by [21]

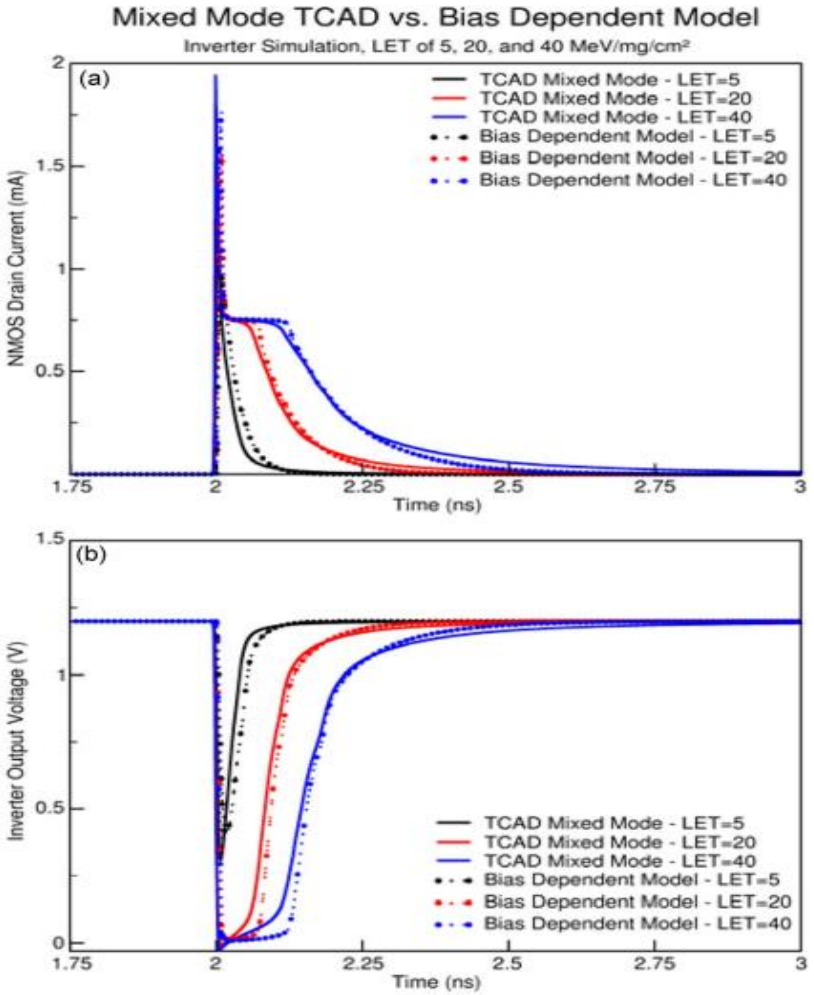


Figure 4-9 Comparison between the proposed model by [21] and mixed mode simulations.

This method alleviated the burden of the computationally intensive TCAD simulations and the need to use another simulator outside the SPICE environment, hence easier use model for the designer. Its accuracy proved to be close to mixed mode simulations, as shown in Figure 4-9

4.2.3 Full Analytical Modelling

An analytical model was developed by [22] that devices two sets of equations. The first set, equations (4-8) to (4-11), is used to estimate the output voltage perturbation, which denotes the pulse height of the output signal. The model describes the output vs the input voltage perturbation as:

$$\sigma_V = \frac{V_{out}}{V_{dd}} \qquad (4\text{-}8)$$

This equation is mapped to the sigmoid surface that describes the output pulse height of the simulated gate. the output pulse height denoted as (V_{out}), the input pulse height is denoted as (V_{in}) and the input pulse width is denoted as (tw_{in}) in the following equations:

$$\sigma_V = \frac{1}{1 + e^{-k(V_{in} - V_0)}} \qquad (4\text{-}9)$$

Where k and V_0 determine the shape of the sigmoid surface that is based on the incoming pulse characteristics. k is the slope of the function at $V_{in} = V_0$ and has V^{-1} unit. V_0 is the magnitude of the incoming pulse that results in $V_{dd}/2$ voltage at the output. These parameters demonstrate the dependency of the output pulse height on the input pulse width as follows:

$$k = c\left(1 - e^{-\frac{tw_{in}}{T}}\right) \qquad (4\text{-}10)$$

$$V_0 = V_{DC}\left(1 + \left(\frac{t_{d1}}{tw_{in}}\right)^{\alpha}\right) \qquad (4\text{-}11)$$

The coefficients included in the previous equations are specific to each gate. V_{DC} is the voltage value where the input voltage is the same as the output voltage while the coefficients c, T, t_{d1}, α are the fitting parameters obtained by fitting the sigmoid surface to the data from simulations. The second set of equations, (4-12) to (4-14), are used to estimate the output pulse width. Like the previous set of equations that described the pulse height propagation, the input vs output pulse width perturbation is described as $\sigma_t = tw_{out}/tw_{in}$. The dependency of the output pulse width on the input pulse height and width is described as:

$$tw_{out} = a\, tw_{in} + t_0\, e^{\frac{-tw_{in}}{t_i}} + b \qquad (4\text{-}12)$$

The coefficients a and b unfolds into:

$$a = a_0 + a_1.V_{in} \qquad (4\text{-}13)$$

$$b = b_0 + b_1.V_{in} \qquad (4\text{-}14)$$

Where a_0 , a_1 , b_0 , b_1 are technology dependent parameters. The plots of the voltage and width transfer functions are shown in Figure 4-11.

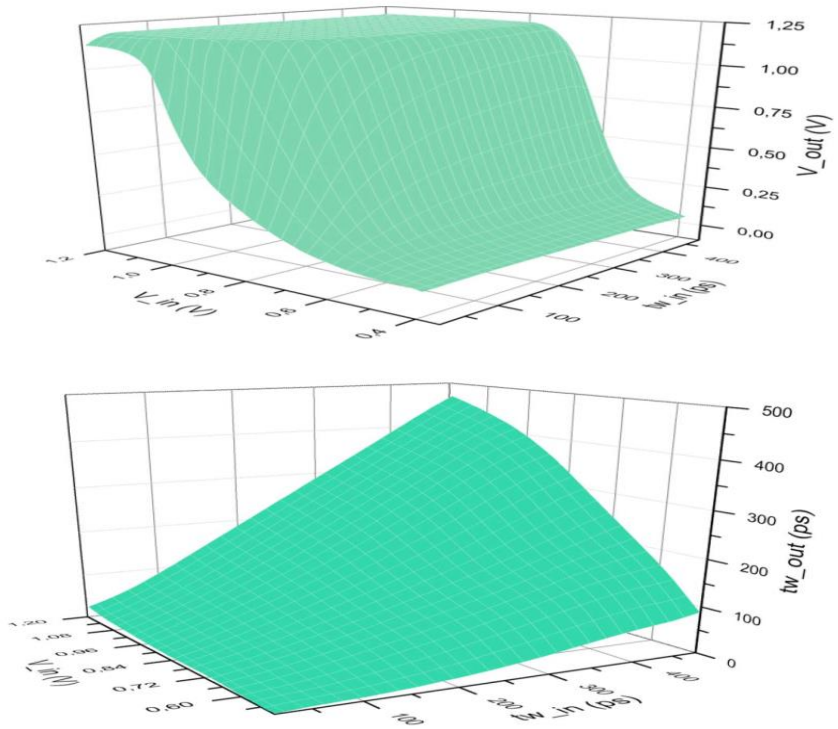


Figure 4-11 Voltage and pulse width transfer functions [22]

To extract the values of the fitting parameters, several simulations must be performed to produce enough data for accurate model fitting. This step is called the model extraction step or characterization. It involves many SPICE simulations and can be time and resources consuming, but it is done only once for each standard cells' library gate. Once the fitting parameters are obtained, no need for further simulations, and the model can be used directly to predict the shape of the output pulse given the input pulse parameters, and this is just substitution in an equation so it can be done in instantaneous time. The setup used for the model extraction simulations is shown in Figure 4-12. The pulse source used is an ideal square pulse followed by an inverter to make the pulse shape more realistic. The simulated data is used to fit the analytical expression using the Marquardt–Levenberg algorithm [23].

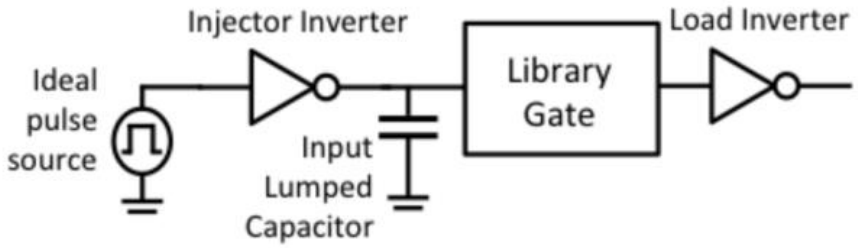


Figure 4-12 Model extraction circuit proposed by [22]

Chapter 5

New SET Propagation Modelling Methodology

The final goal of this work is to identify a set of test vectors that can test a target design immunity to an SET incident. The first stages of the process of generation of test vectors, which will be detailed in the next chapters, depend on the logical functionality and structure of the design but do not consider the electrical properties of the design. The sensitized paths due to generated vectors must be simulated electrically to determine whether the particle strike will be able to reach the output of the circuit or it will be filtered out. Due to the large number of paths that can be generated, simulations will be infeasible computationally. So, making use of the analytical approach of modeling SET propagation, which was described in Chapter 3, will be beneficial because it enables determining the output pulse shape of output SET pulse from a given gate in constant time complexity.

5.1 Flow Description

The first step in the flow is the model extraction step, which can be called Standard Cell Library Characterization. Each cell inside the standard cells' library needs to be subjected to many transient simulations to study how it responds to different shapes of pulses, i.e., pulses with different widths (tw_{in}) and heights (V_{in}). The different pulses

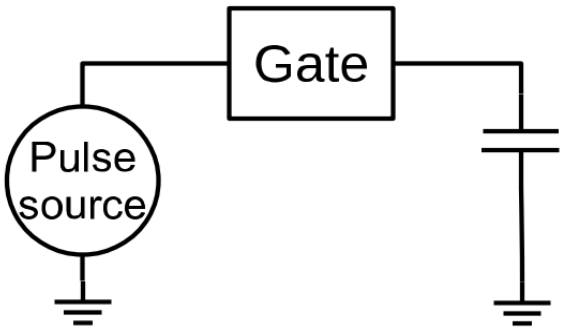


Figure 5-1 Gate characterization test bench

are injected into the cell inputs, and the output pulse width (tw_{out}) and height (V_{out}) are observed and recorded. The input pulses are produced by a sweep over the pulse source

parameters. The used circuit for this, called the model characterization circuit, is shown in Figure 5-1. The capacitance in the model extraction circuit is to account for the loading effect of the gates. As the load capacitance increases, the propagation of the SET pulse through the gate will be more attenuated, hence this characterization setup always uses the smallest inverter in the library as the load gate which provides the worst case scenario of SET pulse propagation. The aim of these simulations is to extract the output pulse width and height to be, along with the input width and height, used in fitting the analytical models described in Chapter 4. Cadence Spectre circuit simulator is used for this purpose to perform the required sweeps. Then for each sweep, the output pulse shape is analyzed to measure its amplitude and width, which is defined to be the time width of the pulse at half the height as shown in Figure 4-4.

After many sweeps, we have a set of input and output widths and heights that are ready to be used to find a model for the cell under test. This process needs to be performed on every cell of the library to characterize a given library. It will be a long and tedious process if the simulations were done manually, so Matlab, Ocean, and SKILL scripts were employed to automate the process given the path to the library.

5.2 Flow Automation

The first step is to iterate over each cell in the library and generate its netlist. This is done through a SKILL script that is available in Appendix A. Then create a testbench for it, and the test bench is just the model extraction circuit mentioned above. Then the netlist of the whole testbench is generated. Now we have a testbench for each cell ready for simulation, but it needs the value of the parametric sweep to be defined for Spectre to run the simulations. The range of input pulses needed is given to Matlab, which generates a compact Ocean script for each cell where all the simulation parameters are defined. Spectre can now be invoked through the Ocean script, and the output waveforms are saved from being processed by Matlab to extract the output pulse widths and heights. The last step is to fit the models using the extracted parameters and save the models so that they can be used during the test vector generation steps. All the codes used for characterization are detailed in Appendix A for reference.

