University of Thessaly

Department of Computer & Communication Engineering

MASTER THESIS

SOFT ERROR RATE (SER) ANALYSIS OF DIGITAL INTEGRATED CIRCUITS

Paliaroutis Georgios – Ioannis
Tsoumanis Pelopidas

Supervisor:
George Stamoulis

July 2013
Volos, Greece

To our families and friends.

ACKNOWLEDGEMENTS

First of all, we would like to thank our families because all these years they give us courage and support us in every moment of our undergraduate/graduate studies in order to succeed our goals and dreams.

Of course we could not omit our thanks to Dr. George Stamoulis, our supervisor professor, as he has supported our work for at least three past years and advised us about a lot of issues arising from our project. We feel very lucky because we have cooperated with him and learned many things that dealt with Hardware and Computer Architecture area.

Finally, we would like to thank Dr. Nestoras Evmorfopoulos and Dr. Ioannis Moudanos, the other two supervisor professors, because they have helped us whenever we faced problems but also anybody that helped us to complete successfully this thesis.

Contents

Chapter 1 – Introduction 5
- 1.1 Digital circuits…… 6
- 1.2 Integrated circuits……7
- 1.3 Logic gates…… 8
- 1.4 Flip-flop…… 8
  - 1.4.1 D flip-flop…… 9

Chapter 2 – Soft error analysis 10
- 2.1 Silicon Atom……10
- 2.2 SEU (Single Event Upset) – SER (Soft Error Rate)……11
- 2.3 Causes of Soft Errors……12
- 2.4 Soft Error Rate on Circuits……12

Chapter 3 – Masking effects 16
- 3.1 Logical masking……16
  - 3.1.1 Modeling of Logical Masking……17
- 3.2 Electrical masking……17
  - 3.2.1 Modeling of Electrical Masking……18
- 3.3 Timing masking……19
  - 3.3.1 Modeling of Timing Masking……20
- 3.4 Overall circuit SER estimation……20

Chapter 4 – Implementation issues 22
- 4.1 Optimization issues……22
  - 4.1.1 Data structure…… 22
  - 4.1.2 Multiple simulations…… 23
- 4.2 Modeling issues…… 24
  - 4.2.1 Inertial delay of logic gates…… 24
    - 4.2.1.1 Logical effort…… 25
    - 4.2.1.2 Parasitic delay…… 27
  - 4.2.2 Multiple particle hits…… 28

Chapter 5 – Results and Conclusion 29
- 5.1 Results……29
- 5.2 Conclusion……30
- 5.3 Future challenges – Prospects……30

References 32

Chapter 1 - INTRODUCTION

A significant issue that had firstly arised in the 1970s and is referred to reliability of VLSI circuits is Single Event Upsets. A SEU is a change of state caused by many reasons such as cosmic rays creating high energy neutrons striking a sensitive node. Those SEUs that occur on microelectronic devices may affect their functionality and thus they could turn into Soft Errors. In this master thesis we represent a tool that estimates Soft Error Rate (SER) in combinational logic of various benchmark circuits.

For the purpose of SER analysis we use ISCAS'85 and ISCAS'89 benchmark circuits that describe netlists consisting of logic gates such as AND, OR, NOT, NAND, NOR, XOR, XNOR and memory elements like D Flip-flops. Our approach includes Monte-Carlo simulations to preserve an accurate estimation of SER that help us to understand the effects of SEUs on circuits. The basic part of our work is the modeling of the three masking phenomena in order to create a reliable tool. These are logical, electrical and timing or latching window masking and they determine whether a SEU will propagate to become a soft error. Finally, we obtain the latching probabilities of every node of the circuit due to a particle hit and thereafter the overall SER of the circuit.

The tool that we represent in this thesis has been implemented in C programming language. We choose this language because it provides a lot of advantages in contrast with others. First of all C is a very simple, general purpose and easy learning language but at the same time efficient and powerful and it is the best used for data structures and designing system software. Furthermore, the writing code runs quickly and the program is very " close to the hardware " which means that we can access low level facilities in our computer quite easily, without the compiler or run-time system stopping us from doing something potentially dangerous. Finally, portability is an important plus as we can compile and execute C programs in any operating system (unix, windows). The function libraries are standard for all versions of C so they can be used on all systems. We have developed our program in openSUSE UNIX environment but also in Windows platform using DEV-C/C++ compiler.

1.1 DIGITAL CIRCUITS

Digital electronics, or digital (electronic) circuits, represent signals by discrete bands of analog levels, rather than by a continuous range. When we refer to digital electronics we mean that they are electronic devices which process information in binary format (0 or 1). Computer systems process information in binary format (0 or 1). For this reason computing systems are using digital electronic circuits for processing the data. Main building blocks of digital electronics are electronic circuits called logic gates. Digital techniques are useful because it is easier to get an electronic device to switch into one of a number of known states than to accurately reproduce a continuous range of values. If we combine the basic logic gates, we can create a series of basic digital circuits, which can be used in order to manufacture more complex digital circuits like the one shown in Figure 1.1 below.

Digital circuits are divided, as we know, in two categories: combinational and sequential. In combinational circuits, the value of outputs depends on the value of inputs at that time. In sequential circuits, that we use in order to analyze Soft Error Rate (SER), the output depends on the value of input not only at that time, but the value had been in the earlier times.

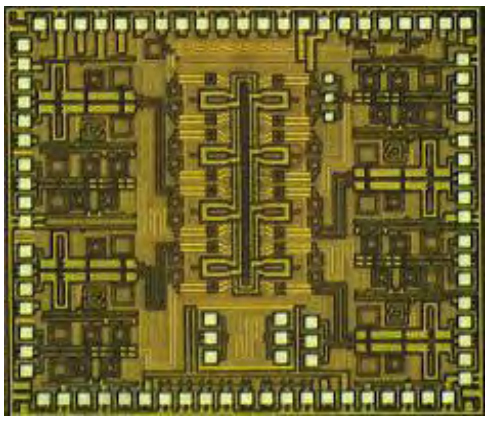


Figure 1.1 a VLSI circuit

1.2 INTEGRATED CIRCUITS

In the previous paragraph we mention that sequential circuits call circuits which contain elements of memory structure which store binary information as the output of these systems depends on the value of inputs not only that time, but the value had been in earlier times. These memory structures are flip-flop and lacthes. Every time the binary information which are stored in the memory elements is the situation (state). Additional values of input and status determine the value of outputs and the next state as. The Figure 1.2 describe us this situation. The sequential circuits are divided into two categories synchronous and asynchronous.

Soft errors induced in memory structures and latches have been extensively studied and there are empirical models covering their contribution to the overall SER of the circuit. Soft errors have important effect in circuits. For this reason many designers can choose to accept that soft errors will occur, and design systems with appropriate error detection and correction to recover gracefully. Typically, a semiconductor memory design might use forward error correction, incorporating redundant data into each word to create an error correcting code.

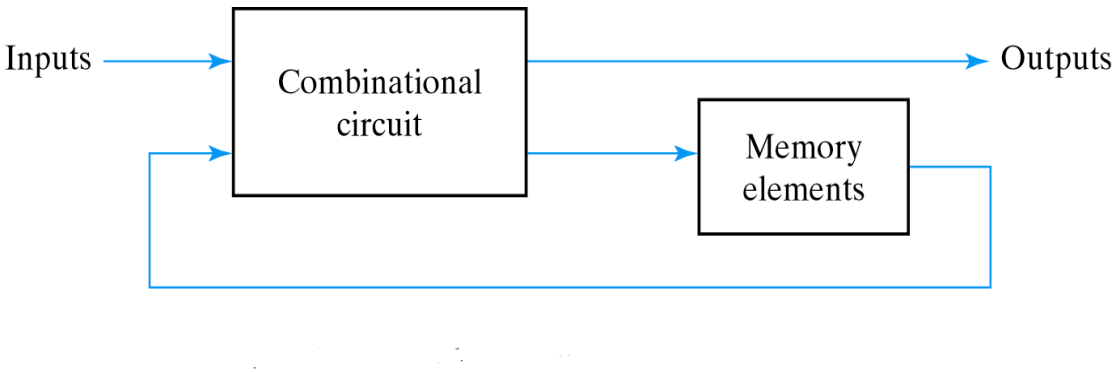


Figure 1.2 a sequential circuit

1.3 LOGIC GATES

Digital systems are said to be constructed by using logic gates. In order to analyse SER in our project we use circuits which include logic gates. These gates are the AND, OR, NOT, NAND, NOR, XOR and ENOR.

