Fault Tolerant Pipelined Multiplier for an In-Order RISC-V Processor

A Project Report

submitted by

THONTA SIRISHA

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

Under the guidance of

Prof V. Kamakoti

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2016

THESIS CERTIFICATE

This is to certify that the thesis titled **Fault Tolerant Pipelined Multiplier for an In-Order RISC-V Processor**, submitted by **THONTA SIRISHA(EE14M067)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof
Dr V. Kamakoti
Professor
Dept. of Computer Science
IIT-Madras, 600 036

Place: Chennai

Date: 28th April 2016

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my guide, Dr. V.Kamakoti for his valuable guidance, encouragement and advice. His immense motivation helped me in making firm commitment towards my project work.

My special thanks to Mr. G.S. Madhusudan,Neel Gala for their encouragement and motivation through out the project. His valuable suggestions and constructive feedback were very helpful in moving ahead with my project work.

I would like to thank my co-guide Dr.Nitin Chandrachoodan and faculty advisor Dr.Deleep R Nair. who have patiently listened, evaluated, and guided us through out the program.

My special thanks to my project team members Arnab,Rahul, for their help and support.

ABSTRACT

KEYWORDS: Fault Tolerance,Reliability,Multiplier,Speed,Area

Silicon reliability has reemerged as a very important problem in digital system design . As voltage and device dimensions shrink, combinational logic is becoming sensitive to temporary errors caused by single event upsets, transistor and interconnect aging and circuit variability. In such cases fault tolerant devices are helpful to have the reliable results.

Multipliers play an important role in today's digital signal processing and various other applications.Compared to addition and subtraction, multiplication operation has more latency. So pipelining the multiplier reduces the overall computation time when more than one multiplication operation is to be performed by the multiplier.Also speed of the multiplier increases.

This thesis aims to design a Fault Tolerant Pipelined Multiplier for an in-order RISC-V processor. The entire signed multiplication operation is divided into 4 stages of pipeline.Fault tolerance is applied to all the four stages present in the pipeline and they are in Triple Modular Redundancy(TMR).Fault handling logic is implemented using both spatial and time redundancy techniques.Time redundancy technique is used to isolate the faulty block if there is any permanent fault.Multiplication is carried out using Wallace Tree architecture and Booth Multiplier algorithm.Reduction of the partial products is done by using carry save addition(3:2 compressor) and 5:2 compressor.Summation of the final summands is done using 4-bit carry look ahead adder,carry select adder and Brent-Kung adder.Making use of all these combinations 12 4-stage pipelined multipliers are designed.Comparision of speed and area is done for all these multipliers.

Contents

List of Figures

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
BSV	Bluespec System Verilog
HDL	Hardware Description Language
FIFO	First In First Out
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Language

NOTATION

r	Radius, m
α	Angle of thesis in degrees
β	Flight path in degrees

Chapter 1

INTRODUCTION

1.1 Overall Microcontroller Architecture

The processor design team of Reconfigurable and Intelligent Systems Engineering[RISE] lab in the computer science department of IIT-Madras has been actively involved in building few processors for academic purposes and other applications. The processor strictly follows the RISC-V instruction set architecture[ISA].Entire design of the processor is done using a Hardware Description Language[HDL] named Bluespec System Verilog[BSV]. The I-Class processor is a 32-bit in-order variant aimed at 50-250MHz micro-controller variants which have an optional memory protection and the design consumes very low power. The integration forms the basis for synchronizing the core to different peripherals in the microcontroller, which have various operating frequencies.My project work involves designing a fault tolerant pipelined multiplier to improve reliability and speed of the multiplier.

Chapter 2

SOURCE OF ERRORS

This chapter discusses the cause of reliability issues faced in the present industry and the source of errors resulting those problems.

2.1 The Hardware Reliability Problem

Computer hardware reliability problems date back to the earliest computers.During the 1940s and 1950s, the vacuum tubes that made up computers had very short life spans, while long wires within the computer often picked up electromagnetic noise [112]. The development of transistors and the evolution of integrated circuits in the following decades led to a dramatic improvement in reliability. Infact, for most applications, once a CMOS IC has been tested, it is expected to operate error-free throughout its life.As CMOS continues to scale, however, it has become less and less reliable.The reduction of power supply voltage and the dimensions of CMOS transistors with every new technology generation reduces the noise immunity and increases the variability of digital circuits. Also, the number of transistors that can be packed on the same chip continues to grow, increasing the number of devices that are susceptible to errors. In fact, today's engineers have to cope with a wide range of errors that could be safely ignored in the past, or only posed a problem for designers of high reliability, availability, and serviceability (RAS) systems.Understanding how these different error categories manifest is critical to devising efficient mitigation techniques. In this chapter we analyze current and emerging sources of errors in deep submicron technologies.

2.2 Error Categories

Hardware errors are split into two main categories :

- Permanent errors

- Temporary errors

Permanent errors, or defects, are irreversible flaws in the manufactured circuit operation, such as stuck-at faults caused by shorts and opens, or design errors such as the infamous Pentium FDIV bug [62]. They manifest every time the circuit is used.Components with permanent errors are not fully operational and have to be replaced.

Temporary errors do not manifest every time the circuit is used. Instead,they occur under certain operational conditions, typically at certain frequencies (f) or power supply voltages (Vdd) or environmental conditions such as temperature (T) and radiation. Their effect on system operation varies depending on their frequency and type, as well as the usage model of the system itself. For example, single event upsets caused by highly energetic neutrons can often be ignored in desktop and laptop computers, because their error rates are small enough to be accepted by most users of these systems. On the other hand, if a chip does not meet timing specifications due to manufacturing process variations, then it is either rejected or labeled as a lower frequency part.

Constantinescu further categorized temporary errors into :

- Intermittent
- Transient

Intermittent errors are caused by an instance of marginal hardware and disappear upon replacement or repair of the faulty part. They tend to occur in bursts and at the same location and are typically activated in a given voltage (Vdd) and temperature range (T) - for example, a part that does not always meet the target frequency due to process variations typically causes intermittent errors.

Transient errors, on the other hand, are caused by external sources rather than marginal hardware, and thus do not disappear upon replacement of the faulty part. They are not bursty, and tend to occur in different places and at random times. Particle strikes are the most typical cause of transient errors.

The harsh environment of space or nuclear plants can introduce a variety of faults in the logic circuits. The radiations and high energetic particles poses causes soft and hard errors in the system and becoming more prominent now a days. This radiation causes two types of failures namely Total Ionising Dose (TID) Effects and single event effects

(SEE) which posing design and test challenges. TID is basically the cumulative long term ionising damage due to electrons and protons deposited in a device. It is measured in Radiation Absorbed Dose (Rads). The device that collects a charged particle for a long time may leads to functional failure.SEE is basically, any measurable effect on the circuit due to an ion strike and these are instantaneous events.SEE basically categorised into soft and hard errors. Both TID and SEE are studied first since they make anomalies in satellite, avionics and nuclear equipments. These effects become one of the most challenging issues that impact the reliability of modern electronic systems used for space applications or even at ground-level applications.

2.2.1 Soft Errors Overview

Radiation-induced soft errors are an increasingly important threat to the reliability of integrated circuits processed in advanced CMOS technologies. Soft error is any measurable or observable change in the state or performance of a microelectronic device, component, subsystem, or system resulting from energetic particle strike. Soft errors are non permanent, random, nonrecurring in nature. When a particle strikes on the transistor then it displaces electrons and holes, thus ionizing a part of the silicon substrate. The displaced electrons and holes begin to recombine and creates a current pulse. The current pulse propagates to other parts of the circuit. When the displaced charge, Qcoll, is more than Qcrit, the pulse is large enough to create a change in state or upset. Qcoll is a function of the ionizing particle's energy, trajectory, point of impact, and the local electric field. The current transient lasts for around 200 picoseconds. Most of the impact is within 2-3 microns of the impact site.

If the effect of soft errors is manifesting up to the system level, it is generally in the form of a sudden malfunctioning of the electronic equipment. Soft errors can not be traced, once memory is updated with new data where corrupted bits were stored earlier. Therefore, failure analysis is not able to conclude that soft errors as the root cause of the problem. Furthermore, the problem is not reproducible or recreated, due to its stochastic or random nature. Therefore,it is usually very difficult to show that soft errors are causing the observed failures.

2.2.2 Soft Errors Classification

Single-event effects (SEEs) are associated with the change of states or transients in a device that energetic external radiation particles induce. Normally, SEEs can be classified into soft and hard errors. Soft errors are non destructive, because resetting or rewriting the device restores normal behaviour thereafter; hard errors are permanent. Soft errors are a subset of single-event effects and can be classified into the following categories:

1. Single-bit upset (SBU) : It is also called Single Event Upset (SEU). The event causes a bit flip in a memory cell or a register due to the particle strike. This event causes temporal loss of device functionality and shall be recovered by rewritten the memory or register.

2. Multiple-bit upset (MBU) : The event causes the upset of two or more bits in the same word of memory or registers.

3. Single-event transient (SET) : The event causes a voltage glitch in the net or wire of the circuit, which propagates and becomes a bit error when captured by a storage element.

4. Multiple-event transient (MET) : The event causes multiple voltage glitches in the nets or wires of the circuit, which propagate and becomes multiple bit errors when captured by storage elements.

5. Single-event functional interrupt (SEFI) : The event causes the lock-up, reset, or other detectable malfunctioning of a component.

2.2.3 Soft Error Specification

Generally, Soft Error Rate (SER) is the rate at which a device or system encounters or is predicted to encounter soft errors. SER is measured in units of Failures In Time (FIT), where 1 FIT denotes one failure per billion device hours (i.e. one failure per 114,115 years). The additive property of FIT makes it convenient for calculations, but mean time to failure (MTTF) is often more intuitive. MTTF is inversely related to FIT and it is not additive. A FIT rate of 1000 is equivalent to MTTF of 114 years. A single bit has fault rate of about 1-10 mFIT then 1 GB memory has fault rate of 109*8*0.01 = 8*107 FIT i.e. an error every 12.5 hours.Researchers expect about an 8 percent increase in soft-error rate per logic state bit each technology generation. Since the number of logic

state bits on a chip is following Moores law, and the aggregate effect on soft-error FIT on a chip is shown in Figure 2.1. Notice that by the 16nm generation, the failure rate will be almost 100 times that at 180nm technology. In an electronic component, the failure rate induced by soft errors can be relatively high compared to other reliability issues.