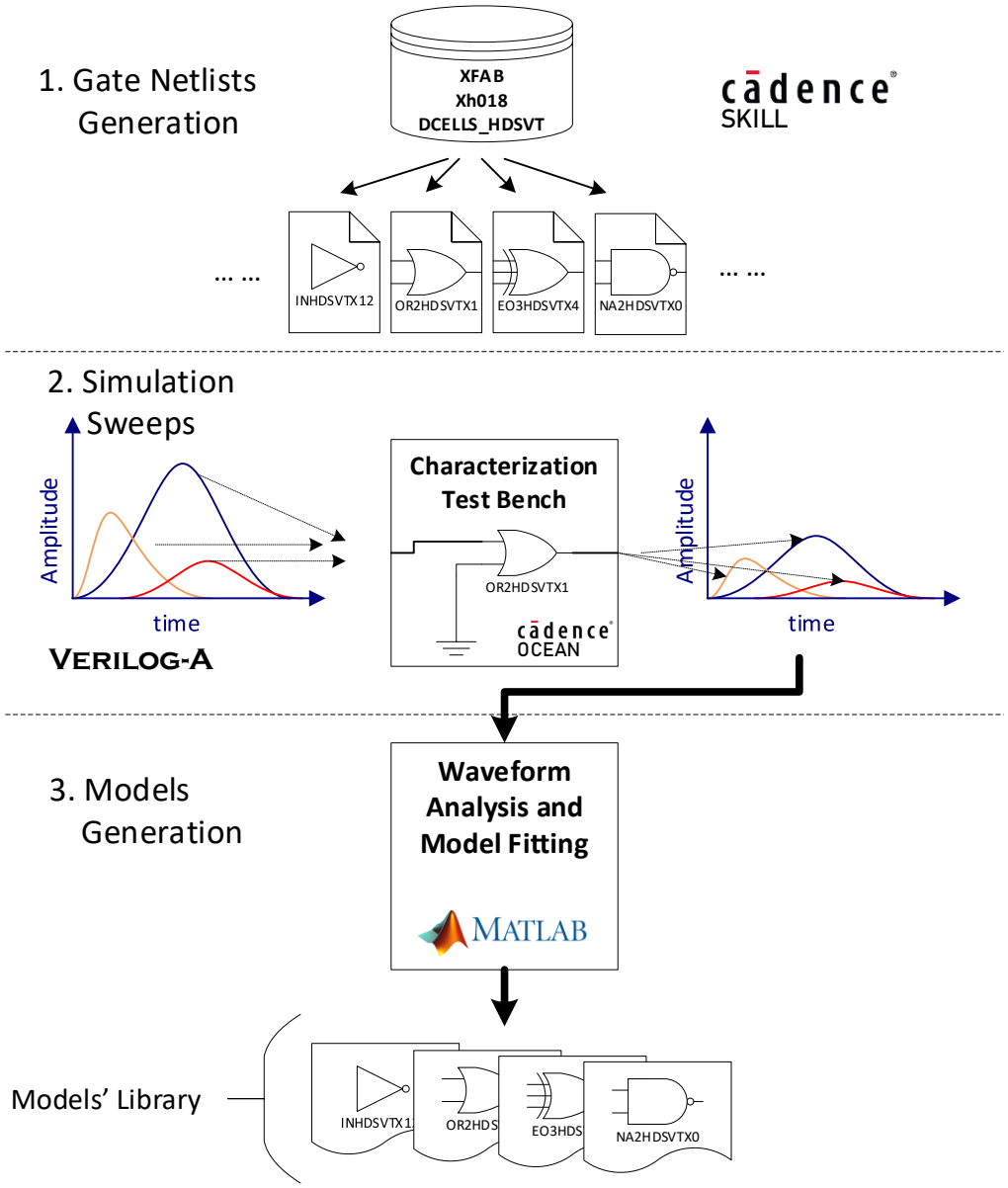


Figure 5-2 The implemented library characterization flow

5.3 Results

The analytical model of [22] was applied in the characterization of XFAB Xh0.18 um technology standard cell libraries. It exhibited relatively high accuracy for most of the voltage regions. One of the model's noticeable drawbacks is the incorrectness of the expected values of the pulse width in the complex gates. The cells which are composed of multiple cascaded gates are referred to as complex gates. For instance, the two-input

AND gate of the high density and low leakage (HDLL) flavor of the same technology, "AND2HDLLX1" reported a maximum pulse height error of 0.6V and maximum pulse width error of 98.7ps. The value is deemed inaccurate since the V_{dd} for this cell was 1.8V. In addition, the radiation community considers pulses greater than 50ps to be effective pulse. Accordingly, 98.7ps error is intolerable. Hence, a valid methodology needs to be presented to report values of tolerable error.

Fig 5-3 shows the simulation data of NO2HDSVTX4 cell besides the two fitting models as an example that illustrates the inaccuracy of the previous model [22] for pulses with height around 0.6 V. This can be observed from equations which show a linear dependence of tw_{out} on V_{in} which is not always true. The proposed interpolation method succeeds in producing a more accurate model for the pulse. The inaccuracies that is found in the analytical model proposed by [22] are a motivation to find another method that achieves the same purpose of not requiring complex simulations during path evaluation steps.

Model fitting using the cubic interpolation method is used since it provides smooth interpolation with a minimal error in our case where the plotted surfaces do not consist of sharp transitions. Computationally, the interpolation method is implemented via a cubic convolution algorithm on Matlab®, which breaks down the surface interpolation into 1D interpolation kernels to make it computationally feasible [24]. Some explanation for the basic concepts behind cubic interpolation is presented in Appendix B.

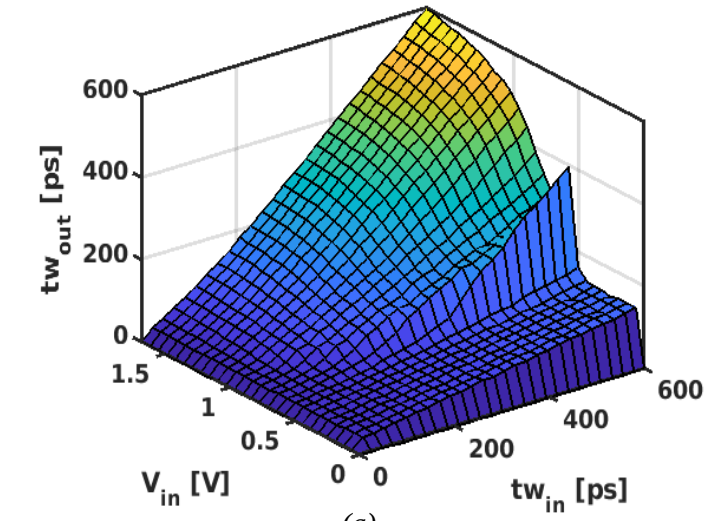
Finally, to test the proposed interpolated model, another simulation run with test input pulse width and height values is performed. The next subsection provides a quantitative comparison of the proposed model against the previous model.

A comparison between the previous model [22] and our cubic interpolation method has been performed, and the results are presented summarized in Table 5-1, where samples of several standard cells with different process variants from XFAB® XH018 technology [25]. Cells with the "_3V" suffix use a supply voltage V_{dd} of 3 volts, while those with "HDLL" suffix are high density and low leakage cells using 1.8 volts supply and the remaining with "HDSVT" are high speed/medium threshold voltage with 1.2 volts supply. The "Xn" suffix denotes the sizing of the CMOS network used in the implementation of each cell within the library. Data in the table are extracted by running

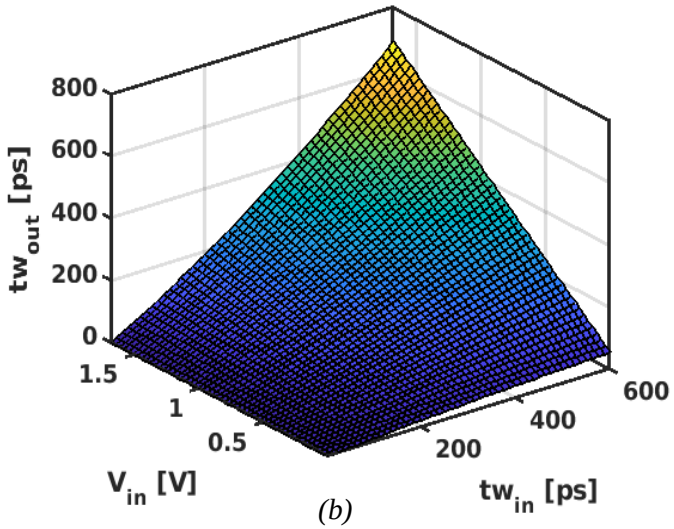
test simulations with sweep values that are different from ones used in the characterization step to test the accuracy of the generated model on new data, and the reported values are the maximum errors in the predicted pulse heights and widths compared to simulation data.

Table 5-1 Comparison between the two presented SET propagation models

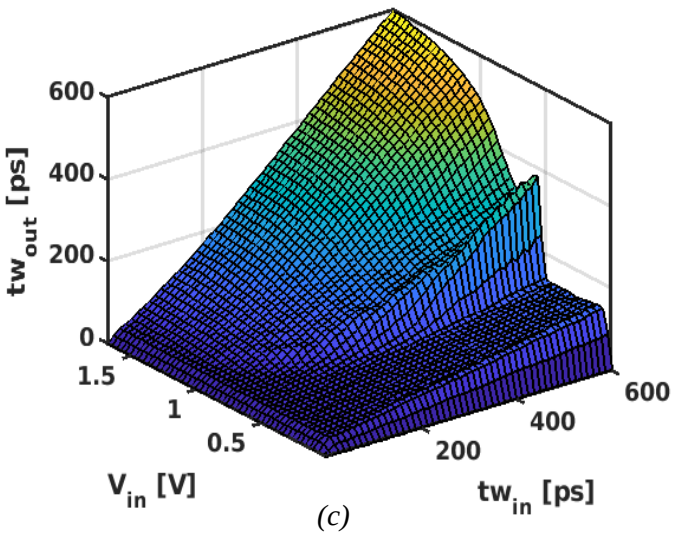
Cell Name	***Description***	***Model in*** [22]		***Cubic Interpolation***	
		Max error in V_{out} [mV]	***Max Error in tw_{out} [ps]***	***Max error in V_{out} [mV]***	***Max Error in tw_{out} [ps]***
IN_3VX1	Inverter	140	123	45	25
NO2_3VX1	2-input NOR	180	102	61	14
NA2_3VX2	2-input NAND	228	108	85	53
EO2_3VX2	2-input XOR	313	103	135	76
NA3HDLLX1	3-input NAND	111	99	4	45
NA2HDLLX2	2-input NAND	183	105	28	57
NO2HDLLX4	2-input NOR	204	65	49	15
NO2HDSVTX0	2-input NOR	78	74	10	12
OR3HDSVTX4	3-input OR	68	280	38	34
OR3HDSVTX0	3-input OR	63	177	20	41
NA4HDSVTX4	4-input NAND	56	54	2	63
INHDSVTX1	Inverter	126	85	23	26
INHDSVTX12	Inverter	294	92	43	32



(a)



(b)



(c)

Fig 5-3 Pulse output width from NO2HDSVTX4 cell characterization. (a) simulation result, (b) model from [22] and (c) cubic interpolation.

Chapter 6

Defects in VLSI Systems and Fault Models

6.1 Introduction

Scaling down Transistors makes them faster and smaller, which causes the cost to reduce. On the other hand, the fabrication defects are not scalable down because they persist or even increases. The effects caused by the scaling downtrend are known as Deep Sub Microns effects (DSM). DSM effects include excessive voltage drop, delay variation, substrate noise, logic errors (stuck faults), and thermal noise. So, fault models are needed to address these issues. Moreover, process variation effects across the chip became serious. All of these effects can cause serious problems and can cause the chip to fail [26]. According to Moore's law, the transistor sizes are decreasing every year, and this

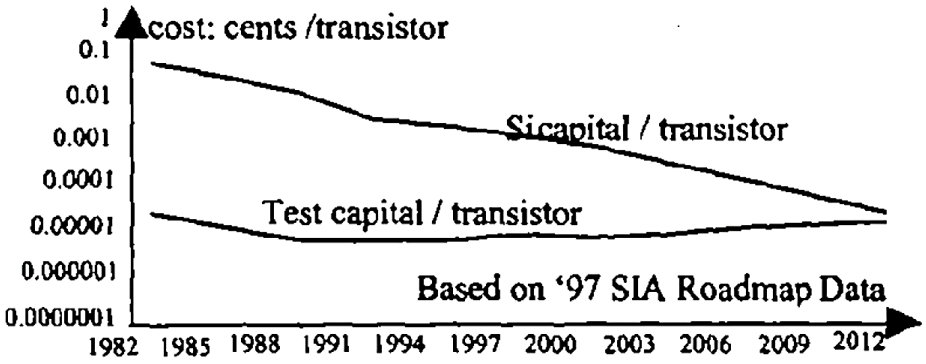


Figure 6-1 transistor price for chip vs transistor price for testing [26]

causes the fabrication prices for each transistor to become less. On the other hand, the price of the transistor which is used in testing is almost the same as the number of transistors needed to validate the circuits is going higher [26].

In addition, the power scaling and power distribution difference must be examined and analyze the power needed by the added test units to meet the requirements of low power applications. Moreover, the defects have gone way beyond stuck at fault,

and scaling down causes more soft defects (defects that is sensitive to frequency, temperature, and V_{dd}).

Moving from aluminum to copper for lower resistance will increase the need for new ways to analyze faults and generate automatic test patterns and fault simulations because copper needs different processes hence different fault distribution across the chip. On the other hand, these new algorithms will require more time because of the complexity of the chips and the number of gates, so unless there is a breakthrough, the time required by the available CAD tools will be infeasible. In short, submicron devices will be more sensitive to process variations and fabrication defects. In addition, there are more techniques used in designing circuits to meet the time and power requirement (such as using different Vth cells). This technique will present different current distribution, which will make it hard for (ATPG) and fault simulation, and coverage analysis using CAD tools [27].

There has always been some confusion between the terms error, defect, and fault in VLSI systems. A defect in a VLSI chip is an unintended inconsistency between the implemented design and fabricated hardware. An error can be defined as a wrong output value produced by a defective system. An error is the symptom of the disease, which is the defect.

A fault is an abstraction of a possible failure at a certain level of circuit representation that is performed to enable the process of generating test vectors. Consider a circuit consisting of two inputs A and B, one output C, and one two-input OR gate. Input A is meant to be connected to the first input of the OR gate, while B is connected to the second input of the gate, and C is at the output of the gate. Due to some malfunction during the fabrication process, the second input of the gate is connected directly to the supply voltage (logic 1). So, the output of the circuit will always remain 1 instead of the intended functionality, which is $C = A + B$. This situation can be described in terms of the notations above as follows: -

- This circuit has a *defect* of a short supply.
- Stuck-at 1 fault for input B.