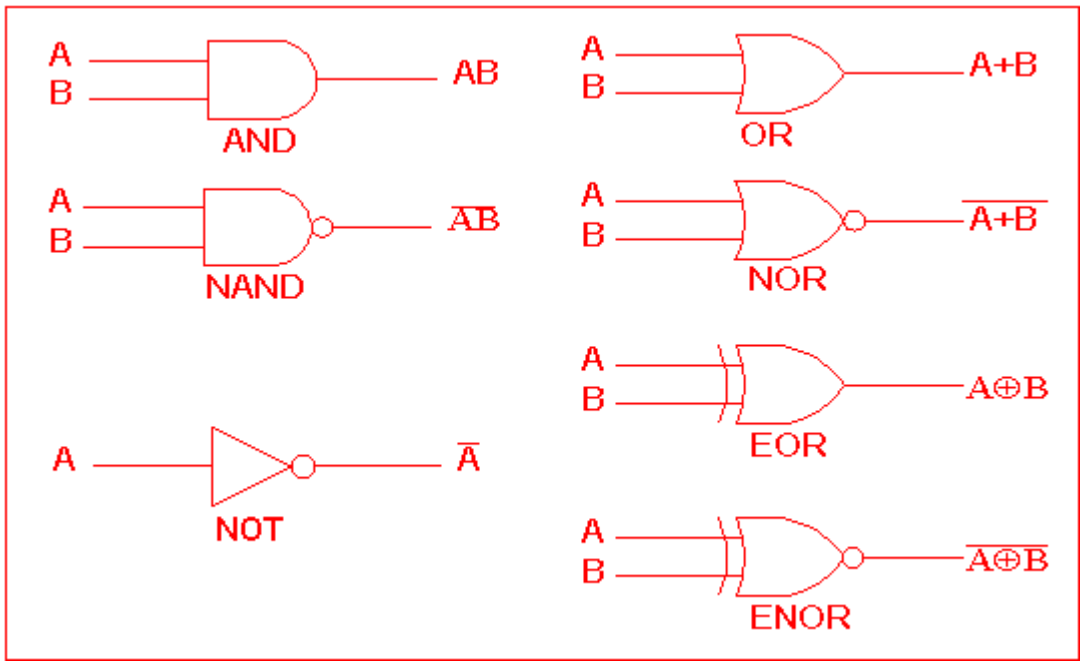


Figure 1.3 main logic gates

1.4 FLIP-FLOP

In electronics, a flip-flop or latch is a circuit that has two stable states and can be used to store state information. A flip-flop is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. It is the basic storage element in sequential logic. Flip-flops and latches are a fundamental building block of digital electronics systems used in computers, communications, and many other types of systems. Flip-flops and latches are used as data storage elements. Such data storage can be used for storage of *state, and such a circuit is described as sequential logic. When used in a finite-state machine,* the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronizing variably-timed input signals

to some reference timing signal. Flip-flops can be either simple (transparent or opaque) or clocked (synchronous or edge-triggered) the simple ones are commonly called latches. The word *latch* is mainly used for storage elements, while clocked devices are described as *flip-flops*. A latch is level-sensitive, whereas a flip-flop is edge -sensitive. That is, when a latch is enabled it becomes transparent, while a flip flop's output only changes on a single type (positive going or negative going) of clock edge.

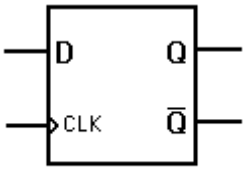
Flip-flops can be divided into common types: the **SR** ("set-reset"), **D** ("data" or "delay"), **T** ("toggle"), and **JK** types are the common ones. The behavior of a particular type can be described by what is termed the characteristic equation, which derives the "next" (i.e., after the next clock pulse) output, Qnext in terms of the input signal(s) and/or the current output, Q.

1.4.1 D FLIP-FLOP

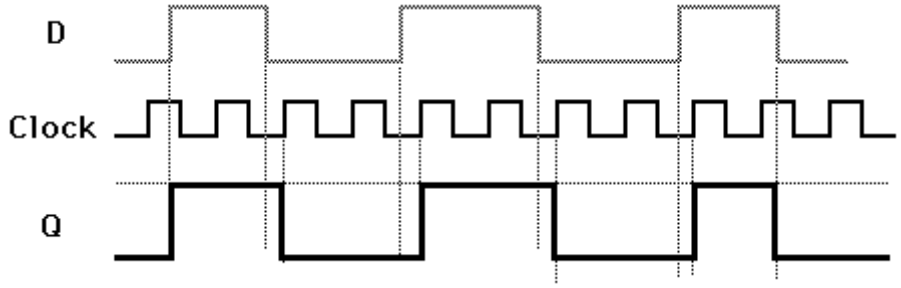
The D flip-flop is widely used. It is also known as a *data* or *delay* flip-flop. The D flip-flop captures the value of the D-input at a definite portion of the clock cycle (such as the rising edge of the clock). That captured value becomes the Q output. At other times, the output Q does not change. The D flip-flop can be viewed as a memory cell, a zero-order hold, or a delay line.

The D flip-flop tries to follow the input D but cannot make the required transitions unless it is enabled by the clock. Note that if the clock is low when a transition in D occurs, the tracking transition in Q occurs at the next upward transition of the clock.

There are two important parameters that characterize a flip-flop about the timing restrictions and we use in our tool. Setup time and hold time. Setup time is the minimum amount of time the data signal should be held steady before the clock event so that the data are reliably sampled by the clock. This applies to synchronous input signals to the flip-flop. Hold time is the minimum amount of time the data signal should be held steady after the clock event so that the data are reliably sampled. This applies to synchronous input signals to the flip-flop.



D-Type



Chapter 2 – Soft Error Analysis

We assume a standard sequential digital circuit that can be described by an array of flip-flops controlled by one clock and combinational logic driven by flip-flop outputs and driving flip-flop inputs. SEUs or soft errors are caused by concentrated bursts of excess charge generated at random locations in a semiconductor substrate and subsequently collected by the drain diodes of the MOS transistors. In this project we try to include the Soft Error Rate (SER) as a design parameter of our circuits.

2.1 Silicon Atom

If we want to create an integrated circuit this process base on a semiconductor material, which conducts more or less electrical current depending on external conditions. It is known that we can use also conduits and insulators if we would like to manufacture an integrated circuit. Insulators don't conduct electrical current unlike conduits have a large number of current carriers free electrons. However semiconductors are more useful because they have an intermediate number of current carriers. The main semiconductor material is silicon. We use silicon in order to create CPUS, memories and for the most chips. Silicon has same advantages that make it deal to use in semiconductor manufacturing. Obviously, it is semiconductor material. There is abundant and for this reason is cheap. The most important thing is that it can be deployed in single crystal. In Figure 2.1 we can see that each atom in the crystal of pure silicon has four valence electrons which are associated with an equal number of electrons from neighbouring atoms (in grid layout) by means of covalent bonds. Si is a symbol of silicon.

