Design of Soft Error Robust High Speed 64-bit Logarithmic Adder

by

Jaspal Singh Shah

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2008

©Jaspal Singh Shah 2008

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Jaspal Singh Shah

Abstract

Continuous scaling of the transistor size and reduction of the operating voltage have led to a significant performance improvement of integrated circuits. However, the vulnerability of the scaled circuits to transient data upsets or soft errors, which are caused by alpha particles and cosmic neutrons, has emerged as a major reliability concern. In this thesis, we have investigated the effects of soft errors in combinational circuits and proposed soft error detection techniques for high speed adders. In particular, we have proposed an area-efficient 64-bit soft error robust logarithmic adder (SRA). The adder employs the carry merge Sklansky adder architecture in which carries are generated every 4 bits. Since the particle-induced transient, which is often referred to as a single event transient (SET) typically lasts for 100~200 ps, the adder uses time redundancy by sampling the sum outputs twice. The sampling instances have been set at 110 ps apart. In contrast to the traditional time redundancy, which requires two clock cycles to generate a given output, the SRA generates an output in a single clock cycle. The sampled sum outputs are compared using a 64-bit XOR tree to detect any possible error. An energy efficient 4-input transmission gate based XOR logic is implemented to reduce the delay and the power in this case. The pseudo-static logic (PSL), which has the ability to recover from a particle induced transient, is used in the adder implementation. In comparison with the space redundant approach which requires hardware duplication for error detection, the SRA is 50% more area efficient. The proposed SRA is simulated for different operands with errors inserted at different nodes at the inputs, the carry merge tree, and the sum generation circuit. The simulation vectors are carefully chosen such that the SET is not masked by error masking mechanisms, which are inherently present in combinational circuits. Simulation results show that the proposed SRA is capable of detecting 77% of the errors. The undetected errors primarily result when the SET causes an even number of errors and when errors occur outside the sampling window.

Acknowledgements

First and foremost, I would like to express my deepest sense of gratitude to Prof. Manoj Sachdev for providing me an opportunity to be a member of CMOS Design and Reliability Group (CDR). It has been a wonderful experience having him as my mentor and working in the group. His guidance, support, and patience during my research are greatly appreciated. I would also like to thank my thesis readers Prof. Andrew Kennings and Prof. Karim Karim for their valuable time and feedback.

I am thankful to my friends in CDR; Jahin, David Li, and David Rennie for their help with my research. I would also like to thanks - Jahin, Tahseen, Sumanjit, and Tasreen for various thoughtful moments shared in the lab.

Sincere thanks to Phil Regier and Fernando Hernandez for taking care of all the software and hardware problems.

I am very grateful to my parents, brother, and his family for their love and support. Special thanks to my wife Inderpreet for her continuous love, support, and encouragement and to my son Harnamit for bringing joy to our life. I also thank my parents-in-law and all my friends.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter will provide an overview of various failure mechanisms, discuss the source and physical causes of soft errors and examine the effects of soft errors in integrated circuits. At the end, the chapter is summarized and thesis organization is provided.

1.1 Failure mechanisms

The exponential growth in the number of transistors on a chip has resulted in new obstacles. Technology scaling has resulted in more permanent failures of devices and interconnects, and more temporary failures such as errors due to transients in the signaling and storage of logic values [Kahng03]. Permanent failures occur when there is a physical imperfection in the chip. These imperfections may be the result of manufacturing or they may occur during the lifetime of the chip from effects such as electromigration or oxide breakdown. Temporary failures, also called soft errors, are errors caused by cosmic rays or alpha particles.

1.1.1 Permanent failures

Electromigration refers to the migration or displacement of metal atoms due to the impact of moving electrons. A metal line will fail or form voids or may form extrusions that project from one of the surfaces if sufficient current density and high temperature is applied. In this phenomenon, the electrons transfer enough momentum to the metal atoms forcing the atoms out of their lattice site and then move them via diffusion. For the failure process to begin the metal needs an imperfection. The metal has unavoidable vacancies and irregular grain boundary patterns that can initiate electromigration. Eventually the failure is in the form of an open circuit or a defective bridge formed by extrusion [Segura04].

The reliability of gate oxide is important in scaled down technologies. The purpose of the oxide is to isolate the gate from the substrate. With shrinking CMOS technology, thickness of the gate oxide is reduced to increase the switching speed. High-k dielectric materials are used to reduce the leakage when the transistor is off. Two main types of oxide failures are oxide wear out and hot carrier injection. Thin oxides may wear out because of electron traps at the oxide interface. With scaling, power supply and operating voltages have not scaled accordingly. For an electron to become hot, an electric field of 10^4V/cm is necessary. This condition is easily met in devices with lengths in the sub-

100 nm regime. An increased electric field provide enough energy to the electron to tunnel into the gate oxide and cause a threshold voltage (V_T) shift leading to reliability issues. PMOS transistor oxide reliability issues are called negative bias temperature instability (NBTI). When the PMOS is negatively biased at elevated temperature, it results in a V_T shift causing problems similar to hot carrier effects.

1.1.2 Temporary failures/Soft errors

When a high energy particle (e.g., neutron, alpha particle, heavy ion) strikes a silicon substrate, it results in an ionization event. Such an event that can upset a data state is dependent upon several factors such as the energy of the incident particle, the location of the strike, the potential of the node, and the amount of charge collected. Such an event is referred to as a single event transient (SET). An SET leading to a false logic evaluation in a combinational circuit which is latched or an SET resulting in a bit flip in a memory cell, register, or flip-flop is called a soft error. The error is 'soft' because if new data is written to the bit, it will be stored correctly. The term soft error is also called a single event upset (SEU). The rate at which soft errors occur is called the soft error rate (SER) [Baumann01], [Baumann05b]. The unit for measuring the SER and other reliability mechanisms is the failure in time (FIT). A FIT is equivalent to one failure in 10^9 device hours. Ever increasing demand for high density and low power have resulted in decreasing transistor size and smaller node voltages. If uncorrected, failures due to soft errors can be higher than all the reliability mechanisms combined [Baumann05a]. With scaling, the node capacitance has decreased. In order to keep the electric field constant, operating voltage is scaled as well. Thus, the total charge required to toggle a node from a particle induced transient decreases [Shivakumar02]. A recent work predicted that the SER per chip of logic circuits will increase nine orders of magnitude from 1992 to 2011, and the impact of SER on combinational logic will be comparable to that of unprotected memory elements [Shivakumar02].

In the following section the main sources of soft errors will be described.

1.2 Soft Error Sources

The main sources of soft errors are: alpha particles, high energy cosmic neutrons, and low energy neutrons. Electromagnetic interference can also cause soft errors by producing alpha-particles.

1.2.1 Alpha particles

Alpha particles are emitted from packaging materials and the interaction of thermal neutrons with the boron present in p-type semiconductors. An alpha particle, composed of two neutrons and two protons, is a doubly ionized helium atom emitted from the nuclear decay of unstable isotopes. The most common source of alpha particles is from naturally occurring ^{238}U, ^{235}U, and ^{232}Th. These impurities emit alpha particles over a range of energies from 4 to 9 Mev.

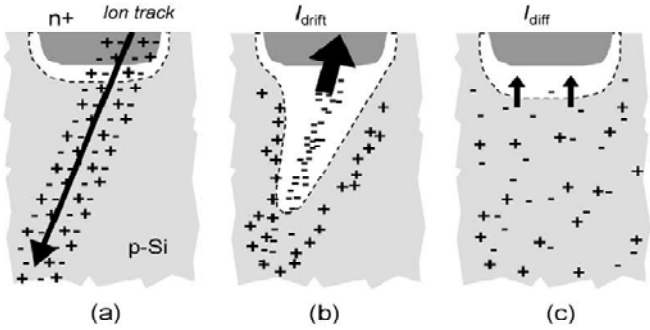


Figure 1.1: Charge generation and collection by heavy ions. Adapted from [Baumann05b].

The interaction of an alpha particle with the silicon substrate is electronic in nature. A particle travelling through the substrate creates electron hole pairs. Figure 1.1 shows the charge generation and the collection phase in a reverse biased junction. Electrons drift to higher potential of n-diffusion and the holes drift to the lower potential of p-diffusion. This sudden burst of charge collection results in a current pulse which can upset the data state. The higher the energy of the alpha particle, the farther it travels into the substrate and more number of electron hole pairs it will generate. Consequently, the higher will be the peak of current pulse. For silicon, the range of a 10MeV particle is <100µm [Baumann05b]. In lead solders, ^{210}Pb is chemically inseparable from ^{208}Pb. ^{210}Pb does not emit an alpha particle when it decays, however, due to short half life of ^{210}Pb, growth of ^{212}Po from $^{210}Pb \rightarrow {}^{210}Bi \rightarrow {}^{210}Po \rightarrow {}^{212}Po$ is possible which has a very high alpha particle emission rate. The semiconductor manufacturing process and the packaging materials are purified to a point of diminishing returns.

1.2.2 High energy neutrons

Primary cosmic rays are thought to be of galactic origin. Due to the interaction of cosmic rays with the earth's atmosphere, cascades of secondary particles are produced. The secondary particles then create tertiary particles and so on. About 1% of this flux reaches sea level. Neutrons constitutes

majority of the flux and they do not cause ionization in silicon by itself. They interact with chip materials elastically and inelastically. Inelastic reaction causes the silicon atom to break into lighter ions with additional particles (protons, and neutrons, and alpha particles). The energy transfer in this case is much higher as compared to an alpha particle; giving a typical burst energy of 15 MeV. A few soft error effects such as multi-bit upsets (MBUs) and single event latch-up (SEL) can only be induced by higher energy neutrons. Unlike alpha particles, neutron flux cannot be reduced by process modifications. However, concrete has been shown to reduce the radiation rate by 1.4x per foot of concrete thickness [Baumann05b].

1.2.3 Low energy neutrons

Another source of ionizing particle is the interaction of low energy (< 1.5MeV) neutrons with boron. Boron is extensively used as a p-type dopant and is also used as boron doped phosphosilicate glass (BPSG) dielectric layers. Figure 1.2 shows Boron fission by a neutron. Boron is composed of two isotopes: ^{11}B (80.1% abundance) and ^{10}B (19.9% abundance) [Baumann05b].

$$^{10}_{5}\mathrm{B} + {}^{1}_{0}\mathrm{n} \rightarrow {}^{7}_{3}\mathrm{Li} + {}^{4}_{2}\mathrm{He} \qquad (1.1)$$

^{10}B is unstable when exposed to neutrons and it breaks into a lithium ion and an alpha particle (1.1). The alpha particle and the lithium ion are emitted in opposite directions to conserve momentum.

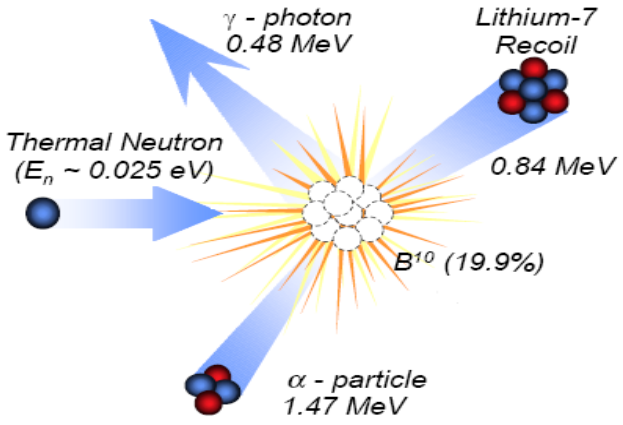


Figure 1.2: Boron Fission by neutron. Adapted from [Web01].

The emitted alpha particle and the lithium have enough energy to induce soft errors in a similar fashion as explained earlier. The SER due to boron activation is mitigated by replacement of BPSG in 0.25µm and beyond processes.

1.2.4 Electromagnetic interference

Electromagnetic interference (EMI) can result in soft errors which are caused by production of α-particles. [Wilkinson05] has shown cancer radiotherapy equipment as a source of soft errors in nearby electronics through interaction of thermal neutrons with boron. The boron component here is BPSG used as dielectric material in earlier CMOS fabrication processes. The α-particles that interact with silicon are capable of producing soft errors as explained earlier.

Table 1-1 : Soft Errors for Different Exposure Conditions and Durations [Wilkinson05]

Exposure Condition	Device Minutes	Errors
50cm, no shielding	30	3
50cm, EMI shielding	10	1
50cm, Thermal neutron shielding	20	0
10m, outside treatment room	20	0

For different exposure conditions, Table 1-1 summarizes the effect of EMI on soft errors. As can be seen, with proper shielding it can be eliminated.

1.3 Soft error in memories

Increased memory density scaling has made memories more vulnerable to single event transients. An SEU stays in the memory unless the bit is written again or is corrected by some other technique.

1.3.1 DRAMs

The normal effect of a transient is to deplete the charge from the cell's storage capacitor. Initially, DRAMs used to store the data on a planar capacitor which had a large area and were very sensitive to an SEU. With the development of 3-D capacitor the critical charge of the node has increased significantly by decreasing the collection efficiency. The collection efficiency decreases with scaling with the decrease in volume of the junction. However, the cell capacitance has remained constant as it

is dominated by the 3-D capacitor cell [Baumann05b]. The result is, the SER of a single bit has decreased, but the system SER is almost constant as shown in Figure 1.3.

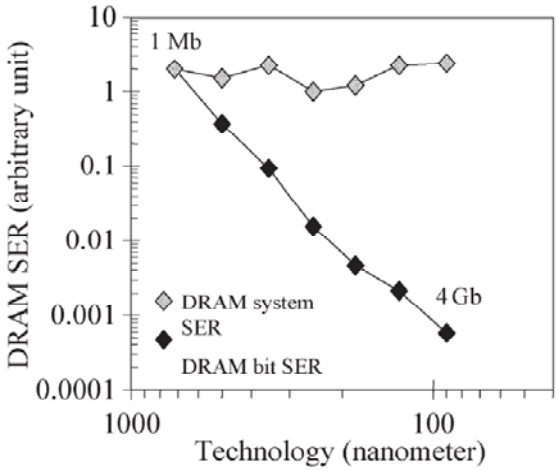


Figure 1.3: DRAM single bit SER and system SER. Adapted from [Baumann05b].

1.3.2 SRAMs

SRAMs store data on the active nodes of a cross coupled inverter pair. A particle strike may flip the state of a memory cell (0 to 1 or 1 to 0). This flip is temporary and the cell can be written back with the correct data in a write operation. The sensitive regions are the driver and the load transistors, which are off. The reverse biased junction of an OFF transistor is sensitive to a particle strike.

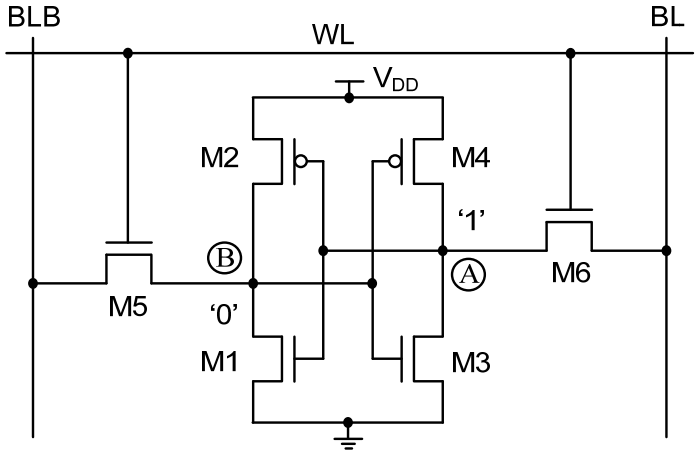


Figure 1.4: A typical SRAM cell.

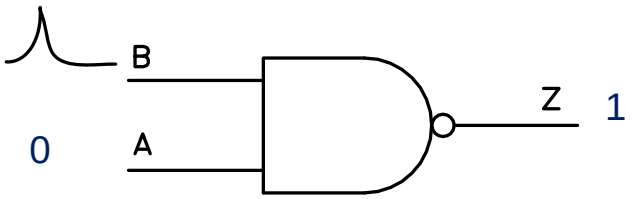
In Figure 1.4 the drain terminal of transistor M3 is sensitive to 1 to 0 transitions and the drain terminal of transistor M2 is vulnerable to 0 to 1 transitions. The minimum charge required to flip the cell is called critical charge (Q_{crit}). Early SRAMs were more robust against the SER because of higher operating voltages. With scaling, SRAM area is minimized to reduced capacitance, leakage, and cell area and the operating voltage is reduced to minimize power. Thus, increased memory density has led to increased SER with each generation of scaling.

1.4 Soft error in logic circuits

An upset in the state of a logic circuit will not affect the computation unless it is latched into a memory element. Thus, a soft error in a combinational circuit is defined as a transient error which will be stored in a memory element [Shivakumar02]. However, unlike in memory circuits there are several phenomena in logic circuits which can mask soft errors.