- An error of incorrect output value for any input combination rather than $A = 0$ and $B = 0$, hence the error is not permanent as it will be observed only if the correct output was supposed to be 1.

Some typical defects in VLSI chips are [28]:

1. Process Defects

   These are related to the steps performed during the IC fabrication process like transistor latch-up, metal shorts/opens, oxide breaks down.

2. Material Defects

   It is related to the bulk silicon wafer material structure like crystal imperfections and surface roughness.

3. Aging Defects

   Which appears slowly as the chip is aging down. The effect maybe is due to the internal operation of the circuits like dielectric breakdown and electromigration and can be due to the exposure of the circuit to environmental factors like temperature, mechanical stress, and radiation.

4. Packaging Defects

   Defects that happen in the packaging structure itself, like mismatch in wire bonds/pads or because of packaging material on the chip, like the spontaneous emission of particles from lead materials in the package.

6.2 Fault models

Fault modeling is an abstraction of a possible failure at a certain level of circuit representation that is performed to enable the process of generating test vectors [29]. There are several fault model categories in digital circuits, such as stuck-at faults, leakage current faults, open faults, bridging faults, and delay faults. Stuck-at and delay faults will be discussed in some detail as they serve the purpose of this work.

6.2.1 Stuck-at Faults

The connection between gates in a design is described as a netlist of interconnections. When a connection between the gates is fixed at a value, the interconnection is said to be a stuck-at fault. Whether the interconnect is stuck at high

logic or low logic, which are called stuck-at one or stuck-at 0, respectively, determine the type of the stuck-at fault. An interconnect.

Consider having n interconnects in a design each has one of three fault values - stuck-at 0, stuck-at 1, and stuck free - will give (3^n) fault combinations with only one stuck-free combination. Hence, for a small number of interconnections, the design will have an enormous number of multiple stuck-at faults. However, this number is reduced to $2n$ if we assume only one fault at a time, single stuck-at. The scope of this section will be the single stuck-at fault, which means that a single port of a gate has a fixed value of either 0 or 1.

The single stuck-at model assumptions are: -

1. Only one node is faulty.
2. The faulty node wrong value is permanently set to 0 or 1.
3. The fault can be located at the input or output of the gate.

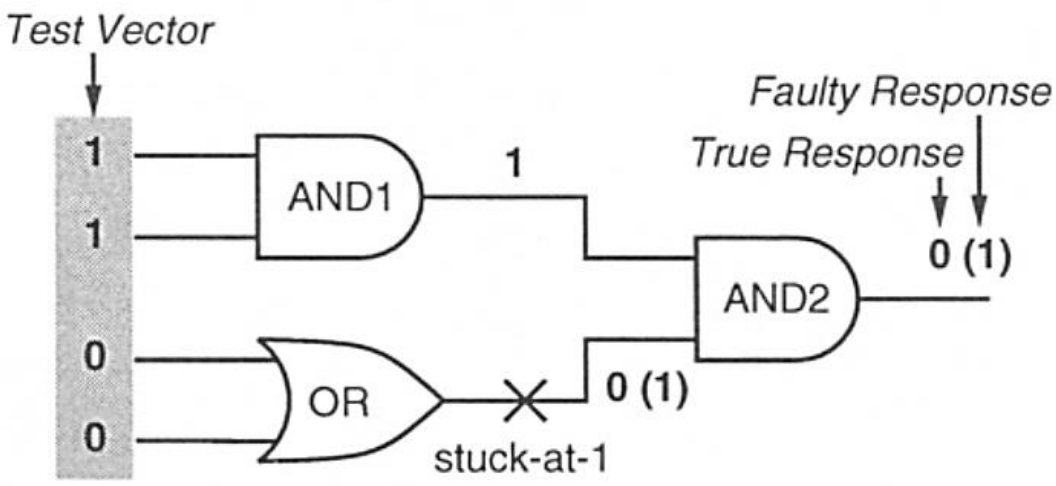


Figure 6-2 Testing stuck-at faults [28]

6.2.2 Fault Equivalence

Two faults are said to be equivalent if and only if they have the same effect on the function of the circuit, i.e., given the same input excitation, the two faulty states of the circuit will give the same output values. Applying the previous definition on a multiple input single output circuit. When exciting the inputs with an input vector V the

circuit response is expected to be $f_0(V)$. If the circuit has two fault responses $f_1(V)$ and $f_2(V)$ then to see that the fault response is different from the correct response using the excitation vector V_1, we must have

$$f_0(V_1) \oplus f_1(V_1) = 1 \qquad (6\text{-}1)$$

This equation simply means that $f_0(V_1)$ can never be equal to $f_1(V_1)$. Similarly,

$$f_0(V_2) \oplus f_2(V_2) = 1 \qquad (6\text{-}2)$$

If the test vectors for fault 1 are the same for fault 2, $V_1 = V_2 = V$ we get

$$\big(f_0(V) \oplus f_1(V)\big) \oplus \big(f_0(V) \oplus f_1(V)\big) = 1 \qquad (6\text{-}3)$$

By simplifying, we get

$$f_1(V) \oplus f_2(V) = 0 \qquad (6\text{-}4)$$

meaning they are equivalent.

Equivalent faults are also indistinguishable because they are produced by the same tests and show the same output. Therefore, we cannot distinguish between them. Consider and a two-input AND gate, a stuck-at 0 fault at either of its inputs forces the gate to always have a 0-logic output, which resembles the situation if the gate had single stuck-at 0 on its output while they are still physically two different faults, but this is not true in the case of stuck-at 1. So, we can say that the stuck-at 0 fault at the input for the AND gate is equivalent to the single stuck-at 0 faults at its output. The same situation is found for the OR gate where stuck-at 1 fault on the input is equivalent to stuck-at 1 in

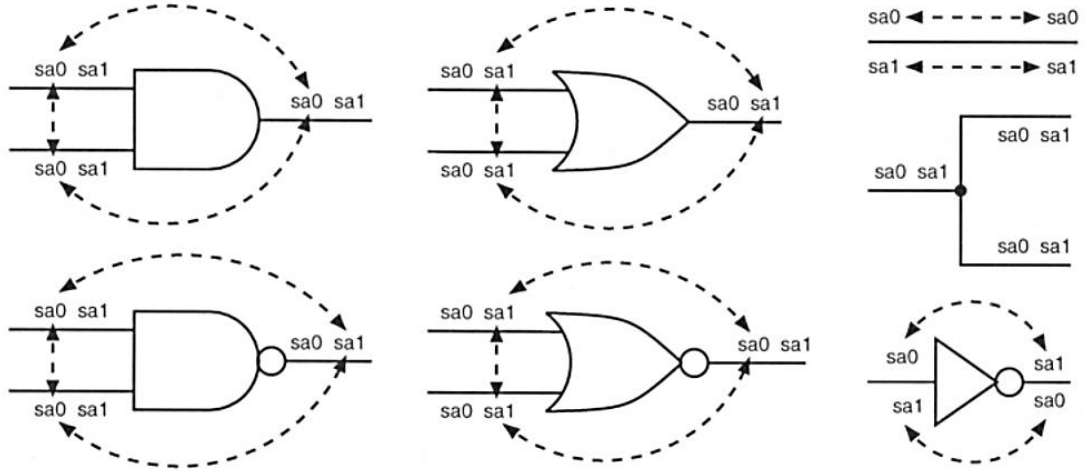


Figure 6-3 Examples of equivalent faults [28]

the output. An inverter cell will have stuck-at 0 at input equivalent to stuck-at 1 at the output. Figure 6-3 shows several examples for equivalence; the double-sided arrow denotes pair-wise equivalence between faults. Notice that fanout structure does not imply equivalence between the stem and branch fault sites because each branch may drive a different gate with different functionality.

6.2.3 Fault Collapsing

Equivalent faults are grouped in sets called equivalence sets. If a fault is found in two equivalence sets, this means the two sets are equivalence and can be merged into one set. Adding faults to sets is called fault collapse, where the collapsing ratio is given by the formula. The faults in the same equivalence groups are said to be collapsed into one fault, and the original faults are deleted, so the effective number of faults will be smaller. Figure 6-4 shows an example of fault collapsing in two different circuit structures. The presence of fanouts hinders the process of collapsing, and the design will have a smaller number of faults deleted by collapsing.

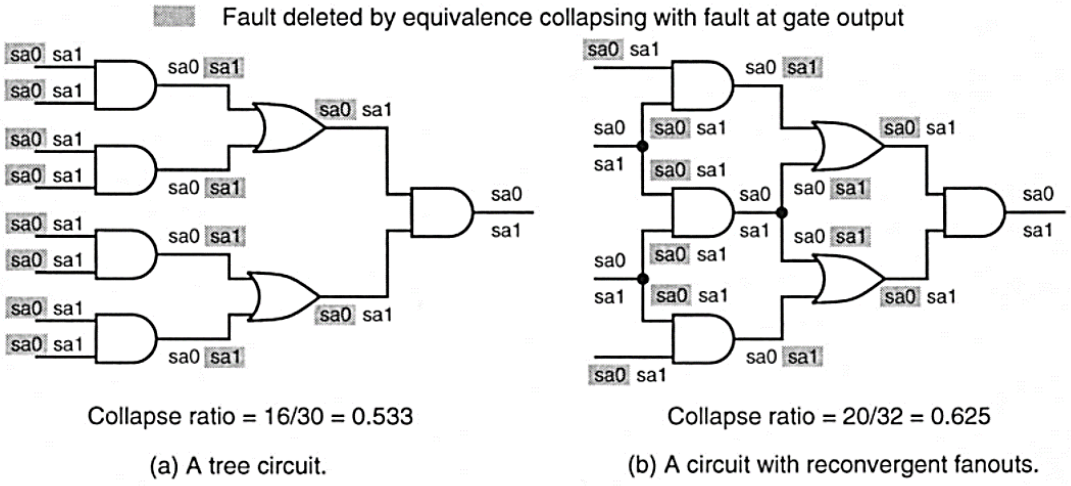


Figure 6-4 Effect of fanouts on collapse ratio [28]

$$Collapse\ Ratio = \frac{Number\ of\ collapsed\ faults}{Total\ Number\ of\ faults} \qquad (6\text{-}5)$$

Collapse ratios are typically about 50% ~ 60%, according to [28], so it will be more efficient for the tool analyzing the circuit for faults to consider collapsing first to reduce the number of faults of interest. ISCAS85' benchmark circuits show the typical collapse ratios as shown in Table 6-1.

Table 6-1 Collapse ratio in for ISCAS85' benchmarks [28]

Circuit name	No. of gates	No. of inputs	No. of outputs	Number of faults		
				All	Collapsed	Collapse ratio
c432	160	36	7	864	524	0.61
c499	202	41	32	998	758	0.76
c880	383	60	26	1,760	968	0.55
c1355	546	41	32	2,710	1,606	0.59
c1908	880	33	25	3,816	2,041	0.54
c2670	1,193	233	140	5,340	2,943	0.55
c3540	1,669	50	22	7,080	3,651	0.52
c5315	2,307	178	123	10,630	5,663	0.53
c7552	3,513	207	108	15,104	8,084	0.54
s27	10	4	1	52	32	0.62
s9234	5,597	19	22	10,572	3,862	0.37
s38584	19,257	12	278	78,854	36,303	0.47

6.2.4 Delay Fault

The stuck-at-fault means that the net is always tied to one logic value, either high or low. The fault will always have one logic value and cannot have the opposite logic value. Another way to interpret the stuck-at fault is to realize that the stuck-at-fault takes an infinite time for the logic value to alternate between the digital logic values. That is why circuits that pass a test designed for stuck-at-faults are more likely not to have infinite delays, which means that the stuck-at-fault could be interpreted as a particular case of delay faults. Circuit designers always look for steady-state values within a precise clock period. Unfortunately, testing circuits for infinite delay faults, stuck-at-faults, is not sufficient for circuits. Work in [30] shows that it is essential to build tests specified only to detect delay effects.

In sequential circuits, the delay of the combinational gates must not exceed the clock period specified early since all inputs change at the clock edge, and the outputs

should have steady values at the next clock edge after the clock period. The combinational path that consumes the most delay in time is defined as the critical timing path that should always be less than the clock period. The transition of input vector applied to the combinational circuit from a certain input vector to another would be appropriate for testing delay faults that do not take infinite time.

The example in Figure 6-5 shows the signal transition in inputs from 010 →100. In ideal situations, all input pins would change instantaneously, as shown in the figure; however, this is not always the case in real situations. The transient region at the output, highlighted in grey, contains multiple transitions that are separated in time. Each transition is corresponding to certain changes in the output vector due to the different combinational paths. Therefore, most right edge transition in the grey region is determined by the last transition of the critical path due to the input vector pair. A necessary timing condition is that the transition region should net go beyond the clock period, or the circuit would attain a delay fault.

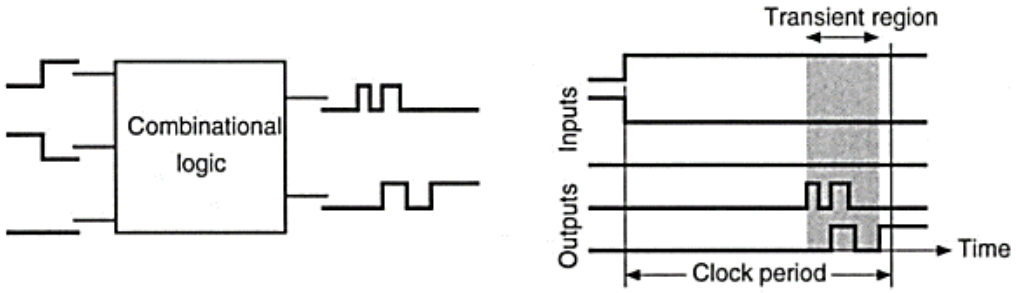


Figure 6-5 Delay fault example [28]

The next figure shows how the input vector pair causes a logic transition from inputs to the primary output. The three dashed lines are the only paths through which the logic transition propagates when the input vectors change from 010 to 100. At the output node K, the logic must not exhibit any sort of transitions to attain a steady-state value before the next clock edge. Otherwise, this input-vector pair transition would cause a delay fault.