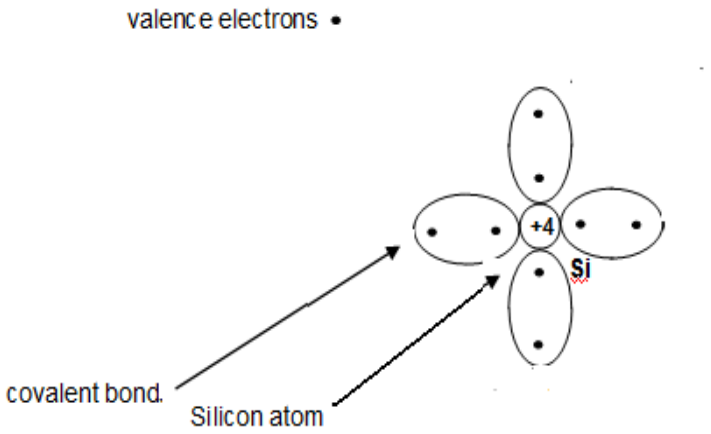


Figure 2.1 a silicon atom

Here we will describe how silicon conducts electricity. At low temperature, all the covalent bonds are intact and the material exhibits no electrical conductivity. At room temperature, several of the covalent bonds are broken and show two different entities of the stream, the free electrons and holes. The movement of a hole is generated by the reverse movement of valence electrons. In silicon crystal concentrations of free electrons and holes are always equal to each other.

2.2 SEU (Single Event Upset) – SER (Soft Error Rate)

Electronics circuits encode information in the form of charge stored on a circuit node or as a current flowing between two circuit nodes. So, one node bit of static memory contains two values that complementary charge respectively to logic “1” and logic “0”. So the value of nodes can be “1” or “0” zero. A single event upset (SEU) is a change of state caused by ions or electro-magnetic radiation striking a sensitive node in a micro-electronic device, such as in a microprocessor, semiconductor memory or tansistors. The state change is a result of the free charge created by ionization in or close to an important node of a logic element. The error in device output or operation caused as a result of the strike is called an SEU or Soft Error. In electronic and computing, a soft error is an error in a signal or datum that is wrong. In a computer's memory system, a soft error changes an instruction in a program or a data value. There are two types of soft errors. The first type is chip-level soft error. These errors occur when the radioactive atoms in the chip's material decay and release alpha particles into the chip. The second type is a system-level soft error. These errors occur when the data being

processed is hit with a noise phenomenon, typically when the data is on a data bus. The computer tries to interpret the noise as a data bit, which can cause errors in addressing or processing program code. The bad data bit can even be saved in memory and cause problems at a later time.

2.3 Causes of Soft Errors

Our project focuses on Soft Errors. These are important pieces of circuits because they affect the operation of the circuit. These are errors in processor execution that are due to electrical noise or external radiation rather than design or manufacturing defects. Moreover soft errors caused by high energy neutrons resulting from cosmic rays colliding with particles in the atmosphere. The existence of cosmic ray radiation has been known for over 50 years, and the capacity for this radiation to create transient faults in semiconductor circuits has been studied since the early 1980s. As a result, most modern microprocessors already incorporate mechanisms for detecting soft errors.

Cosmic ray flux depends on altitude. The average rate of cosmic-ray soft errors is inversely proportional to sunspot activity. Energetic neutrons produced by cosmic rays may lose most of their kinetic energy and reach thermal equilibrium with their surroundings as they are scattered by materials. Thermal neutrons are also produced by environmental radiation sources such as the decay of naturally occurring uranium or thorium. The thermal neutron flux from sources other than cosmic-ray showers may still be noticeable in an underground location and an important contributor to soft errors for some circuits.

2.4 Soft Error Rate on Circuits

Here we present how a high-energy particle hit can interact with silicon atom and their effect on circuit. This phenomenon may be modeled by a time varying double-exponential current pulse.

$$I(t) = I_0(e^{-t/a} - e^{-t/b})$$

We can say that a is the collection time constant of the junction and b is the time constant for initially establishing the ion track. Different particle energy levels will produce a different number of electron-hole pairs which will be reflected into Io. The circuit node that is affected can be either a gate output or an internal node of a gate. In both cases the particle hit appears as voltage pulse at the gate output. Since the particle energy level can be

described as a random variable, the width of the output voltage pulse can also be described as a random variable with a probability density function (pdf) that can be approximated by Monte-Carlo electrical simulations of the charge injection circuits.

There are four possible cases of charge injection scenarios for circuits as shown in Figure 2.2 where the resistors represent conducting transistors, the rectangles represent the drain regions of transistors and the direction of arrows inside the current sources correspond to the direction of the current flow. For cases I and II, the voltage at the p+ node will go up, and for cases III and IV, the voltage at the n+ node will go down. Cases II and IV will not affect the logic state of the circuit because the node is already at the logic value toward which the injected charge will drive the node. However cases I and III may affect the logic value of the node. This situation could affect an output value.

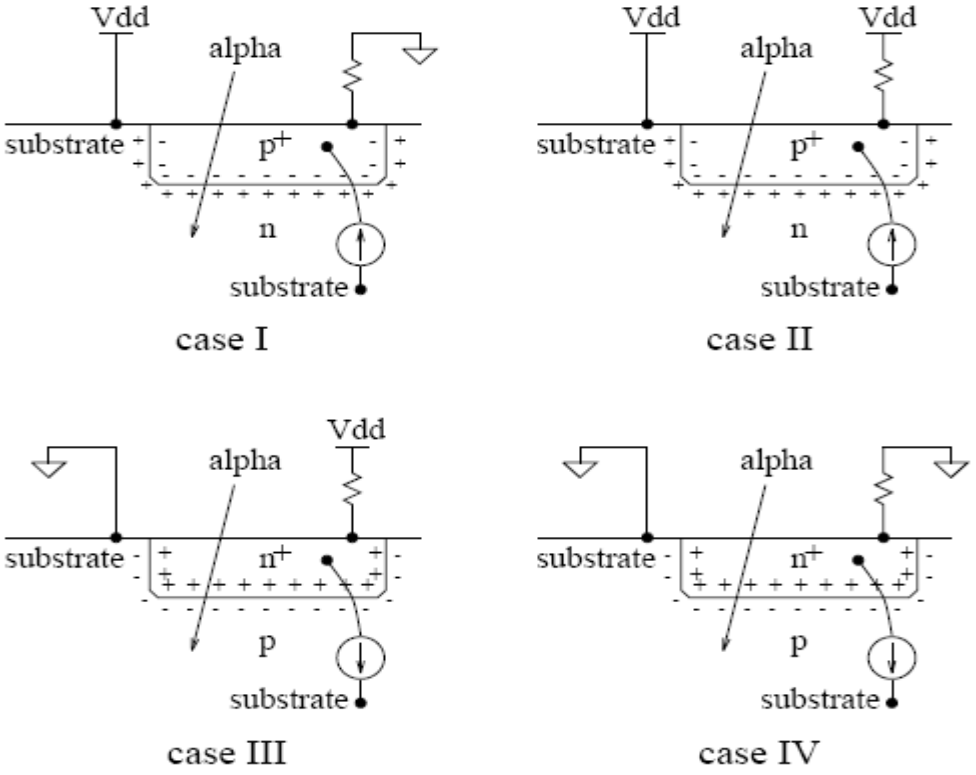


Figure 2.2 charge injection scenarios

The positively charged alpha particle travels through the semiconductor and disturbs the distribution of electrons there. If the disturbance is large enough, a digital signal can change from a 0 to a 1 or vice versa. In combinational logic, this effect is transient, perhaps lasting a fraction of a nanosecond, and this has led to the challenge of soft errors in combinational logic mostly going unnoticed. In sequential logic such as latches and RAM, even this transient upset can become stored for an indefinite time, to be read out later.