1.4.1 Logical Masking

An SEU at a node in a combinational circuit will not affect the output of the circuit when its result is determined by another input. The other input is called the controlling input. This can be better explained with the help of a NAND gate.



Controlling Input

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Figure 1.5: Logical masking in NAND gate.

In Figure 1.5 an error affects the input B of the NAND gate while input A is at logic 0. Thus, the error does not affect the output Z. In such cases, the error is said to be logically masked.

1.4.2 Electrical Masking

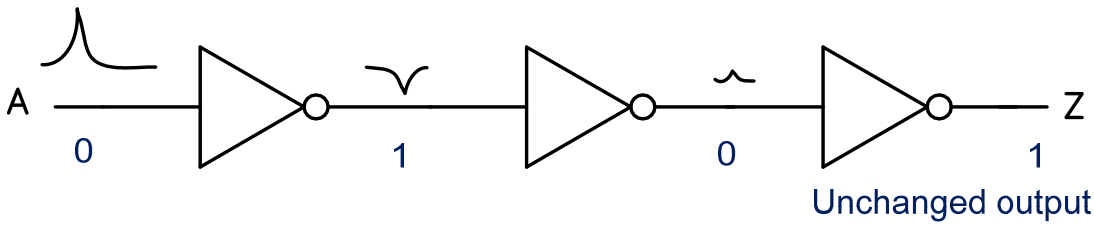


Figure 1.6: Electrical masking in inverter. Adapted from [Karnik04].

The particle strike may be attenuated by the logic gates at subsequent stages due to electrical properties of the gate. This phenomenon is called electrical masking. Figure 1.6 shows the pulse attenuation by a chain of inverters.

1.4.3 Latching Window Masking

The period during which the latch is transparent to the data is called the latching window. The pulse resulting from the particle strike may not reach a latch at the clock transition such that it is not stored in the latch. This effect is called latching window masking (Figure 1.7). The period during which the latch is sensitive to the pulse is called the window of vulnerability [Seifert04].

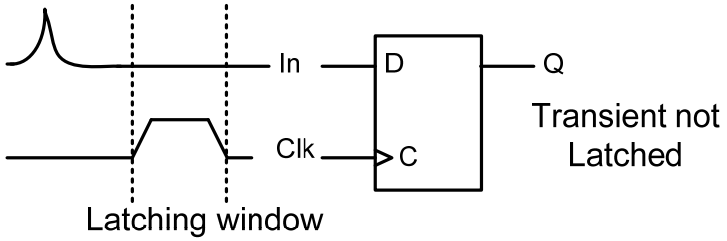


Figure 1.7: Latching window masking

These masking effects lower the soft error rate in combinational logic. Nevertheless, with decreasing feature sizes and increasing in the number of pipeline stages, these masking effects diminish considerably. Electrical masking could be reduced by device scaling because smaller transistors are faster and thus may have less attenuation effect on the pulse. Also, at higher clock rates, latches will cycle more frequently, which may reduce the latching window masking.

1.5 Scaling and Soft Errors

Soft errors were first discovered in DRAMs, and after many generations it is currently a more robust device because of various enhancements. It can be seen that as the technology scales (Vdd goes down and memory density goes up) the SER for DRAMs goes down as shown in Figure 1.8.

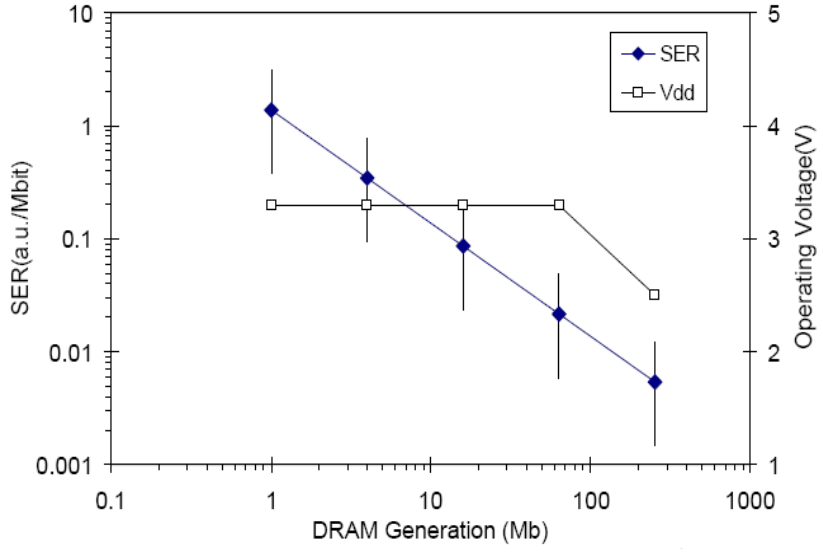


Figure 1.8: DRAM SER and scaling. Adapted from [Web01].

However, SRAMs are becoming more susceptible to soft errors as technology scales. This is because with scaling, the node capacitance and supply voltage are decreasing.

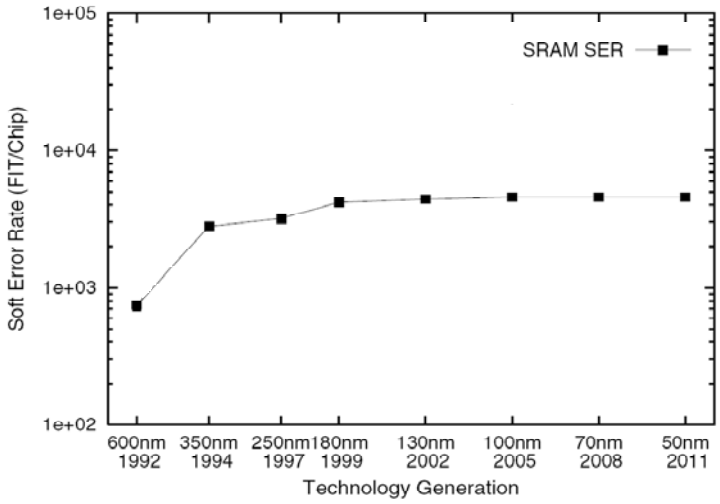


Figure 1.9: SER of a constant area SRAM array. Adapted from [Shivakumar02].

The sensitivity of SRAMs is determined by the critical charge storage and collection efficiency. The collection efficiency is determined by the process while the critical charge depends on both the process and the circuit design. In deep submicron technologies, both critical charge and collection efficiency decrease with scaling [Hazucha00]. Figure 1.9 shows the SER for a constant area SRAM array. The area dedicated to memory is expected to increase by 70% in the next decade [Semico07]. Thus, the overall SER for SRAMs increases with scaling. With decreasing feature size, the critical charge of the node decreases making logic circuits more susceptible to the SER.

1.6 Motivation

Addition is a basic operation in an arithmetic logic unit (ALU). An error effect in an adder will only be known after several clock cycles and correction at that time is practically not feasible. With technology scaling, the critical charge required to upset a logic level in a combinational circuit has been reduced. In order to have a reliable system which is immune to soft errors some method needs to be employed in terms of space or time redundancy or a combination of both [Nicolaidis99], [Anghel00]. Space redundancy implies running the same inputs in parallel through two sets of hardware. Sometimes dual rail logic is used for this purpose. The outputs are then latched and compared through a parity generation circuit. Space redundancy generally results in 100% hardware overhead. However, some techniques are proposed in literature which save area by some extent as discussed later in the thesis. Another methodology for reliable system design is time redundancy. In this approach, the same output is sampled at two different time intervals. The results are latched and then compared with an XOR tree. A mismatch between two samples indicates an error. In order to have a comparison either of the two samples should have a correct value. Hence, the input needs to be evaluated again for the second sample. For example, in dynamic circuits, to evaluate again a precharge cycle is needed. Hence, the dynamic circuit implementing time redundancy requires two clock cycles.

In the proposed research, time redundancy is explored to design a high speed 64-bit soft error robust logarithmic adder using a single precharge and evaluate cycle. The aim of this work is to explore circuit techniques that can be applied to design a reliable adder. In this thesis, pseudo-static logic style is used to implement the compound domino logic. The pseudo-static logic is capable of evaluating twice without an extra pre-charge. This circuit is realized in 90nm CMOS technology.

1.7 Summary and Thesis Organization

In this chapter basic failure mechanisms in integrated circuits have been discussed. The main focus area from reliability of circuits and systems perspective is the vulnerability of the integrated circuits to soft errors. Soft errors were earlier believed to be issues only in memories. However, with scaling technology and shrinking transistor size, the critical charge required to upset a logic node has reduced. Soft errors in combinational circuits need to be addressed for reliable operation of integrated circuits. Different sources of soft errors are analyzed for memories and logic circuits. Major thesis contributions are:

- Proposed pseudo-static logic as a choice for time redundant circuit design.
- Time redundancy is implemented for the first time to design a soft error robust adder.

The thesis is organized as follows: in Chapter 2, various adder architectures are discussed. Existing soft adder robust adders are also discussed. In Chapter 3, design of the proposed soft error robust adder is presented. Circuit techniques are compared for design of a robust adder and the components of the design are explained. In Chapter 4, simulation results of the proposed adder are critically analyzed in terms of energy and delay. In Chapter 5, conclusion is provided with the possibility of future work.

Chapter 2
Adder Architectures

In this chapter, essential background information of the adder will be provided; various adder architectures will be discussed and compared. Known soft error robust architectures will be presented and analyzed. And finally, the chapter is summarized.

Addition has always been a great subject of research with the focus being architectures which are smaller, faster and energy efficient. There are a variety of architectures available which are good in optimizing one or the other parameter. Addition forms the basis of any computer architecture; performance and reliability of the adder can dominate the performance of the architecture. Much attention has been paid to minimize the chip area and to optimize the speed of operation. However, with shrinking transistor size the computational unit has become more susceptible to SEUs.

2.1 Basic Full Adders

Consider a two bit adder with carry input such that A, B, C_i are the inputs which generates a sum S and carry out C_o. The truth table and the expression are as below:

Table 2-1 : Truth Table for Adder.

A	B	Ci	S	Co	Carry Status
0	0	0	0	0	Delete
0	0	1	1	0	Delete
0	1	0	1	0	Propagate
0	1	1	0	1	Propagate
1	0	0	1	0	Propagate
1	0	1	0	1	Propagate
1	1	0	0	1	Generate/Propagate
1	1	1	1	1	Generate/Propagate

$$\boldsymbol{S = A \oplus B \oplus C_i} \tag{2.1}$$

$$\boldsymbol{C_o = AB + BC_i + AC_i} \tag{2.2}$$

If propagate (P) and generate (G) signals are defined such that

$$\boldsymbol{P_i = A_i + B_i} \tag{2.3}$$

$$\boldsymbol{G_i = A_i B_i} \tag{2.4}$$

The generate signal indicates whether a carry is generated (0 or 1) at the i^{th} bit location and the propagate signal indicates whether an incoming carry from $(i-1)^{th}$ bit is propagated to i^{th} bit. In terms of full adder, logic is given as:

$$\boldsymbol{C_i = G_i + P_i C_{i-1}} \tag{2.5}$$

and

$$\boldsymbol{S_i = P_i \oplus C_{i-1}} \tag{2.6}$$

where C_i and S_i represents the carry and the sum results for i^{th} bit in an n-bit adder. It is important to mention that Pi as defined in (2.3) is only valid for carry calculation. The propagate signal for the sum calculation must be implemented as

$$\boldsymbol{P_i = A_i \oplus B_i} \tag{2.7}$$

In the present day architectures, two intermediate sum signals are calculated anticipating an incoming carry; $C_{i-1} = 0$ and $C_{i-1} = 1$ and a multiplexer is used to select the appropriate sum, once the actual carry becomes available. This has resulted in some speed improvements. Some of the architectures are discussed in the following sections.

2.1.1 Ripple Carry Adder

A ripple carry adder for an n-bit operand can be constructed by cascading n-full adders as shown in Figure 2.1. At i^{th} bit location the carry output and the sum are generated by using carry from (i-1) stage. In this case, the carry ripples from the least significant bit to most significant bit. Thus, the adder is called ripple carry adder. The delay is a function of the number of stages. The propagation delay of this network is also a function of the input vector. For an n-bit adder, the worst case delay will occur when the carry has to ripple from the least significant bit to the most significant bit and is given by (2.8)

$$t_p = (n-1)t_{carry} + t_{sum} \tag{2.8}$$

where t_{carry} is the carry propagation delay from the input to the output and t_{sum} is the propagation delay of the sum block. The delay of ripple carry adder is a linear function of the number of bits. Thus, as the number of bits increases the delay increases as well.

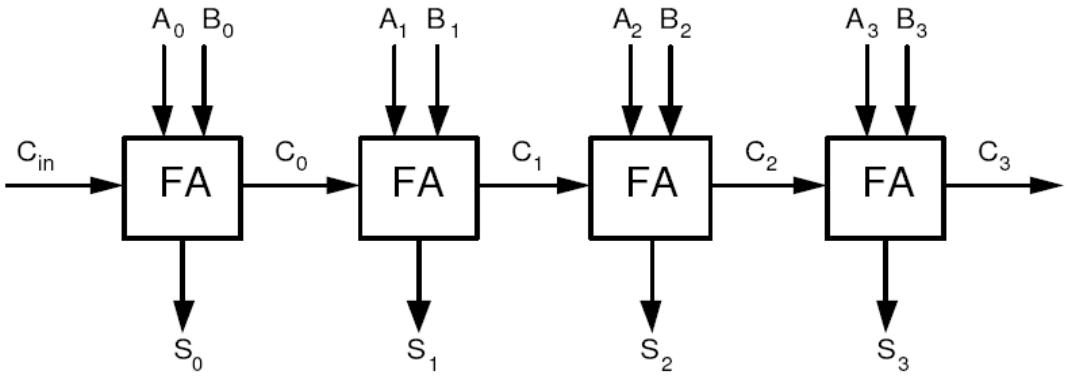


Figure 2.1: Four bit ripple carry adder. Adapted from [Rabaey03].

2.1.2 Carry Select Adder

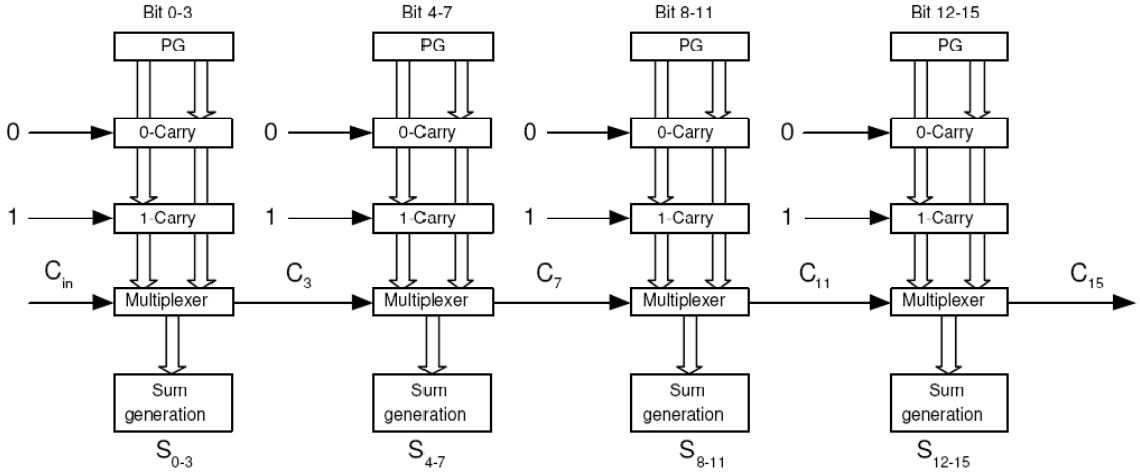


Figure 2.2: 4-bit carry select adder. Adapted from [Rabaey03].

The carry select adder anticipates the outcome of the carry based upon possible values of the input carry and evaluates the results for both the possibilities in advance. Once the real value is known an appropriate result is selected using a multiplexer. As shown in the Figure 2.2 the carry output from the previous block controls the multiplexer that selects the appropriate carry. If the number of bits are

N, we can divide all the bits into N/M groups with M bits in each, the worst case propagation delay is given by (2.9)

$$t_p = t_{setup} + Mt_{carry} + \left(\frac{N}{M}\right) t_{bypass} + t_{sum} \quad (2.9)$$

2.1.3 Carry Skip Adder

In a ripple carry adder, the longest path is from the carry in to the carry out. Suppose the input vector is such that all propagate signals is logic 1 then the ripple process can be bypassed using additional circuit controlled by propagate logic. The structure of the carry skip adder (also called carry bypass adder) is shown in Figure 2.3.