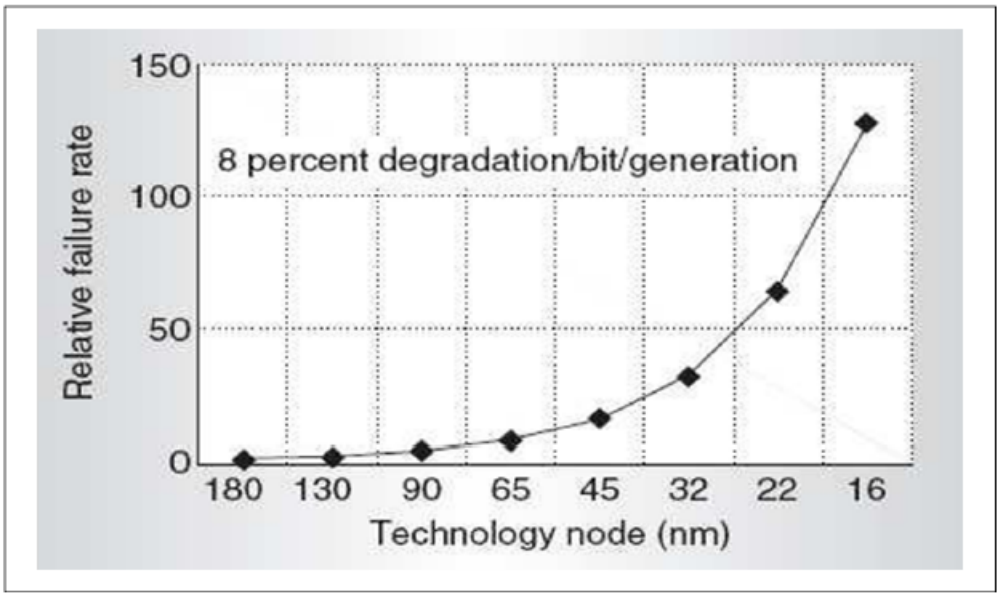


Figure 2.1: Soft error failure-in-time of a chip

2.2.4 Hard Errors Overview

Hard errors are the events which interrupt device function and permanently damage the device. They are destructive in nature and cause permanent damage to the device. Under hard error when a bit can never be read or written reliably, the processor might not be able to correct and recover from the error. Hard errors may be caused by wire wear out due to electro-migration,gate oxide wear-out and ageing defects that increase over time. This cause a open or short mode failure.

2.2.5 Hard Errors Classification

Hard errors are a subset of single-event effects and can be classified into the following categories :

1. Single Event Latch Up (SEL) : An abnormal flow of high-current in a device caused by the passage of a single energetic particle through sensitive regions of the device structure and resulting in the loss of device functionality. It is triggered by heavy ions, protons,neutrons and cause catastrophic failure.

2. Single Event Gate Rupture (SEGR) : An event in which a single energetic-particle strike results in a breakdown and subsequent conducting path through the gate oxide of a MOSFET.An SEGR is caused by an increase in gate leakage current which finally cause either the degradation or the complete failure of the device.

3. Single Event Burn out (SEB) : An event causes direct path between source and drain of the transistor. This cause a local hotspot and finally failure of the device.

2.2.6 Impact of soft errors in sequential and combinational circuits

The circuit of modern processor or other electronic system falls into two basic classes: sequential circuit and combinational circuit. Soft errors in these two circuits have different impact. Thus, different approaches are required to protect the sequential circuit and the combinational circuit.

Errors in Sequential Circuits

The main contribution to the soft error rate (SER) comes from sequential circuits in current microprocessors. Sequential circuits always refer to different storage elements, such as registers,memories, counters and flip-flops in general. A soft error in these circuits may result in a bit flip in the saved state, which may lead to a wrong execution. Storage elements take up a large part of the chip area in modern microprocessors. As a result, most modern microprocessors already incorporate mechanisms for detecting soft errors, like the triple modular redundancy technique.

Errors in Combinational Circuits

A particle that strikes a p-n junction within a combinational circuit may alter the value produced by the circuit. However, a transient change in the combinational circuit will not affect the results of a computation unless it is captured by a sequential circuit.

Transient changes on the clock signal or reset signal will definitely cause the circuit incorrectly executed. Past research has shown that combinational logic is much less susceptible to soft errors than memory elements [11] and the probability of the glitch from the combinational circuit captured by the sequential circuit is very small. With the trends of reduced feature sizes, supply and threshold voltages, soft error tolerance of combinational logic circuits is affected more than memory elements. In addition, higher clock frequencies increase the chance of a glitch being captured by a sequential element [7-12]. For processors where the sequential elements have been protected, combinational logic will quickly become the dominant source of soft errors.

2.3 Error Sources

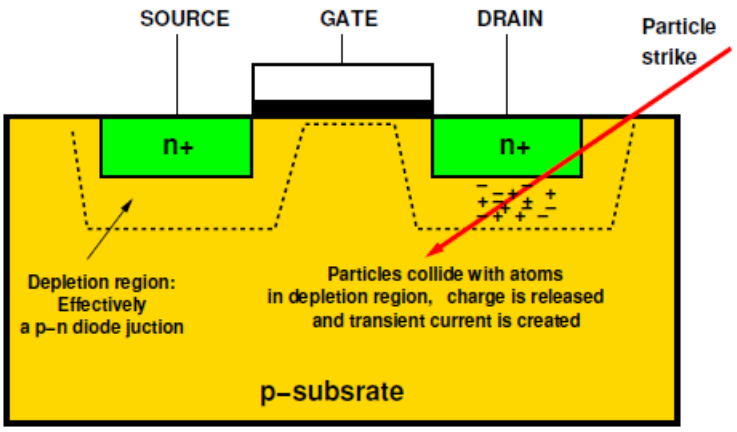
In this section we describe the most important current and emerging sources of errors in deep submicron technologies and focus how they affect combinational logic structures.

2.3.1 Single Event Upsets

Single Event Upsets (SEUs) or soft errors are the most important source for transient errors in CMOS ICs. They are not caused by marginal hardware,but are random in nature and are currently the main reason that fully operational chips fail. The sensitivity of a chip to SEUs is called Soft Error Rate (SER).

Soft errors may occur when the active volume of the chip substrate is hit by highly energetic particles. If these particles penetrate the chip and have sufficient energy (> 1MeV), they can collide with silicon nuclei in the substrate. The area that is most sensitive to particle strikes is the depletion region under the drain of an MOS transistor, as shown in Figure 2.1. The accumulated charge can cause a transient current that momentarily flips the logic state of the transistor drain node. In rare occasions, a single particle strike can cause a bit flip on the drain nodes of two different transistors. Double-flips have been observed in the field but their occurrence rate is two orders of magnitude smaller than single-bit flips.

Particles are generally created by either alpha radiation or cosmic rays.Alpha radiation consists of highly charged helium nuclei, also known as alpha particles.These are



N-type MOS transistor under a particle strike. The cor
p-substrate and the n-drain (source) forms the depletion region o
When the particle strike happens, p-n pairs break and the released
transient current from the substrate to the drain.

Figure 2.2: Cross-sectional view of MOSFET

emitted by radioactive impurities in packaging, or from radioactive contamination of the semiconductor. If the silicon substrate is not exposed to radioactive material, either from within the chip packaging or from external sources,then no alpha particle strikes will occur.

Cosmic rays consist of either solar wind or galactic particles. Solar particles have been the dominant source of errors in satellite electronics. They have very high flux, but their energies are very low and thus they cannot enter the atmosphere. On the other hand, galactic particles have energy in the order of GeV, therefore they penetrate the atmosphere. In high altitudes they collide with atoms in the air and generate secondary particles. These secondary particles keep on colliding with other particles in the atmosphere and gradually lose their energy. Still, many reach the surface of the earth. The most dangerous of them are neutrons. Because they are not charged, they do not lose energy in the electron sea of the atmosphere. Therefore,when they reach the surface of the earth they have enough energy to create collisions with silicon atoms in the chip substrate. Neutrons are responsible for most SEUs inside the earth's atmosphere. Since their flux increases with altitude, SER is larger at high elevation.

SER $\alpha\ F * A * K * exp(Qcrit/Qs)$

where F is the particle flux, A is the area sensitive to particle strikes, K is a constant and Qs is the charge collection efficiency of the device, which is proportional to the gate length.

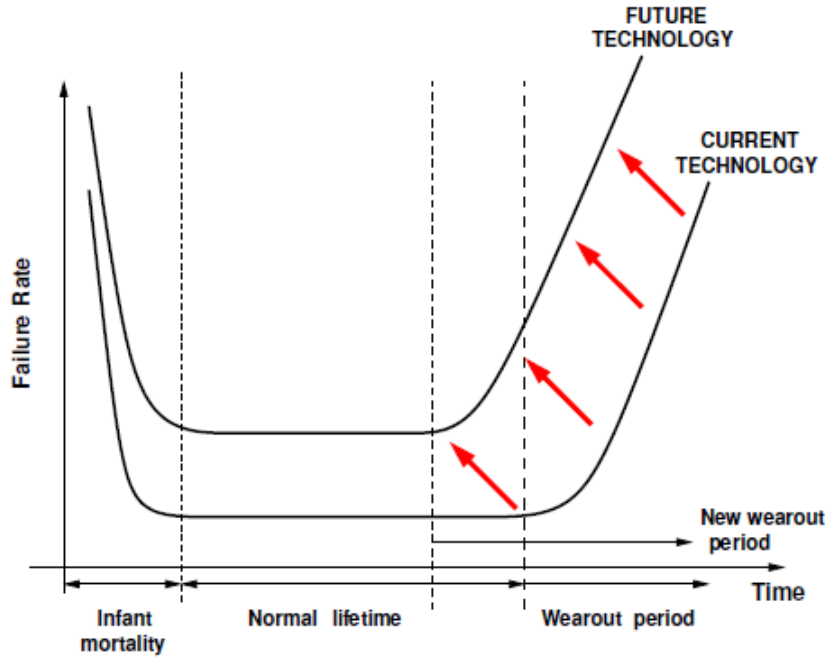


Figure 2.3: Bath Tub Curve

2.3.2 Wearout Errors

Failure rates during the lifetime of a digital IC are modeled with the bathtubshaped curve, shown in Figure 2.3. There is an initial period of high error rates,known as infant mortality period. To avoid such high error rates in the field, all manufactured chips are stressed at high Vdd and T before shipping. This process,called burn-in, accelerates chip aging, thus fast-forwarding marginal chips to infant mortality. During the normal lifetime the error rates are very low. The final period is called wearout, where error rates progressively increase due to device aging.

Technology and voltage scaling have moved the bathtub curve upwards and made it narrower. This means that error rates increase, the projected chip lifetime is shorter and

wearout failures start happening much earlier. The wearout process is gradual. The circuits become progressively slower until they break down. During the onset of wearout errors, slight timing violations creep in. Therefore, they behave as intermittent delay errors, they are caused by marginal hardware, usualy activated for specific ranges of power supply voltage and temperature, typically low Vdd and high T, and only occur for certain data input combinations. Also, since the aging process cannot be stopped, wearout errors are inevitable. Typical wearout mechanisms are electromigration, hot carrier injection, negative-bias temperature instability, and time-dependent dielectric breakdown.