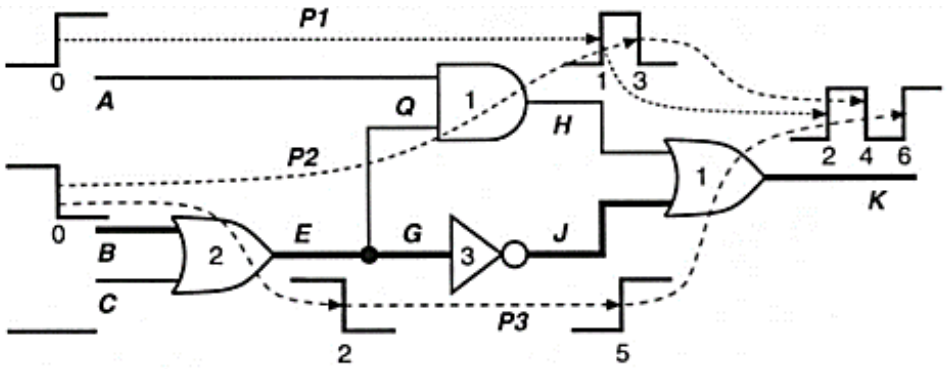


Figure 6-6 Example of delay fault examining logic propagation from inputs to outputs [28]

The delay faults are dependent on the direction of the transition, either a rising or falling. The sum of possible delay faults is always double the number of the physical paths. The upward arrow ↑ will be used to resemble a rise transition, but the ↓ downward arrow for a fall transition. Considering more than one delay fault at a time would be faulty, so only one delay fault needs to be examined to tackle the problem.

6.2.5 Transition Fault

The most common form of delay faults is the transition fault. When a path in the circuit experiences slow signal transition after applying input vector pair, a transition fault occurs—there two types of transition faults: slow-to-rise and slow-to-fall. For example, a certain node has a LOW steady-state value while applying the first input vector and should attain a HIGH steady-state value because of the second input vector, so this node experiences a rise transition. In addition, transforming the voltage value from HIGH to LOW is considered a fall transition. Any transition could be faulty when transitions occur slowly, resulting in the transition region mentioned above exceeding the clock period.

The simplest way to apply a transition fault test is to make use of the stuck-at test vectors. To detect a slow-to-rise fault on a certain circuit node, apply a test vector stuck-at-0 (V_1) to the circuit, resulting in HIGH voltage on this node. This will cause signals on primary outputs of the circuits to reach a certain state. Then, apply a stuck-at-1 test vector V_2 that is responsible for setting a LOW voltage on the specified node and changing the value of the primary outputs from the previous state invoked by V_1 to make sure that the transition should appear from the fault site to at least one of the primary outputs. Also,

the input vector pair need to make sure that the transition happened in the node under test propagates in a path through different logic till it reaches a primary output node. Following the same analogy, to detect also a slow-to-fall fault would be applying a stuck-at-1 then stuck-at-0 vectors on one condition that the primary output should experience a transition as well. A necessary assumption is used while detecting the right input vector pair, which is the transition should take a large delay in time, although the observation path is short. The designer should expect the logic output state produced by V_2 . Therefore, any mismatch between the expected output and the resulting output after a clock period is considered a slow-to-rise fault. Assuming that there is one primary output in the circuit and it must attain a LOW logic value after applying the second vector, so the HIGH logic value would be considered as a fault.

6.3 ATPG

Automatic test-pattern generation (ATPG) is the process of generating input vectors to test a circuit for specific faults. The circuit is described at the logic gate level. ATPG algorithms usually operate with a fault generator program, which creates the minimal collapsed fault list so that the process becomes more efficient and alleviates the burden of determining the significant faults from the designer. Controllability, observability, and testability measures are insured in all significant ATPG algorithms.

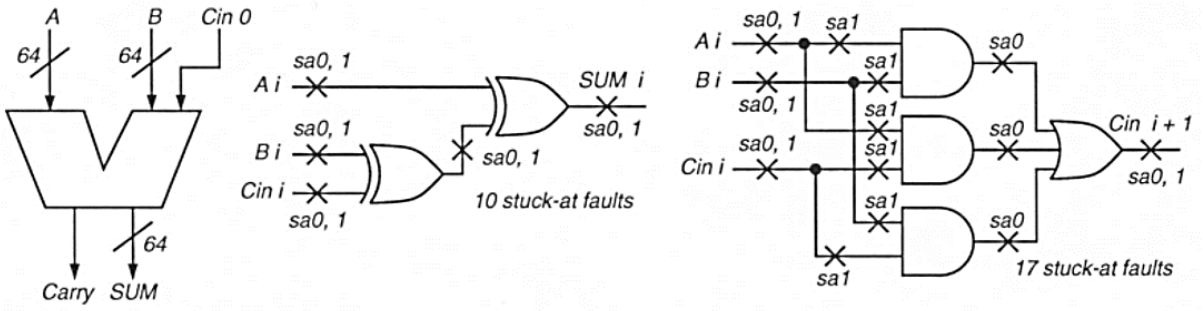


Figure 6-7 64-Bit adder composed of parallel simple full adders [28]

A possible method for ATPG can be to generate a complete set of test patterns to ultimately stimulate all the circuit truth table. A 64-bit adder is shown in Figure 6-7, and

its structure is a repeated full adder circuit for each bit which is inefficient logic design. However, this example helps the demonstration of the functional versus structural methods paradigm. From the functional testing point of view, the adder has 129 inputs and 65 outputs. So, to completely test all its functionality, we need 2^{129} input patterns that produce 2^{65} output patterns. These numbers of patterns are enormous. At present, the available automatic test equipment that feeds the input vectors to the design and compares the output to the expected output cannot operate at speeds higher than one *GHz.* Assuming we can operate on the 1 GHz, the process would take 2.16×10^{22} years to apply all the test patterns to the design under test. The exhaustive functional testing is impractical for large designs, and it can be employed in small designs.

Structural testing uses a different approach to this. It only simulates a minimal set of stuck-at faults on each node of the circuit after discarding the equivalent faults. If we employed the fault equivalence concept discussed above, each bit full adder has 27 faults. This adder has no redundant hardware, and the total structural fault list will have no more than 64×27 faults. So, we need 1728 test patterns. The 1 *GHz* testing setup would apply these patterns in 1728 n*s*, and since this test set covers all the possible structural stuck-at faults in the adder, it achieves the same fault coverage as the exhaustive functional test. ATPG algorithms produce vectors that can have 98% or higher coverage.

6.3.1 Operation of Automatic Test Pattern Generation tool

ATPG algorithms start by injecting a fault into the design under test and try to activate the fault and propagate its effect through the hardware to be observable a circuit output. If the output signal changes from the value expected for the fault-free circuit, the fault can be detected. Fault effects are propagated from gates like AND or NAND input to its output by setting other inputs to 1, which are called the off-path inputs. While for OR and NOR gates, the propagation from input to output is done by setting other inputs to 0, which is a non-controlling value for the OR gate. The values of internal nodes of a circuit can be observed using an advanced technique called the E-beam testing [31], which allows observation of internal signals by a picture of the circuit that lights different colors according to voltage levels so logic 0 will have spot color and logic 1 will have different spot colors. This eliminates the need for propagating the faults to the circuit's

primary outputs (PO). However, this method is expensive and can be only used also for small designs because now we are observing rage number of nodes which will be inefficient as the chip will require many image processing efforts to test the chip while injected the input test vectors. For this, we can see the value of ATPG algorithms as they can be made to expose a faulty voltage value from the internals by propagating it to a primary output of the circuit where it will be easy to measure the voltage and determine if there is a problem.

6.3.2 Data Structures

Like any other software algorithms, ATPG algorithms need a data structure that can represent the possible search space for the procedure of testing. Binary Search Trees are used for that purpose. Figure 6-8 shows that according to the circuit *primary inputs* in (a), the tree can be traversed to arrive at the correct output value. The tree represents all possible (8) choices for circuit input patterns. Starting at the root node, if the left branch is selected, then signal A is 0, but if the right one is selected, then A is 1. At the second and third levels in the tree, next levels values are selected for other circuit inputs, B and C. So, it is another representation for the truth-table of this circuit that covers all possible input/output patterns. The leaf nodes represent the correct value for the output. All ATPG algorithms need to search a similar tree according to the design under test to find test patterns, and it may need to search the entire tree to prove that a fault cannot be detected. Noticing that the number of leaf nodes is $2^{No.\ of\ Inputs}$ which grows exponential, we must avoid the complete search situation.

A Binary Decision Diagram (BDD) can fully represent any switching function. Figure 6-8 shows the BDD for the circuit of Figure 6-8 (a).

The diagram can be read as follows:

1. Start from the root and try to reach the leaf nodes.
2. Traverse through all the possible paths from the root to leaves.
3. The values of branches represent a possible input combination which results in the output value on the leaf node.

For example, the rightmost path in the BDD is $A\ \bar{B}\ \bar{C}$ which produces the circuit output 1, which means we need input $ABC = 100$ to have output 1. It is straightforward to verify that all BDD paths produce the correct output logic. BDDs have been widely used in ATPG algorithms. However, it was reported to suffer problems of computational intractability for multiplier circuit designs which had extremely long runtimes that depend on the primary input patterns [32].

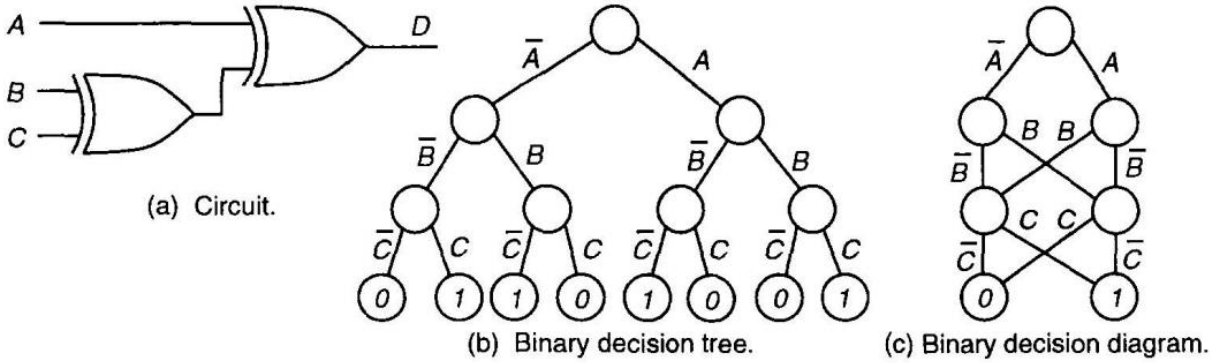


Figure 6-8 Binary decision tree and diagram [28]

6.3.3 Algorithms categories

Exhaustive

The simplest approach is to exhaustively search all the possible input patterns to find a set that successfully renders the fault sites observable at the primary outputs. But this approach is infeasible for the reasons discussed before.

Random

A random pattern generator starts by generating random input patterns, then with the help of a logic simulator, it simulates the fault effects in the circuit. The generated vector must contribute to increasing the test coverage, or it will be discarded as being not useful. The test coverage from all the randomly generated vectors is observed, and the procedure continues until the target coverage is reached. Figure 6-9 shows a flowchart for the typical random algorithms flow.

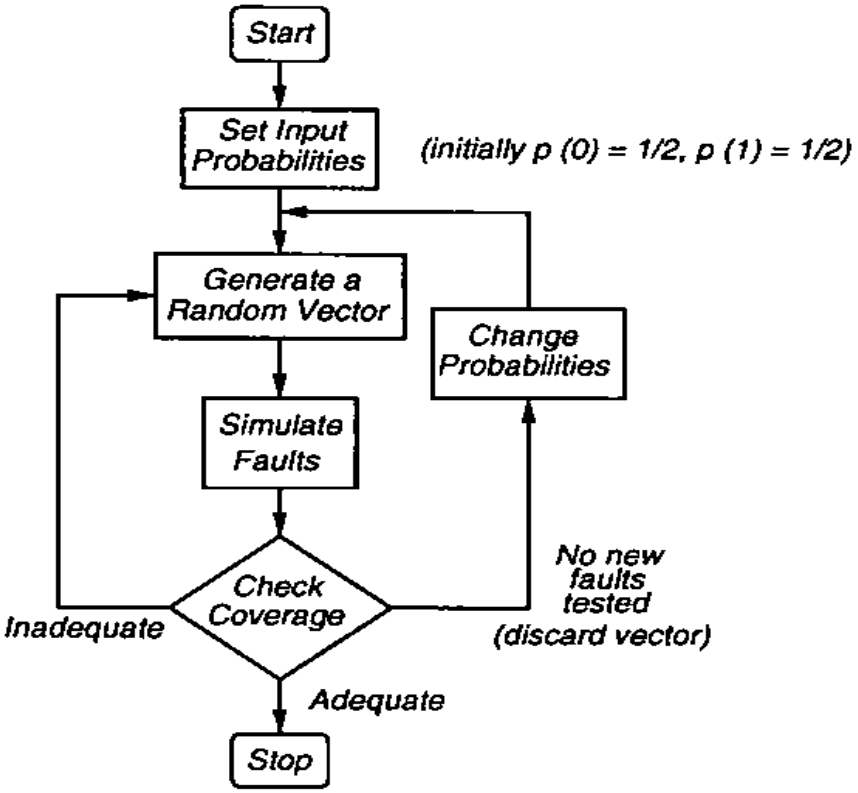


Figure 6-9 A typical random algorithm flowchart [33]

6.3.4 5-Valued Algebra

Before proceeding with the next method, an important concept needs to be explored, which is the ATPG algebra. It is a high-order Boolean set of notations to represent both the "good" and the "failing" circuit at the same time. This enables determining signal values for both circuits in on pass of ATPG. Since finding a test vector

Table 6-2 Roth's 5-valued algebra symbols [33]

Symbol	Meaning	Roth's 5-valued algebra	
		Good machine	Failing machine
D	(1/0)	1	0
$\overline{D}$	(0/1)	0	1
0	(0/0)	0	0
1	(1/1)	1	1
X	(X/X)	X	X

requires that a difference be maintained between the two machines, it is more efficient to represent both machines in the algebra rather than dealing with them separately. Roth [33] formulated the 5-valued algebra described in Table 6-2. The valued algebra is simple, a new truth table is obtained for, AND gate is shown in Figure 6-10. Similar operations can be extended for other gate types.