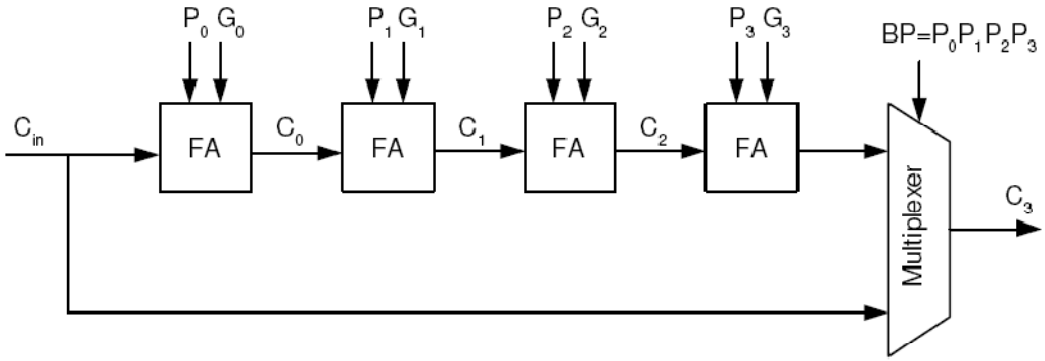


Figure 2.3: Carry skip adder. Adapted from [Rabaey03].

The worst case delay of N-bit inputs such that it is divided into N/M groups of M bits each is given by (2.10).

$$t_p = t_{setup} + Mt_{carry} + \left(\frac{N}{M} - 1\right) t_{bypass} + (M-1)t_{carry} + t_{sum} \quad (2.10)$$

where t_{setup} is the time to create propagate and generate signals through one block, t_{carry} is the propagation delay through one block, t_{bypass} is the delay through multiplexer of single stage and t_{sum} is the time to generate the sum of final stage.

2.1.4 Carry Look-ahead Adders

Fast adders look-ahead to predict the carry out in an N-bit adder. The look-ahead adders remove the ripple carry effect by generating a carry for each bit simultaneously. The delay to add two N bit

numbers no longer depends upon N, but on the logarithm of N, which is smaller. Here, all the required carry outputs are computed in parallel based on propagate and the generate signals which are given by (2.11). The dependency between C_i and C_{i-1} can be eliminated by expanding the equation.

$$\boldsymbol{C_{o,i} = G_i + P_i\left(G_{i-1} + P_{i-1}\left(\ldots + P_1\left(G_0 + P_0 C_{i,0}\right)\right)\right)} \tag{2.11}$$

Where $C_{i,0}$ is typically 0. It is clear from (2.11) that the carry at intermediate level can be computed independently. As N grows, the carries require gates with larger fan-ins, which will slow down the adder computation.

2.1.5 Hybrid Adder Architectures

Tree adders are look-ahead adders with multilevel look-ahead. The main components of a tree adder are the number of logic stages, the number of logic gates, the maximum fanout on each gate, and the number of wiring tracks between each stage. An ideal tree adder would have $log_2 N$ level of logic with fanout of 2 at each stage and 1 wiring track between each level. Based on the work done by [Harris03] and [Patil07] the carry propagation logic has four parameters:

1. Radix (R): In tree adders, R is defined as the average number of bits combined at each logic stage of carry propagation logic (CPL). In a linear carry-skip or carry select adders, R refers to the average number of bits combined per stage to generate a block propagate-generate term.
2. Logic Depth (L): L indicates the total number of stages in the CPL, and is at least $log_2 N$ for an N-bit adder. It is important to note that the number of stages in the adder can be more than L.
3. Fanout (F): F represent the maximum logical branching seen by any stage in the CPL.
4. Wiring Track (T): T measures the maximum number of wires running across the bit pitch between any successive levels of the CPL.

The carry select adder generates the sum based on an input carry of 1 and 0. When the tree adder is combined with a carry select adder, it is called a hybrid adder. Hybrid adders select the correct sum which is computed in parallel with the carry merge tree based on the incoming carry. The timing of signals is an important consideration in this case such that the sum and the incoming carry are available at the same time.

2.2 Comparison of Adder Architectures

For the carry generation, R, L, T, and F are interdependent. The Brent-Kung tree [Brent82] minimizes F. The Sklansky [Sklansky60] tree reduces L to a minimum at the expense of F. The Kogge-Stone tree [Kogge73] minimizes both L and F at the cost of long wiring tracks i.e. T. The tree has more propagate generate (PG) cells which will increase power consumption. The Kogge-Stone is still widely used for high performance 32-bit and 64-bit adders [Weste05]. The Han-Carlson Tree [Han87] is a hybrid between Kogge Stone and Brent Kung. It performs Kogge Stone for the odd number of bits and then uses an additional stage to evaluate carries at the even bits reducing T. The Knowles tree [Knowles01] is a network between Kogge-Stone and Sklansky. It reduces T by increasing F with Kogge Stone as the reference. The Ladner-Fisher [Ladner80] tree is network between Sklansky and Brent-Kung. It computes prefixes at odd numbered bits and uses an additional stage to calculate even bits. A modification of Kogge Stone is a 32-bit sparse-tree adder [Mathew03] which is divided into critical and non-critical sections. Instead of generating carry for each bit (C0, C1, C2 and so on), the sparse tree adder generates carries every fourth bit (i.e., C0, C3, C7 and so on). As a result, the critical path is reduced to a pruned carry merge tree that consists of a PG generator followed by five stage carry-merge logic for 32-bit adder. This approach reduces both F and T significantly.

2.3 Existing Soft Error Robust Adders

Parity based error detection is suggested [Gaisler97] to detect SEUs in microprocessors. This observation is based on heavy ion testing of a 32-bit SPARC compatible processor for space application. Duplication with comparison and triple modular redundancy (TMR) is well known for single error detection. [Mesquita07] proposed a TMR adder for FPGA devices using a carry-select adder. As explained in section 2.1.2 such a scheme performs two additions on the same numbers assuming an incoming carry of 1 and 0. TMR requires three inputs of the block that should be protected. These three inputs are then connected to a voter circuit which by majority election gives the correct output. It is obvious that this idea has an overhead of 200% in terms of area and will affect the power (Figure 2.4).

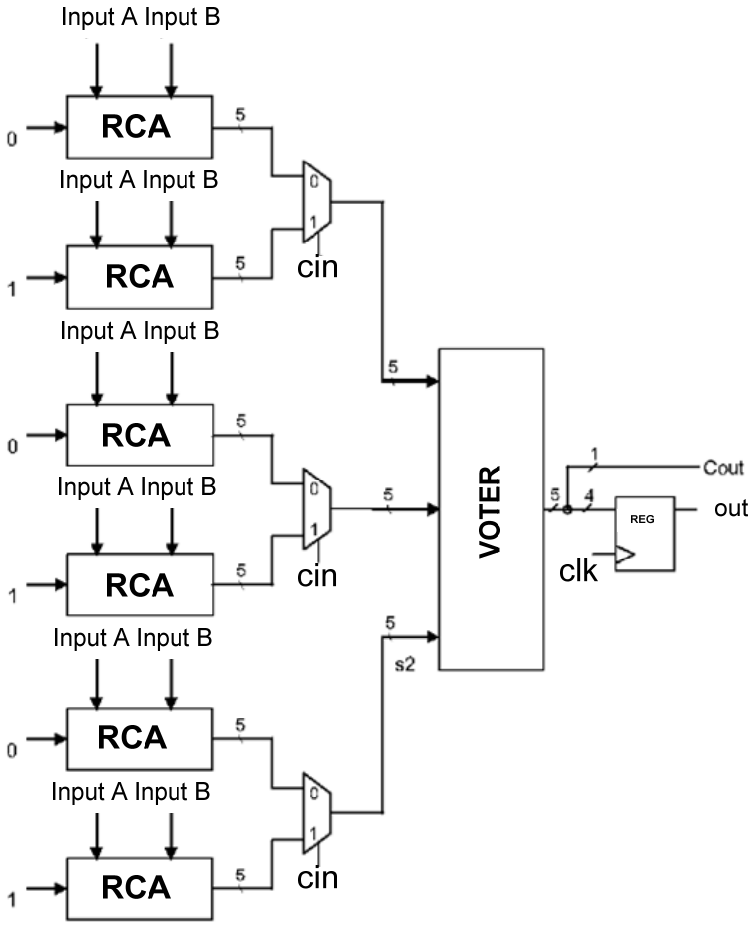


Figure 2.4: Carry select adder with TMR. Adapted from [Mesquita07].

[Mesquita07] reduced the overhead by suggesting an alternative CSA by using the same adder with an inverted carry input. It is called re-computing with inverted carry (RIC). This concept works well assuming the incoming carry is error free so that it selects the correct input at the multiplexer.

In the current trends, where 64 bit adders are the norm, the carry merge tree is huge as compared to the sum circuit, and soft errors originating in the carry merge tree will select the wrong sum in the CSA. The TMR will not be able to detect this. Secondly, if there is an SET event in the voter circuit itself, it will result in a spurious output which the current design will not be able to detect.

In another theoretical work [Townsend03b], a1-out-of-3 coding scheme is presented as a possible method of detecting errors in adder computation. The valid code words are {100,010,001}, where $\bar{1}$ is represented by the code word 100, 0 is represented by the code word 010, and 1 is represented by the

code word 001. For example, to add +4 and -2, the operands can be represented in decimal, 2's complement, and 1-out-of-3 encoding as:

Addend +4	0100	010 001 010 010
Augend -2	1110	100 001 001 010

The block diagram of this method is shown in Figure 2.5**.**

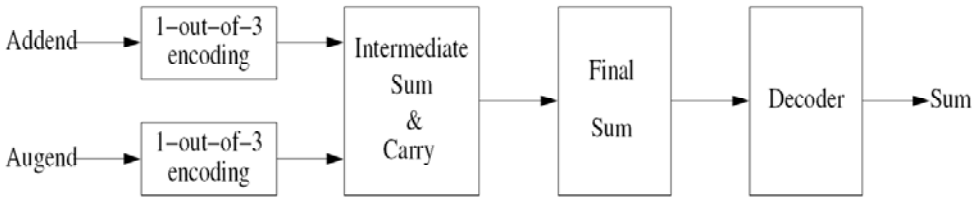


Figure 2.5: Block diagram for addition with encoding. Adapted from [Townsend03b].

A 1-out-of-3 checker is required to detect errors in the code words. If an error occurs then there is either more than one 1 or less than one 1 in the code word. The author was unable to find a hardware implementation of this scheme and believes there are three major issues. One, encoding and decoding is done in-line with adder computation which will result in significant delay. Only simulation results or a silicon implementation will justify weather the delay is competitive compared to time redundant or space redundant approaches discussed earlier. Second, operands are stored in flip flops and any SEU event affecting the operand will produce false encoding, which will transport erroneous computation. Third, the adder is a part of an arithmetic logic unit (ALU), and the implementation of such codes will result in increased complexity and considerable area overhead.

The carry checking/parity prediction adder presented by [Nicolaidis03] has laid out some goals which such a circuit should meet in order to be efficient.

1. It should be totally self checking.
2. It requires low-hardware overhead.
3. It has a compact checker.
4. It can be combined with parity checked datapaths and memories without using code translators.

Further, three schemes are proposed by [Nicolaidis03] which will result in low hardware cost. Figure 2.6 (a) shows the block diagram for this case. The sum output of this adder is given by Equation (1) [Nicolaidis03] and is also true for [Mathew07].

$$P_{sum} = \sum_{i=0}^{n-1} S_i = \sum_{i=0}^{n-1} (A_i \oplus B_i \oplus C_{i-1})$$
$$= \sum_{i=0}^{n-1} A_i \oplus \sum_{i=0}^{n-1} B_i \oplus \sum_{i=0}^{n-1} C_{i-1}$$
$$= P_A \oplus P_B \oplus P_{Carry} \quad (2.12)$$

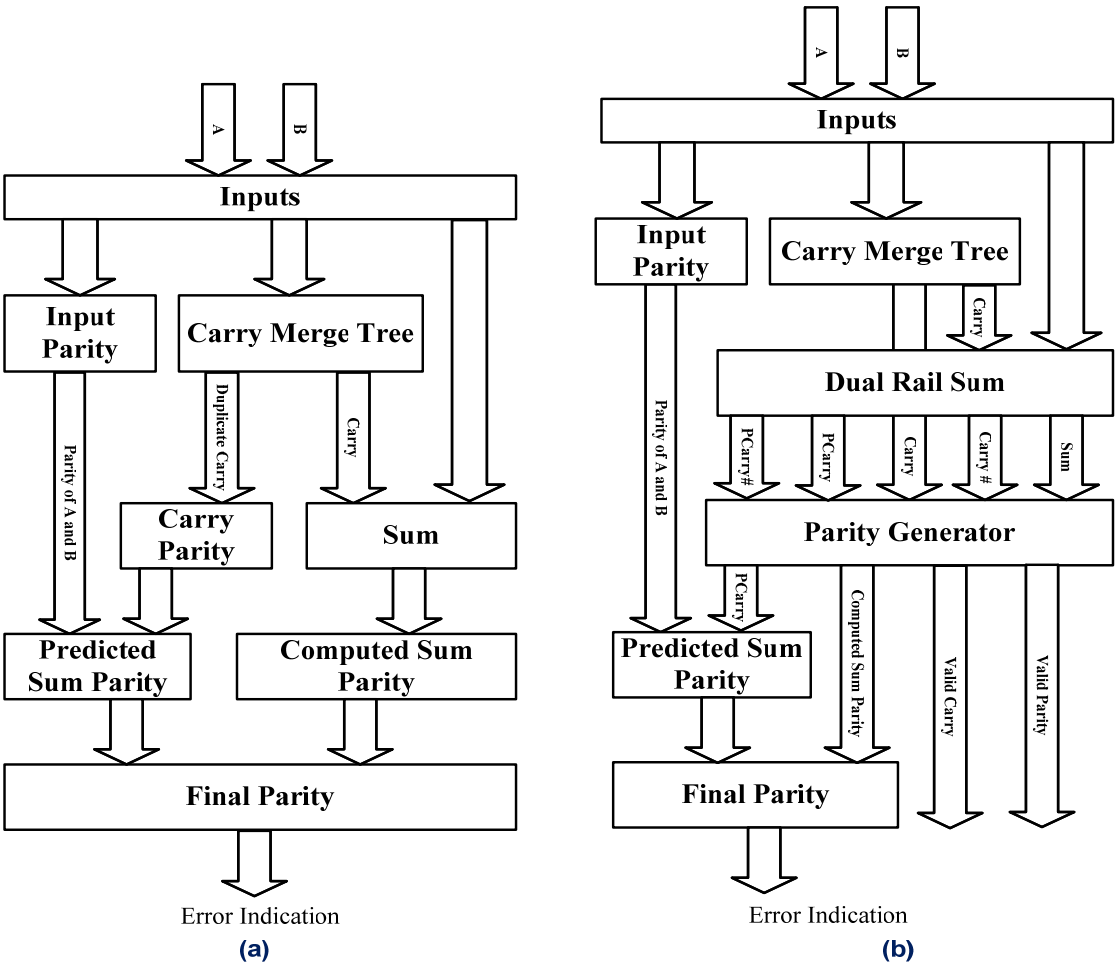


Figure 2.6 : Block diagrams: (a) [Nicolaidis03] and (b) [Mathew07]

Where A_i , B_i are operands and S_i is the sum for i[th] bit. P_A, P_B are the input parities. P_{sum} is the output sum parity and P_{carry} is the carry parity. When the sum is available at the end of the cycle, its parity can be computed by an XOR tree.

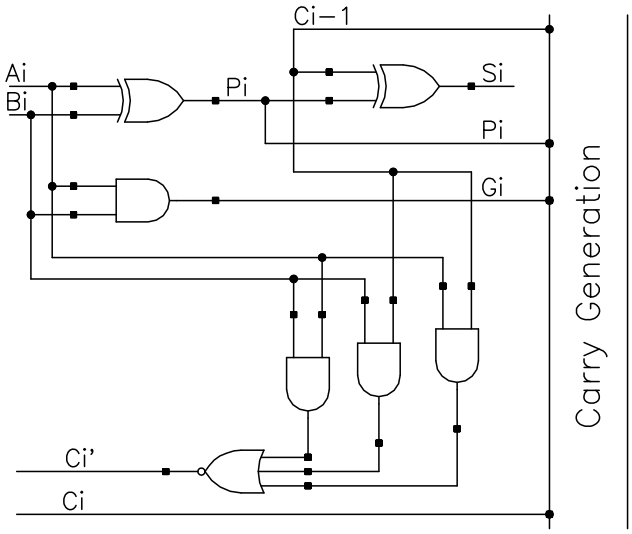


Figure 2.7: Adder bit slice. Adapted from [Nicolaidis03].