Electromigration (EM) : This happens in the interconnect and is due to high density currents. The currents create electron winds that cause metal atoms to migrate over time, gradually removing metal atoms from wires, thus increasing interconnect resistance. EM eventually results in an open circuit, creating a permanent error.

Hot Carrier Injection (HCI) : This is caused by hot carriers that are injected into the gate dielectric during transistor switching and remain there. This increases the threshold voltage and effectively reduces the transistor switching speed. If a large number of carriers accumulates in the gate dioxide, the device fails permanently. In general HCI affects nMOS transistors more than pMOS transistors because electrons have higher mobility and can more easily penetrate the gate dioxide.

Negative Bias Temperature Instability (NBTI) : This on the other hand, affects predominantly pMOS transistors. When a negative voltage is applied to the gate of a pMOS transistor, a number of Si-H bonds will break at the silicon-oxide interface.This leads to the creation of trapped holes that capture electrons owing in the channel, The threshold voltage increases and the pMOS transistor becomes slower and fails to meet timing constraints.

TDDB : This is the breakdown of the gate dioxide. At the onset, a tunneling current starts owing from the gate to the drain (source) of the transistor. An increase in this leakage current increases the voltage between the gate and the drain (source)and thus reduces the switching speed of the transistor. As more of the gate dioxide breaks down, the current increases until the transistor becomes non-responsive.

Environmental Variations : Environmental (or operational) variations are fluctuations in the temperature and power supply voltage of the chip. Temperature variations are the result of different switching activities and leakage currents in different chip

modules as well as problems in the chip cooling system (e.g. dust in the fan and heat sink).High temperatures can cause permanent failures. One common method to protect against overheating is to place thermal sensors on the die. When a certain temperature is exceeded, dynamic voltage and frequency reduction and microarchitectural throttling techniques are employed to lower power consumption . Also, circuits become slower at high temperatures and are more prone to delay errors.Power supply variations happen due to inductive noise, resistive voltage drops and variations in leakage current . Sophisticated power supply networks are being used to maintain the voltage within a specified margin and avoid potentially harmful spikes.

Current designs strive to eliminate all errors caused by environmental variations and chips always specify a range within which they are guaranteed to operate reliably. If an error occurs in combinational logic due to very high temperature or low voltage, it will be a intermittent delay error. Since gradual temperature increases and small power supply droops only slightly increase circuit delay,temporary errors due to operational variations are expected to only slightly increase cycle time.

2.3.3 Manufacturing Process Variations

Manufacturing process variations are static variations in device and interconnect properties that affect the circuit operation, performance, and power consumption.Again, voltage and transistor/interconnect dimension scaling is the reason for that trend. Transistors and wires are so small that small changes in their geometries or their electrical characteristics have a big effect on their behavior. Geometry fluctuations occur because the wavelength of the lithography process has not scaled with feature size. Therefore the quality of printed patterns on the silicon die continues to deteriorate.In current MOS transistors, variability has been observed in all geometries,such as the effective length (Leff), the width (W) and the oxide thickness (tox). Similarly, the interconnect width, thickness and spacing may also vary. Also, random dopant fluctuations in the channel affect the threshold voltage.

2.3.4 Design Errors

So far we have considered failure mechanisms that were caused by external sources or by marginal hardware, and did not take into account errors caused by the designers themselves. Design errors are practically unavoidable given the complexity of modern microprocessors. After a chip has been taped out, a substantial amount of time is spent in post-silicon debugging, typically 35% of the total design and development time. Only when all functional bugs have been fixed, can the chip be released to the market. A delayed release, or even worse, errors in the shipped parts, often has a huge impact on the profits and the reputation of the manufacturer.

2.3.5 Other Sources of Errors

The above mechanisms are not the only possible causes of temporary errors.In fact, there are many other error sources, such as electrostatic discharge (ESD), electromagnetic interference (EMI), and crosstalk noise , just to name a few. Sometimes, it is not clear whether to classify these errors as transient or intermittent. For example, crosstalk noise may cause a transient spike or make the signal become slower causing intermittent delay errors.Also, moving into smaller dimensions may expose vulnerabilities to noise sources that can currently be safely ignored. Finally, emerging electronic nanotechnologies,such as carbon nanotubes and silicon nanowires have proved to be very error prone. Overall, the importance of the reliability problem is expected to increase in the future.

Chapter 3

FAULT TOLERANCE AND EXISTING MITIGATION TECHNIQUES

3.1 Need for Fault Tolerance

The task of making a processor function correctly according to its design specification spans different stages of the chip life cycle. The design validation stage ensures that there are no implementation mistakes due to which the transistor level implementation of the processor varies from the design specification. The manufacturing testing stage ensures that there are no manufacturing process related defects due to which the physical hardware may not match the transistor level implementation of the processor. After these two stages, the processor is deemed fit to be shipped to the customers. However, the processor still may not function correctly on the field due to many reasons. There may be some implementation mistakes or manufacturing defects which escaped detection from the validation or testing stages. There may be some new defects introduced in the chip after it has been shipped. There may be also some external disturbances which induce incorrect behavior in the processor. Steps need to be taken both at design time and on the field to tolerate such mistakes, defects, and disturbances. The ability of a processor to function correctly in spite of these hindrances is called fault tolerance .

3.2 Fundamental definitions regarding fault tolerance

The three fundamental terms in fault tolerance are

- fault
- error
- failure

A fault is a physical defect, imperfection, or flaw that occurs within some hardware component. It represents a defect in the physical universe such as in semiconductor devices, mechanical elements, displays, printers, power supplies, or other physical entities which make up the system.

An error is the manifestation of a fault. Error is a deviation from accuracy in the information universe such as in the data words within a computer or digital voice or image information. If the error results in the system performing one of its functions incorrectly then a failure has occurred. A failure occurs in the external universe and can be witnessed by the users of the system.

The goal of fault tolerance is to prevent any failures from happening due to faults. Fault tolerance is defined as the ability of a system to continue to perform its tasks after the occurrence of faults. Common terms associated with fault tolerance are fault masking, fault detection, and fault recovery. Fault masking is the process of preventing faults in the system from causing errors. Fault detection is the process of recognizing the presence of a fault and fault recovery is the process of remaining operational in presence of the fault. The fault recovery process generally depends on the type of the fault that is present in the system. The terms error masking, error detection, and error recovery can be similarly defined.

3.2.1 Worst-Case Design Rules

The traditional approach to faults taken by the industry is to rely on error avoidance, rather than on error detection and recovery. The correct operation of the chip is achieved by adopting design rules that guarantee that the probability of an error is below an acceptable upper bound. These Worst-Case design rules have been traditionally used by every digital IC manufacturer. In memory arrays the dimensions and the electrical characteristics of the cells must be within a certain range to make sure that single event upsets are rare. In pipelined combinational logic, after timing analysis is performed and the worst-case delay is found, designers add a guardband, so that the chip is guaranteed to work after manufacturing for a specified range of power supply and temperature.

Though worst-case design has been historically successful, as feature sizes shrink, it becomes far too conservative leading to the investigation of new circuit design tech-

niques to cope with component unreliability and unpredictability.In SER-tolerant designs, silicon on insulator (SOI) and triple-well CMOS processes are being used to reduce the frequency of soft errors by reducing the size of the active region where the particle strikes occur. SOI processes are estimated to be one order of magnitude more resilient to particle strikes and are standard for space applications.

3.2.2 Structural Redundancy

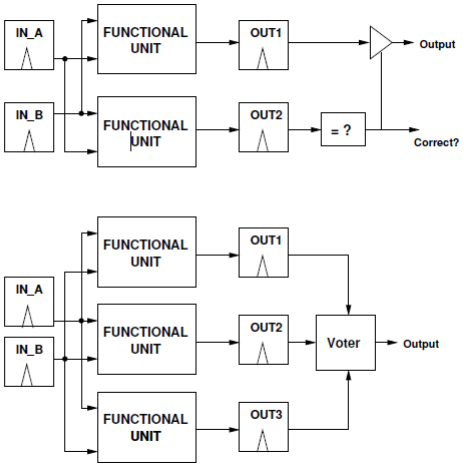


Figure 3.1: Structural Redundancy

Structural, or space redundancy (SR), is a simple form of redundant computation. The computational resources are duplicated and instructions are executed twice. The results of the two functional units are compared. If they are different,at least one of the two redundant FUs has computed the incorrect result. SR is very easy to implement, but its area and power overhead is more than 100%, since the computational logic is replicated and a comparator is placed at the FU output.This overhead may be prohibitive for using SR for large, power-hungry FUs, such as floating point units. SR is more tractable for FUs with multiple instances in the processor pipeline, such as integer ALUs. However, it doubles resource utilization, Finally, if the logic is just duplicated, recovery is impossible because we do not know which one of the two values is correct. To guarantee recovery, triple-modular redundancy (TMR) with a voter is used.FUs enhanced with SR and TMR are shown in Figure.

3.2.3 Temporal Redundancy

Temporal, or time, redundancy (TR), performs two or more redundant computations in a single FU, by executing the same instruction at different times. Such an FU is shown in Figure 2.5. After the two computations have been made, the two output results in latches OUT1 and OUT2 are compared. If the computation is correct, the output is propagated to consumer operations, otherwise an error signal is raised. Redundant execution may be controlled either by hardware [94] or by software [66].

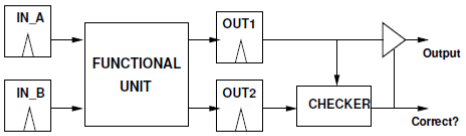


Figure 3.2: Temporal Redundancy

Even though the structural duplication of the computational logic is avoided, TR has the following limitations

- Additional storage and control
- Slowdown
- Power overhead
- Fault Coverage

3.3 Previous work to enhance reliability

Many solutions to the hardware reliability problem have been proposed.There is a wide range of errors to be handled, as we described in Section 2.2, but different levels of the design stack can be combined to provide efficient detection and recovery. We first overview circuit solutions, then focus on microarchitectural and architectural techniques and finish with better than worst-case design methodologies.

3.3.1 Circuit-level Solutions

This technique relies on changes in the circuit design to reduce soft error sensitivity. It involves designing the circuit in such a way that it can combat the radiation effects. This is achieved by adding redundancy at transistor level or circuit level.

3.3.2 Software-level Solutions

Many software implemented hardware fault tolerance (SIHFT) techniques have been proposed in the recent past. These techniques are attractive due to their low cost. Extra code is embedded inside the program which performs the fault tolerant actions.A faulty data structure of the program is examined and repaired to satisfy certain pre-specified constraints.