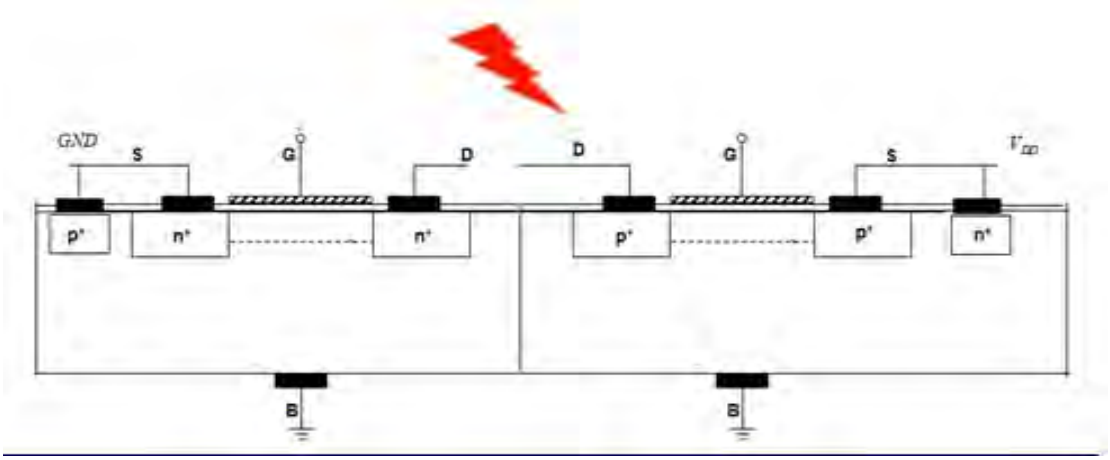


Figure 2.3 a particle hit

In order to describe a particle hit we initially suppose that we have a not gate and p+ drain at 0 and a n+ drain 1 as shown in Figure 2.3. A particle hit occur. This situation generates a number of electron-hole pair. So the behaviour of gates begins to change.

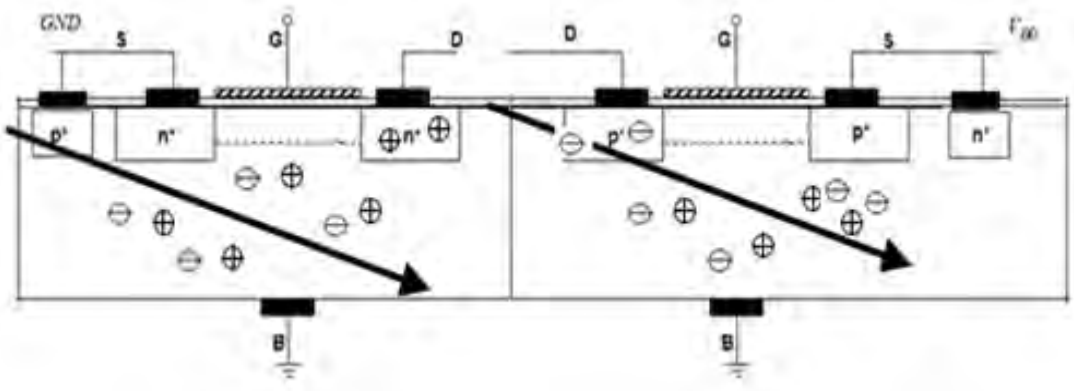


Figure 2.4 electron-hole pairs

In Figure 2.5 we can see that electrons go towards VDD (supply voltage) and get observed by the VDD. Holes have the opposite movement because they go towards GND (ground) and get observed by GND. In additional, we observe that the drain of p+ and n+ have a different load than the previous Figure 2.4 as holes accumulate into p+ drain and electrons accumulate into n+ drain. Finally we can conclude that a digital signal of gate change because the p+ drain is now at logic 1 and n+ drain is at logic 0.

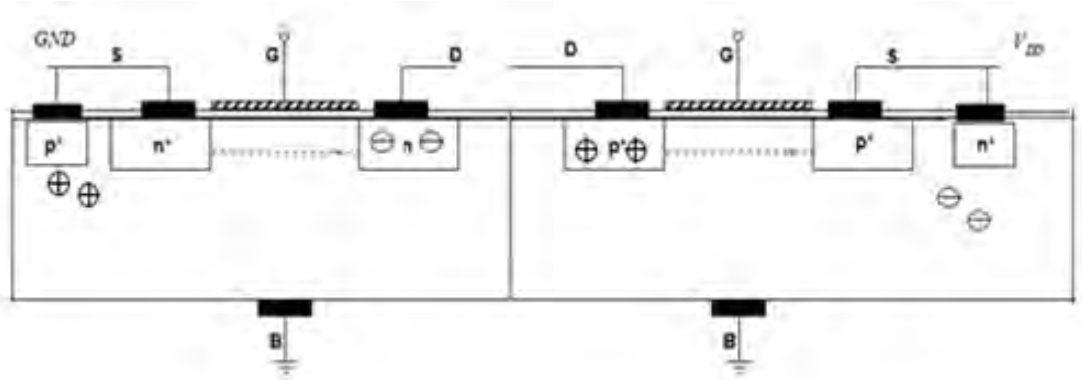


Figure 2.5 final state of gate

Chapter 3 – SER IN COMBINATIONAL LOGIC

Soft error rate analysis in combinational logic is much more complicated unlike in memory circuits that has been studied extensively and has models covering the contribution of soft errors to the overall SER of the circuit.

SEUs create a voltage transient on any node of the circuit. This transient may propagate through the circuit until the inputs of sequential elements (e.g. flip-flops) and at last be latched by one or more of them. However, many transient errors will not be captured in a memory circuit because they could be masked by one of the following phenomena. There are three phenomena that provide to circuits a kind of natural resistance to SEUs. The most important part of our thesis is the incorporation to our program of these masking effects, in order to succeed an accurate SER analysis of digital integrated circuits. In this chapter we will explain what exactly are the masking phenomena, how they are modeled and also in which way they have been imported to our tool.

The three natural masking effects in combinational logic that determine whether a SEU will propagate to become a soft error or not are logical, electrical and timing or temporal or latching window masking.

3.1 LOGICAL MASKING

Logical masking occurs when a SEU on a particular node of the combinational logic is blocked from reaching the inputs of latches due to a subsequent gate whose result is completely determined by its other input values. For instance, if an error occurs on “NAND2” gate due to a particle hit, as shown in Figure 3.1, the logic state of the output node will be changed from logical 0 to logical 1. However, this node drives two NAND-2 gates whose the other input’s states are logical 0. Consequently, the SEU will be completely masked at the outputs that we show with arrows in the figure. That SEU essentially will be eliminated and will not be latched in flip-flops that follow the combinational part of circuit causing a soft error. A particle hit is masked respectively through the other types of logic gates.

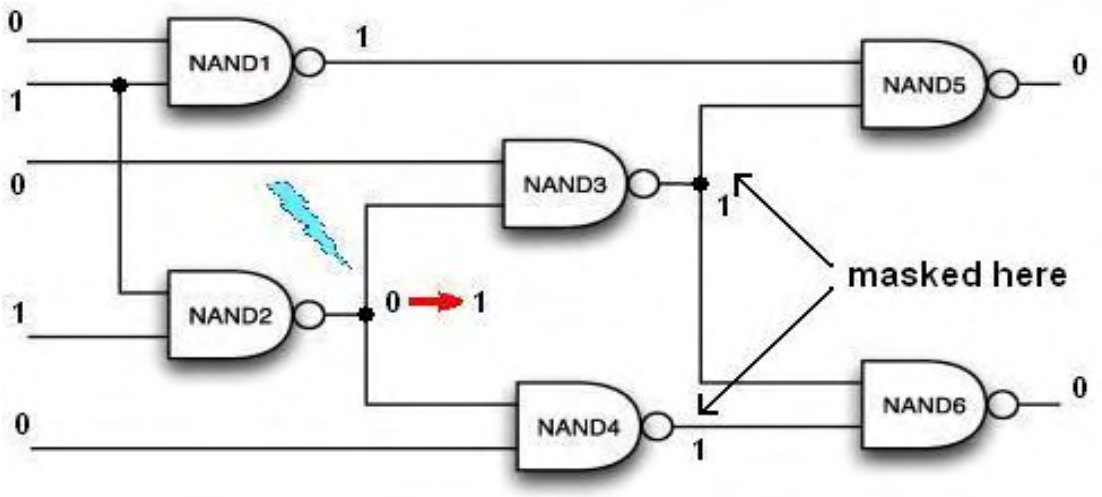


Figure 3.1 logical masking

3.1.1 Modeling of Logical Masking

In order to model logical masking we used a zero-delay fault simulator in logic level. This simulator takes as input a vector of logic states for the primary inputs of circuit and forwards the vector, through the circuit, to the inputs of latches. Afterwards, it considers also the SEU that caused a logic state switch on a particular node. At the end of each simulation we see if the inputs of flip-flops finally have changed logic state and we proceed to the next step which is the examination of the other masking phenomena and the mixing with logical masking. Finally, we have created 10.000 input vectors for these Monte-Carlo simulations, which is a magnitude that obtains fairly good accuracy to our SER analysis. As Monte-Carlo simulation in large circuits, like those we use in our program, is a time expensive procedure we can decrease the number of input vectors for a faster simulation. Nevertheless, simulation is less accurate than before.