6.3.5 Path Sensitization Methods

This approach consists of three steps which will be described in some details

1. Fault Sensitization
2. Fault Propagation
3. Line Justification

These three steps are the constituents of what is widely known as the D-Algorithm.

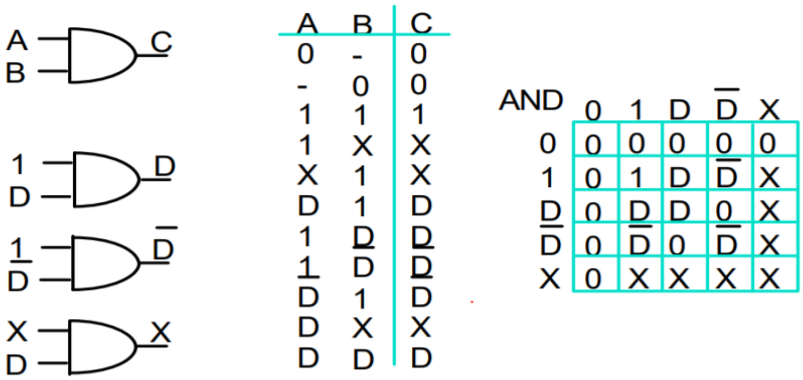


Figure 6-10 5-valued algebra of an AND gate [33]

The fault sensitization step aims at forcing the fault site with an opposite value to the fault, i.e., force it to 1 in case of stuck-at 0 faults. This is to distinguish good from the faulty circuit. The second step is fault propagation, where the fault is propagated through all the possible paths by adjusting the off-path internal signals to allow the propagation through the gate, for example, in Figure 6-11. If we want to propagate a fault through the path $A - h - k - l$, the f, j and E are called the off-path inputs and their logic should be $f = 1, j = 0$ and $E = 1$. Line justification step is where the assignments we made in the previous step are justified by the primary output, i.e., make sure that we can have those values of internal signal we assumed during the propagation step. In the second and third steps, we may be faced with a conflict, where a needed signal assignment contradicts some previously-made assumptions. To illustrate this issue, consider the circuit in Figure 6-11. Note that input *B* fans out to two AND gates. The fanout branches from *B* to the inputs of the two AND gates are labeled f and g, and the output of those gates are h and i. Our target is to generate a test for *B* stuck-at-0. The first step, as discussed, is to sensitize the fault by setting *B* to 1, and this leads to the signal assignments $f = D$ and $g = D$. The next step is Fault propagation requires to

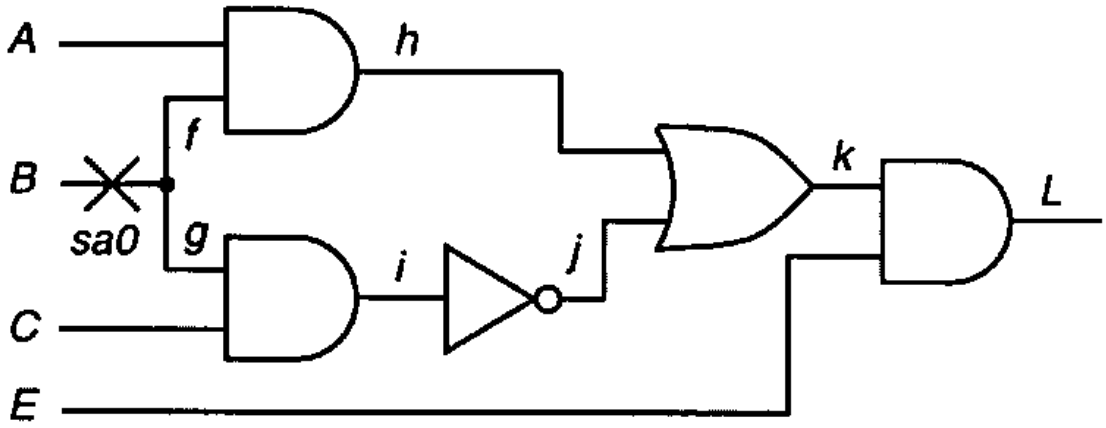


Figure 6-11 Circuit to demonstrate path sensitization [28]

select one of the following paths :-

1. Propagation along the path $f - h - k - L$.
2. Propagation along the path $g - i - j - k - L$.
3. Simultaneous propagation along both paths $f - h - k - L$ and $g - i - j - k - L$.

Let us choose path $f - h - k - L$ for propagation. This means that for every AND gate along the path, the off-path inputs should be set to logic 1, while for every OR gate along the path, the off-path inputs should be set to logic 0. This results in the signal assignments $A = 1, j = 0\ and\ E = 1$ to line justify (step 3). This we can assign $i = 1$ to justify $j = 0$ by backward logic simulation of inverter j but AND gate i then needs to have an output of 1, but it already has input $g = D$. Using the 5-valued algebra table reveals that there is no way to get $i = 1$ when an input of AND already was set to D, so we retract the assignment $j = 0$ and try the alternative assignment $j = 1$, but this blocks the fault propagation through the path $f – h – k – L$. So we neglect that path and try the remaining 2 and 3 listed above. We choose the third path. We change our *fault propagation* approach and now make the assignments $A = 1$, $C = 1$, and $E = 1$ to ensure fault propagation. Forward logic simulation from these assignments yields $i = D, j = \overline{D}$ and $h = D$, $k = 1$ Notice that k has a fixed value which means that the fault is untestable this path because it should have a variant value "D" so that we can observe a difference between good and failing circuits. The only remaining path is $g - i - j - k - L$. 5-valued logic simulation gives $i = D,\ j = \overline{D}, k = \overline{D}\ and\ L = \overline{D}$, we still need to justify $h = 0$. This is achieved by backward simulation for AND gate h, with input $f = D$ by setting input $A = 0$. The only pattern to test for B stuck-at-0 is $ABCE = 0111$, and this produces the output $\overline{D}$ i.e., $L = 0$ in the good machine, and $L = 1$ in the failing machine.

Chapter 7

Proposed SET Test Vector Generation Flow

7.1 Description

The methodology of obtaining SET test vectors is very similar to the one employed to investigate the transition fault model as the typical way of testing for a transition fault is to find a pair of input vectors such that the first vector's, which is called the initialization pattern, objective is to set up an initial state before any transition, whereas the other vector, known as propagation pattern, guarantees that the transition occurs and is observed at one of the primary outputs [9] [10]. The propagation pattern can be used directly for SET to find the pulse propagation path. The predefined SET model is then used to test whether the input SET pulse can reach the primary output from an electrical point of view. The Automatic Test Pattern Generation (ATPG) capability of Mentor Graphics Tessent FastScan® tool is applied on circuits from ISCAS '85 benchmarks [36]. The tool is provided with the gate-level netlist of the design under test and list of fault sites, and then propagation vectors are extracted from the automatically generated test vectors for transition faults. These propagation vectors serve as the test vectors for SET faults.

The sensitized paths by the generated vectors are then analyzed using the electrical modeling approach discussed in Chapter 4 to determine whether the given input pulse will be able to propagate to the outputs of the circuit or not.

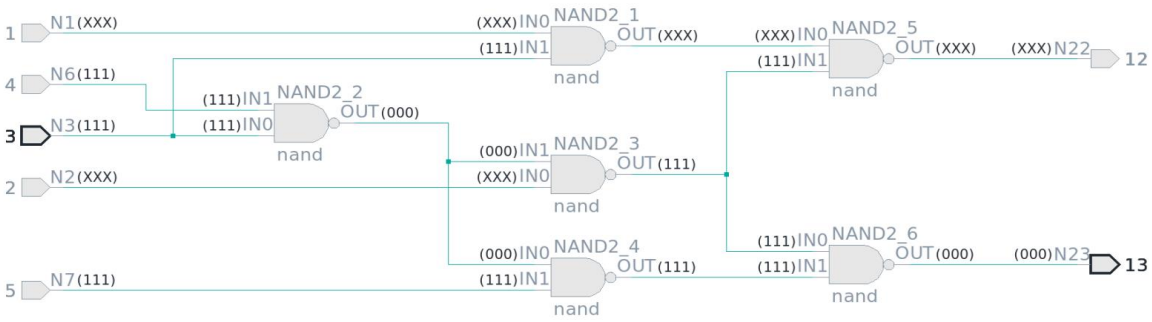


Figure 7-1 c17 benchmark

7.2 Results

7.2.1 c17 benchmark

A simple Verilog netlist with only five inputs, two outputs, and composed of six 2-input NAND gates, shown in Figure 7-1. A SET fault at the input node N3 is to be tested. So, the tool is run such that it generates a test pattern to expose a transition fault at N3. The tool generated the pattern XX111 for inputs N1, N2, N3, N6, N7, respectively, and the fault should be observed on N13, and its correct state should be 0. This input pattern sensitizes the logic path shown in Figure 7-2. Notice that the off-path inputs are all set to logic 1, as the path is composed of NAND gates, so the off-path inputs must be set to 1, or the fault will be logically masked from appearing at the output. Now the path should be electrically simulated to see whether a given pulse will propagate through it. Therefore, the path is then synthesized using the XFAB Xh018 HDSVT standard cells library and then our SET model approach is applied to predict the output pulse and is also simulated using Cadence Spectre simulator to verify the accuracy of the model.

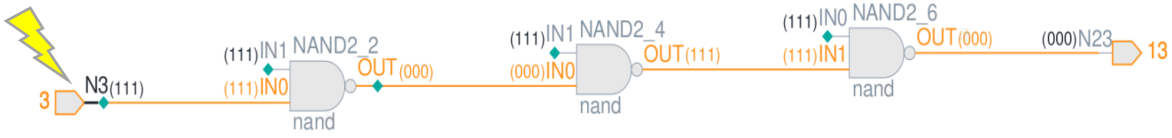


Figure 7-2 Sensitized logic path to test SET at input node N3.

Table 7-1 SET modeling result for the path sensitized for testing fault at N3.

#	Source benchmark	Test vectors		Input pulse		Output pulse (simulation)		Output pulse (model)	
		Primary input	Primary output	Height [V]	Width [ps]	Height [V]	Width [ps]	Height [V]	Width [ps]
1	c17	X X 1 1 1	X ***0***	1.1	300	1.865	168	1.91	286
2	c17			1.1	100	0.004	-	0.023	10

The table contains two scenarios with only different input pulse parameters. The first case is a relatively wide SET pulse,300 ps width, and 1.1 V height, which arrived at the output with amplified height and narrower width. So, this pulse can be considered to have successfully caused an SEU if the output of this combinational circuit is connected to the input of a memory element. By this, we can say that for this design, a SET pulse with width 300 ps and height 1.1 V generated due to particle strike at N3 will cause a fault in the system, and this fault can be tested using the excitation input pattern X X 1 1 1.

In the second case, a narrower pulse was used so and observing the output pulse reveals that it has a negligible height and width, which means this SET pulse has been electrically masked, although the circuit logic allows it to propagate. This input vector should be disregarded if the user is confident about the radiation environment will not cause SETs wider than 100 ps.

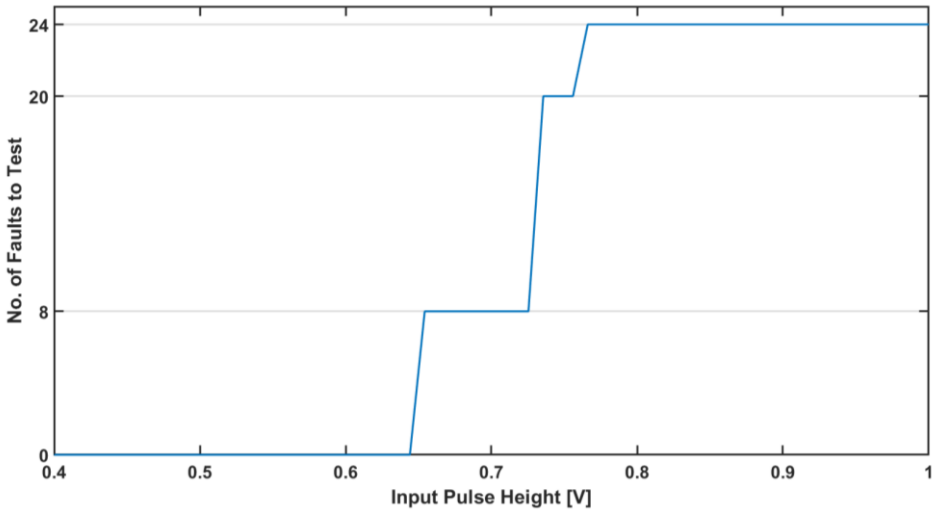


Figure 7-3 Effect of pulse shape on the required testing effort in c17 benchmark

As the input SET pulse energy decreases, which is manifested by reduction in width or height, it becomes less probable for the induced pulse to propagate to the output. Figure 7-3 shows this dependency for a fixed pulse width which is 200 ps. The figure can be interpreted as follows;

1) If the pulse height is less than 0.65 V, then no need to perform any test for SETs as the pulse will not be able to propagate through any single gate in this circuit.
2) If the pulse height is around more than 0.65 V and less than about 0.75 V, then there will be only 8 fault locations (from a total of 24 for this circuit) will be prone

to SETs but the testing effort now is reduced as the fault locations are saved by about 67% hence fewer test vectors and steps are required. Similarly, for a pulse around 0.75 V, the SET can affect more locations and the number of faults to be tested is increased to 20.