3.3.3 Microarchitecture and Architecture-level solutions

Architectural error mitigation techniques that have been proposed in the research literature and in industrial products all rely on SR, TR, or a combination of them and use information redundancy for array structures. Computation reliability enhancements fall into the following categories: parity prediction and residue checking, signature checking, lockstepping, redundant multithreading, microarchitecturebased instruction replication, software-based instruction replication, and reconfigurable designs

Parity Prediction and Residue Checking

In modern fault-tolerant microprocessors, error detection relies on two lightweight error detection techniques: parity prediction and modulo residue checking for multipliers/dividers in particular. In parity prediction, the parity of the output is predicted based on the input operand parities and selected FU internal signals.After the output is produced, its parity is computed and compared to the predicted parity. In modulo-p residue checking, a p-bit multiplier (divider) operates on the modulo-p values of the N-bit inputs ($p < N$). The modulo-p output of the multiplier is then compared to the output of the p-bit multiplier. Both these techniques avoid full structural duplication, but they

(i) tend to extend the clock cycle,
(ii)are not equally efficient for all FUs, and
(iii) have lower fault-coverage than plain re-execution.

Signature Checking

Signature checking techniques create a signature of the execution stream by monitoring the executed code and specific signals in the processor. The signature is usually a CRC checksum of the monitored bits , but can also be generated by additional instructions embedded in the code. The run-time signature is compared against a value generated at compile time. The checking is implemented either in hardware or in software and may be triggered at different code granularities, although in most cases it happens at basic block boundaries.

Lockstepping

Lockstepping is a structural redundancy technique to enhance combinational logic reliability. Two modules operating in lockstep receive the same inputs on every cycle and, in the absence of errors, produce the same outputs. The lockstepped system needs to provide a way to do recovery when an error is detected. Lockstepping has been extensively used in high reliability, availability, serviceability (RAS) systems in the past decades.

Chapter 4

DESIGN AND IMPLEMENTATION

Fault tolerance is achieved in computer systems using some sort of redundancy.Redundancy techniques can be categorized as hardware, information, time,and software redundancy techniques.

Hardware Redundancy Hardware redundancy is usually achieved by physically replicating the hardware. Passive hardware redundancy uses fault masking to hide the faults and to prevent the occurrence of errors. Triple modular redundancy is a common passive redundancy technique. This technique uses three copies of the same hardware and

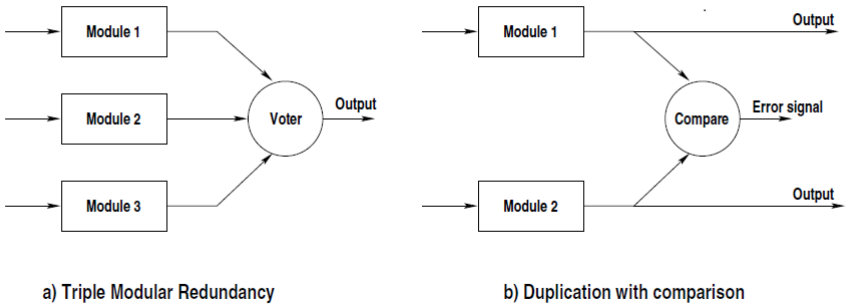


Figure 4.1: Illustration of hardware redundancy

uses a voter (or a set of voters) to select the correct output. No corrective action is required on part of the user for this sort of redundancy. Active hardware redundancy, on the other hand, uses reconfiguration to achieve fault tolerance. Reconfiguration involves detecting the presence of an error and thus deducing the presence of a fault (fault detection), determining where the fault has occurred (fault location), and then remaining operational in spite of the fault (fault recovery). Duplication with comparison is a common active redundancy technique. This technique uses two copies of the same hardware and compares the result to detect an error (and hence the fault). Hybrid hardware redundancy is a combination of passive and active redundancy techniques. Passive redundancy is used to mask the faults and active redundancy is used to locate and replace the faulty hardware.

Information Redundancy Information redundancy is achieved by providing extra information along with data to allow for fault detection and fault recovery. Figure 1.3 shows the scheme used for information redundancy. Data is first encoded before the computation is performed. Error detecting codes and error correcting codes are the

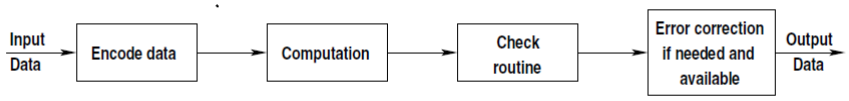


Figure 4.2: Information redundancy

common encoding schemes used. Suppose an error is introduced due to some fault during the computation. If the code is error detecting for the type of error introduced, the result of the computation will be invalidly encoded. This invalid code is detected in the checking routine. If the code is also error correcting for the type of error introduced, enough information is provided in the encoding such that the error can also be corrected. Parity encoding is an example of an error detecting code. An extra bit is added to the data word such that the total number of 1 bits in the word is either odd (odd parity encoding) or even (even parity encoding). An error which results in one of the bits of the word being flipped will result in an even parity (if odd parity encoding is used) or odd parity (if even parity is used). This difference in parity is then detected in the checking routine. The error detecting/correcting capability depends on the type of the error introduced. For example, if even number of bits are flipped in a data word due to an error, parity encoding will be incapable of detecting the error. Other encoding schemes commonly used include m-of-n codes, duplication codes, checksums, cyclic codes, hamming codes, berger codes and arithmetic codes .

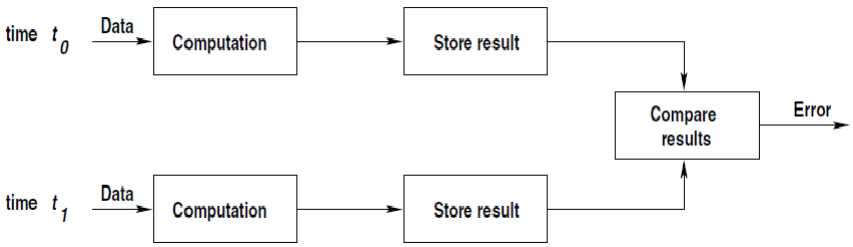


Figure 4.3: Time redundancy

Time Redundancy Time redundancy is usually achieved by running the same computation multiple times on the same computer system and comparing or voting the re-

sults. Figure 4.3 shows an example time redundancy technique which performs error detection.To perform error correction, the same computation needs to be run three or more times and a voter needs to be used to select the correct output. Time redundancy techniques incur lower cost compared to hardware and information redundancy techniques since no modifications/additions are required to the hardware.However, they involve a performance penalty since the same computation is performed multiple times. Transient faults can easily be detected by time redundancy techniques assuming the fault is resident in the system only during one of the computations.A permanent fault cannot be detected by just time redundancy if it corrupts all the computations in the same manner. To detect permanent faults, the computations performed are varied such that a fault affects each of the computation outputs differently.

4.1 Fault Tolerant and Correction System Using Modular Redundancy

Basically, a TMR system is composed of three identical devices and voting logic. The voting logic is the majority voter which takes the majority of inputs to be the output value. Since Device B and Device C are replication of Device A and they all accept the same input value, the output of A, B and C should be consistent in theory. Due to the fault in system, one of these three devices may have an error inside and generate a different output. [4]This inconsistency will be caught and corrected by voting logic .Thus the voted output is always a correct value under the assumption of single error. Thus, the voted output is always a correct value under the assumption of single error. When the TMR concept is applied to a processor (system), all output signal of the CPU are voted; therefore no error should exist at output of voters. Any error that occurs represent that one of the CPUs has an error inside .If that error is not corrected by some way, it may result in more errors and finally become unrecoverable.

Terminology

Let us call the devices performing operations in each stage as Funcional Units.As the design is implemented in TMR,there are 3 FUs in each stage.

4.1.1 Implemented Fault Handling Logic

- For each stage (of all 4 stages)separate error count is assigned for each Functional Unit.
- If there is transient error only in FU1,then error count of FU1 is incremented.The same follows for FU2 and FU3.
- If the transient error occurs for 3 times or when the error count reaches 3,then it signifies a permanent fault happened in the respective FU .The idea of assigning error count for each FU rather than single error count for all the FUs is to uniquely track the permanent fault in a particular FU.
- If there is a permanent fault in a function unit,then we need to isolate the functional unit.This is done by setting the PF to 1 for that FU.SO from the next run this FU do not evaluate the result and is set idle.

4.2 Multiplier

Multiplication is a basic arithmetic operation which is present in many parts of the digital computer especially in signal processing systems such as graphics and computation system. Digital signal processors are designed in such a way that it has a feature of DSP algorithm in real time. The basic building blocks of DSP are multiplier, arithmetic logic unit and multiply and accumulate unit. It requires more hardware resources and processing time than addition and subtraction requires.For real time applications the key issues in the design of computational building blocks of digital signal processor are speed and accuracy.Hence various multiplier architectures have been proposed to increase the performance of the multiplier.

However area and speed are two important conflicting constraints. So improving speed results always in larger areas. The number of gates per chip area keeps on increasing, while the gate switching energy does not decrease at the same rate. So the power dissipation rises and removal of heat becomes difficult and expensive.Here are different multiplier structures which can be classified as Serial Multipliers, Parallel multipliers, Array multipliers, Tree multipliers and so on. Multipliers are categorized in relative to their architecture,applications, and the way of producing partial products and summing up of partial products to produce the final result.

4.3 Some Multiplier Architecures

All multiplication methods share the same basic procedure -

- Generation of partial products
- Addition of partial products
- Final addition

A number of different methods can be used to add the partial products. The simple methods are easy to implement,but the more complex methods are needed to obtain the fastest possible speed.The following are some multiplier architectures.

4.3.1 Iterative

The simplest method of adding a series of partial products is shown in figure below. It is based upon an adder-accumulator, along with a partial product generator and a hard wired shifter.This is relatively slow, because adding N partial products requires N clock cycles. The easiest clocking scheme is to make use of the system clock, if the multiplier is embedded in a larger system. The system clock is normally much slower than the maximum speed at which the simple iterative multiplier can be clocked, so if the delay is to be minimized an expensive and tricky clock multiplier is needed, or the hardware must be self-clocking.

4.3.2 Linear Arrays

A faster version of the basic iterative multiplier adds more than one operand per clock cycle by having multiple adders and partial product generators connected in series.This is the equivalent of "unrolling" the simple iterative method. The degree to which the loop is unrolled determines the number of partial products that can be reduced in each clock cycle, but also increases the hardware requirements. Typically, the loop is unrolled only to the point where the system clock matches the clocking rate of this multiplier. Alternately,the loop can be unrolled completely, producing a completely combinatorial multiplier (a full linear array). When contrasted with the simple iterative scheme, it will match the system clock speed better, making the clocking much simpler.