A mismatch between the predicted sum parity and the computed sum parity implies there is an error. If the same circuit is used to generate P_{sum} and P_{carry} , there may be an SEU affecting both the parities. Hence, the carry parity circuit should compute parity in parallel with the sum circuit. If an SET affects the carry at an arbitrary location, further assuming that the input operands are conducive for carry propagation, the result will be multiple errors in the carry propagation. Thus, it can be safely concluded that for accurate parity prediction some kind of carry redundancy is required [Nicolaidis03]. A double rail technique is used by [Nicolaidis03] to generate two set of carries which are the complement of each other. Figure 2.7 shows how a duplicate carry, also called a check carry, can be generated. Here, one carry is coming from the carry generation circuit and the other one is being calculated by the duplication block as described in Equation (2.13) .

$$C_i' = (A_iB_i + B_iC_{i-1} + A_iC_{i-1})' \quad (2.13)$$

The duplication block receives its input from the carry generation circuit. Thus, an error affecting the carry generation circuit will also affect the check carry. If we consider C_i such that it has been affected by a fault, C_i' will remain unaffected because it is generated from C_{i-1} which is again unaffected.

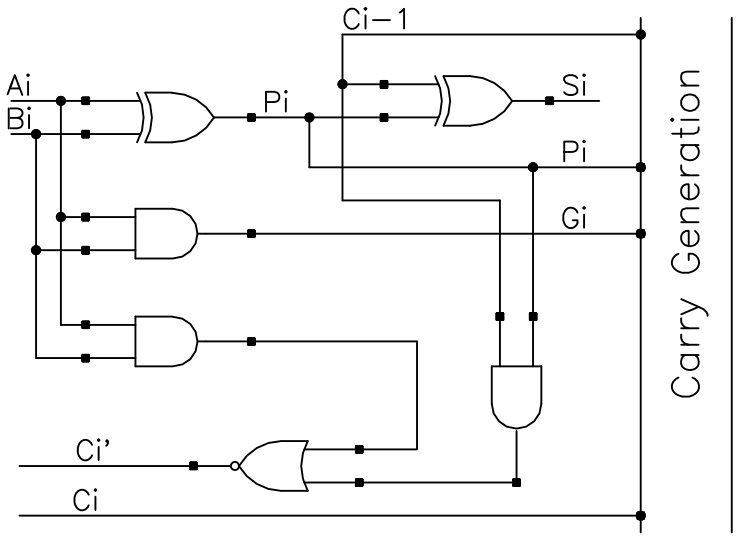


Figure 2.8: Adder bit slice with partial carry duplication. Adapted from [Nicolaidis03].

Hence, the error is detected by comparison of $\boldsymbol{C_i}$ and $\boldsymbol{C_i}'$ even though $\boldsymbol{C_{i+1}}$ and $\boldsymbol{C_{i+1}}'$ are both faulty [Nicolaidis03]. The partial carry duplication scheme as expressed in Equation (2.14) is shown in Figure 2.8. The main drawback of this scheme is that for a 64 bit adder the carry duplication will have hardware overhead and it will increase T and affect F of the previous stage.

$$C_i' = (A_i B_i + P_i C_{i-1})' \tag{2.14}$$

In the case when a propagate signal is affected by an SET, the sum signal is also affected which can be detected. This test case is shown in Figure 2.9. Another scheme proposed by [Mathew07] is shown in Figure 2.6 (b). In this case, a dual rail sum is generated in parallel with the carry merge tree. True and complementary carry and carry parity is generated. A parity generation circuit generates computed sum parity which is compared with the calculated sum parity. The true and the complementary carry and the parity signals are also compared to indicate if the results are valid. Both [Nicolaidis03] and [Mathew07] compute parity multiple times and they do not account for the increased area and hence the increase in susceptibility to SETs.

	Pi=Ai xor Bi
	Gi=Ai and Bi
Original	Ci=Gi+PiCi-1
Duplicate	Ci=Gi+PiCi-1
	Si=Pi xor Ci-1

A3	A2	A1	A0	PA
1	0	1	0	0
B3	B2	B1	B0	PB
0	0	1	1	0
P3	P2	P1	P0	
1	0	0	1	
G3	G2	G1	G0	
0	0	1	0	
C3	C2	C1	C0	Pcarry
0	0	1	0	1
C3	C2	C1	C0	Pcarry
0	0	1	0	1
S3	S2	S1	S0	Sum Parity
1	1	0	1	1
SumParity= S3 xor S2 xor S1 xor S0 = 1				
Predicted Sum Parity = PA xor PB xor Pcarry = 1				
No Error !				

A3	A2	A1	A0	PA
1	0	1	0	0
B3	B2	B1	B0	PB
1	0	1	1	0
P3	P2	P1	P0	
0	0->1	0	1	
G3	G2	G1	G0	
1	0	1	0	
C3	C2	C1	C0	Pcarry
1	0	1	0	1
C3	C2	C1	C0	Pcarry
1	0	1	0	1
S3	S2	S1	S0	Sum Parity
1	1->0	0	1	0
SumParity= S3 xor S2 xor S1 xor S0 = 0				
Predicted Sum Parity = PA xor PB xor Pcarry = 1				
Error!				

Figure 2.9: Test case for [Nicolaidis03].

2.4 Summary

In this chapter, common adder architectures are discussed and compared. However, due to the nature soft errors, they require special consideration and traditional architectures cannot be used per se without some modifications. Important background information on robust adder architectures is provided. To achieve fault tolerance, redundant information is required. It can be achieved either by space, time or a combination of time and space redundancy (section 1.6). Space redundancy always results in hardware overhead while time redundancy results in delayed outputs. Considering fault tolerant circuits, [Nicolaidis03] proposed a method to generate parity for the operands and the carry with some optimizations in redundant carry generation circuit. Calculated sum parity is compared with the computed sum parity and a mismatch indicates an error. In this scheme, carry parity is generated using each carry from the tree. Using a sparse tree architecture, carry is calculated every 4^{th} bit using dual rail sum trees, and a parity generation circuit resulting in reduced wiring tracks [Mathew07]. Consequently, use of dual rail can translate to higher area cost and increased susceptibility to SETs.

Chapter 3
Design of Soft Error Robust Adder

In this chapter, the focus is on the design of a soft error robust adder [SRA]. Logic family is comprehensively investigated for sensitivity to SETs and suitability for robust design. Other components of design such as sum circuits, and parity circuits are presented and optimized for power and delay. The proposed adder is summarized in the end.

An error can be defined as the probability of a failure over the entire design. In order to optimize the reliability of a design there must be a figure of merit or a methodology which can quantify the reliability. To the author's best knowledge no such approach exists. The design of an adder with soft error robustness has certain challenges. The important metrics which are considered is delay, area, and energy.

3.1 Logic Family

For the design of a high speed adder compound domino logic and hybrid architecture is a practical choice. A carry merge architecture proposed in [Mathew03] and used in [Mathew07] is used to demonstrate and compare energy and delay of the SRA. This architecture is similar to Sklansky in that it computes carries in 4-bit groups [Weste05] as shown in Figure 3.2.

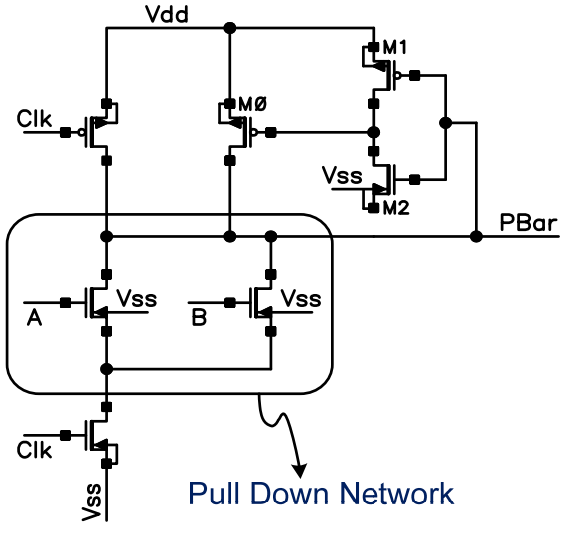


Figure 3.1: Dynamic propagate circuit.

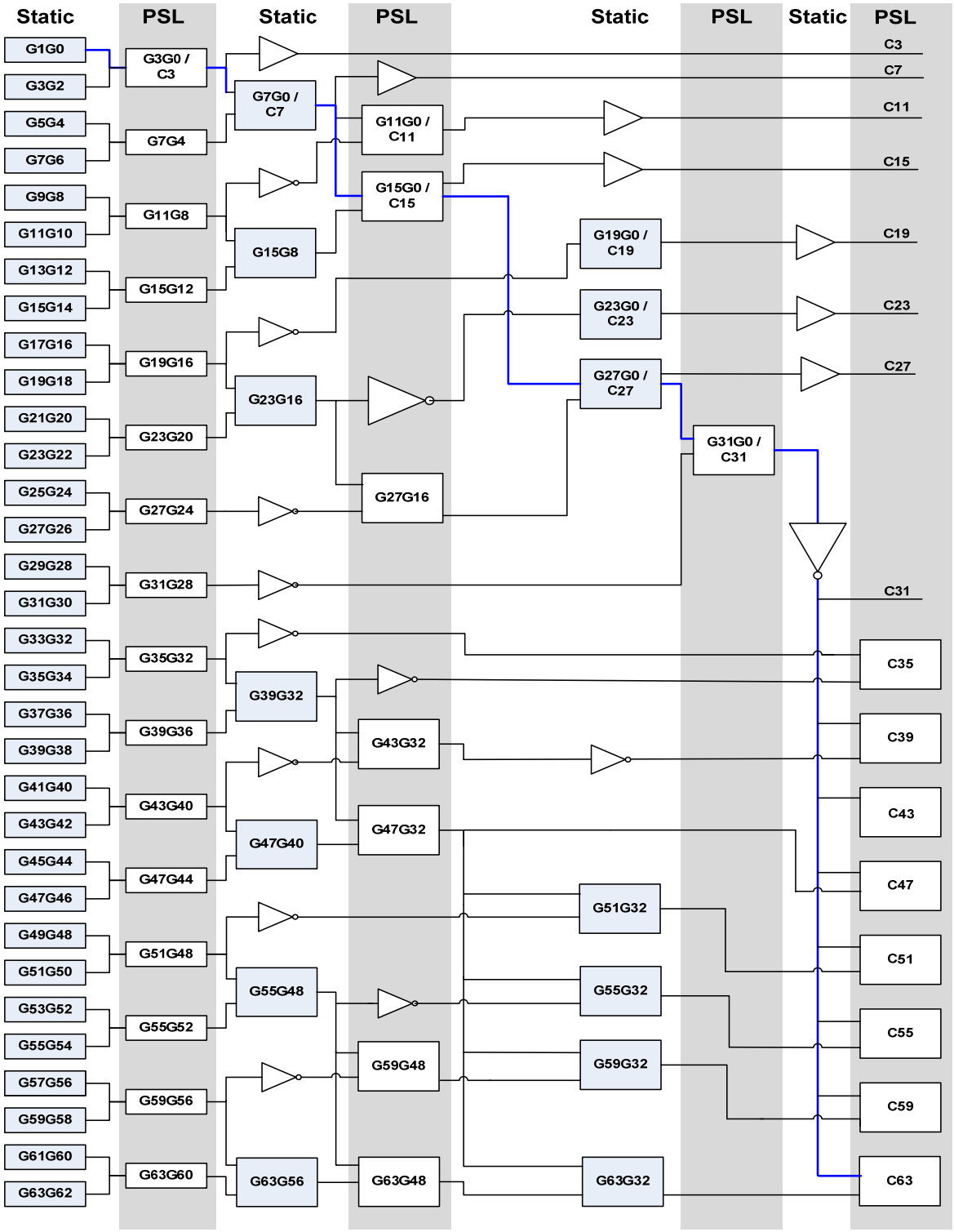


Figure 3.2: 64-Bit carry merge Sklansky adder.

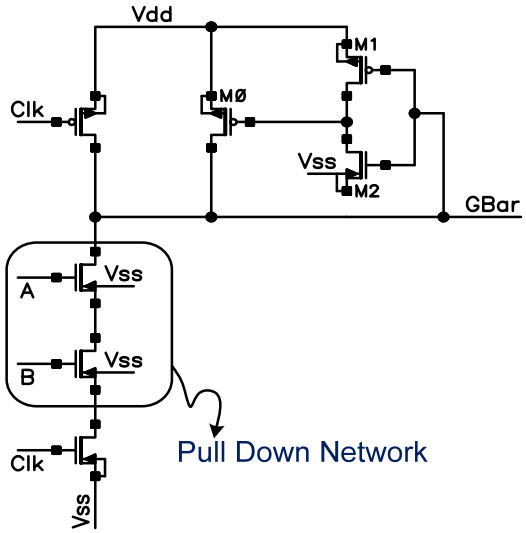


Figure 3.3: Dynamic generate circuit.

In this design, the pseudo-static logic style is used in the compound domino logic instead of the dynamic logic gates. The switching speed of a static circuit depends upon two factors; the current conduction through the MOSFET and the parasitic capacitance. During switching there exists a path between the power supply and the ground which leads to short circuit current dissipation. The dynamic circuits differ from the static circuits in that they use capacitance to hold a logic level. When clock is 0, the pFET charges the output node capacitance to logic 1. This is called a precharge event. During the evaluate event (when clock signal is 1) the pull down network (PDN) is evaluated. If the input vector is such that the nFETs conduct then the output is pulled to the ground. In other words, the output node capacitance which was precharged during the evaluation period now discharges to ground. This circuit is called dynamic because its output is valid only for small period of time before leakage corrupts the value. The charge sharing and leakage during evaluation period determines the switching frequency of the circuit. A typical problem of dynamic gates is the issue of cascading which leads to reduced noise margin and there is a chance of the circuit malfunctioning. This problem can be addressed by using Domino logic. In domino logic, the output of the dynamic gate is taken at the output of the inverter which can further be connected to the PDN of the next stage. The cascaded domino stages eliminate the possibility of a glitch in the next level. It still suffers from the charge sharing and the charge leakage problems. Using a feedback loop to control the charge leakage is valuable. Hence, an inverter is used in the output path to derive a pFET which in the case of discharge at the output node will charge it back to supply voltage. The pFET in this design is called a charge

keeper and it helps to keep the signal free from parasitic effects. By combining the outputs of multiple dynamic gates it is possible to eliminate the need of an inverter, and instead complex static logic can be inserted in between. The static logic can perform some logic operation in addition to inversion. Such a structure is called Compound Domino Logic. Figure 3.1 and Figure 3.3 shows the dynamic propagate and the generate circuits which are a part of the compound domino logic.

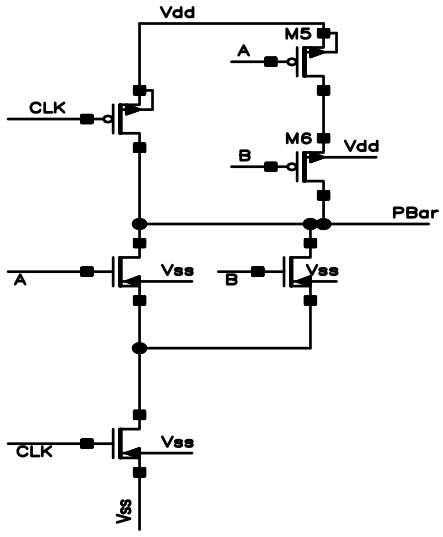


Figure 3.4: Pseudo-static propagate circuit.

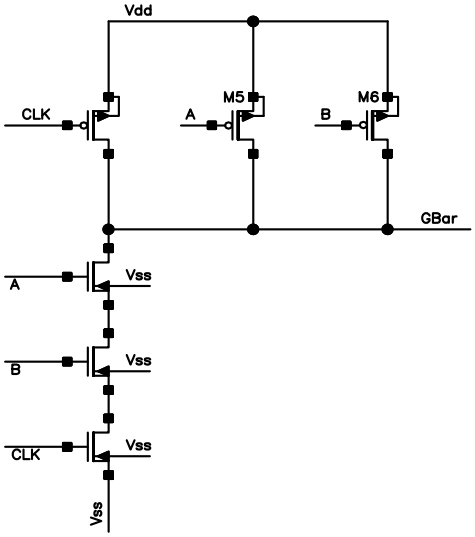


Figure 3.5: Pseudo-static generate circuit.

In Figure 3.1 and Figure 3.3 transistor M0 is the keeper transistor. In the proposed soft error robust adder pseudo-static logic (PSL) is used. The propagate and the generate circuit in this logic is shown in Figure 3.4 and Figure 3.5 respectively.