3.2 ELECTRICAL MASKING

Electrical masking is another one factor that prevents a fault from reaching flip-flop inputs and causing a soft error. A SEU is electrically masked if the pulse resulting from a particle hit is attenuated due to the electrical properties of gates on its propagation path such that the resulting pulse is of insufficient magnitude to be reliably latched. As illustrated in Figure 3.2, the pulse that generated at the output of the first gate has been eliminated as it passed through a number of gates.

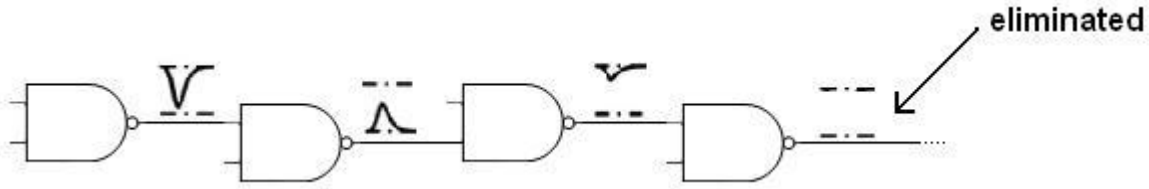


Figure 3.2 electrical masking

3.2.1 Modeling of Electrical Masking

As we already mentioned, when a high energy particle hits a transistor it generates a voltage pulse at the gate output. This pulse can be described as a random variable with a particular probability density function (pdf). In the context of our work we consider that the width of output voltage pulse W follows uniform distribution which is discretized so as the propagation through the circuit will be feasible. Thus, as illustrated in Figure 3.3 the widths of pulses are discretized in N levels while the largest quantization value can be equal to the period of the clock, since for pulse widths larger than that, timing masking has no effect.

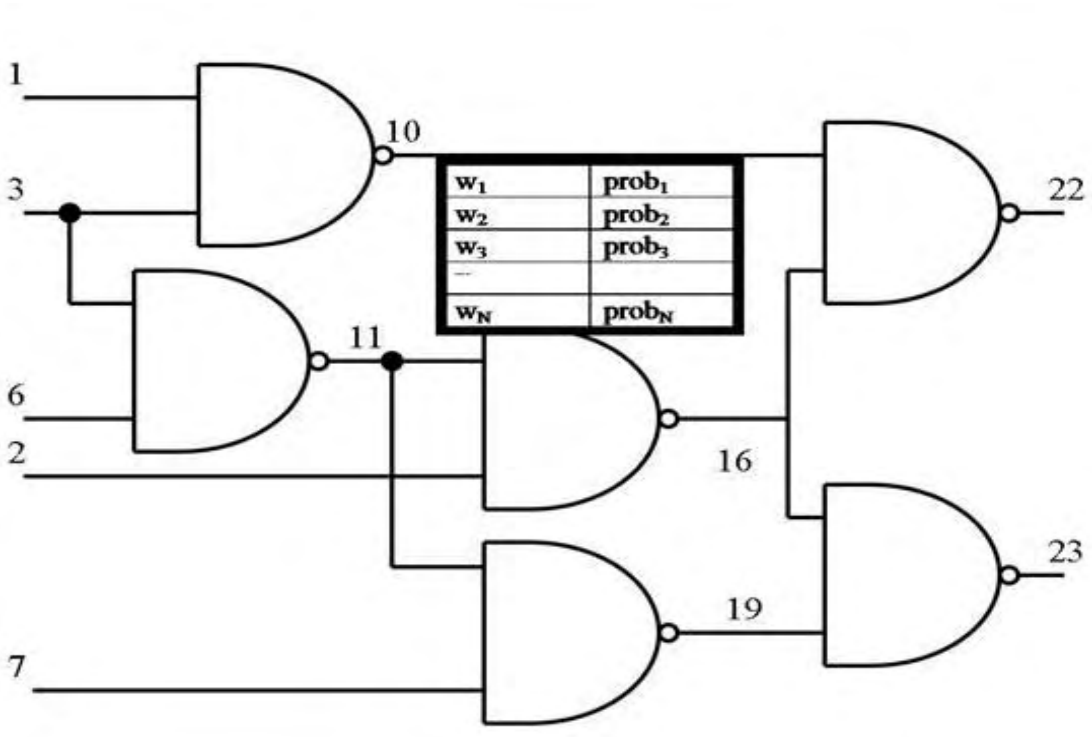


Figure 3.3 modeling of electrical masking

For the propagation of the pdf until the inputs of flip-flops we used the modeling of ASERTA and CARROT according to which, when a particle strikes a node of an intergrated circuit the duration of the generated pulse is dependent on the delay of the gate that is driving the node. A slow gate will attenuate a pulse at its output more compared with a fast gate. Consequently, slow gates have better pulse attenuation characteristics. This is reflected at the following linear function of the output pulse duration of a gate:

$$w_{out} = \begin{cases} 0 & , \quad w_{in} < d \\ 2(w_{in} - d) & , \quad d < w_{in} < 2d \\ w_{in} & , \quad w_{in} > 2d \end{cases}$$

where w_{in} is the input pulse duration and d is the gate's propagation delay.

This function is applied through the circuit but only to the gates that are affected from the particle hit as well as to all the entries of the discretized pdf of the pulse width. A significant point for the reliable simulation is the case in which more than one pdf are inputs of a specific gate. Then we choose the pdf with the greatest pulse width as shown:

$$w_{in} = max\,(w_{in\,1}, w_{in\,2}, \ldots w_{in\,k})$$

3.3 TIMING MASKING

As the pulse resulting from a particle hit propagates through the circuit and reaches a latch may be outside of the latching window, where the memory element capture its input value. Thus, this SEU will not be latched and therefore it will not be changed into a soft error. That is the timing or latching window masking as it is shown in Figure 3.4.

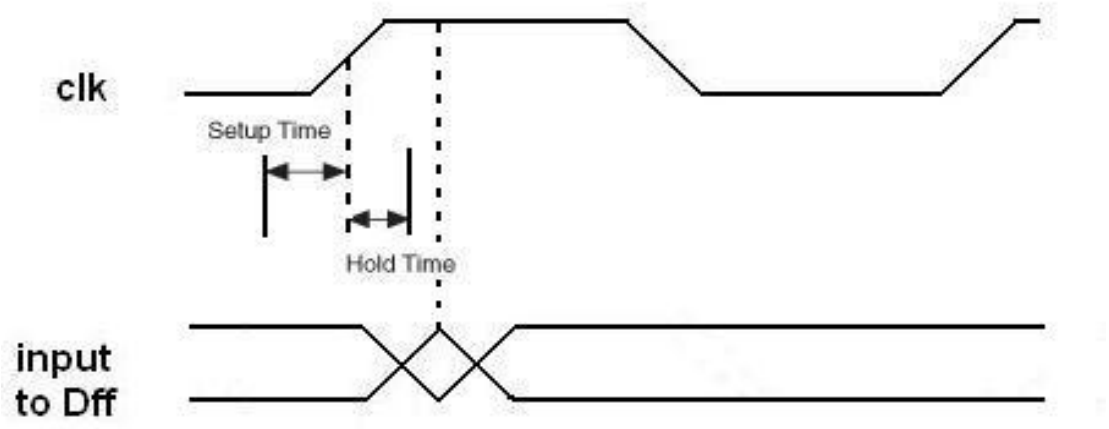


Figure 3.4 timing masking

3.3.1 Modeling of Timing Masking

Together with the analysis and modeling of the other masking effects, that presented before we introduce a simple timing analysis for our circuit. Thus, we compute the minimum and maximum arrival times of the erroneous pulse at each node from the charge injection node, where both times are zero, till the inputs of memory elements. The function *double timing masking(struct node * a)* returns the times at the output of each gate as follows:

$$min\_output\_time = min\ (all\ min\ times\ of\ each\ input) + d$$
$$max\_output\_time = max\ (all\ max\ times\ of\ each\ input) + d$$

where *d* is the inertial delay of the corresponding gate which was calculated on the fly and it will be described later.