3) If the pulse height is above 0.8 V, then all the fault sites can be prone to this pulse and all paths should be tested and examined, also the generated test vectors will be the same set as the transition fault test vectors set.

The staircase shape of Figure 7-3 is due to the nature of the benchmark, as all its paths are composed of only one, two or three gates and all the gates are similar.

7.2.2 c432, c3540, c499, and c7552 benchmarks

These netlists are larger and provide a variety of gate types to test our flow. Table 7-2 contains examples of the logic paths that propagate the fault upon applying the automatically generated input test vector generated to detect a specific fault site from c432, c3540, c499, and c7552 benchmarks. The fault sites for these examples are arbitrarily chosen. Table 7-2 contains the input vector, the expected correct output, and the characteristics of the output pulse as predicted by our electrical model, which is also compared to the circuit simulation of the same path. 'X' in input and output columns denotes do-not-care values, and the bits in italic, bold font in the primary output column highlight the location where the output pulse will be observed. In rows 1, 2, 4 and 6, a propagation of the pulse to the primary output will cause an SEU if latched into a memory instance. However, the pulse failed to reach the primary output in cases 3, 5 and 7. Accordingly, no failure incident occurred. The characteristics of the induced electrical pulse due to radiation interaction with a chip are determined by several factors like linear energy transfer and impact ionization which are outside the scope of this work, but the models used are flexible enough to handle any shape of SET pulses generated. The sensitized paths for both input vectors are shown in Figure 7-4.

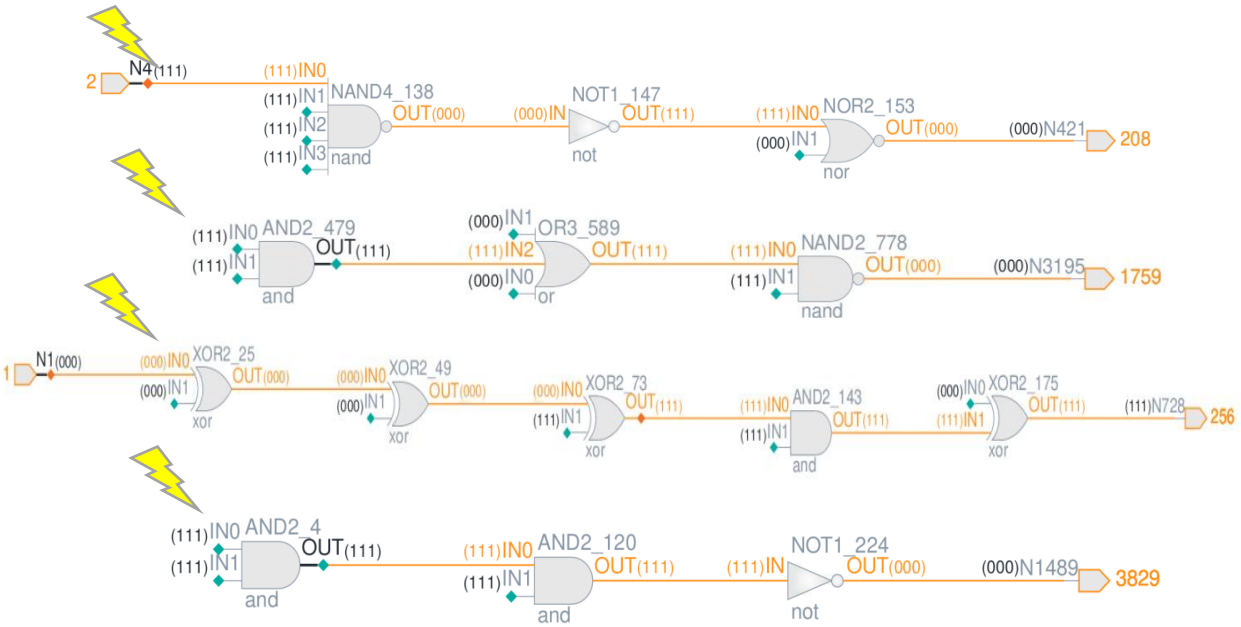


Figure 7-4 Sensitized paths from c432, c3540, c499 and c7552 benchmarks, respectively

Table 7-2 The predicted output pulse values vs. SPICE simulation upon applying the generated test vectors.

#	Source benchmark	Test vectors		Input pulse		Output pulse (simulation)		Output pulse (model)	
		Primary input	Primary output	Height [V]	Width [ps]	Height [V]	Width [ps]	Height [V]	Width [ps]
1	c432	1 0 X 0 (27'X)	111***0***111	1.2	300	1.26	185	1.3	286
2	c3540	0(5'X)1(22'X)1(20'X)	0X***0***(19'X)	1.2	200	1.21	534	1.26	505
3	c3540			0.96	200	0.018	102	0.091	43
4	c499	(7'0)1(13'0)1(15'0)1001	(4'0)***1***001(13'0)1(10'0)	1.05	350	1.25	255	1.25	380
5	c499			0.9	350	0	-	0	-
6	c7552	X0(64'X)11(139'X)	(50'X)0X***0***(55'X)	1.2	300	1.25	298	1.28	340
7	c7552			0.9	300	0	-	0	-

The flow can be summarized in the following flow chart.

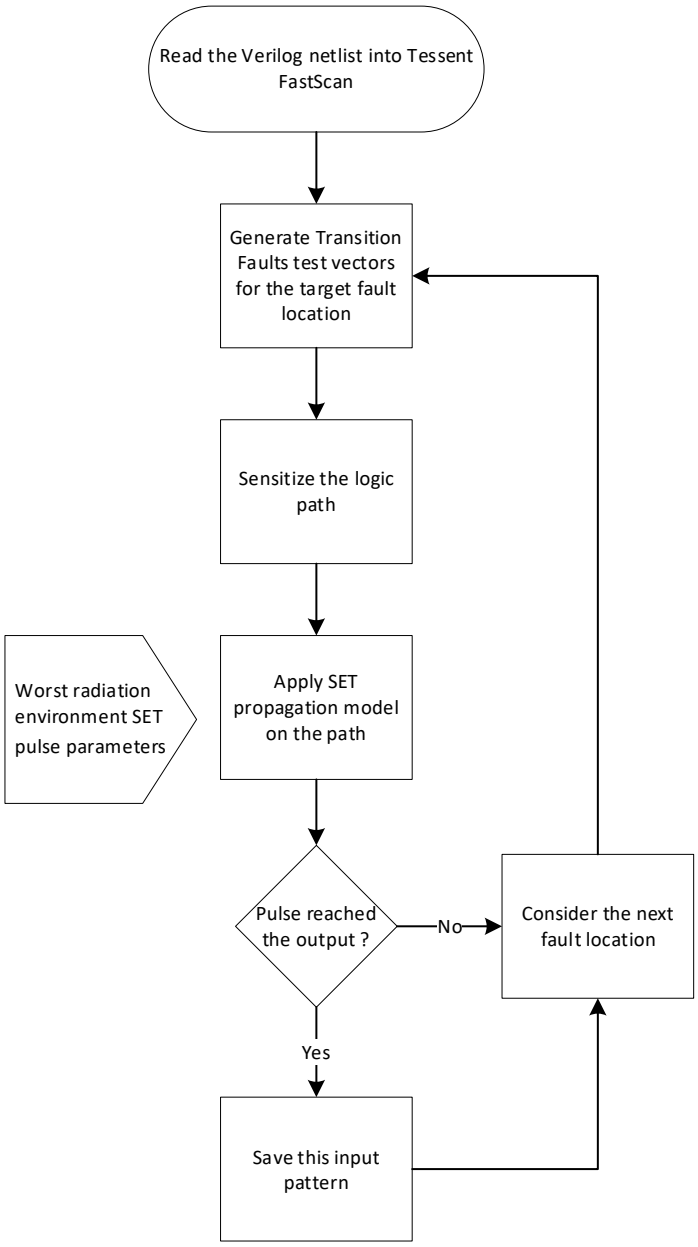


Figure 7-5 Test vector generation flow.

The above flow can be used in the case of a known input pulse shape to determine the test vectors for the design under test. A modification can be done to the flow to be capable of determining the sensitivity of the design in terms of the minimum pulse that can cause an SEU. The flow should start with an initial pulse that is not energetic enough to propagate through a path, then iteration over the propagation modeling step until

reaching a minimum pulse that starts to propagate, any larger pulse (higher amplitude or width) will be guaranteed to propagate.

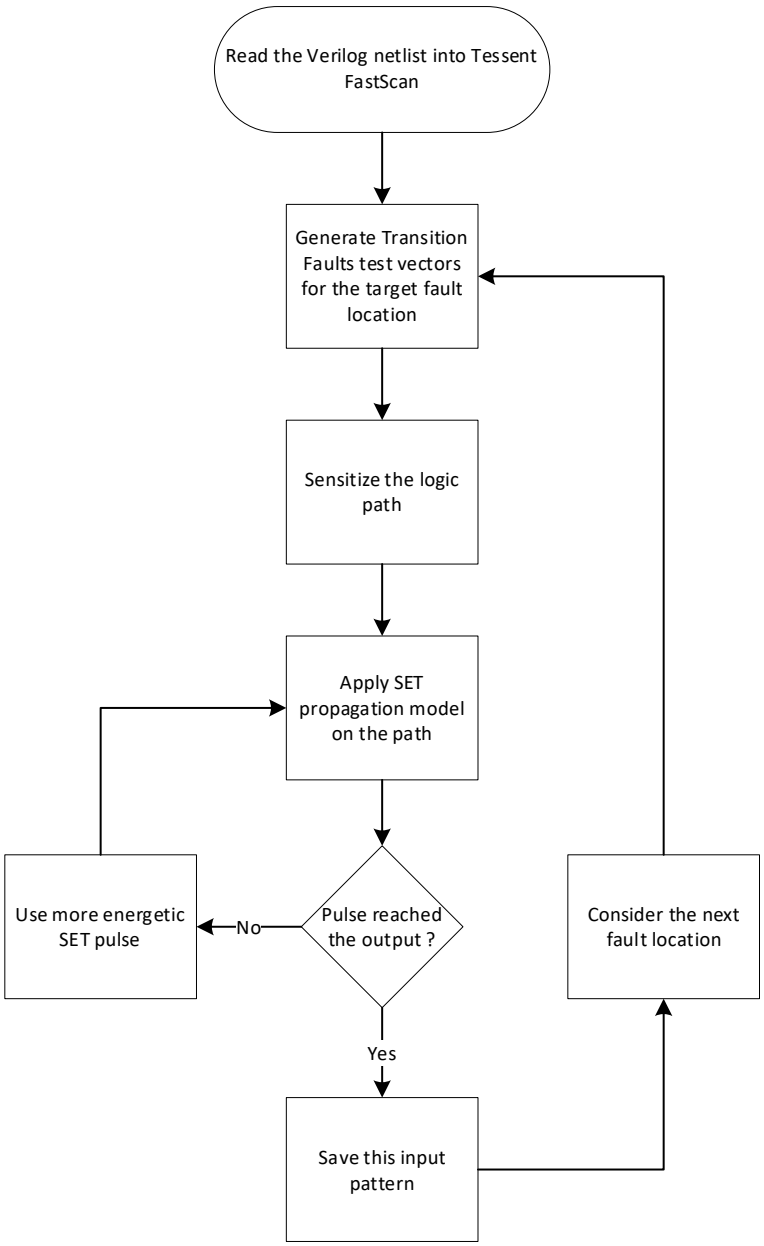


Figure 7-6 Path sensitivity characterization flow.

Chapter 8

Conclusion and Future Work

In this work, a full flow for modeling the SET propagation through ASIC standard cells and automatic generation of test vectors for combinational circuits are proposed and tested over commercial standard cell libraries. Introduction of a new method for fitting the SET propagation model, which showed higher accuracy in matching the electrical simulations results model than the method reported in literature Automation scripts were developed to automate the process of characterization of a complete library with minimal manual intervention, which enables easier and quicker characterization of complete standard cells libraries. A modification to the normal flow of ATPG from the industrial tool is presented to fit the problem of SET test vectors generation.

To make the most use of this flow, a methodology should be developed to identify the worst-case test vector conditions which need augmentation of the vectors generation flow with a smart process to identify true worst-case input patterns among the generated vectors. This process should also guarantee high fault coverage with the smallest number of vectors needed. This is currently ongoing research.

A test chip will be fabricated using TSMC 65nm node to be used for experimental verification of the proposed flows.

Appendix A

This appendix describes and lists the different scripts used throughout this work.

I. Verilog-A

Verilog-A is part of the Verilog-AMS Hardware Description Language (HDL) language standard [37], which defines a behavioral language for analog and mixed-signal systems. It is derived from the IEEE 1364 Verilog HDL specification. Verilog-A is used in this work to model the SET pulses from the discussed equation in Chapter 4, to be used in electrical simulations.