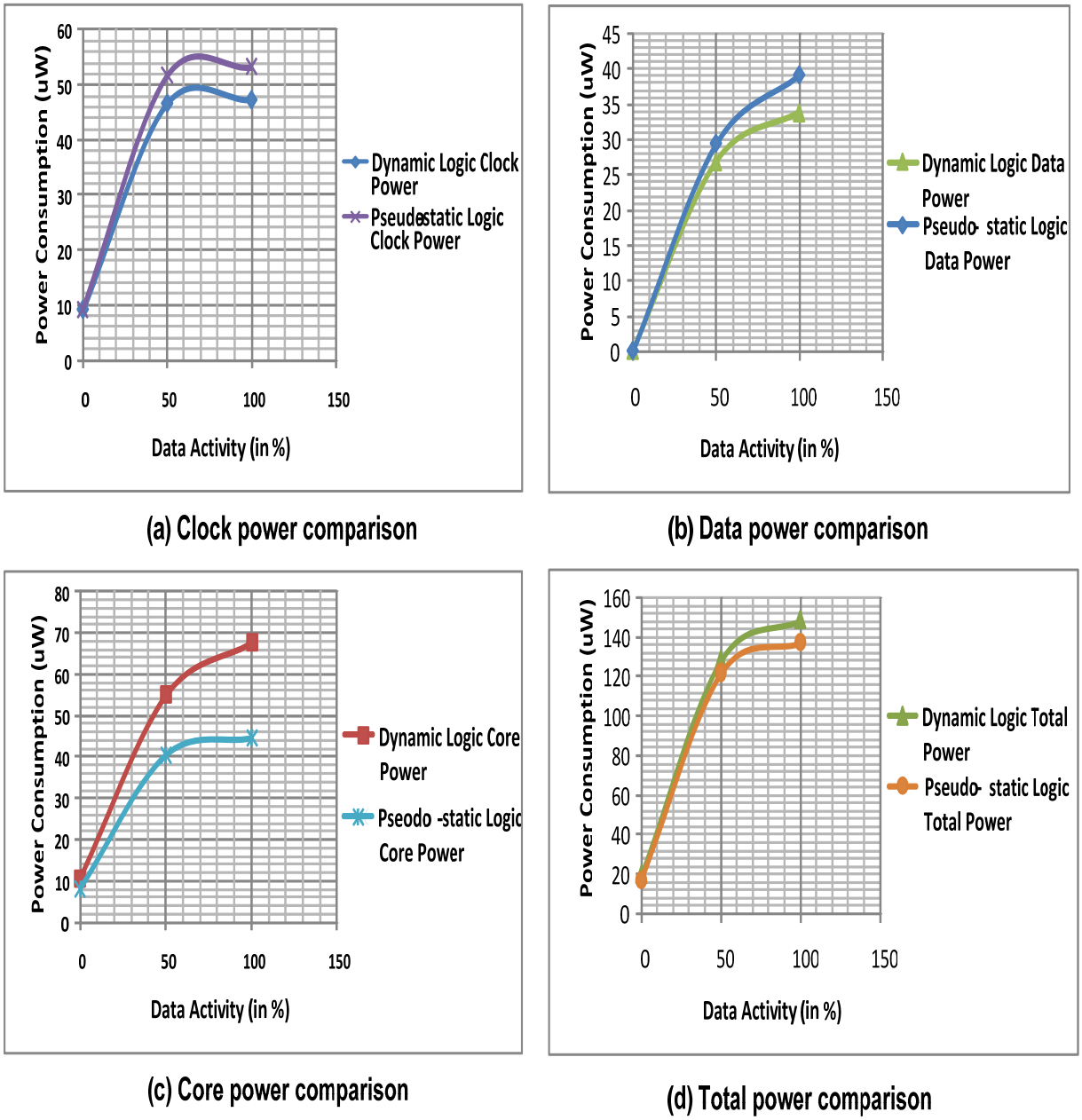


Figure 3.6: Dynamic vs. PSL power comparison.

In pseudo-static logic the keeper transistor is replaced with a static pull up network (PUN) which is a complement of the PDN. The PSL works in the same fashion as the dynamic logic i.e., during a precharge event output capacitance is charged to logic 1 and during the evaluation period, the pull down network is evaluated. The use of a complementary pull up network has two advantages. First, it helps in charge leakage reduction as there is a static path between the output and the power supply in the evaluation window. Second, in an SET event it helps to recover the logic back to the pre SET state. Figure 3.6 shows the clock power, the data power, the core power and the total power comparison for the propagate-generate circuit using dynamic logic and PSL logic. Use of PSL saves a transistor in each propagate stage resulting in energy saving as can be seen in the energy delay comparison plots shown in Figure 3.7.

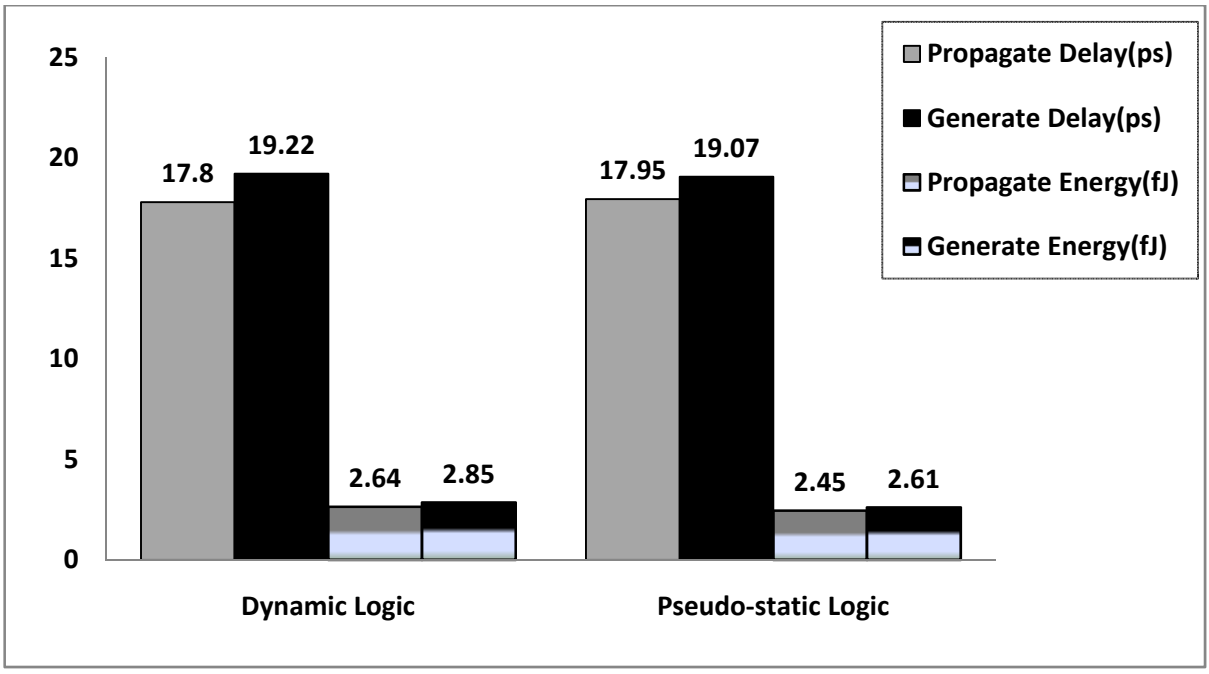


Figure 3.7: Energy delay performance comparison.

In all comparison plots, the difference in power consumed by dynamic logic and PSL increases with increased data activity. The main objective of this research is to develop a soft error robust adder. Hence, some SET simulations are carried out on the propagate generate (PG) block designed in each logic style. The method of performing such simulations will be explained in the next chapter. Again, the results are compared for dynamic logic and PSL. It has been observed that the 0 to 1 transition at

the input of the PG block results in a propagated SET when the input vectors are favorable for such a transition. Figure 3.8(a) shows the output of a PG block implemented using dynamic logic for all possible input combinations of inputs A and B.

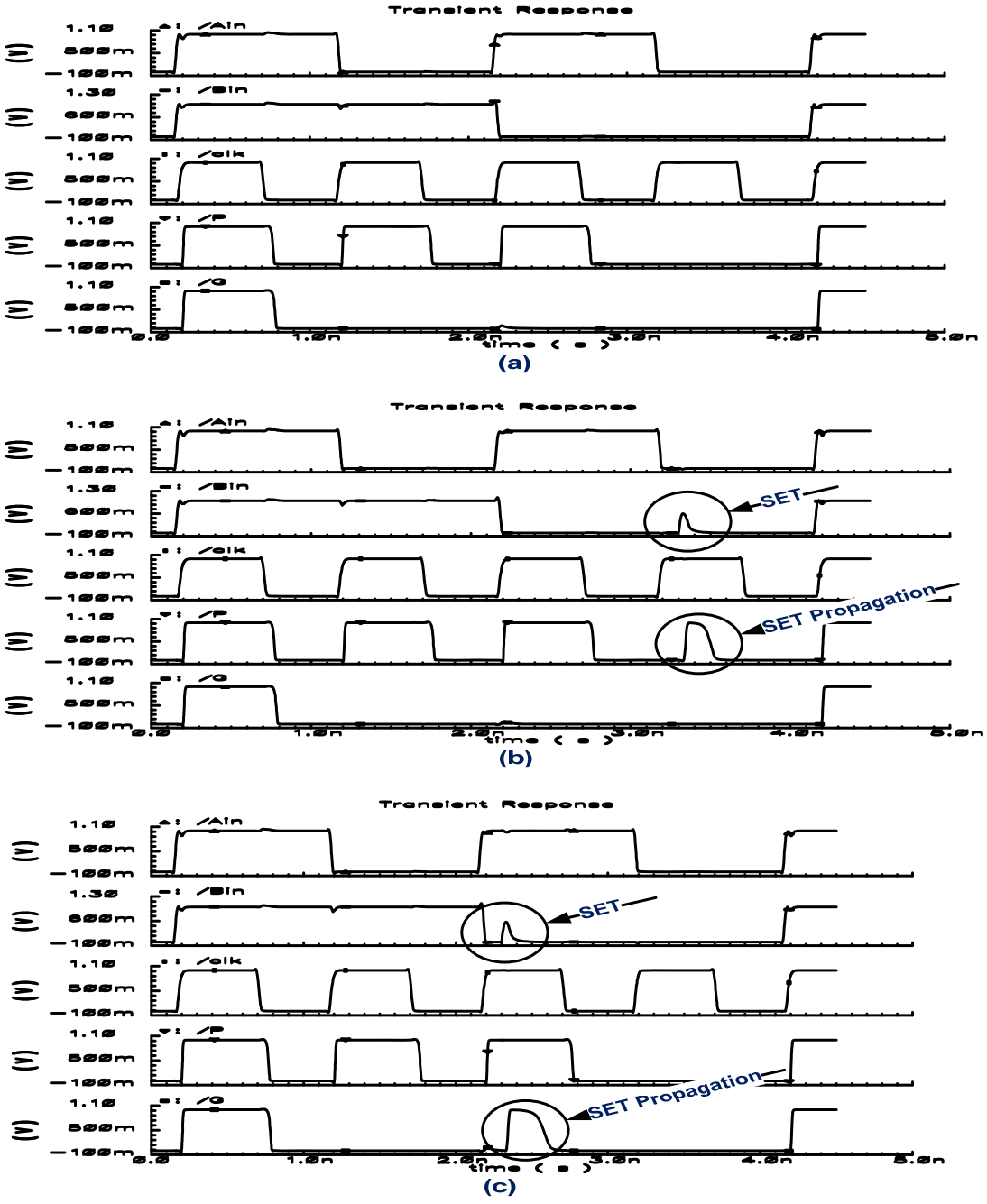


Figure 3.8: SEU (0 to 1) observations in dynamic logic

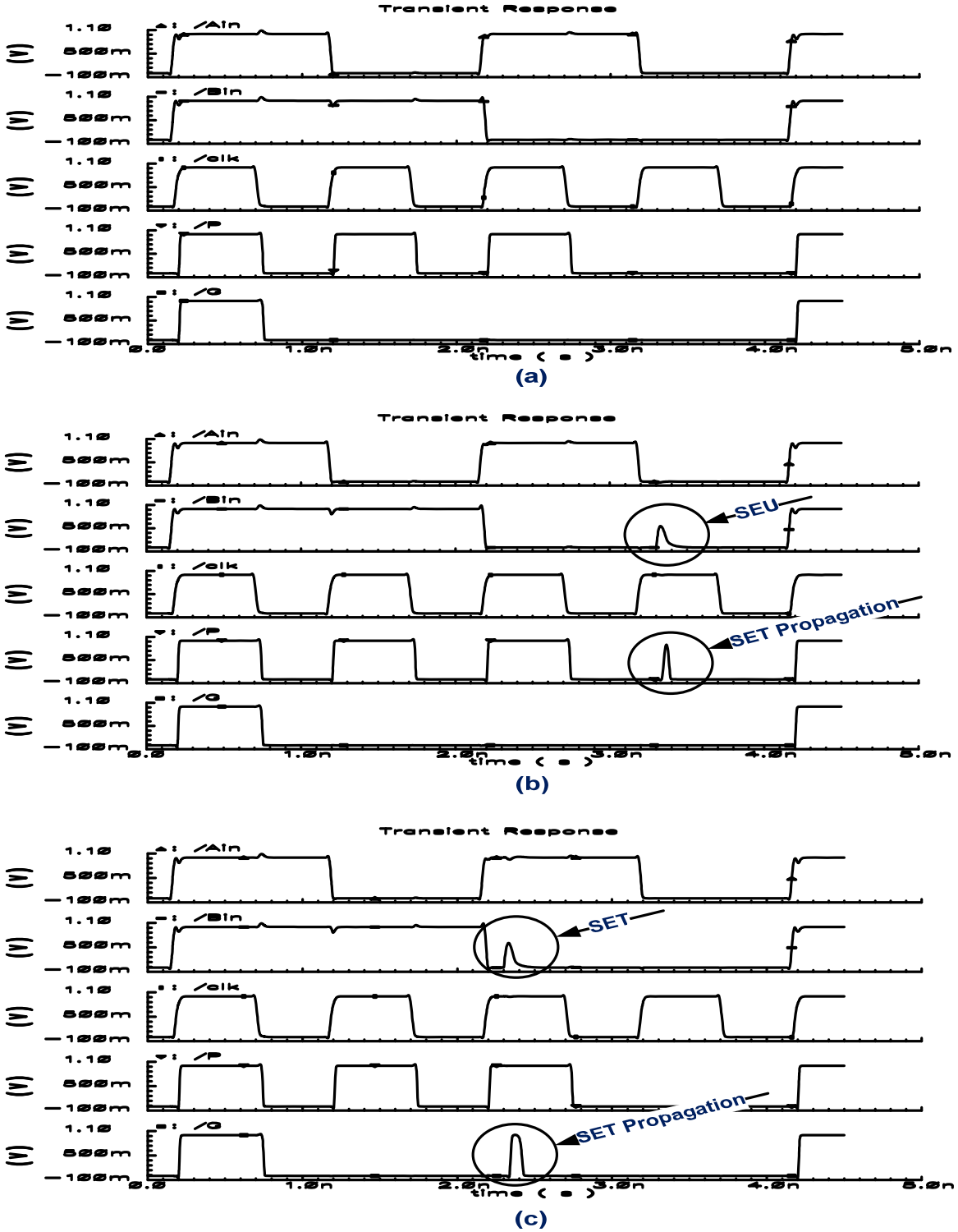


Figure 3.9: SEU (0 to 1) observations in pseudo-static logic

The outputs P and G are governed by equations (2.3) and (2.4). In Figure 3.8(b) an SET occurs at the input B such that it gives rise to a glitch. This glitch gets evaluated by the propagate circuit ensuing into a false propagate signal. The false propagate signal can subsequently lead to a false logic evaluation in further stages of the carry merge tree. When a similar SET occurs at a different time interval, it can result in an SET in the generate block, which is capable of producing a false generate signal (Figure 3.8(c)). The false generate signal can result in a false carry in the carry merge tree. The PSL, when simulated under similar conditions leads to similar results. An important observation is that the duty cycle of the propagated SET is much smaller when compared with dynamic logic. This property of PSL is exploited to design a time redundant adder as explained later. These are the confidence building measures that the adder designed with this logic can have some soft error robust capabilities. Extensive simulations and results at the complete architecture level will be discussed in the next chapter. Thus, the carry merge tree as shown in Figure 3.2 is implemented with PSL logic.