3.4 OVERALL CIRCUIT SER ESTIMATION

The above analysis is repeatedly applied to the fanouts of gates being activated from the particle hit at a specific gate until we reach the flip-flop inputs. At the end of this procedure we can calculate the latching probabilities and overall circuit SER considering the results that have been originated from the simulation of the three masking effects over the circuit. These are the logic vectors, the pdf of pulse width and the minimum/maximum output time at the inputs of every flip-flop.

To calculate timing latching probabilities for each pdf entry of each latch we check from logical masking, for each simulation, which inputs to flip-

flops are at an erroneous state. From these we select the ones whose pulse width entries of pdf are not zero, which means that pulse has not been eliminated because of electrical masking and we find a *minimum* and a *maximum time window* as shown below:

$$min\_time = min\ (of\ the\ selected\ nodes)$$
$$max\_time = max\ (of\ the\ selected\ nodes)$$

Thus, the latching probability taking into account *min_time* and *max_time* is:

$$p\_latching = \begin{cases} 1 & , \quad t\_latch > T \\ t\_latch / T & , \quad t\_latch < T \end{cases}$$

where $t\_latch = max\_time - min\_time + wi + setup\_time + hold\_time$, *setup_time* and *hold_time* are the known timing parameters of flip-flops and T is the clock period of the flip-flop.

So, the total latching probability is the sum of the above probabilities for all Monte-Carlo experiments over the number of experiments. The overall SER due to an upset at a specific node is the sum of the product of the pulse width probabilities by their respective total latching probabilities. Consequently, the overall SER of the circuit is the sum of the products of the SER for each node by the SEU probability calculated by the particle flux.

Summing up, if all the three masking effects fail to occur, the propagated pulse becomes latched and the output of the logic circuit will be an erroneous value. In the context of circuit operation, this erroneous output value may be considered a soft error event. However, from a microarchitectural-level standpoint, the affected result may not change the output of the currently-executing program. For instance, the erroneous data could be overwritten before use, masked in subsequent logic operations, or simply never be used. If erroneous data does not affect the output of a program, it is considered to be an example of microarchitectural masking.

Chapter 4 - IMPLEMENTATION ISSUES

In this chapter we are going to analyze some important implementation issues that have been resulted from several parts of our thesis. We can categorize them according to their nature. One category is the one with the code optimization issues, so that the program can succeed better execution time. The other one is the modeling issues that concerning SER analysis of ISCAS'85 - ISCAS'89 benchmark circuits.

4.1 OPTIMIZATION ISSUES

4.1.1 Data Structure

This SER Analysis tool takes as input a gate level circuit that consists of thousands of nodes and gates which means that we should use an appropriate data structure to store the whole information. This data structure has to be accessed and manipulated as fast as possible. Our first approach was to create a simple linked list which had as elements the nodes of the circuit. Unfortunately, this implementation was really expensive because every time a new gate and therefore some nodes (inputs and output) were read, a new linear search had to take place over the list in order to create the circuit in right way and to avoid double entries.

Thus, we used a more complicated data structure called hash table. This structure consists of an associative array of particular size which in every slot contains a pointer to *struct node* that we have implemented. So, every bucket has a simple linked list of nodes as shown in Figure 4.1. Nodes are allocated to the appropriate bucket with the help of a well defined hash function that takes advantage of the whole table. This hush function is applied to every node and creates an index - from node name - which determines the position into the array.

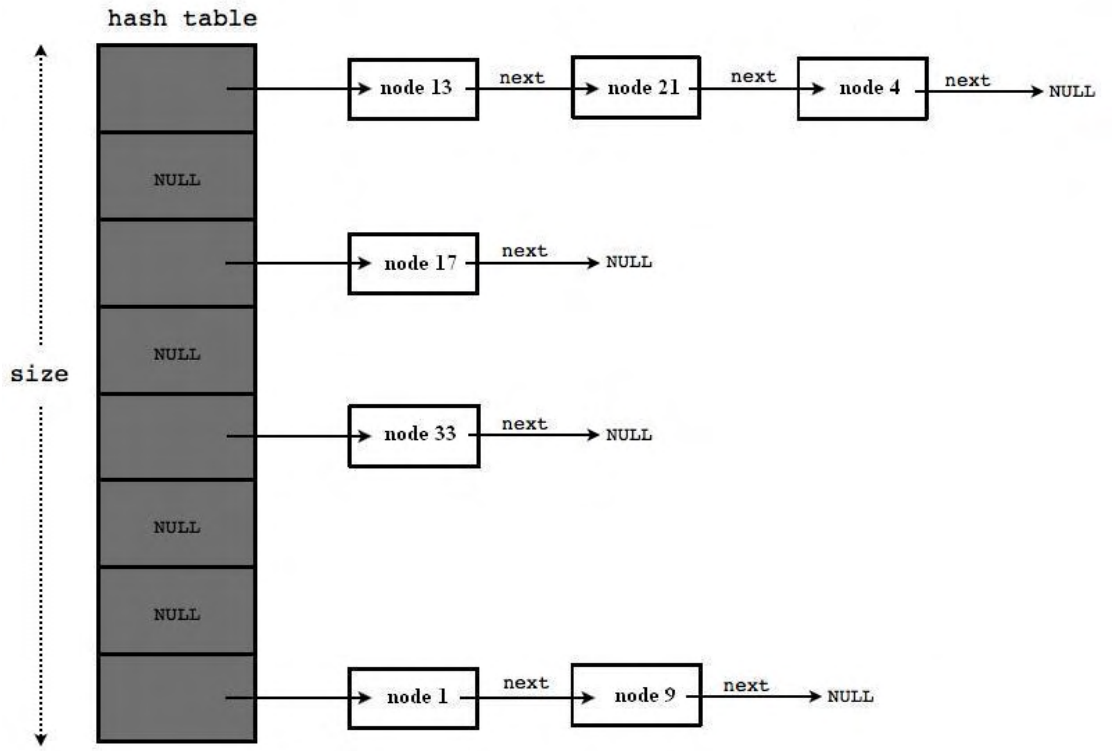


Figure 4.1 hash table data structure

Our hash function gets round allocating the nodes in a uniform way through the table and with the suitable size of table gives an extreme speed up compared with the previous implementation. So, complexity from O (n) becomes O(1 + n/100) where 100 is the number of buckets.

4.1.2 Multiple simulations

Our program spends the most of the time executing Monte-Carlo simulations which is a very time consuming operation considering the size of the circuits. For this reason we use unsigned integer instead of integer as inputs to the circuit for logic simulations. In this way we take advantage of each bit of 32-bit unsigned integer and we succeed speed up of 32 times. The operations that take place in the program are done with bitwise operations like the XOR operation at the last step of logical masking to determine which of 32 bits were masked logically at the inputs of flip-flops.

4.2 MODELING ISSUES

4.2.1 Inertial delay of logic gates

In our project it is necessary to find the delay for each gate. We use delay for electrical and timing masking in order to estimate the output pulse and timing at the output of each gate. We estimate delay using logical effort method. In next paragraphs we excuse this method. It is useful method and has many advantages.