A. Code for the double exponential current pulse

```
// VerilogA for SEE, DECS, veriloga

`include "constants.vams"
`include "disciplines.vams"

module DECS(p,n,trig);
    // my terminals
    inout p, n,trig;
    electrical p, n,trig;
    real I_d;
    real trig_time;
    real abs_time;
real td1;
real td2;
parameter real Ipeak = 100e-6;
parameter real tr = 2p;
parameter real tf = 10p;
parameter real rise_offset = 10p;
parameter real fall_offset = 20p;

    analog
    begin

        @(initial_step)
        begin

            I_d = 0;
            trig_time=-5000;

        end

@(cross(V(trig),+1))begin
 trig_time = $abstime ;
    td1 = rise_offset;// - trig_time;
    td2 = fall_offset;// - trig_time;
end
abs_time = $abstime - trig_time; // to shift the time by trigger time

//single exponent (test)
```

```
//I_d =Ipeak * (exp(- (abs_time) / tf));

//double exponential
if(abs_time <= td1)
I_d = 0;
else if (abs_time > td1 && abs_time < td2)
I_d =Ipeak * (1-exp(- (abs_time - td1) / tr));
else
I_d =Ipeak * (exp(- (abs_time - td2) / tf) - exp(- (abs_time - td1) / tr));

        I(p,n) <+ I_d;
    end
endmodule
```

B. Code for the double sinusoidal voltage pulse

```
// VerilogA for SEE, DSVP, veriloga

`include "constants.vams"
`include "disciplines.vams"

module DSVP(p,n,trig);
    // my terminals
    inout p, n,trig;
    electrical p, n,trig;
    real V_d,pi=3.14;
    real omega_1,omega_2,t2,t3,t1;
    real abs_time,trig_time;

parameter real ref =1;
parameter real A =1;
parameter real t0 = 10p;
//parameter real t1 = 20p;
parameter real tw_in = 50p;

    analog
    begin
        @(initial_step)
        begin

            V_d = 0;
            trig_time=-5000;
            t1 = t0+tw_in/3;
            t2 = tw_in+t0;
            t3 = t1+t2-t0;
            omega_1 =pi/(t1-t0);
            omega_2 =pi/(t3-t2);
        end

@(cross(V(trig),+1))begin
 trig_time = $abstime ;

end
abs_time = $abstime - trig_time; // to shift the time by trigger time
```

```
if (abs_time <= t0 || abs_time >= t3)
    V_d=0;
else if ( abs_time >= t0 && abs_time < t1)
    V_d=0.5*A*(sin(omega_1*(abs_time-t0)-pi/2)+1);
else if ( abs_time >= t1 && abs_time < t2)
    V_d=A;
else if ( abs_time >= t2 && abs_time < t3)
    V_d=0.5*A*(sin(omega_2*(abs_time-t2)+pi/2)+1);

        V(p,n) <+ abs(ref - V_d);
    end
endmodule
```

II. SKILL and OCEAN

SKILL is a scripting language that is adopted by Cadence to allow for automation of various EDA processes in its suite of tools [38]. OCEAN is a subset of SKILL language with a higher-level syntax that simplifies the used model of some SKILL functionalities.

A. OCEAN script to run characterization for the specific gate.

The script starts with defining the model files for the used devices, then places the netlist of the design (characterization test bench). The simulation options are then defined along with the parametric sweep values.

```
simulator( 'Spectre )
modelFile(
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/config.scs"
"default")
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/param.scs" "
3s")
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/bip.scs" "tm
")
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/cap.scs" "tm
")
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/dio.scs" "tm
")
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/mos.scs" "tm
")
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/photo.scs" "
tm")
```

```
    '("/pdks/Xfab/Xh018/xh018/cadence/v8_0/spectre/v8_0_1/lpmos/res.scs" "tm
")
)
analysis('tran ?stop "1n"  ?errpreset "conservative"  )
desVar(   "ref" 0   )
desVar(   "tw_in" 100p  )
desVar(   "cap_val" 10f )
desVar(   "v_in" 2.5    )
envOption(
    'analysisOrder  list("tran")
)
temp( 27 )

design("~/simulation/EO2_3VX4/spectre/schematic/netlist/netlist")
paramAnalysis("v_in" ?values '(1e-
05     0.33334     0.66667           1      1.3333      1.6667           2
    2.3333      2.6667           3)
paramAnalysis("tw_in" ?values '(1e-13  3.3422e-11  6.6744e-11  1.0007e-
10  1.3339e-10  1.6671e-10  2.0003e-10  2.3336e-10  2.6668e-10       3e-10)
))
paramRun()
selectResults('tran)
out=outfile("~/simulation/EO2_3VX4/my_run/sim_test.out" "w" )
ocnPrint(?output out ?numberNotation 'engineering VT("/out_dut"))
```

B. SKILL script to netlist a complete standard cells library

The script assumes that the library is installed and defined in the cds.lib file.

```
simulator('spectre)

gates=ddGetObj("<Target_Library_Name>")~>cells~>name

foreach(g gates

      dcv=dbOpenCellViewByType("propagation_modeling" g "schematic" "sch
ematic" "w"

      )

xfbinst=dbOpenCellViewByType("D_CELLS_3V" g "symbol")

new_inst=dbCreateInst( dcv xfbinst "myinst0" list(0 0) "R0" 1)

dbClose(xfbinst)

schCheck(dcv)
```

```
dbSave(dcv)

dbClose(dcv)

design("propagation_modeling" g "schematic")

nn=buildString('(g "scs") ".")

createNetlist()
)
```

III. Matlab

Matlab is employed mainly to perform the modeling and data manipulation steps, such as sweeps generation, model fitting, parameters extraction from transient simulations. In addition, it is used in generating other used scripts and simulation files besides handling the communication between different tools and scripts. It is used to create a customized ocean script for every library gate.

A. Flow control scripts

i. Top level

```
cells = {'NA2_3VX4'};%{'IN_3VX4'};%{'AND2_3VX0'};%{'NO2_3VX0'};%{'IN_3VX0'};%,
'AND4_3VX1'};
ports_conns = { {'A' 'B' 'Q'; 2 1 3 } ,{'A' 'B' 'C' 'D' 'Q'; 1 0 1 2 3 } };%%%%%%%
netlists_path = '~/simulation';

for cell_id = 1:length(cells)
    cells{cell_id}
    my_run_dir = [netlists_path,'/',cells{cell_id},'/my_run'];
    gate_netlist_path =
[netlists_path,'/',cells{cell_id},'/spectre/schematic/netlist/netlist'];
    mkdir(my_run_dir);


[sim_netlist,instnce_name]=create_sim_netlist(gate_netlist_path,ports_conns{cell_id});
    [ocn_script]= create_sim_ocean(sim_netlist,my_run_dir);
    cmd = ['./run_ocean.sh ',ocn_script , ' > ',[my_run_dir,'/my_run.log']];
    system(cmd);
end
```

ii. Preparing the final simulation netlist

```
function [sim_netlist,instnce_name]=create_sim_netlist(gate_netlist_path,ports_conns)
```

```
% ports_conns = { 'A' 'B' 'C' 'Q'; 0 1 1 2};
%0 gnd
%1 vdd
%2 test_input
%3 test_output
vdd_name='vdd3!';
TB_Netlist_path='./templts/TB.scs';
sim_netlist =gate_netlist_path;

if ~isfile([gate_netlist_path,'_gen'])
    movefile(gate_netlist_path,[gate_netlist_path,'_gen']);
end
 t_gate = fileread([gate_netlist_path,'_gen']);
t_tb = fileread(TB_Netlist_path);
indx =strfind(t_gate,'myinst0');
instantiation_line = t_gate(indx:end);%get the myinst0 line as it's the last line in
the gate netlist
tokens = strsplit(instantiation_line,{' ','(',')'});
instnce_name = tokens{end};
if(size(ports_conns,2) ~= length(tokens)-4)
    error('size of conns provided doesnt match the gate ports');
end
ports={};
for i = 1:length(tokens)-4
    if(ports_conns{2,i}==0)
        ports{i}='0';
    elseif(ports_conns{2,i}==1)
         ports{i}=vdd_name;
    elseif(ports_conns{2,i}==2)
         ports{i}='in_dut';
    elseif(ports_conns{2,i}==3)
         ports{i}='out_dut';
    end
end
new_instnce_line=[newline ,tokens{1},' ( ',strjoin(ports),' ',strjoin(tokens(end-2:end-
1)),' ) ',tokens{end}];
t_gate(indx:indx+1)=['//'];%comment the old myinst0 line
t_all= [t_gate t_tb new_instnce_line];
f_out = fopen(sim_netlist,'w');
fprintf(f_out,'%s',t_all);
fclose(f_out);
```

iii. Generation of OCEAN scripts to control the simulator

```
function [ocn_script]= create_sim_ocean(sim_netlist,out_dir)

[tw_in, v_in] = gen_sweep_pars(3,300*10^-12,27,27);

ocnHeader = './templts/ocnHeader.ocn';
out_waveform_file = [out_dir,'/sim.out'];
ocn_script = [out_dir,'/sim.ocn'];
```

```
t_head = fileread(ocnHeader);

t_design = [newline ,'design("',sim_netlist,'")'];

t_param = [ newline,'paramAnalysis("v_in" ?values ''(',num2str(v_in),')'...
                ,newline,'paramAnalysis("tw_in" ?values ''(',num2str(tw_in),')'...
                ,newline,'))'];

t_footer= [newline ,'paramRun()',...
            newline,'selectResults(''tran)',...
            newline,'out=outfile("',out_waveform_file,'" "w" )',...
            newline,'ocnPrint(?output out ?numberNotation ''engineering
VT("/out_dut"))'];%,...
            %newline,'ocnPrint(?output out ?numberNotation ''engineering
VT("/in_dut"))'];
f_out = fopen(ocn_script,'w');
fprintf(f_out,'%s',[t_head t_design t_param t_footer]);
end
```

B. Data manipulation and modeling

i. Models' fitting

```
gili_model=0;
testing_mode=1;
gf=fittype('prop_model(alpha,c,T,VDC,td1,Vdd,tw_in,v_in)','coefficients',{'c','T','alph
a','VDC','td1'},'independent',{'tw_in','v_in'},'problem','Vdd');
options = fitoptions('Method','NonlinearLeastSquares','Algorithm','Levenberg-
Marquardt','Robust','Off');

options.StartPoint = [20 150 1 Vdd/2 50];
% fit height
if(~testing_mode)
    if(gili_model)
        v_model=fit([tw_in',v_in'],v_out',gf,options,'problem',Vdd);
    else
        v_model=fit([tw_in',v_in'],v_out','cubicinterp','Normalize','on');
    end
else
    if(gili_model)
        load(['./models/',dut_name,'_gimodel.mat']);
    else
        load(['./models/',dut_name,'_model.mat']);
    end
end


% fit width
gf=fittype('tw_out_model(a0,a1,b0,b1,t0,ti,tw_in,v_in)','coefficients',{'a0','a1','b0',
'b1','t0','ti'},'independent',{'tw_in','v_in'});
%tw_out_l=reshape(tw_out,1,size(tw_out,1)*size(tw_out,2));
```

```
figure(4)
options = fitoptions('Method','NonlinearLeastSquares','Algorithm','Levenberg-
Marquardt','Robust','Off');

if(~testing_mode)
    if(gili_model)
        tw_model=fit([tw_in',v_in'],tw_out',gf,options);
    else
        tw_model=fit([tw_in',v_in'],tw_out','cubicinterp');
    end
end
% save the model
if(testing_mode)

save(['./models/',dut_name,'_test_model.mat'],'v_model','tw_model','max_err_v','max_err
_t','Vdd');
else
    if(gili_model)

save(['./models/',dut_name,'_gimodel.mat'],'v_model','tw_model','max_err_v','max_err_t'
,'Vdd');
    else

save(['./models/',dut_name,'_model.mat'],'v_model','tw_model','max_err_v','max_err_t','
Vdd');
    end
end
```

ii. Reading and analyzing the simulation output waveforms

```
function [tw_in,v_in,tw_out,v_out]=parse_sim(sim_out_file, correct_output)
Vdd=3;
plot_trans=0;
v_threshold=0.04;
fid = fopen(sim_out_file);
sim_count=0;
while(~feof(fid))
sim_count=sim_count+1;
data_1= textscan(fid,'v_in = %f\ntw_in =
%f','CommentStyle','\n','CommentStyle','time');
v_in(sim_count)= data_1{1};
tw_in(sim_count)=data_1{2};
data_2 = textscan(fid,'%f %f\n','CommentStyle','\n');
data_X{sim_count}= data_2{1};
data_Y{sim_count}= data_2{2};
end
fclose(fid);

for indx =1:sim_count%[390 391 392]

    data_X_filtered=data_X{indx};
    data_Y_filtered=data_Y{indx};
```

```
    if(correct_output > 0)
        data_Y_temp=correct_output-(data_Y_filtered);
    else
        data_Y_temp=data_Y_filtered;
    end
    if(plot_trans)
     figure
     %hold on
        plot(data_X_filtered,data_Y_filtered,'.');
    ttl=['tw\_in= ',tw_in_val{indx},'ps v\_in= ' ,v_in_val{indx}];
    title(ttl)
    end
    [v_out(indx), indx_max]=max(data_Y_temp);
    %tw_in(indx)=str2double(tw_in_val{indx});
    %v_in(indx)=str2double(v_in_val{indx});
    if(abs(v_out(indx))>v_threshold)
        difference = abs(v_out(indx)/2-data_Y_temp);
        [~,first_edge] = min(difference(1:indx_max-1));

        [~,secnd_edge] = min(difference(indx_max+1:end));
        if(indx_max < length(difference))
            secnd_edge=secnd_edge+indx_max;
        else
            secnd_edge = indx_max;
        end
        if(isempty(first_edge))
                first_edge=secnd_edge;
        end
        tw_out(indx)=abs(data_X_filtered(first_edge)-data_X_filtered(secnd_edge));
     if(plot_trans)
    hold on
    plot(data_X_filtered(first_edge),data_Y_filtered(first_edge),'ro');
    plot(data_X_filtered(secnd_edge),data_Y_filtered(secnd_edge),'ro');
     end
    else
        tw_out(indx)= 0;
    end

end
tw_in=tw_in*10^12;
tw_out=tw_out*10^12;
nn=length(unique(v_in));
v_in =reshape(v_in,nn,length(v_in)/nn);
tw_in =reshape(tw_in,nn,length(tw_in)/nn);
v_out =reshape(v_out,nn,length(v_out)/nn);
tw_out =reshape(tw_out,nn,length(tw_out)/nn);
end
```

Appendix B

I. Piecewise cubic interpolation.

Assume we have a set of data in the form of (x,y) pairs, and it is needed to find values for y that aren't within the data set. Interpolation is a widely used method for this purpose. Here, piece-wise cubit interpolation is described. The whole data set is divided into subintervals (pieces) on which we try to find an interpolating polynomial for each subinterval. This assumes that the function has a smooth behavior inside each subinterval.