3.2 Transistor Sizing

Sizing allows for fast optimization of the path delay. The delay of a logic gate depends upon two parameters, namely; the parasitic capacitance and the load that the gate will drive, called electrical effort (f), and the logical effort (g) which explains that for a given load, the complex gate has to work harder to produce the same response as that of an inverter. The electrical effort is given by:

$$\textbf{Electrical effort } f = \frac{C_{load}}{C_{in}} \tag{3.1}$$

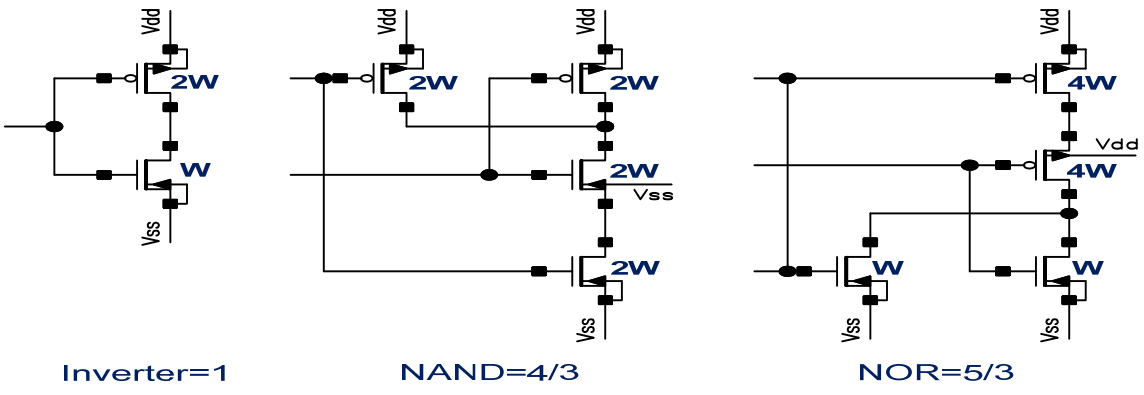


Figure 3.10: Logical effort of basic cells

The electrical effort helps in calculating the load driving capability of the gate while the logical effort explains the ability of the gate to drive current in comparison with an inverter. To calculate g, transistors are sized appropriately so as to have the same resistance as an inverter and then we take the ratio of the input capacitance of each input to that of an inverter. As the gates become complex their logical effort increases. The logical effort of the simple gates is shown in Figure 3.10. This concept can be utilized for optimizing transistors in a chain. The major drawback of logical effort is it doesn't account for slew rate effects and the interconnect delay. Second, the optimizations are geared for high speed while neglecting area and power concerns. Third, the logical effort for dynamic gates can be misleading as output falls faster than the input rises [Sutherland99]. This is the general background for transistor sizing, in this research our focus is on compound domino logic with dynamic gates replaced with PSL.

First, the sizing of PSL will be discussed and then the interface from PSL to static logic and then the interface of static logic back to PSL will be explained. In the PSL (shown in Figure 3.5) pull down path, the transistors are designed to give a unit resistance. If the inputs A and B are 1, during precharge event there will be contention between pFET and nFETs. In the first stage of a carry merge tree, an extra transistor which is controlled by a clock is used to avoid contention between the pull up and the pull down paths. This transistor is called the footer transistor. The precharge transistor is designed to give two unit resistances at the cost of rise time. The charge leakage issue problem is solved by using a PUN as already explained in section 3.1. The sizing of PUN is kept at a unit resistance to avoid contention with PDN.

Static logic requires that the inputs are in a steady logic state until they are sampled. In other words, the outputs from different PSL gates which are inputs into static logic have the same delay. The static logic is designed such that the inverter at the output of the PSL is avoided. This results in true inputs again at the next PSL level. Also, for static to PSL interface the clock is delayed to avoid glitches. It can be explained as: the output of first PSL transitions from 1 to 0, then the output of first a static stage is 0 to 1, at second PSL stage which is footerless if the clock already enters the evaluation phase and its input is 1, there will be a contention.

It is important to mention that when a gate is connected to more than one gate in a chain there is another parameter called branching effort that comes into play. The branching effort is the ratio of the total capacitance of a stage to the capacitance of the path. The logical effort calculation also helps in deciding which logic family should be used in a given design. Domino circuits are better because they avoid static power consumption and the PMOS current issues during evaluation.

3.3 Sum Generation/ Selection Circuit

In an SRA, the sum is calculated in a group of 4-bits. Two additions are performed simultaneously anticipating an incoming carry of 1 and 0 respectively. When the actual carry is available from the carry merge tree a multiplexer selects one of the conditional sums. It is important to mention that the worst delay of the 4-bit sum block should be less than the minimum delay of the carry merge tree. This implies that the conditional sum should be available before the arrival of the fastest carry signal. A very good comparison of different sum circuits has been carried out in [Alioto02]. In the design of the SRA, three different adder structures are considered and their delay is compared. The comparison is shown in Figure 3.11.

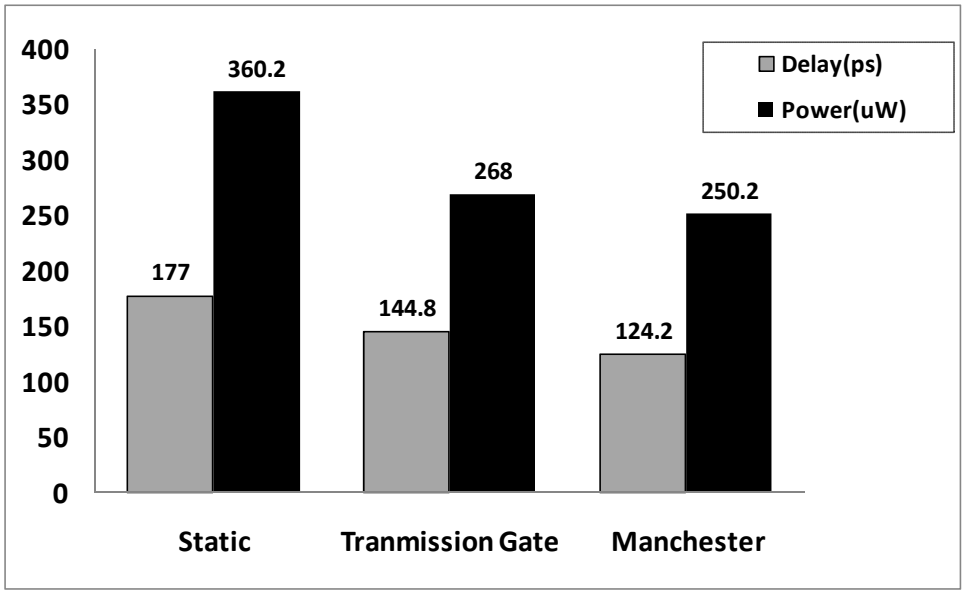


Figure 3.11: Power delay comparison

From the results, it can be concluded that the Manchester adder offers the best power delay performance. These results also agree with [Alioto02]. Hence, the Manchester adder architecture is implemented in the SRA.

3.4 Time redundant samples

One of the ways to achieve fault tolerance in a circuit is by using time redundancy. In time redundancy, the same outputs are computed twice for error detection and computed thrice for error correction. Since the time redundant approach uses the same circuit, the penalty in terms of hardware

is minimal; the cost in terms of delay can be a computation delay. In the design of an SRA, time redundancy is used to detect soft errors. The same output is sampled at two different time intervals and both values are stored in a flip flop. Then the flip flop outputs are compared for parity. A mismatch between the two samples indicates that an SET has affected the circuit. Important aspects of this approach are; one, the timing of first sample, and two, how much delay after the first sample, the second sample is taken. In the case of the SRA the first sample is taken when the worst delay output is available. The second sample is taken 110 ps after the first input. Based on a large number of device simulations carried out with different levels of deposited charge, it is reported by [Walstra05] that nearly all the charge is deposited in the first 10 ps and in a waveform span of 100 ps. Thus, the number 110ps is chosen assuming that any SET event in the adder will not last longer than this duration.

3.5 Clock network

The clock signals in the SRA are used to derive the PSL and the flips flops which store the time redundant samples. The voltage controlled delay elements [Nummer03] are used to generate different clock phases. The delay between the different clock phases is chosen to match the delay of the PSL plus the static stage delay (refer to Figure 3.2). The clock network is designed to derive an equal load at each stage of the carry merge tree. A similar technique is used to design the clock network for the time redundant sampling block (TRSB). The clock network consumes 40% of the total power of the complete adder.

3.6 Flip Flop

The time redundancy implemented in SRA requires that the samples are latched for parity generation. Different latching techniques are considered such as C^2MOS, TSPC, and DFF from the technology library. The power and the delay response of these are considered and the results are presented in Figure 3.12. The C^2MOS based DFF offers the best delay; however, its power consumption is comparable to a TSPC latch. Thus, a C^2MOS flip flop was designed for the TRSB and its schematic is shown in Figure 3.13.

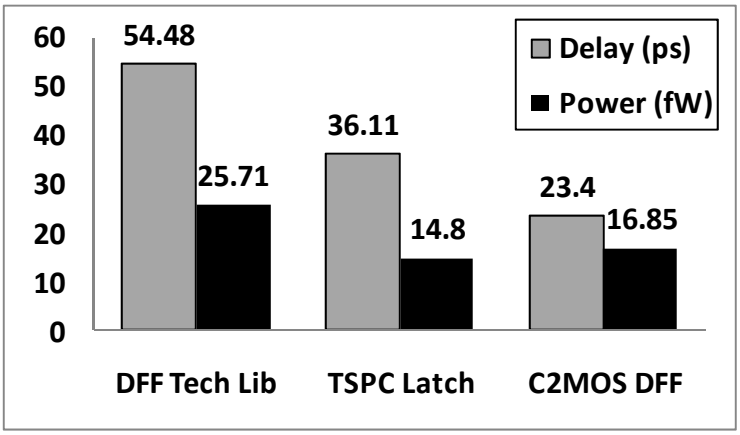


Figure 3.12: Power delay comparison of different flip flops.

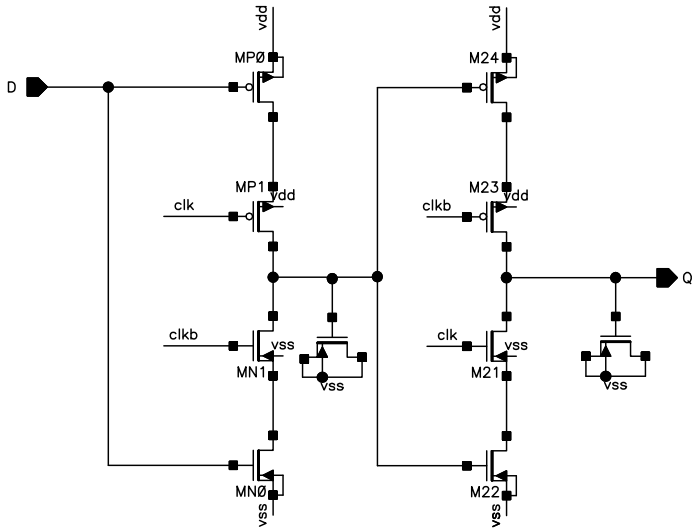


Figure 3.13: C^{2}MOS flip flop.

3.7 Parity Circuit

The parity circuit is an XOR tree which compares the output of TRSB. If two samples taken at different times for the same bit are different, the corresponding XOR gate will have complementary inputs. It will result in parity 1 at the output, indicating an error.

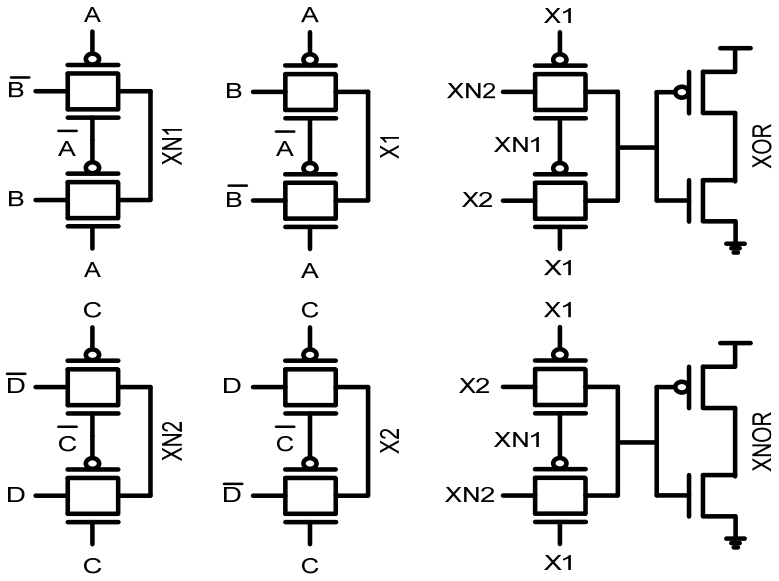


Figure 3.14: 4-input TG XOR

Different XOR design styles are considered and a 4-input transmission gate (TG) [Jahinuzzman08] based design (Figure 3.14) gives the best results for power and delay and is used in this design.

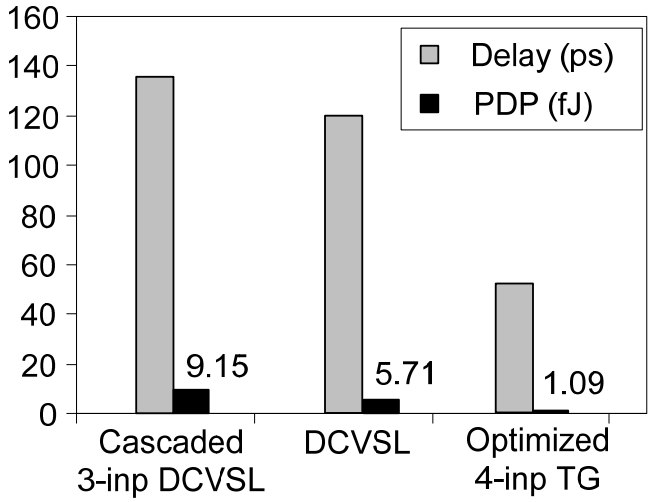


Figure 3.15: PDP for 4-input XOR

The beauty of this circuit is that it provides true and complementary outputs which are a necessity for a multilevel XOR tree. Figure 3.15 shows the power delay performance of a 4-input TG based XOR. A 64-input XOR tree is designed to compare the outputs of the TRS block.

3.8 Summary

In this chapter the proposed SRA design components are discussed. A detailed analysis of its components such as logic family, sum circuit, sampling technique, and parity generation circuit is carried out. The PSL used in the SRA has the ability to recover from an SET event. This characteristic of the SRA is exploited to design a time redundant sampling technique. The same output is sampled twice at two different time intervals and captured in the TRSB. The time interval is chosen such that any SET is captured by either the first sample or the second sample. Afterwards, these samples are compared with an XOR tree. The parity of a 64 bit adder is computed to indicate if an SET has affected the circuit. There are certain cases when this circuit will not be able to detect an error. These cases will be discussed in the next chapter.

Chapter 4

Simulation Results

Based on the design described in Chapter 3, the 64-bit SRA is implemented in ST microelectronics 90nm CMOS technology. The design is simulated for various test cases and results are examined. Power delay analysis is carried out and compared.

4.1 Testing of Soft Error Robust Adder

Testing of the SRA is one of the critical parts of this research. During an ionization event a cylindrical track of electron hole pairs is formed. When the track comes closer to the depletion region, the carriers are collected by the electric field resulting in a current transient at that node. In the literature, a few current pulse (CP) models have been presented [Baumann05b], [Walstra05]. A typical CP resulting from an ionization event is shown in Figure 4.1

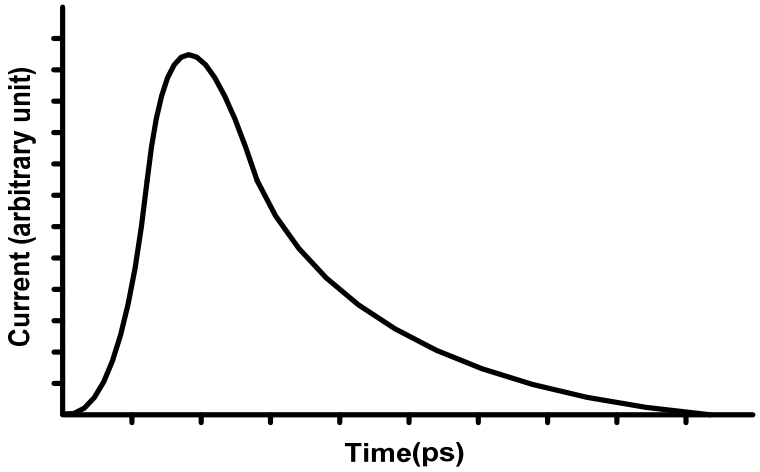


Figure 4.1: Current pulse resulting from ionization event

A similar pulse is used for simulating the SRA using a double-exponential current source. For the purpose of simulation the CP is inserted at different nodes of Figure 3.2. It is then observed at various output blocks, which can be PSL or a static generate or propagate and eventually the final parity is checked to see if the circuit indicates an error.

From the logic perspective, an SET can affect the adder circuit in three ways: one, it can affect the operands; two, it can affect an intermediate carry node (when the carry is available to select conditional sum); three, it affects some intermediate location. It is pragmatic that an SET affecting

any of the operands is more likely to also affect the sum circuit because it will directly affect the conditional sum generation.