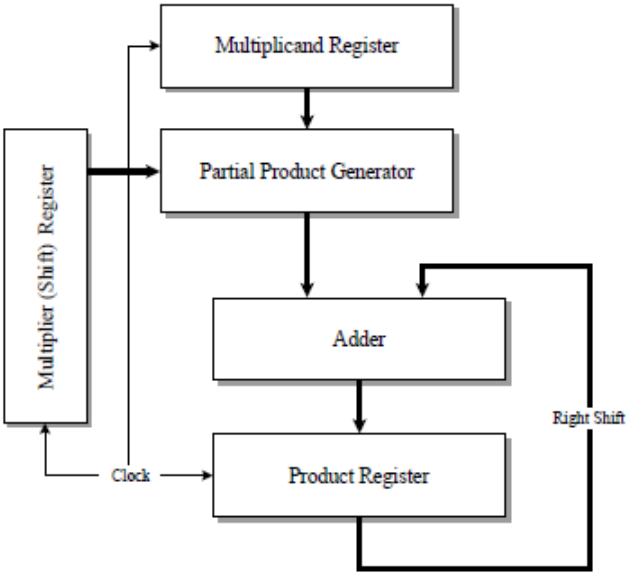


Figure 4.4: Simple iterative multiplier

There is also less overhead associated with clock skew and register delay per partial product reduced.

4.3.3 Serial Multiplier

Where area and power is of utmost importance and delay can be tolerated the serial multiplier is used. This circuit uses one adder to add the m * n partial products. The circuit is shown in the fig. below for m=n=4. Multiplicand and Multiplier inputs have to be arranged in a special manner synchronized with circuit behavior as shown on the figure. The inputs could be presented at different rates depending on the length of the multiplicand and the multiplier. Two clocks are used, one to clock the data and one for the reset. A first order approximation of the delay is O (m,n). With this circuit arrangement the delay is given as D =[(m+1)n + 1] tfa.

As shown the individual PP is formed individually. The addition of the PPs are performed as the intermediate values of PPs addition are stored in the DFF, circulated and added together with the newly formed PP. This approach is not suitable for large values of M or N.

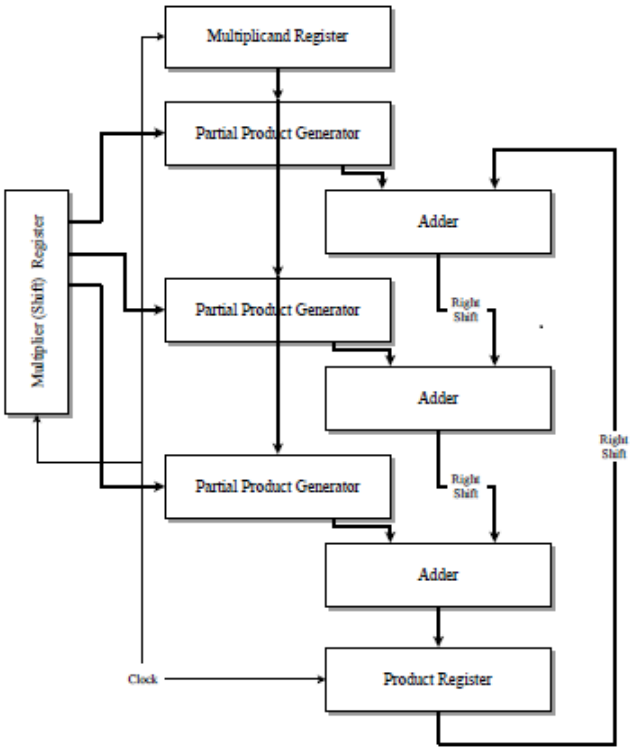


Figure 4.5: Linear Array Multiplier

4.3.4 Parallel Addition (Trees)

When a number of partial products are to be added, the adders need not be connected in series, but instead can be connected to maximize parallelism, as shown in Figure 1.5. This requires nomore hardware than a linear array, but does havemore complex interconnections.The time required to addNpartial products is now proportional to log N, so this can be much faster for larger values of N. On the down side, the extra complexity in the interconnection of the adders may contribute to additional size and delay.

4.3.5 Wallace Tree Multiplier

For real-time signal processing, a high speed and throughput Multipliers-Accumulator (MAC) is always a key to achieve high performance in the digital signal processing system. The main consideration of MAC design is to enhance its speeds. That high speed is achieved through this well-known Wallace tree multiplier. Wallace introduced parallel multiplier architecture to achieve high speed. Wallace Tree algorithm can be used to reduce the number of sequential adding stages. The advantage of high speed

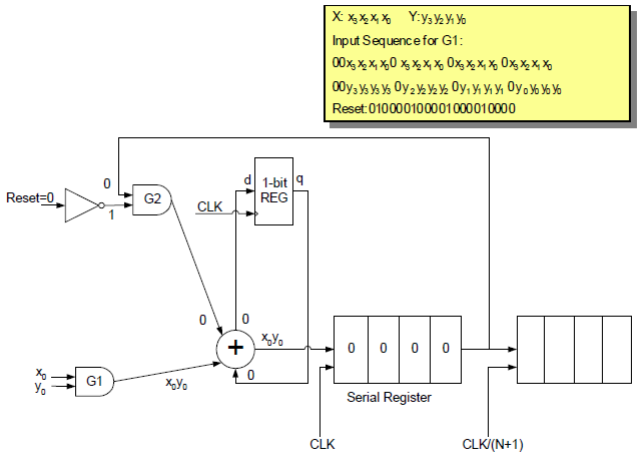


Figure 4.6: Serial Multiplier

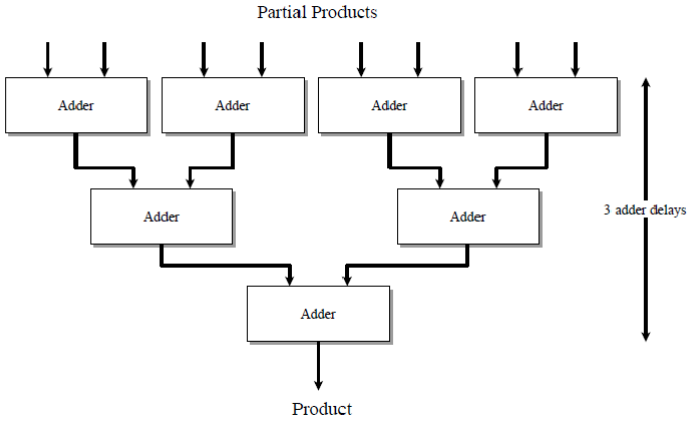


Figure 4.7: Adding 8 partial products in parallel

becomes an enhanced feature for multipliers having operand of greater than 16 bits. The Wallace tree was being constructed using carry save adder(CSAs also known as full adders or 3-2 counters) to reduce an N row bit product matrix to an equivalent two row matrix that is then fed into carry propagating adder to sum up those rows of bits and to produce the product. The carry save adders are those conventional full adders in which carries are not connected and three bits of inputs are taken in and two bits are given as output.Many different adder tree structures have been used to reduce the computation time of the multipliers. The computation time of the Wallace tree has achieved the lower bound of O (log3/2 N). For n-bit Wallace tree multiplier, the number of steps needed is (log3/2(n/2) + 1). Wallace tree multipliers have significant complexity and timing advantages over traditional matrix multipliers. The main advantage of this multiplier is its Logarithmic circuit delay. Delay is log (n).

4.3.6 Modified Booth Algorithm

Booth encoding is a method of reducing the number of partial products required to produce the multiplication result. To achieve high-speed multiplication, algorithms using parallel counters like modified Booth algorithm has been proposed and used. This type of fast multiplier operates much faster than an array multiplier for longer operands because it's time to compute is proportional to the logarithm of the word length of operands. By recoding the numbers that are to be multiplied,Modified Booth multiplier allows for smaller, faster multiplication circuits. The number of partial products is reduced to half, by using the technique of Radix-4 Booth recoding. Reduction in the number of partial products depends upon how many bits are recoded and on the grouping of bits.

Booth Radix-4 Algorithm

The grouping considers each three bits of the multiplier bits starts from the LSB bit and the first considers only two bits. From the next it considers three bits in which one bit will be overlapped on the previous group. Thus grouped multiplier will result in the production of bits between these five bits as follows as -2,-1, 0, +1, and +2.Essentially, three multiplier bits [Y (i+1) ,Y (i) and Y (i-1)] are encoded into eight bits that are used to select multiples of the multiplicand [-2X,-X,0,+X,+2X].

Radix-4 Modified Booth's algorithm [6] is:

- Y (i-1) = 0; Insert 0 on the right side of LSB of multiplier.
- Start grouping each three bits with overlapping from Y (i-1).
- If the number of multiplier bits is odd, add a extra 1 bit on left side of MSB and generate partial product from Table 1.
- When new partial product is generated, each partial product is added two bit left shifting in regular sequence.
- It is then sign extended.

It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly.It is a method that will reduce the number of multiplicand multiples. For a given range of numbers to be represented, a higher representation radix leads to fewer digits. Since a k-bit binary number can be interpreted as

K/2-digit radix-4 number, a K/3-digit radix-8 number, and so on, it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication. This is shown for Radix-4 in the example below.

$X_{i+1}X_iX_{i-1}$	Z_i
0 0 0	0
0 0 1	1*multiplicand
0 1 0	1*multiplicand
0 1 1	2*multiplicand
1 0 0	-2*multiplicand
1 0 1	-1*multiplicand
1 1 0	-1*multiplicand
1 1 1	0

This algorithm is widely used to reduce the area of multiplier and to increase the speed.Here –2*multiplicand is actually the 2s complement of the multiplicand with an equivalent left shift of one bit position. Also, +2 *multiplicand is the multiplicand shifted left one bit position which is equivalent to multiplying by 2.

To enter 2*multiplicand into the adder, an (n+1)-bit adder is required. In this case, the multiplicand is offset one bit to the left to enter into the adder while for the low-order multiplicand position a 0 is added. Each time the partial product is shifted two bit positions to the right and the sign is extended to the left. During each add-shift cycle, different versions of the multiplicand are added to the new partial product depends on the equation derived from the bit-pair recoding table above.

Since the Booth Method applies to 2's complement arithmetic, care must be taken to insure sign extensions are in place. Once the table of the partial products are drawn, all the rows of the partial products have to be arithmetically extended to 2*N, where N is the length of the multiplicand. This is necessary to obtain correct results but it increases the capacitive load, the area and the computational time.

4.4 Adders for Multiplication

Fast carry propagate adders are important to high performance multiplier design in two ways. First, an efficient and fast adder is needed to make any "hard" multiples that are needed in partial product generation. Second, after the partial products have been summed in a redundant form, a carry propagate adder is needed to produce the final nonredundant product. The delay of this final carry propagate sum is a substantial portion of the total delay through the multiplier, so minimizing the adder delay can make a significant contribution to improving the performance of the multiplier.

4.4.1 Carry-lookahead Adder

For a carry-lookahead group of N bits,the transistor implementation has N+1 transistors in the stack.Since wide gates and large stacks display poor performance,the carry-lookahead computation has to be limited to upto two or four bits in practice.Generally we consider four bits.Addition of two 64-bit summands can be achieved by using cascade of 16 4-bit Carry-look ahead adders(CLAs).