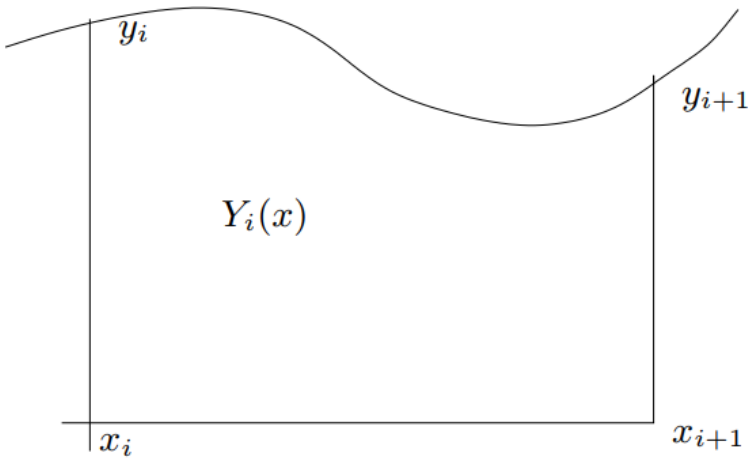


Figure 0-1 Piece of the function to be interpolated

We want to find the curve $Y_i(x)$ that interpolates the two data points between y_{i+1} and y_i.

Cubic interpolation method assumes that we have a cubic polynomial,

$$Y_i(x) = a\,(x - x_i)^3 + b\,(x - x_i)^2 + c\,(x - x_i) + d$$

Our target is to calculate the coefficients a, b, c, d.

Firstly, we get the first derivative of our interpolant,

$$Y_i'(x) = 3a\,(x - x_i)^{\,2} + 2b\,(x - x_i) + c$$

We can find d and c directly to be

$$Y_i(x = x_i) = y_i = d$$

$$Y'(x = x_i) = y_i' = c$$

Now we try to find expressions for the coefficients a and b

$$Y_i(x_{i+1}) = \; a\,(x_{i+1} - x_i)^{\,3} + b\,(x_{i+1} - x_i)^2 + c\,(x_{i+1} - x_i) + d$$

Let $h_i = x_{i+1} - x_i$

$$Y_i(x_{i+1}) = y_{i+1} = a\,h_i^3 + b\,h_i^2 + c\,h_i^2 + d$$

$$Y_i'(x_{i+1}) = y_{i+1}' = 3a\,h_i^2 + 2b\,h_i + c$$

The previous two equations can be re-written as,

$$\begin{bmatrix} h_i^3 & h_i^2 \\ 3h_i^2 & 2h_i \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_{i+1} - ch_i^2 - d \\ y_{i+1}' - c \end{bmatrix}$$

Solving this system of equations yields the values of a and b as,

$$a = \; -2\left(\frac{y_{i+1} - d}{h_i^3}\right) + \frac{y_{i+1}' + c}{h_i^2}$$

$$b = 3\left(\frac{y_{i+1} - d}{h_i^2}\right) - \frac{y_{i+1}' + 2c}{h_i}$$

We can calculate the derivatives as follows,

$$Y_i'(x_i) = y_i' = \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}$$

$$Y_{i+1}'(x_{i+1}) = y_{i+1}' = \frac{y_{i+2} - y_i}{x_{i+2} - x_i}$$

Hence, given the values $y_{i-1} \to y_{i+2}$ and $x_{i-1} \to x_{i+2}$, which are the measured data points, we can find the complete cubic interpolant expression in the interval $[x_i, x_{i+1}]$. This is repeated for all the "pieces" of the data.

Bi-cubic interpolation is an extension of this method and is used for surface interpolation. The algorithm is defined completely in Matlab and is used throughout this thesis for the proposed model of electrical modeling of the SET propagation.

References

[1] "Radiation Basics," *NRC Web*. https://www.nrc.gov/about-nrc/radiation/health-effects/radiation-basics.html

[2] J. Perez, "Why Space Radiation Matters," *NASA*, Apr. 13, 2017. http://www.nasa.gov/analogs/nsrl/why-space-radiation-matters

[3] M. Pazianotto *et al.*, "Influence of clouds on the cosmic radiation dose rate on aircraft," *Radiation protection dosimetry*, vol. 161, Jun. 2014, doi: 10.1093/rpd/ncu186.

[4] F. Cortese *et al.*, "Vive la radiorésistance!: Converging research in radiobiology and biogerontology to enhance human radioresistance for deep space exploration and colonization," *Oncotarget*, vol. 9, Feb. 2018, doi: 10.18632/oncotarget.24461.

[5] "What Is the Solar Cycle? | NASA Space Place." https://spaceplace.nasa.gov/solar-cycles/en/

[6] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 583–602, Jun. 2003, doi: 10.1109/TNS.2003.813129.

[7] A. P. Mouritz, Ed., "4 - Strengthening of metal alloys," in *Introduction to Aerospace Materials*, Woodhead Publishing, 2012, pp. 57–90. doi: 10.1533/9780857095152.57.

[8] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design Test of Computers*, vol. 22, no. 3, pp. 258–266, May 2005, doi: 10.1109/MDT.2005.69.

[9] A. Abubakr, A. Ibrahim, Y. Ismail, and H. Mostafa, "The Impact of Soft Errors on Memristor-Based Memory," in *2017 New Generation of CAS (NGCAS)*, Sep. 2017, pp. 229–232. doi: 10.1109/NGCAS.2017.72.

[10] M. Andjelkovic, A. Ilic, Z. Stamenkovic, M. Krstic, and R. Kraemer, "An overview of the modeling and simulation of the single event transients at the circuit level," in *2017 IEEE 30th International Conference on Microelectronics (MIEL)*, Nis, Oct. 2017, pp. 35–44. doi: 10.1109/MIEL.2017.8190065.

[11] P. V. Dressendorfer, "Basic mechanisms for the new millennium," Sandia National Labs., Albuquerque, NM (United States), SAND-98-1388C; CONF-980705-, Sep. 1998. Accessed: Apr. 09, 2021. [Online]. Available: https://www.osti.gov/biblio/658465-basic-mechanisms-new-millennium

[12] T. Heijmen, D. Giot, and P. Roche, "Factors that impact the critical charge of memory elements," in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, Jul. 2006, p. 6 pp.-. doi: 10.1109/IOLTS.2006.35.

[13] C. Lazzari, G. Wirth, F. Kastensmidt, L. Anghel, and R. Reis, "Asymmetric transistor sizing targeting radiation-hardened circuits," *Electrical Engineering*, vol. 94, pp. 11–18, Mar. 2011, doi: 10.1007/s00202-011-0212-8.

[14] G. C. Messenger, "Collection of Charge on Junction Nodes from Ion Tracks," *IEEE Transactions on Nuclear Science*, vol. 29, no. 6, pp. 2024–2031, Dec. 1982, doi: 10.1109/TNS.1982.4336490.

[15] T. Merelle *et al.*, "Criterion for SEU occurrence in SRAM deduced from circuit and device Simulations in case of neutron-induced SER," *IEEE Transactions on*

Nuclear Science, vol. 52, no. 4, pp. 1148–1155, Aug. 2005, doi: 10.1109/TNS.2005.852319.
[16] C. Hu, "Alpha-particle-induced field and enhanced collection of carriers," *IEEE Electron Device Letters*, vol. 3, no. 2, pp. 31–34, Feb. 1982, doi: 10.1109/EDL.1982.25467.
[17] L. B. Freeman, "Critical charge calculations for a bipolar SRAM array," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 119–129, Jan. 1996, doi: 10.1147/rd.401.0119.
[18] S. Barcelo, X. Gili, S. A. Bota, and J. Segura, "An SET propagation EDA tool based on analytical glitch propagation model," in *2013 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, Oxford, United Kingdom, Sep. 2013, pp. 1–5. doi: 10.1109/RADECS.2013.6937388.
[19] P. E. Dodd, M. R. Shaneyfelt, J. A. Felix, and J. R. Schwank, "Production and propagation of single-event transients in high-speed digital logic ICs," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 6, pp. 3278–3284, Dec. 2004, doi: 10.1109/TNS.2004.839172.
[20] M. Turowski, A. Raman, and G. Jablonski, "Mixed-Mode Simulation and Analysis of Digital Single Event Transients in Fast CMOSICs," in *2007 14th International Conference on Mixed Design of Integrated Circuits and Systems*, Jun. 2007, pp. 433–438. doi: 10.1109/MIXDES.2007.4286199.
[21] J. S. Kauppila *et al.*, "A Bias-Dependent Single-Event Compact Model Implemented Into BSIM4 and a 90 nm CMOS Process Design Kit," *IEEE Transactions on Nuclear Science*, vol. 56, no. 6, pp. 3152–3157, Dec. 2009, doi: 10.1109/TNS.2009.2033798.
[22] X. Gili, S. Barcelo, S. à A. Bota, and J. Segura, "Analytical Modeling of Single Event Transients Propagation in Combinational Logic Gates," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 4, pp. 971–979, Aug. 2012, doi: 10.1109/TNS.2012.2187071.
[23] P. Gregory, *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with Mathematica® Support*. Cambridge: Cambridge University Press, 2005. doi: 10.1017/CBO9780511791277.
[24] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 6, pp. 1153–1160, Dec. 1981, doi: 10.1109/TASSP.1981.1163711.
[25] "Technology," *X-FAB:* https://www.xfab.com/technology (accessed Feb. 27, 2021).
[26] Kwang-Ting Cheng, S. Dey, M. Rodgers, and K. Roy, "Test challenges for deep sub-micron technologies," in *Proceedings 37th Design Automation Conference*, Jun. 2000, pp. 142–149. doi: 10.1109/DAC.2000.855293.
[27] A. K. Stamper, T. L. McDevitt, and S. L. Luce, "Sub-0.25-micron interconnection scaling: damascene copper versus subtractive aluminum," in *IEEE/SEMI 1998 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop (Cat. No.98CH36168)*, Sep. 1998, pp. 337–346. doi: 10.1109/ASMC.1998.731585.
[28] M. L. Bushnell and V. D. Agrawal, Eds., "Fault Modeling," in *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, Boston, MA: Springer US, 2002, pp. 57–80. doi: 10.1007/0-306-47040-3_4.
[29] A. A. Abou-Auf, "Total-Dose Worst-Case Test Vectors for Leakage Current Failure Induced in Sequential Circuits of Cell-Based ASICs," *IEEE Transactions*

on Nuclear Science, vol. 56, no. 4, pp. 2189–2197, Aug. 2009, doi: 10.1109/TNS.2009.2019275.
[30] M. A. Breuer, "The Effects of Races, Delays, and Delay Faults on Test Generation," *IEEE Transactions on Computers*, vol. C–23, no. 10, pp. 1078–1092, Oct. 1974, doi: 10.1109/T-C.1974.223808.
[31] Y. Tokunaga and J. Frosien, "High performance electron beam tester for voltage measurement on unpassivated and passivated devices," in *Proceedings. "Meeting the Tests of Time"., International Test Conference*, Aug. 1989, pp. 917–922. doi: 10.1109/TEST.1989.82383.
[32] Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Comput.*, vol. C–35, no. 8, pp. 677–691, Aug. 1986, doi: 10.1109/TC.1986.1676819.
[33] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 5, pp. 567–580, Oct. 1967, doi: 10.1109/PGEC.1967.264743.
[34] A. K. Majhi and V. D. Agrawal, "Tutorial: Delay Fault Models and Coverage," p. 6.
[35] P. Agrawal, V. D. Agrawal, and S. C. Seth, "Generating tests for delay faults in nonscan circuits," *IEEE Des. Test. Comput.*, vol. 10, no. 1, pp. 20–28, Mar. 1993, doi: 10.1109/54.199801.
[36] F. Brglez and H. Fujiwara, *A neutral netlist of 10 combinational benchmark circuits and a targeted translator in FORTRAN*. 1985.
[37] "Verilog-AMS (Analog/Mixed-Signal)." https://www.accellera.org/downloads/standards/v-ams (accessed Apr. 16, 2021).
[38] T. J. Barnes, "SKILL: a CAD system extension language," in *27th ACM/IEEE Design Automation Conference*, Jun. 1990, pp. 266–271. doi: 10.1109/DAC.1990.114865.