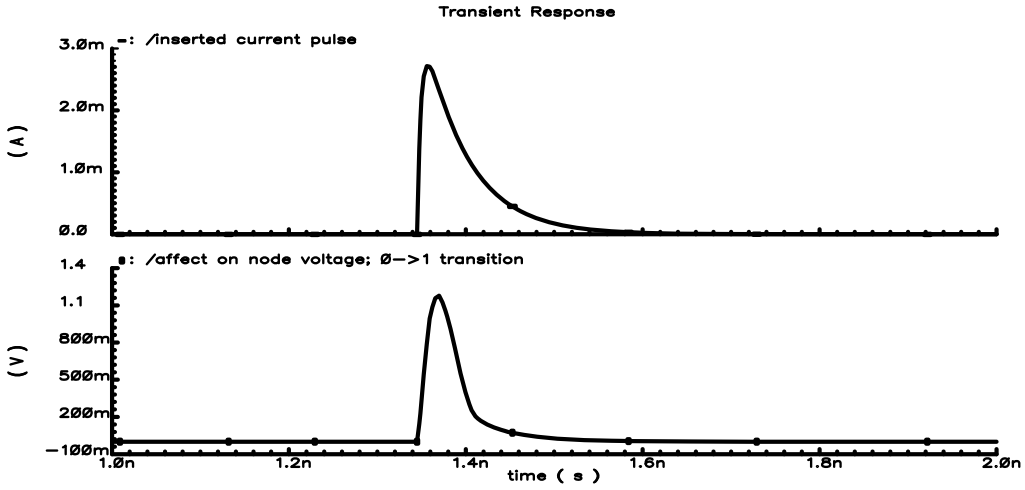


Figure 4.2: Typical current pulse (0 to 1) from simulation environment

If an error occurs at a carry node, it can have two implications. One effect is that the multiplexer can select the wrong sum, if the incoming carry is 0 and an SET gives rise to a spurious 1. At the output multiplexer where this carry will select the conditional sum, a wrong selection will be made for the duration of the SET. When the carry returns to its original state, it will select the correct sum. The second effect is that it can result in a series of transitions in the form of false carry propagation. For a false carry propagation to happen, there are certain conditions that should be present.

The operands affect the carry merge tree and sum circuit. Consider the 64-bit input vector; an SET (0 to 1) can only propagate to the next significant bit if at least one of the operands is logic 1. Now assume the second operand is a 0 and the SET causes it be to be logic 1. The given bit can generate a carry signal provided the SET occurs when the generate signal is being evaluated. The false carry can further affect a number of stages depending upon the logic level of the next significant bits. It is interesting to note that if the next significant bit is logic 0 for both the operands, the error in the carry merge tree cannot propagate farther than this bit.

In another situation, both the operands are logic 0 and an SET causes one of the operands to be logic 1 (Figure 4.2). This will result in a faulty sum for this bit in the form of a glitch in the output. This is the case under the assumption when the incoming carry is 0. However, if the incoming carry is 1, the theoretical sum for this bit would have been logic 1 (0 + 0 +1), but now it will be 0 and further it will generate a false carry.

Now consider the case when both the operands are logic 1. Under normal calculations, the sum output will be logic 0 and it will generate a carry. This is under the assumption that the incoming carry is 0.

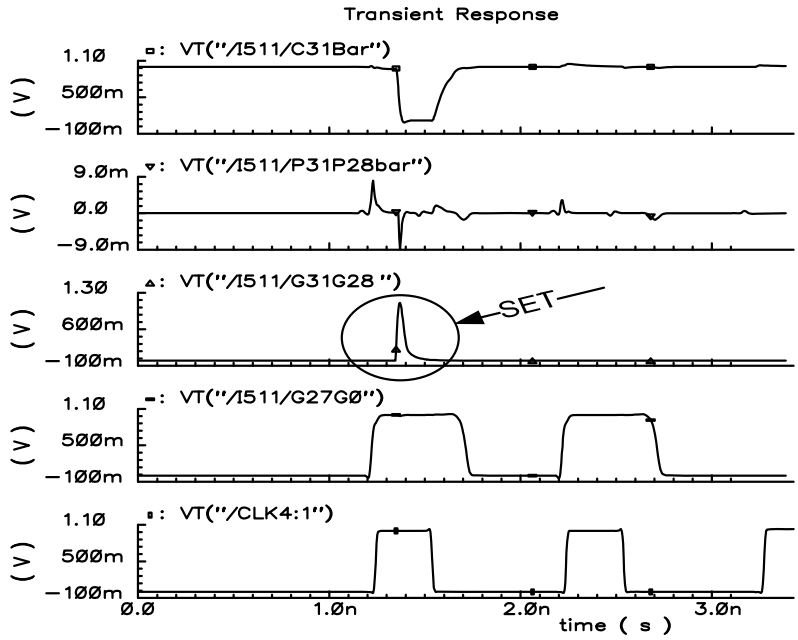


Figure 4.3: False carry generation.

If an SET cause one of the operands to be logic 0, the consequence will be that the sum will have a 0 to 1 glitch and the carry out will be 0. However, with 1 as the incoming carry, the transient will cause a 1 to 0 glitch in the sum, but the carry out will remain unaffected, and in this situation the transient cannot propagate. For the cases when an SET occurs at some intermediate node falls in one of the above categories to result in a propagated SET. In the other cases, it may be covered by logical masking.

The location of the SET is an important factor; however, when the error occurs knowing the location is also vital. In Figure 4.4, C<15> (carry at bit 15) is hit by an SET (1 to 0) and it strikes towards the end of the duty cycle. In this case it results in a duty cycle reduction in S<16> (sum for bit 16). This event is captured by the time redundant samples stored in latches L1<16> and L2<16> and the parity signal, which is not indicated by an "Error" in Figure 4.4. For the same case when SET hit C<15> at a different time, as shown in Figure 4.5, it shows a glitch in S<16>.

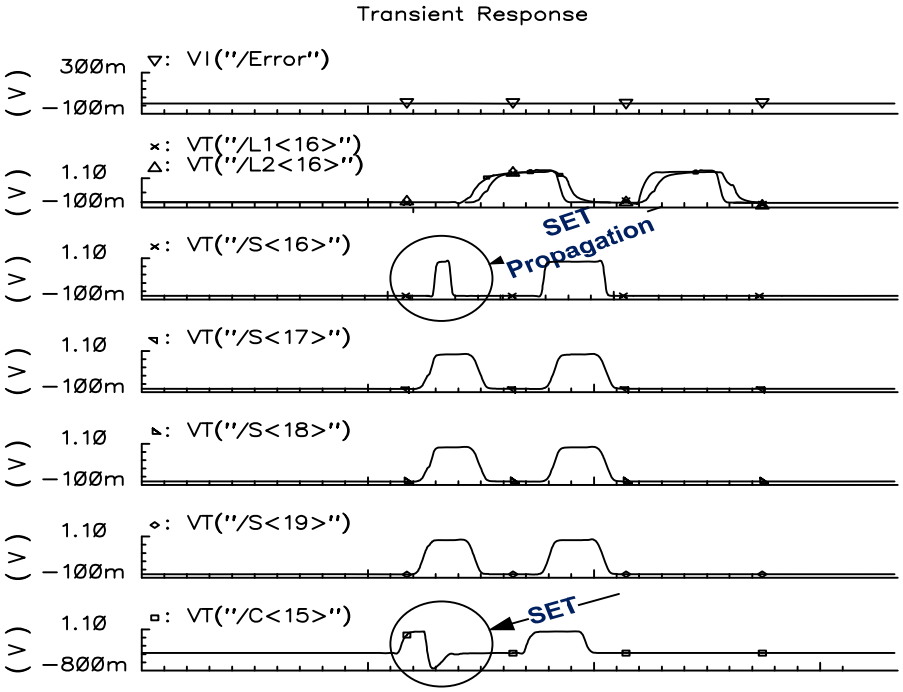


Figure 4.4: SET (1 to 0) affecting C15.

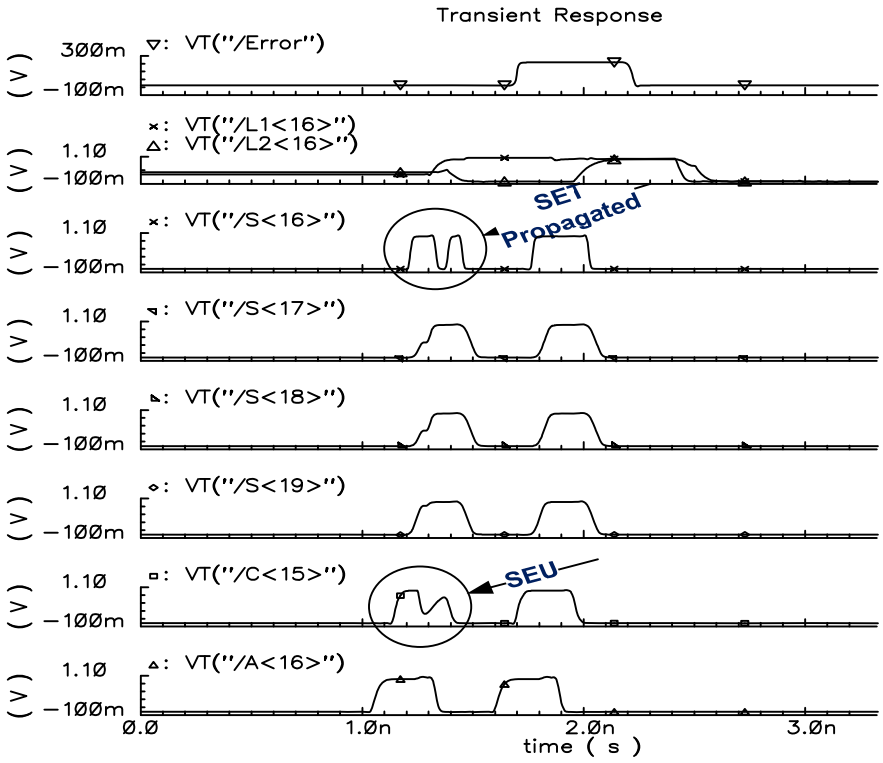


Figure 4.5: SET (1 to 0) affecting C15 at a different time

This event is captured and indicated by an “Error” signal. An alert reader will observe that S<17> and S<18> also see the error transition effect.

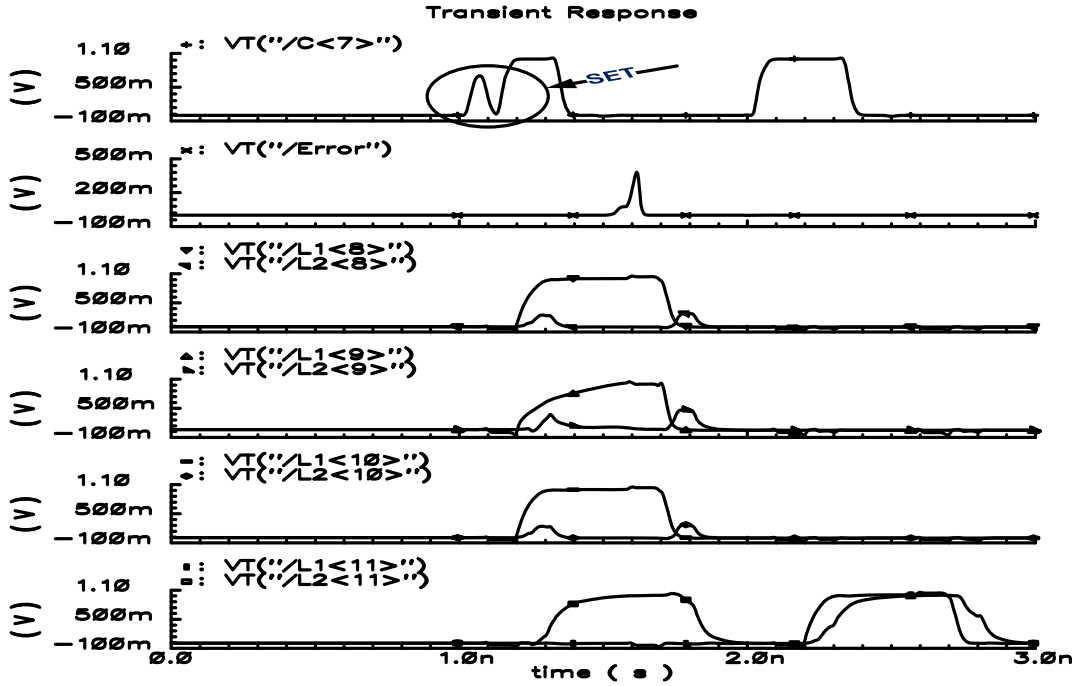


Figure 4.6: SET (1 to 0) affecting C7

Figure 4.6 shows the case when an SET in a carry signal propagates into the sum block. Sum S<8> to S<11> are affected and there are even number of errors hence the parity circuit cannot detect it. More simulation results will be discussed without showing the waveforms. These are the errors that occur in carry propagation path. There are the following observations:

- If the operand is such that bits from <0> to <7> are 1, 0 in any order such that C<7> is 1 and from C<7> to C<56> operand bits are such that it is a carry propagation path and bit <57> onwards there is no specific order for operand. If there is a transient at C<7> it will continue to propagate and result in duty cycle reduction in C<57> and S<57> resulting in an error which is detected.

- If there is an SET at carry C<15> such that it is 0 to 1 now it will affect all the carries till C63 unless there is an explicit kill or generate. This will result in false evaluation of carries and it cannot be detected. The same idea is tested for different depths of carry propagation

and it can be generalized that an error in a carry will propagate all the way to the final carry if the operand is conducive for carry propagation.

Table 4-1: SRA Simulation Results Summary.

Sr. No.	Error Test Case Category	%age
a	Detected	77.46
b	Even Number of Errors	7.04
c	Outside Sampling window	5.63
d	No Error	3.52
e	Glitch	2.82
f	False Carry Generation	2.11
g	Reduced Duty Cycle	1.41

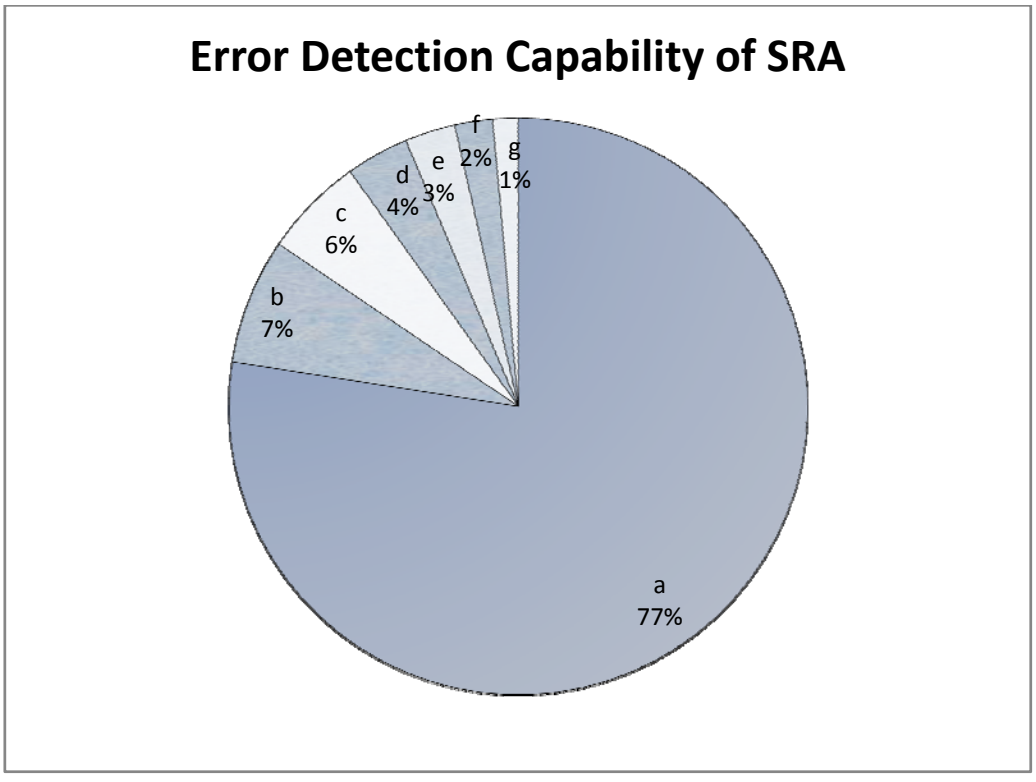


Figure 4.7: Soft error analysis of SRA.

- If it is a carry propagation mode then the carry will affect subsequent carries otherwise it will affect just a single bit which is easier to detect. For example, a G47G44 1 to 0

transition will affect the carries <47>, <51>, <55>, < 59> provided the operand is the worst delay vector. Otherwise, it will be absorbed with a legitimate carry.

- If there is an incoming carry which is affected by an SET, the glitch will propagate. However, if there is a carry generation after the glitch then it will mask the SET with logical masking.

- An error at the input of a static gate is more likely to lead to multiple errors as compared to that of a pseudo-static logic. Because it cannot recover its previous state as it happens in case of pseudo-static logic. This result in a wrong evaluation from the given gate and it cannot be detected in subsequent stages.

- When there is chain of errors it will affect the sum at the last stage as duty cycle reduction (DCR) and it will be an indication that there is an error. For example, if G47G44 has a 1 to 0 or G47G46 has 0 to 1 transition, it will eventually lead to DCR on S<63> , the DCR will be captured by sampling block and will result in a detected error.

- If the error occurs exactly in between the sampling window, then the circuit cannot capture this event and there will be no error indication.