4-bit CLA :

In the case of ripple carry adder,the time required to generate the sum is delayed by the arrival time of carrier from the previous full adder stage.As the carry got rippled from lsb to msb,the entire path is present in the critical path.This results in huge delay in a ripple carry adder.Inorder to reduce the delay fast adders like carry-lookahead adders are used.In this all four carries are generated at the same time rather tan depending on the previous carry signal.But this results in more hardware utilisation,hence area increases.

Carry-look ahead adder hardware may be designed as shown in Figure below. The carry-look ahead logic consists of two logic levels, AND gates followed by an OR gate, for each ci when the adder inputs are loaded in parallel, all gi and pi will be generated at the same time. The carry-look ahead logic allows carry for each bit to be computed independently.

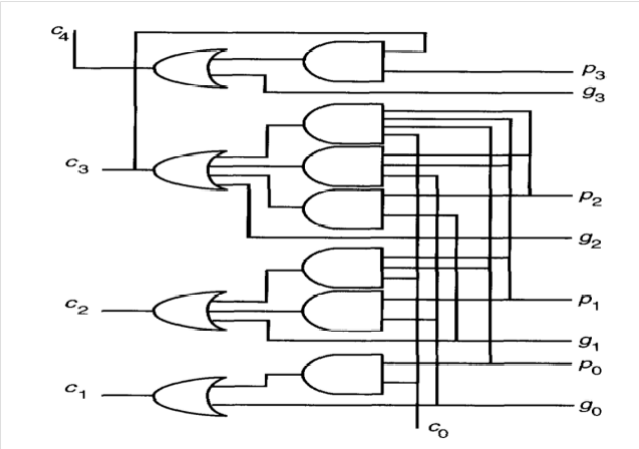


Figure 4.8: 4-bit Carry-lookahead Adder Circuit

Ideally, the carry signal ci will be produced through two-stage logic at the same time, which means that the adder will have a constant time complexity. However, it is impractical to build a two stage full large-size carry-look ahead adder because of the practical limitations on fan-in and fan-out, irregular structure, and long wires delay.

In practice two approaches the block carry-look ahead adder and the complete carry-look ahead adder are used to implement the CLA[1]. In the first implementation, small (4-bit or 8-bit) carry-look ahead logic cells with sections generate and propagate functions are built, and then they are stacked to build larger carry-look ahead adders.

The total delay of the carry-look ahead adder is O (log k) which can be significantly less than the carry chain adder. There is a penalty paid for this gain in term increased area. The carry- look ahead adders require O (k * log k) area.

Let

c_0 *be the carry* $-$ *in of the* $4-$ *bit CLA block.*

g_i, p_i *are the generate and propagate signals of the bit position* i.

Then the 4 *carry bits are derived as follows* :

$c_1 = g_0 + p_0 c_0$

$c_2 = g_1 + p_1(g_0 + p_0 c_0)$

$c_3 = g_2 + p_2(g_1 + p_1(g_0 + p_0 c_0))$

$c_4 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 c_0)))$

4.4.2 Carry Select Adder

The carry-select adder comes in the category of conditional sum adder [9]. Conditional sum adder works on some condition this scheme; blocks of bits are added in two ways: assuming an incoming carry of 0 or 1, with the correct outputs selected later as the block's true carry-in becomes known.

In a ripple-carry adder,every full-adder cell has to wait for the incoming carry before an outgoing carry can be generated.One way to get around this linear dependency is to anticipate both possible values of the carry input and evaluate the result for both possibilities in advance.Once the real value of the incoming carry is known,the correct result is easily selected with a simple multiplexer stage.

An implementation of this idea is shown in the figure below.Consider the block of adders,which is adding bits k to k+3 .Instead of waiting in the arrival of the output carry of bit k-1,both the 0 and 1 possibilities are analyzed.From a circuit point of view,this means that two carry paths are implemented.When Co,k-1 finally settles ,either the result of the 0 or the 1 path is selected by the multiplexer,which can be performed with a minimal delay.The hardware overhead of the carry-select adder is restricted to an additional carry path and a multiplexer,and equals about 30percent with respect to a ripple-carry structure.

A full carry-select adder is now constructed by chaining a number of equal-length adder stages in the figure below.The first-order model of the worst case propagation delay of the module is derived as,

tadd = tsetup + Mtcarry + (N/M)tmux + tsum

where tsetup,tsum and tmux are fixed delays and N and M represent the total number of bits,and the number of bits per stage,respectively.tcarry is the delay of the carry through a single full-adder cell.The carry delay through a single block is proportional to the length of that stage or equals Mtcarry.

We can design a carry select adder using unequal-length adder stages as shown in the figure below.Consider the multiplexer gate in the last adder stage.The inputs to this multiplexer are the two carry chains of the block and the block mutiplexer signal from the previous stage.A major mismatch between the arrival times of the signals can be

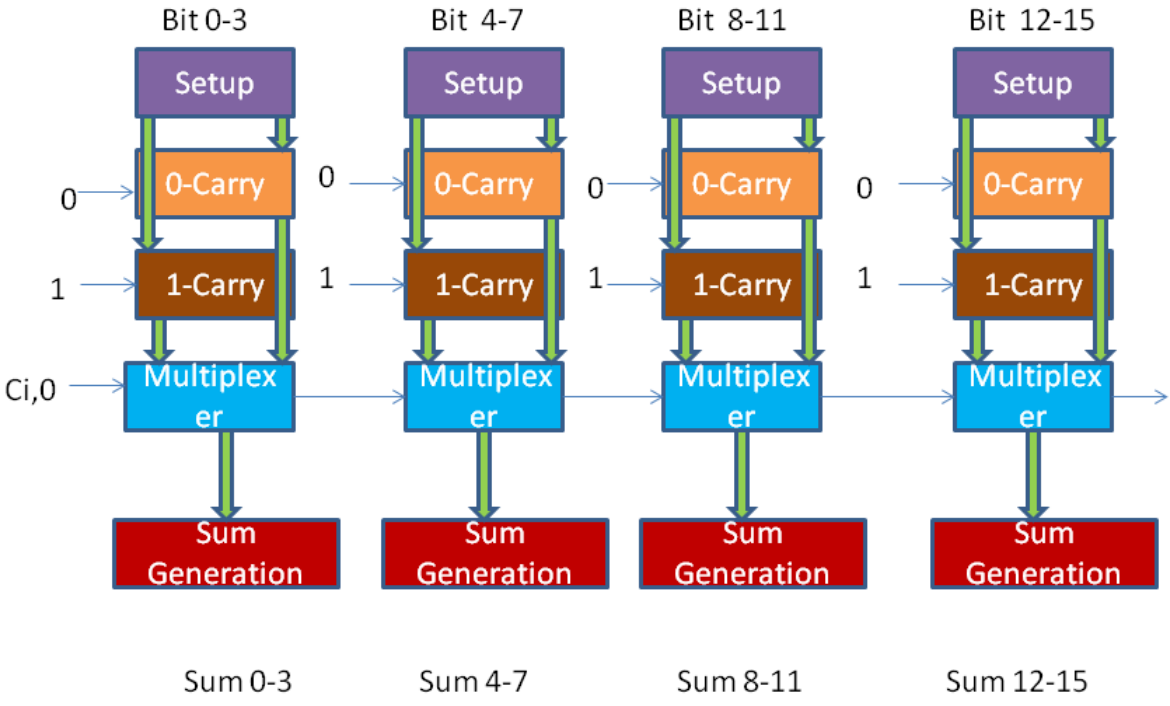


Figure 4.9: 16-bit,linear carry-select adder

observed.The results of the carry chains are stable long before the multiplexer signal arrives.It makes sense to equalise the delay through both paths.This can be achieved by progressively adding more bits to the subsequent stages in the adder,requiring more time for the generation of carry signals.

In the figure below,I implemented this by adding 4-bits in the first stage,8-bits in the second stage,12-bits in the third stage,16-bits in the fourth stage and 24-bits in final stage.

In the above figure there are 5 stages.First stage contains one 4-bit CLA, second stage contains 2 4-bit CLAs in cascade,third stage contains 3 4-bit CLAs,fourth stage contains 4 4-bit CLAs and fifth stage contains 6 4-bit CLAs. The number CLAs in each stage is designed based on the arrival times of the signals to the multiplexers present at each stage.

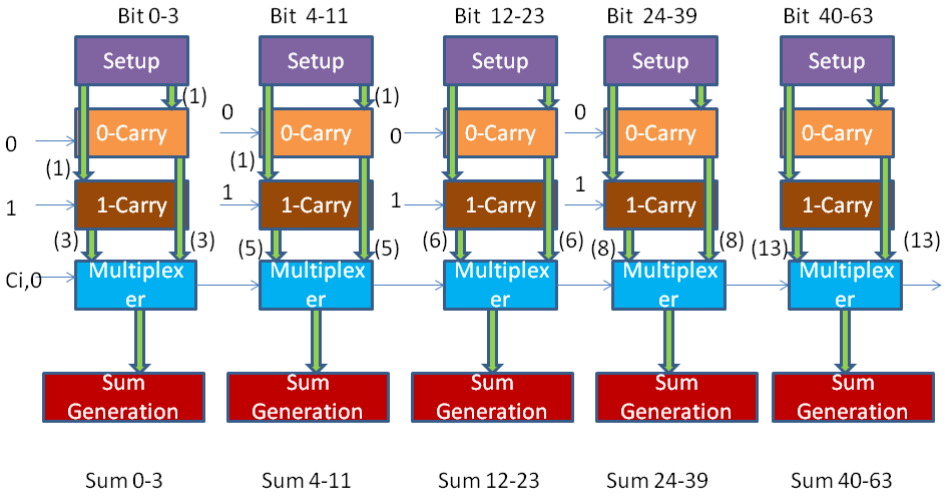


Figure 4.10: 64-bit carry-select adder

4.4.3 Brent Kung Adder

The Brent-Kung adder is a parallel prefix adder.It was developed by Brent and Kung.Parallel prefix adders are fast adders used for high performance arithmetic structures in industries.Parallel prefix additionis done in three steps :

- Pre-processing stage
- Carry generation network
- Post processing stage

1. Pre-processing stage

In this stage we compute,the generate and propagate signals which are used to generate carry input of each adder.A and B are inputs .The following are the equations for propagate and generate:

$P_i = A_i \oplus B_i$

$G_i = A_i.B_i$

2. Carry generate network

In this stage we compute carries corresponding to each bit.Execution is done in parallel form.Carry operator contains two AND gates and one OR gate.It uses propagate and generate as intermediate signals which are given by the following equations:

$P_{(i:k)} = P_{(i:j)}.P_{(j-1:k)}$

$G_{(i:k)} = G_{(i:j)} + (G_{(j-1:k)}.P_{(i:j)})$

3. Post processing stage

This is the final stage to compute the summation of input bits.Sum bit equation is given by :