In general the propagation delay of a gate can be written as:

- d = f + p

P is the parasitic delay inherent to the gate when no load is attached f is the effort delay or stage effort that depends on the complexity and fanout of the gate.

- f = g * h

The complexity is represented by the logical effort, g. An inverter is defined to have a logical effort of 1. More complex gates have greater logical efforts, indicating that they take longer to drive a given fanout. For example, the logical effort of the NAND gate is 4/3. A gate driving h identical copies of itself is said to have a fanout or electrical effort of h. If the load is not identical copies of the gate, the electrical effort can be computed as:

- h = Cout / Cin

where Cout is the capacitance of the external load being driven and Cin is the input capacitance of the gate.

In Figure 4.2 we can see normalized delay and electrical effort for an idealized inverter and 2-input NAND gate. The y-intercepts indicate the parasitic delay, i.e., the delay when the gate drives no load. The slope of the lines is the logical effort. The inverter has a slope of 1 by definition. The NAND gate has a slope of 4/3.

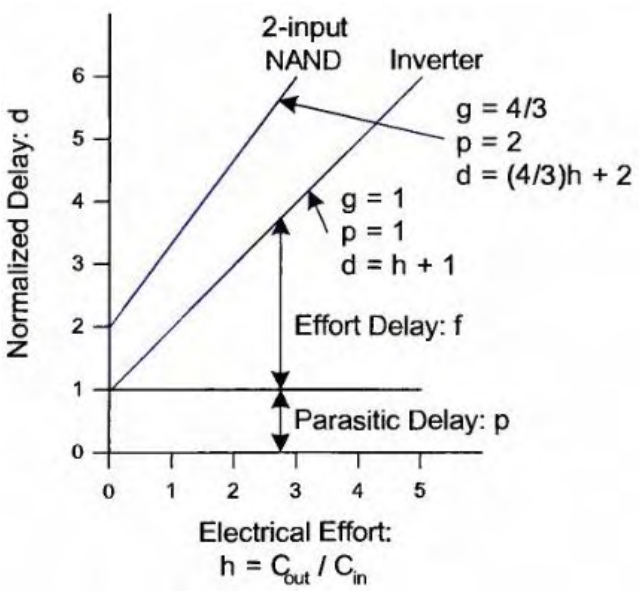


Figure 4.2 normalized delay vs electrical effort

Note that separate rising and falling delays are computed. These parameters are related to the logical effort terms as given in Table 4.1. The effective resistance of a gate increases with the logical effort of the gate but decreases with the gate size.

Logical Effort Term	Synopsys Term
d	delay
p	intrinsic
C_{out}	capacitance
g/C_{in}	resistance

Table 4.1 logical effort terms

4.2.1.1 Logical effort

Logical effort of a gate is defined as the ratio of the input capacitance of the gate to the input capacitance of an inverter that can deliver the same output current. Equivalently, logical effort indicates how much worse a gate is at producing output current as compared to an inverter, given that each input of the gate may only present as much input capacitance as the inverter. Logical effort can be measured in simulation from delay and fanout plots as

the ratio of the slope of the delay of the gate to the slope of the delay of an inverter. Alternatively, it can be estimated by sketching gates. The next figure shows inverter, NAND, and NOR gates with transistor widths chosen to achieve unit resistance, assuming pMOS transistors have twice the resistance of nMOS transistors. The inverter presents 3 units of input capacitance. The NAND presents 4 units of capacitance on each input, so the logical effort is 4/3. Similarly, the NOR presents 5 units of capacitance, so the logical effort is 5/3. This matches our expectation that NANDs are better than NORs because NORs have slow pMOS transistors in series.

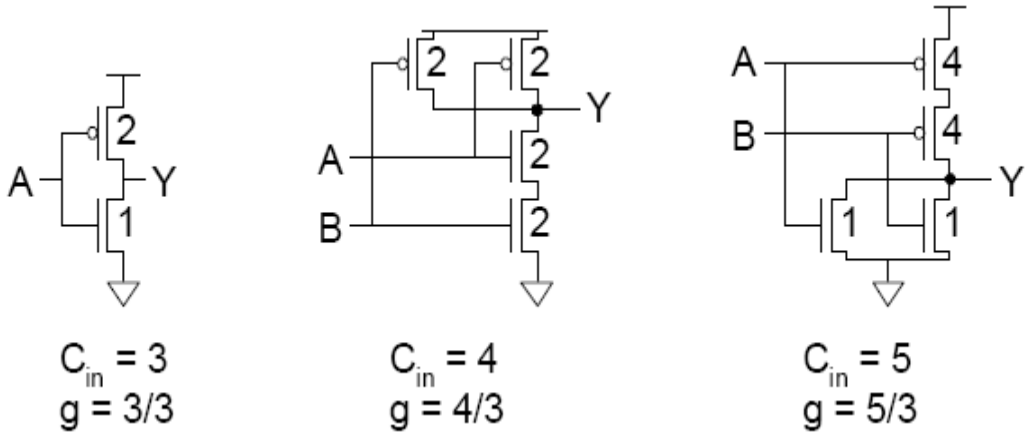


Figure 4.3 NOT, NAND and NOR gates

The Table 4.2 lists the logical effort of common gates. The effort tends to increase with the number of inputs. NAND gates are better than NOR gates because the series transistors are nMOS rather than pMOS. Exclusive-OR gates are particularly costly and have different logical efforts for different inputs. An interesting case is that multiplexers built from ganged tristates, have a logical effort of 2 independent of the number of inputs. This might at first seem to imply that very large multiplexers are just as fast as small ones. However, the parasitic delay does increase with multiplexer size; hence, it is generally fastest to construct large multiplexers out of trees of 4-input multiplexers.

Logical effort of common gates					
Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		4/3	5/3	6/3	$(n+2)/3$
NOR		5/3	7/3	9/3	$(2n+1)/3$
tristate, multiplexer	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

Table 4.2 logical effort of common gates

4.2.1.2 Parasitic delay

The parasitic delay of a gate is the delay of the gate when it drives zero load. It can be estimated with RC delay models. A crude method good for hand calculations is to count only diffusion capacitance on the output node. Transistor widths were chosen to give a resistance of R in each gate. The inverter has 3 units of diffusion capacitance on the output, so the parasitic delay is 3RC = x. In other words, the normalized parasitic delay is 1. In general, we will call the normalized parasitic delay pinv. Pinv is the ratio of diffusion capacitance to gate capacitance in a particular process. It is usually close to 1 and will be considered to be 1 on many examples for simplicity. The NAND and NOR each have 6 units of diffusion capacitance on the output, so the parasitic delay is twice as great (2pinv, or simply 2). The Table 4.3 estimates the parasitic delay of common gates. Increasing transistor sizes reduces resistance but increases capacitance correspondingly, so parasitic delay is, to first order, independent of gate size. However, wider transistors can be folded and often see less than linear increases in internal wiring parasitic capacitance, so in practice, larger gates tend to have slightly lower parasitic delay.

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n
Tristate / mux	2	4	6	8	2n
XOR, XNOR		4	6	8	

Table 4.3 parasitic delay of common gates

Here we ought to refer that we regret that the w (width) for each gate is 8. So, the input capacitance of the NOT gate is cin_inv = 2 * w and the type we use for logical effort for each gate is g = (4 * inputs + w) / 2 * w. We decided to use this type because the w has the same value for pmos and nmos transistor for each gate.

4.2.2 Multiple particle hits

As we have already discussed, soft errors are caused by alpha particle hits or cosmic rays creating energetic neutrons and protons. These kind of hits occur on a particular area of circuits. Consequently, they affect a neighbour of a circuit and thus, a number of neighboring nodes.

Our earliest approach selects randomly a number of nodes, along the circuit, that SEUs occur and continues with the simulations as described before. A better approach will be proposed in the next chapter.

Chapter 5 - Results and Conclusion

5.1 Results

In Table 5.1 that follows we represent the results of the execution of ISCAS'85 and ISCAS'89 benchmark circuits with the tool that we have implemented. Also, it demonstrates details about every circuit like the number of gates, nodes or flip-flops and overall circuit SER. Finally, as illustrated in Figure 5.1 we can see how execution time rises as we simulate more complicated and bigger circuits. The system on which these benchmarks have been executed is an Intel Pentium G645 2.9 GHz with 2 GB of RAM.