The simulation results are summarized in Table 4-1 and a pie chart of results is presented in Figure 4.7. The proposed design detects 77% of the errors. However, the errors, which are not detected, can be classified into different categories. About 30% of the errors which are not detected are the cases where two bits are affected by an SET. In this case, the parity circuit cannot indicate an error. The 25% of invisible errors are related to the timing of the SET with respect to the signal. The propagated SET is outside the sampling window and hence, remains undetected. In another category, 10% of the unobserved errors are due to false carry generation because of an SET. The rest of the unnoticed errors are: one, SET didn't result in an error (17%), two, and it resulted in a small glitch (13%), and three, it resulted in duty cycle reduction of the output sum signal (5%).

4.2 Power Delay Analysis

The energy delay characteristics of an adder depend upon several factors: technology, circuit family, adder architecture, transistor sizes, and the leakage current. Thus, no single metric can thoroughly compare different adder topologies. The energy delay comparison provides one simple way to compare different designs. In Figure 4.8 an energy delay plot (EDP) for the proposed design is presented. It also includes the same plot from [Mathew07] for comparison purposes. The line with diamonds represents the 64-bit SRA simulations results in 90nm CMOS process.

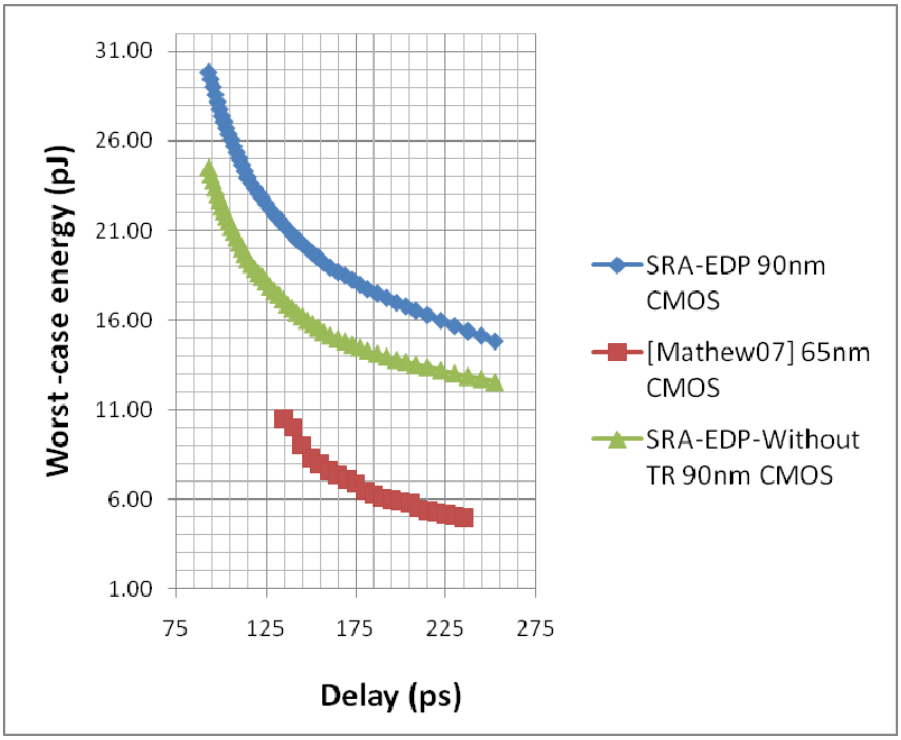


Figure 4.8: Energy delay plot

For the delay of 150 ps the energy consumed is 19.75 pJ. The green line with triangles represents the SRA with soft error detection capability turned off and for the same delay of 150 ps, the energy consumed is 15 pJ. However, when the results are compared with [Mathew07] implemented in 65 nm CMOS with dual rail architecture. For the same delay the power consumed is 8 pJ which is less than half of the proposed design. Considering technology scaling, as we are comparing 65 nm CMOS implementation with 90 nm CMOS implementation we can expect a theoretical reduction of 30% in power. Still the proposed design consumes more energy. However, the author believes, as the two

processes are different, a fair comparison is possible by implementing the SRA in 65nm CMOS process. Another important parameter which can be used to compare the designs is area. The proposed adder implemented with PSL logic will have 50% less area in sum generation as compared to dual rail implementation of [Mathew07]. Also, the proposed adder requires less number of XOR trees to generate parity as compared to multiple parity generation circuits in [Mathew07]. Reduced area of the SRA can be looked upon as less vulnerability to soft errors. A main drawback in the proposed design is that it consumes more power than [Mathew07]. In the SRA, the clock tree is not optimized for power and is contributing 40% of the total power. Hence, there is scope for improvement. A positive aspect can be seen when the SRA is compared with error detection capability (diamonds in Figure 4.8) and without error detection capability (triangles in Figure 4.8). The error detection capability comes at a cost of 24% energy overhead with minimal area overhead.

4.3 Summary

In this chapter, simulation results of the SRA are provided and compared with the literature. The proposed design using pseudo-static logic to implement 64-bit carry merge architecture is presented. A time redundant sampling technique is implemented for the first time to design a soft error robust adder. The SRA operating at 2 GHz frequency can indicate an error of 630 ps from the input. The design works on an idea of time delayed sampling with 66% area saving as compared to the TMR technique proposed in [Townsend03b] and 50% area saving when compared with [Mathew07]. A major drawback of this circuit is its inability to detect an even number of errors. This research suggests that the PSL methodology is best suited for design of soft error robust circuits because of its inherent capability to recover from an SET.

Chapter 5
Conclusion

In this thesis, a major reliability concern in the integrated circuit design, soft errors, is addressed. Based on the background presented in Chapter 1, soft errors in combinational circuits are considered a major area of research. The emphasis is on soft errors in adders since addition is an important part of an arithmetic logic unit (ALU). Briefly, the main findings of this research are outlined.

5.1 Contributions

Various adder architectures are discussed and compared, and existing soft error robust architectures are investigated. Compound domino logic, which is used in contemporary adder designs, is analyzed. Pseudo-static logic is proposed to be an area efficient logic choice for time redundant circuit design. Dynamic and pseudo-static logic are compared in depth for different parameters such as energy, delay, and power. The time redundancy approach for error detection is implemented to design an SRA with 2 GHz speed using 90nm CMOS technology.

The susceptibility of a 64-bit carry merge Sklansky adder to soft errors is evaluated for different operands. Errors are inserted in the inputs, the carry merge tree, and the sum generation circuit. Simulation vectors are carefully chosen so that SET is not masked by inherent logical masking and propagated SET is observed in further stages. Through extensive simulations, it is established that operands which results in carry propagation are more likely to cause multi bit errors (MBU). The time of strike that is, when the transient occurs relative to duty cycle of the signal being affected is observed to be an important metric when simulating such circuits for soft error robustness.

Overall, the proposed adder took half the area for sum generation when compared with an adder from the literature [Mathew07], which employs dual rail design approach for error detection, and is capable of detecting 77% of the errors.

5.2 Future Work

Although, the proposed SRA saves area, the time redundant technique can be investigated and expanded upon for an energy efficient design. In this design, the clock tree was consuming 40% of the total power. An energy efficient clock tree design for adders can be another possibility which can be researched. Techniques have been presented in the literature for soft error detection in memories and logic circuits, the author believes that the soft errors affecting the clock tree of the adder can escape time and space redundant techniques. This can be a possible future work. The SRA and the dual rail logic methodologies only indicate an error at the end of the computation cycle. However, there can be a possibility for analyzing the computation at intermediate stages of the adder to indicate an error. Another desirable feature is to include error correction in adders without carrying out triple modular redundancy.

Glossary

Critical Charge (Q_{CRIT}): The smallest charge that can result in a soft error in an SRAM cell.

Transient Pulse: Transient induced by a single highly ionizing particle in a linear device.

Single Event Transient (SET): Radiation induced perturbation in combinational circuit results in transients called single event transients. If propagated and latched into a memory element, these will lead to a soft error.

Single Event Upset (SEU): Transient induces charge in the voltage of a storage node and causes the data state to change.

Soft Error Rate (SER): The rate at which soft error occurs is called the soft error rate.

Failure in time (FIT): The rate at which the soft errors occur is expressed in terms of FITs which is a measure of the number of failures in 10^9 hours of operation.

Linear Energy Transfer (LET): The magnitude of disturbance an ion causes depends on the linear energy transfer (LET) of that ion is reported in mega electron volt square centimeter per milligram.

Multi Bit Upset (MBU): Large transient causes several nodes to be switched either through charge sharing, parasitic bipolar action, or by depositing $Q > Q_{CRIT}$ at several storage nodes at once.

Single Event Burnout (SEB): Transients induce avalanche breakdown of the junction with thermal runaway causing burn out.

Single Event Latch-up (SELU): Loss of functionality due to a high current state induced by a single highly ionizing particle. SELs produce permanent damage unless a current limitation protects the device. In this case, the cancellation of the high current state requires a power off-on.

Single Event Gate Rupture (SEGR): Destruction of a power MOS due to a conductive path and a high electrical field induced in the gate oxide by a single highly ionizing particle.

Bibliography

[Alioto02] M. Alioto and G. Palumbo, "Analysis and comparison on full adder block in submicron technology," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 10, pp. 806-823, 2002.

[Anghel00] L. Anghel, D. Alexandrescu and M. Nicolaidis, "Evaluation of a soft error tolerance technique based on time and/or space redundancy," *Integrated Circuits and Systems Design, 2000. Proceedings. 13th Symposium on,* pp. 237-242, 2000.

[Baumann01] R.C. Baumann, "Soft errors in advanced semiconductor devices-part I: the three radiation sources," *Device and Materials Reliability, IEEE Transactions on,* vol. 1, pp. 17-22, 2001.

[Baumann05a] R. Baumann, "Soft errors in advanced computer systems," *Design & Test of Computers, IEEE,* vol. 22, pp. 258-266, 2005.

[Baumann05b] R.C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on,* vol. 5, pp. 305-316, 2005.

[Brent82] R. Brent and H. Kung, "A regular layout for parallel adders," *IEEE Trans. Computers,* vol. C-31, no. 3, March 1982, pp. 260-264.

[Dodd03] P.E. Dodd and L.W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *Nuclear Science, IEEE Transactions on,* vol. 50, pp. 583-602, 2003.

[Gaisler97] J. Gaisler, "Evaluation of a 32-bit microprocessor with built-in concurrent error-detection," *Fault-Tolerant Computing, 1997. FTCS-27. Digest of Papers., Twenty-Seventh Annual International Symposium on,* pp. 42-46, 1997.

[Gill06] B.S. Gill, C. Papachristou and F.G. Wolff, "Soft Delay Error Analysis in Logic Circuits," *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings,* vol. 1, pp. 1-6, 2006.

[Han87] T. Han and D. Carlson, "Fast area-efficient VLSI adders," *Proc. IEEE Symp. Computer Arithmetic*, 1987, pp. 49-56.

[Harris03] D. Harris, "A taxonomy of parallel prefix networks," *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on,* vol. 2, pp. 2213-2217 Vol.2, 2003.

[Hazucha00] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *Nuclear Science, IEEE Transactions on,* vol. 47, pp. 2586-2594, 2000.

[Jahinuzzman08] S. Jahinuzzaman, T. Shakir, S. Lubana, J.S. Shah and M. Sachdev, "A multiword based high speed ECC scheme for low-voltage embedded SRAMS," *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European,* pp. 226-229, 2008.

[Johnson88] B.W. Johnson, J.H. Aylor and H.H. Hana, "Efficient use of time and hardware redundancy for concurrent error detection in a 32-bit VLSI adder," *Solid-State Circuits, IEEE Journal of,* vol. 23, pp. 208-215, 1988.

[Kahng03] A.B. Kahng, "Error tolerance," *Design & Test of Computers, IEEE*, vol. 20, pp. 86-87, 2003.

[Karnik04] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in CMOS processes," *Dependable and Secure Computing, IEEE Transactions on,* vol. 1, pp. 128-143, 2004.

[Kaul91] N. Kaul, B.L. Bhuva and S.E. Kerns, "Simulation of SEU transients in CMOS ICs," *Nuclear Science, IEEE Transactions on,* vol. 38, pp. 1514-1520, 1991.

[Knowles01] S. Knowles, "A family of adders," *Proc. IEEE Symp. Computer Arithmetic*, 2001, pp. 277-284.

[Kogge73] P. Kogge and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Computers*, vol. C-22, no. 8, Aug. 1973, pp. 786-793.

[Ladner80] R. Ladner and M. Fischer, "Parallel prefix computation," *J. ACM*, vol. 27, no. 4, Oct. 1980, pp. 831-838.

[Lala01] P.K. Lala and A. Walker, "On-line error detectable carry-free adder design," *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on,* pp. 66-71, 2001.

[Lisboa04] C.A.L. Lisboa and L. Carro, "Arithmetic operators robust to multiple simultaneous upsets," *Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on,* pp. 289-297, 2004.

[Mathew03] S. Mathew, M. Anders, R.K. Krishnamurthy and S. Borkar, "A 4-GHz 130-nm address generation unit with 32-bit sparse-tree adder core," *Solid-State Circuits, IEEE Journal of,* vol. 38, pp. 689-695, 2003.

[Mathew07] S. Mathew, M. Anders, R. Krishnamurthy and S. Borkar, "A 6.5GHz 54mW 64-bit Parity-Checking Adder for 65nm Fault-Tolerant Microprocessor Execution Cores," *VLSI Circuits, 2007 IEEE Symposium on,* pp. 46-47, 2007.

[Mesquita07] E. Mesquita, H. Franck, L. Agostini and J.L. Guntzel, "Soft Error Tolerant Carry-Select Adders Implemented into Altera FPGAs," *Programmable Logic, 2007. SPL '07. 2007 3rd Southern Conference on,* pp. 199-202, 2007.

[Nicolaidis99] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," *VLSI Test Symposium, 1999. Proceedings. 17th IEEE,* pp. 86-94, 1999.

[Nicolaidis03] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 11, pp. 121-128, 2003.

[Nummer03] M. Nummer and M. Sachdev, "DFT for testing high-performance pipelined circuits with slow-speed testers," *Design, Automation and Test in Europe Conference and Exhibition, 2003,* pp. 212-217, 2003.

[Patil07] D. Patil, O. Azizi, M. Horowitz, R. Ho and R. Ananthraman, "Robust Energy-Efficient Adder Topologies," *Computer Arithmetic, 2007. ARITH '07. 18th IEEE Symposium on,* pp. 16-28, 2007.

[Rabaey03] Jan M. Rabaey, A. Chanderkasan, and B. Nikolic *"Digital Integrated Circuits: A Design Perspective"*. Prentice Hall, 2nd Edition, 2003

[Segura04] Jaume Segura, Charles F. Hawkins *"CMOS electronics: how it works, how it fails"*.Wiley-IEEE Press, 2004

[Seifert04] N. Seifert and N. Tam, "Timing vulnerability factors of sequentials," *Device and Materials Reliability, IEEE Transactions on,* vol. 4, pp. 516-522, 2004.

[Semico07] Semico Research Corp. – ASIC IP report; 2007

[Shivakumar02] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger and L. Alvisi, "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,* pp. 389-398, 2002.

[Sklansky60] J. Sklansky "Conditional-sum addition logic," *IRE Trans. Electronic Computers*, vol. EC-9, June 1960, pp. 226-231.

[Sutherland99] I. Sutherland, B. Sproull, and D. Harris, Logical Effort: Designing Fast CMOS Circuits, San Francisco, CA: Morgan Kaufmann, 1999.

[Townsend03a] W.J. Townsend, J.A. Abraham and P.K. Lala, "On-line error detecting constant delay adder," *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE,* pp. 17-22, 2003.

[Townsend03b] W.J. Townsend, J.A. Abraham and E.E. Swartzlander Jr., "Quadruple time redundancy adders [error correcting adder]," *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on,* pp. 250-256, 2003.

[Walstra05] S.V. Walstra and Changhong Dai, "Circuit-level modeling of soft errors in integrated circuits," *Device and Materials Reliability, IEEE Transactions on,* vol. 5, pp. 358-364, 2005.

[Web01] Error Characterization and Modeling Methodologies at TI- Internet resource

[Weste05] Neil H. E. Weste, David Harris *"CMOS VLSI Design: A Circuits and Systems Perspective"*. Pearson Education, 3rd Edition, 2005

[Wijeratne07] S.B. Wijeratne, N. Siddaiah, S.K. Mathew, M.A. Anders, R.K. Krishnamurthy, J. Anderson, M. Ernest and M. Nardin, "A 9-GHz 65-nm Intel Pentium 4 Processor Integer Execution Unit," *Solid-State Circuits, IEEE Journal of,* vol. 42, pp. 26-37, 2007.

[Wilkinson05] J.D. Wilkinson, C. Bounds, T. Brown, B.J. Gerbi and J. Peltier, "Cancer-radiotherapy equipment as a cause of soft errors in electronic equipment," *Device and Materials Reliability, IEEE Transactions on,* vol. 5, pp. 449-451, 2005.