$S_i = P_i \oplus C_i$

Suppose if we assume T be the unit gate delay,then the time required to compute the sum of 64-bits using Brent-kung adder is shown below :

T = 1 : All $G_i s \; and \; P_i s \; are \; generated$

T = 2 : $X_{63:62}, X_{61:60}, X_{59:58}, X_{57:56}, X_{55:54}, X_{53:52}, X_{51:50}, X_{49:48}, X_{47:46}, X_{45:44}, X_{43:42},$

$X_{41:40}, X_{39:38}, X_{37:36}, X_{35:34}, X_{33:32}, X_{31:30}, X_{29:28}, X_{27:26}, X_{25:24}, X_{23:22}, X_{21:20}, X_{19:18},$

$X_{17:16}, X_{15:14}, X_{13:12}, X_{11:10}, X_{9:8}, X_{7:6}, X_{5:4}, X_{3:2}, X_{1:0}, C_0$

T = 3 : $X_{63:60}, X_{59:56}, X_{55:52}, X_{51:48}, X_{47:44}, X_{43:40}, X_{39:36}, X_{35:32}, X_{31:28}, X_{27:24}, X_{23:20},$

$X_{19:16}, X_{15:12}, X_{11:8}, X_{7:4}, X_{3:0}, C_1$

T = 4 : $X_{63:56}, X_{55:48}, X_{47:40}, X_{39:32}, X_{31:24}, X_{23:16}, X_{15:8}, X_{7:0}, C_3, C_2$

T = 5 : $X_{63:48}, X_{47:32}, X_{31:16}, X_{15:0}, C_4, C_5, C_7$

T = 6 : $X_{63:32}, X_{31:0}, C_6, C_{11}, C_{15}, C_9, C_8$

T = 7 : $X_{63:0}, C_{31}, C_{10}, C_{12}, C_{13}, C_{16}, C_{17}, C_{19}, C_{23}$

T = 8 : $C_{63}, C_{14}, C_{18}, C_{20}, C_{21}, C_{24}, C_{25}, C_{27}, C_{32}, C_{33}, C_{39}, C_{47}$

T = 9 : $C_{22}, C_{26}, C_{28}, C_{29}, C_{34}, C_{40}, C_{41}, C_{43}, C_{48}, C_{49}, C_{51}, C_{55}, C_{36}, C_{37}$

T = 10 : $C_{30}, C_{42}, C_{44}, C_{45}, C_{50}, C_{52}, C_{53}, C_{56}, C_{57}, C_{59}, C_{38}$

T = 11 : C$_{46}, C_{54}, C_{58}, C_{60}, C_{61}$

T = 12 : C$_{62}$

T = 13 : SUM

Where

$\mathrm{X}_{i:i-1} = (G_i, P_i).(G_{i-1}, P_{i-1})$

$C_i = (G_i, P_i).(C_{i-1}, 0)$

4.5 Advantage of Pipelining

Pipelining is nothing but doing more than one operation ,in a single data path.When performing a particular operation some part of the combinational logic is idle,and we can make use of that part if we pipeline that combinational logic.So,we divide the entire combinational logic into stages and insert registers in between..This way of execution increases the clock frequency compared to the one which is not pipelined.This also increases the overall throughput of the system.

For example,let the multiplication operation takes 5 clock cycles .Then 10 multiplication operations take 50 clock cycles.Whereas in a 4-stage pipeline it takes 13 clock cycles.This clearly shows the throughput of the system increases.Also clock frequency increases because the the combinational logic between the registers is less compared to the non-pipelined one.

4.6 Proposed Design For Multiplier

Analysed Wallace tree multiplier and Booth Multiplier using 3-2 compressor,5-2 compressor for reduction of partial products and 4-bit Carry-lookahead adder,Carry Select Adder,Brent-kung adder for final addition with respect to area and speed.

Partial Product reduction using 3-2 compressors is done as shown below :

No. of Summands	No. of groups of Three	Remaining Summands
33	11	0
22	7	1
15	5	0
10	3	1
7	2	1
5	1	2
4	1	1
3	1	0

Partial Product reduction using 5-2 compressors is done as shown below :

No. of Summands	No. of groups of 5/3	Remaining Summands
33	6-5:2 comp	3
15	3-5:2 comp	0
6	1-5:2 comp	1
3	1-3:2 comp	0

4.6.1 Wallace Tree Multiplier with 4-bit CLAs

The complete multiplier logic is divided in the 4 stages of pipeline as follows:

Stage-I

- Partial product generation
- Six levels of reduction

Stage-II

- Two levels of reduction
- Lower 16-bits addition

Stage-III

- Lower 24-bits addition of remaining 48-bits

Stage-IV

- Upper 24-bits addition

4.6.2 Wallace Tree Multiplier with 5-2 compressors and 4-bit CLAs

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial Product Generation
- Two levels of reduction using 5-2 compressors and
- One level using 3-2 compressor

Stage-II

- One level of reduciton using 3-2 compressor
- Lower 12-bits addition

Stage-III

- Lower 24-bits addition of remaining 52-bits

Stage-IV

- Upper 28 bits addition

4.6.3 Wallace Tree Multiplier with Brent-kung adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial Product Generation
- Four stages of 3-2 Compressors

Stage-II

- Four stages of 3-2 Compressors
- From T=1 to T=2 as mentioned in section 4.4.3

Stage-III

- From T=3 to T=7 as mentioned in section 4.4.3

Stage-IV

- From T=8 to T=11 as mentioned in section 4.4.3

4.6.4 Wallace Tree Multiplier with 5-2 compressors and Brent-kung adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial Product Generation
- One level of reduction using 5-2 compressor

Stage-II

- Two levels of reduction using 5-2 compressor
- One level of reduction using 3-2 compressor

Stage-III

- From T=1 t0 T=7 as mentioned in section 4.4.3

Stage-IV

- From T=8 to T=13 as mentioned in section 4.4.3

4.6.5 Wallace Tree Multiplier with Carry Select adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial Product Generation
- Three levels of reduction

Stage-II

- Five levels of reduction

Stage-III

- Two stages of Carry Select Adder

Stage-IV

- Remaining three stages of Carry Select Adder

4.6.6 Wallace Tree Multiplier with 5-2 compressors and Carry Select adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial Product Generation
- One level of reduction using 5-2 compressor

Stage-II

- Two levels of reduction using 5-2 compressor

Stage-III

- One level of reduction using 3-2 compressor
- Two stages of Carry Select Adder

Stage-IV

- Remaining three stages of Carry Select Adder

4.6.7 Radix-4 Booth Encoded Wallace Tree Multiplier with 4-bit CLAs

The complete multiplier logic is divided in the 4 stages of pipeline as follows:

Stage-I

- Partial product generation
- Four levels of reduction

Stage-II

- Two levels of reduction
- Lower 16-bits addition

Stage-III

- Lower 24-bits addition of remaining 48-bits

Stage-IV

- Upper 24-bits addition

4.6.8 Radix-4 Booth Encoded Multiplier with 5-2 compressors and 4-bit CLAs

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial product generation
- One level of 5-2 compressor

Stage-II

- One level of 5-2 compressor
- Two levels of 3-2 compressor
- Lower 8-bits addition

Stage-III

- Lower 28-bits addition of remaining 56-bits

Stage-IV

- Upper 28-bits addition

4.6.9 Radix-4 Booth Encoded Wallace Tree Multiplier with Brent-kung adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial product generation
- Three levels of reduction

Stage-II

- Three levels of reduction
- From T=1 to T=3 as mentioned in section 4.4.3

Stage-III

- From T=4 to T=8 as mentioned in section 4.4.3

Stage-IV

- From T=9 to T=13 as mentioned in section 4.4.3

4.6.10 Radix-4 Booth Encoded Multiplier with 5-2 compressors and Brent-kung adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial product generation

- One level of 5-2 compressor

Stage-II

- One level of 5-2 compressor
- Two levels of 3-2 compressor

Stage-III

- From T=1 to T=7 as mentioned in section 4.4.3

Stage-IV

- From T=8 to T=13 as mentioned in section 4.4.3

4.6.11 Radix-4 Booth Encoded Wallace Tree Multiplier with Carry Select adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial product genration
- Two levels of reduction

Stage-II

- Four levels of reduction

Stage-III

- Two stages of Carry Select Adder

Stage-IV

- Remaining three stages of Carry Select Adder

4.6.12 Radix-4 Booth Encoded Multiplier with 5-2 compressors and Carry Select adder

The complete multiplier logic is divided in the 4 stages of pipeline a follows:

Stage-I

- Partial product generation
- One level of 3-2 compressor

Stage-II

- Two levels of 5-2 compression

Stage-III

- Two stages of Carry Select Adder

Stage-IV

- Remaining three stages of Carry Select Adder

Chapter 5

BLUESPEC SYSTEM VERILOG

BSV is a HDL used in design of electronic systems such as FPGA, ASIC etc. It is a very high level language and results in synthesizable hardware which can run on FPGA emulation platforms. BSV substantially extends the design subset of System Verilog and also increases the programmer's coding efficiency. It has more polymorphism than System Verilog.

5.1 Key Features of BSV

- High level atomic rules in place of Verilog's always block.
- High level interfaces instead of Verilog's port list.
- Powerful Parametrization and Polymorphism.
- Powerful static checking.
- Fully synthesizable at all levels of abstraction.

5.2 Study of the Bluespec System Verilog build process

The following are the steps involved in building a BSV design:

- A developer writes a Bluespec System Verilog program. It may be optionally have Verilog, System Verilog, VHDL and C components.
- The Bluespec System Verilog program is compiled in to Verilog or Bluesim. Then it has two different stages:
  1. pre elaboration - It do parsing and also do type checking.
  2. post elaboration -It does code generation.
- The compilation output is either linked into a simulation environment or processed by a synthesis tool. Once the Verilog or Bluesim implementation is generated, the workstation provides the following tools to help analyse your design:
- The figure below illustrates the various steps involved in building a design in Bluespec SystemVerilog. The steps involved in the design are :

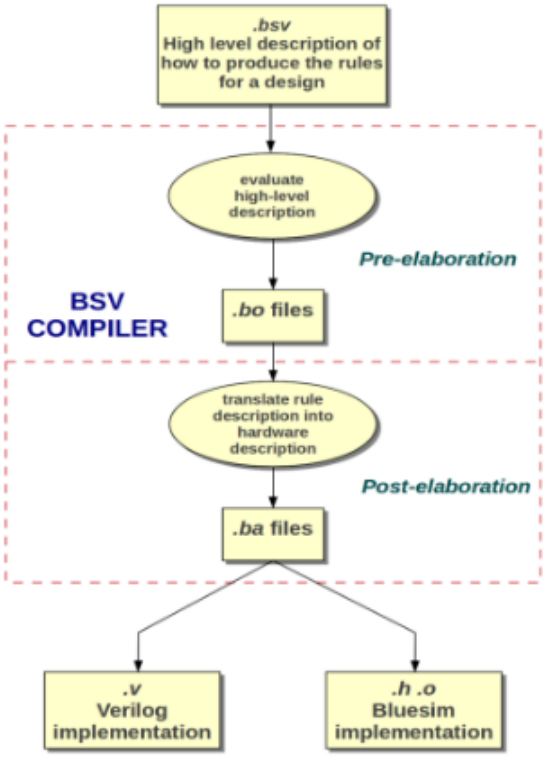


Figure 5.1: BSV Flow

- Designer writes a BSV program. It may optionally include Verilog, System Verilog,VHDL, and C components.
- The BSV program is compiled into a Verilog or Bluesim specification. This step has two distinct stages:
  - Pre-elaboration - parsing and type checking
  - Post-elaboration - code generation
- The compilation output is either linked into a simulation environment or processed by a synthesis tool

5.3 Bluespec SystemVerilog Constructs

5.3.1 Rules

Rules are used to explain how the data shifts from one state to another state, instead of the Verilog methods of uses always blocks. Every Rule has two components:

- Rule conditions : In rule condition we declare condition like in while in c. if condition satisfied then goes to rule body.
- Rule body : It is a set of actions these explains state transitions.