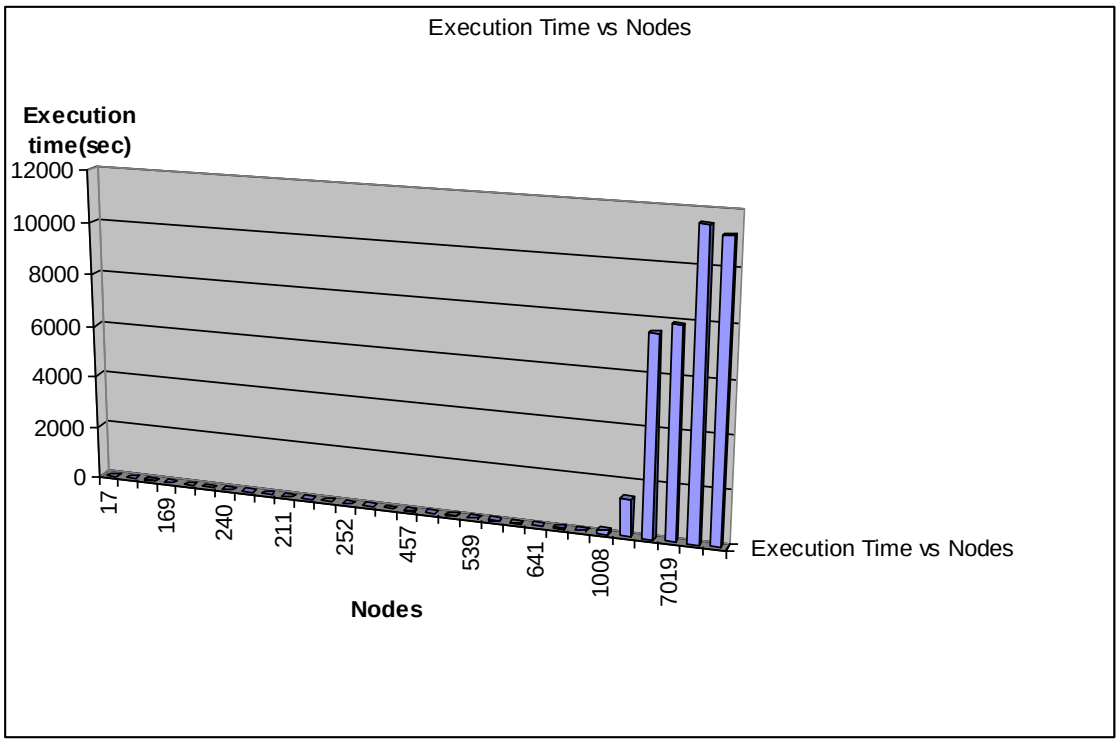


Figure 5.1 execution time vs number of nodes

5.2 Conclusion

In this master thesis we have dealt with Soft Error Rate Analysis in Integrated Circuits. The tool that has been implemented ensures in a big way fast and accurate estimation of SER which is very useful for engineers - especially aeronautical - to come up against cosmic rays or alpha particles that cause soft errors. This estimation comes out of a deliberate process that combines the three masking effects constituting the basic parts of our approach.

5.3 Future Challenges - Prospects

This basic work has many improving potentials that regarding implementation issues which are going to speed up the program's execution but also modeling issues which can make this analysis much more accurate. For instance, a new version of program could parsing DEF/LEF files in order to model the particle hit in a particular area of the circuit that affects neighboring nodes.

Our program can be the beginning of a thoroughly implemented tool that takes other parameters on board such as timing, thermal, power or leakage, which are very important in VLSI circuit design.

Circuit name	Nodes	Inputs	Gates	DFFs	Execution Time	Overall Circuit SER
S27	17	4	13	3	0.07 sec	0.01656584
S208_1	125	10	115	8	1.86 sec	0.04103197
S208	149	10	139	8	1.86 sec	0.04103197
S298	169	3	166	14	3.97sec	0.05298491
S386	284	7	277	6	4.12sec	0.01581466
S382	196	3	193	21	6.92sec	0.14375757
S344	240	9	231	15	6.74sec	0.08889600
S349	224	9	215	15	6.82sec	0.10234129
S400	203	3	200	21	7.23sec	0.14025177
S444	211	3	208	21	8.82sec	0.15434552
S526	280	3	277	21	9.23sec	0.09036216
S526N	280	3	277	21	9.43sec	0.09320419
S420	252	19	233	16	6.23sec	0.11947320
S510	293	19	274	6	9.45sec	0.06999071
S420_1	313	18	295	16	7.12sec	0.11487777
S832	457	28	429	5	16.5sec	0.01760506
S820	443	18	425	5	16.2sec	0.01744302
S641	517	35	482	19	25.4sec	0.17323527
S713	539	35	504	19	28.4sec	0.19093299
S953	496	16	480	29	41.4sec	0.56845114
S838_1	641	34	607	32	29.4sec	0.18963497
S838	641	34	607	32	28.9sec	0.18963497
S1238	768	14	754	18	42.7sec	0.11744645
S1196	762	14	748	18	44.5sec	0.09415392
S1423	1008	17	991	74	1.92min	0.44978881
S5378	3053	35	3018	179	23.6min	0.72483019
S9234	7002	19	6983	228	2.1hours	0.45789101
S9234_1	7019	36	6983	211	2.2hours	0.34572302
S13207	9608	31	9577	669	3.2hours	0.45896532
S13207_1	9609	32	9577	638	3.1hours	0.63582431

Table 5.1 execution results

REFERENCES

[1] http://en.wikipedia.org/wiki/Soft_error

[2] http://en.wikipedia.org/wiki/Single_event_upset

[3] http://en.wikipedia.org/wiki/Flip-flop_%28electronics%29

[4] D. Bountas, G.I. Stamoulis : CARROT – A Tool for Fast and Accurate Soft Error Rate Estimation, Embedded Computer Systems: Architectures, Modeling, and Simulation Lecture Notes in Computer Science, vol. 4017, pp 331-338, 2006

[5] Dhillon , Y.S., Diril, A.U., Chatterjee, A.: Soft-error tolerance analysis and optimization of nanometer circuits, in Proceedings of Design, Automation, and Test in Europe (2005) 288–293.

[6] CMOS VLSI Design: A Circuits and Systems Perspective, 4/E, Neil Weste, David Harris Publisher: Addison-Wesley Copyright: 2011

[7] 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, June 2003

[8] T. Karnik, P. Hazucha and J. Patel: Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes, IEEE Transactions on dependable and secure computing, Vol. 1, No. 2, April-June 2004

[9] Shivakumar P., Kistler M., Keckler S.W., Burger D., Alvisi L. : Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic, Dependable Systems and Networks, 2002. DSN 2002.Proceedings. International Conference on, vol., no., pp.389, 398, 2002

[10] Hungse Cha, Patel J.H.: A logic-level model for α-particle hits in CMOS circuits," Computer Design: VLSI in Computers and Processors, 1993. ICCD '93.Proceedings., 1993 IEEE International Conference on, vol., no., pp.538,542, 3-6 Oct 1993

[11] Paul E. Dodd, Marty R. Shaneyfelt, James A. Felix James R. Schwank: Production and Propagation of Single-Event Transients in High-Speed Digital Logic ICs, IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 51, NO. 6, DECEMBER 2004

[12] Paul E. Dodd : Physics-Based Simulation of Single-Event Effects, IEEE TRANSACTIONS ON DEVICE AND MATERIALS RELIABILITY, VOL. 5, NO. 3, SEPTEMBER 2005

[13] Allan. H. Johnston ,Jet Propulsion: California Scaling and Technology Issues for Soft Error Rates, Presented at the 4th Annual Research Conference on Reliability, Stanford University, October 2000