5.3.2 Modules

A module has of three kind of things: state, rules that operate on that state, and an interface that has inputs and outputs of module. A module definition specifies a scheme that can be instantiated multiple times.

5.3.3 Interfaces

Interfaces give a means to group of wires into bundles with mentioned uses, explained by methods. An interface is a tend to remind one of something of a struct, where each member is a method. Interfaces may have other interfaces also.

5.3.4 Methods

Signals and buses are driven in and out of modules using methods. These methods are grouped together into interfaces. There are three kinds of methods:

- Value Methods: It takes zero or more parameters and returns a value.
- Action Methods: It takes zero or more parameters and It performs an action inside of module.
- Action Value Methods: It takes Zero or more parameters, and performs an action, and returns the result.

5.3.5 Functions

Functions are simply parametrized combinational circuits. Function application simply connects a parametrized combinational circuit to actual inputs.

5.4 Application Areas of Bluespec System Verilog

- Modeling for Software development
- Modeling for Architecture Exploration
- Verification
- IP creation

5.5 Building a design in Bluespec System Verilog

- The designer writes the BSV code and it may contain Verilog, Verilog Hardware Description Language and C components.
- The Bluespec System Verilog code is compiled into either Verilog or a Bluesim. This step has 2 stages:
  1. Pre elaboration does parsing and it also does type checking.
  2. Post elaboration does code generation.

The compiled output is either linked to a simulation environment or processed by synthesis tool.

Chapter 6

SIMULATION AND SYNTHESIS

6.1 Hardware Design Flow

Coding the design in a high level language is job only half complete. The final realization of the hardware is the ultimate goal of any project. The hardware or the VLSI design flow as depicted in Figure 8.1 gives the major steps taking the design towards physical realization. A short detour explaining this flow is in order at this stage.

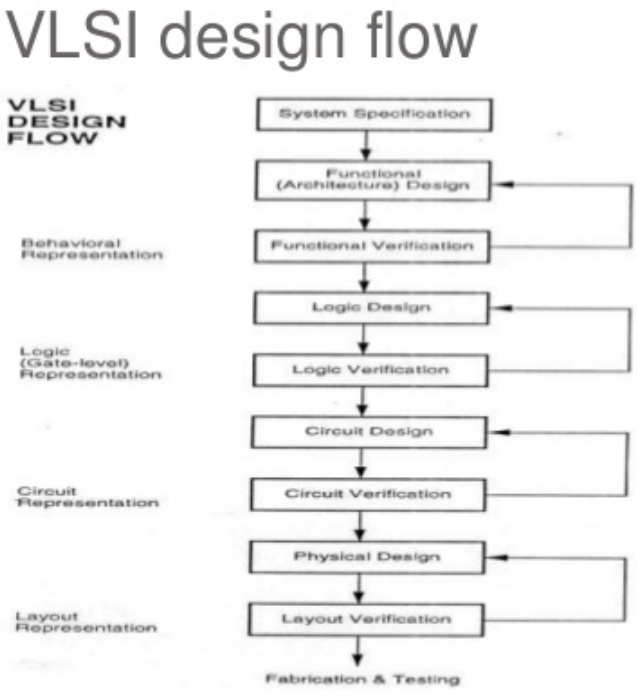


Figure 6.1: Flow of synthesis

The design of any product starts with an idea. The idea is born out of a client requirement. This idea is put down as a higher level behavioural model of the final product using the high level languages like BSV. The behavioural model is then compiled into a RTL using a suitable compiler. RTL are generally the description of the circuit at the module level where input output interfaces, clock and other signals are visible. Any design can be described in RTL using the Huffman's model.

Once the RTL is arrived at, the next step in the design flow is the logic synthesis. Using commercial EDA tools, the designer converts the RTL into a netlist which is

nothing but a list of gates and wires whose input output are specified. The EDA tools gives a lot of options like types of gates to be used, constraints for the design with respect to the power, area and timing, thus a highly optimized netlist is achieved after logic synthesis.

On getting the netlist, more EDA tools are used to do place and route of gates and wires or floor planning as it is popularly called. The result of place and route is the mask that could be handed over to the foundry for carrying out the fabrication of the chip. Two most important part of the design flow are the testing and verification. Testing is done to ensure final chip does not suffer from manufacturing defects and verification is done at each stage of the flow to ensure the design meets the requirements as were originally projected. In our case however, we limit the scope to design, implementation of the design in BSV, logic synthesis and post-synthesis simulations to verify performance.

6.2 Implementation, Synthesis and Simulations

On completion of the BSV coding, the project is compiled with BSV compiler which gives options to compile for BSV simulator or to generate verilog files for further processing. In our case we need both. We simulate the design using the Bluesim simulator and observe the number of hops which was shown to be a good indicator of the latency in the network. The results of the simulation are observed on the BSV GUI and recorded for analysis. Further the design is compiled to generate the verilog files which are required for the EDA tool to complete the logic synthesis as discussed in the design flow diagram. We not only receive an optimized netlist after logic synthesis but also reports for power, area and timing which are required for analysis.

The synthesis tool accepts the verilog files of the design and runs the synthesis algorithm for logic minimization. The synthesis culminates with generation of synthesized design schematic and detailed synthesis report with hardware units used in the final design. It is possible to selectively visualize the flow of the signals and the modules of interest making it convenient for the designer to verify the correctness of the design. The generated netlist is further used for post-synthesis simulations for arriving at power utilization by the design.

Simulation : The following test cases are checked in simulation for all the stages of pipeline.Here the test case followed by the expected result were written :
Stage-I
1.One TE in one of the three FUs(done for all the 3)."Pipeline will stall for one clock cycle and it is considered as TE"

2.TE in one of the FUs for 2 clock cycles(done for all the 3)."Pipeline stalls for 2 clock cycles and it is considered as TE"

3.TE in one of the FUs for 3 clock cycles."Permanent error is recorded for that particular block and is eliminated"

4.TE in all the FUs for 1 clock cycle."Pipeline stalls for 1 clock cycle and is considered as TE"

5.TE in all the FUs for 2 clock cycles."Pipeline stalls for 2 clock cycles and is considered as TE"

Testing is done by making use of Linear Feedback Shift Register (LFSR). Each stage output is ORed with the LFSR value for simulating stuck at 1 and ANDed for simulating stuck at 0.

Simulation Result The below are the screen shots of test cases in the same order mentioned above :

```
-----------------------------------counter:          5------------------
error injected here
evaluating 1
evaluating 2
evaluating 3
no fault
dequing the value from stage 4
product :       82165598252996
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 4
in stage2
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 3
in stage1
evaluating 1
evaluating 2
evaluating 3
transient error in 3
```

Figure 6.2: Test case 1

```
-----------------------------------counter:          6------------------
----------------------
error injected here
product : 00000c350ee9fb80,       13422023015296
evaluating 1
product : 00000c350ee9fb80,       13422023015296
evaluating 2
product : 00000c350ee9fb80,       13422023015296
evaluating 3
no fault
dequing the value from stage 4
product :       13422023015296
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 4
in stage1
evaluating 1
evaluating 2
evaluating 3
transient error in 1
```

```
-------------------------------------counter:          7------------------
----------------------
error injected here
product : 000110a28a46b0a0,      299765267345568
evaluating 1
product : 000110a28a46b0a0,      299765267345568
evaluating 2
product : 000110a28a46b0a0,      299765267345568
evaluating 3
no fault
deguing the value from stage 4
product :      299765267345568
in stage1
evaluating 1
evaluating 2
evaluating 3
transient error in 1
```

Figure 6.3: Test case 2

```
-------------------------------------counter:          45------------------
error injected here
evaluating 1
evaluating 2
evaluating 3
no fault
deguing the value from stage 4
product :      -38125337442489
evaluating 1
evaluating 2
evaluating 3
no fault
enguing the value to stage 4
in stage2
evaluating 1
evaluating 2
evaluating 3
no fault
enguing the value to stage 3
in stage1
evaluating 1
evaluating 2
evaluating 3
transient error in 1
```

```
------------------------------------counter:         46------------------

error injected here
evaluating 1
evaluating 2
evaluating 3
no fault
dequing the value from stage 4
product :       -43019430296150
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 4
in stage1
evaluating 1
evaluating 2
evaluating 3
transient error in 1
```

```
------------------------------------counter:         47------------------
-----------------------
error injected here
evaluating 1
evaluating 2
evaluating 3
no fault
dequing the value from stage 4
product :         -411092000289
in stage1
evaluating 1
evaluating 2
evaluating 3
permanent fault occured in 1
```

Figure 6.4: Test case 3

```
------------------------------------counter:         16------------------
-----------------------
error injected here
product : 000c581858659e0e,     3474561306041870
evaluating 1
product : 000c581858659e0e,     3474561306041870
evaluating 2
product : 000c581858659e0e,     3474561306041870
evaluating 3
no fault
dequing the value from stage 4
product :      3474561306041870
in stage1
evaluating 1
evaluating 2
evaluating 3
can not decide,since more than one block has error
```

Figure 6.5: Test case 4

```
-------------------------------------counter:         19------------------
----------------------
error injected here
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 4
in stage2
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 3
in stage1
evaluating 1
evaluating 2
evaluating 3
can not decide,since more than one block has error


-------------------------------------counter:         20------------------
----------------------
error injected here
evaluating 1
evaluating 2
evaluating 3
no fault
dequing the value from stage 4
product :     3474561306041870
evaluating 1
evaluating 2
evaluating 3
no fault
enquing the value to stage 4
in stage1
evaluating 1
evaluating 2
evaluating 3
can not decide,since more than one block has error
```

Figure 6.6: Test case 5

Synthesis Results

Comparision of the synthesis results for Wallace Tree and Booth Recoded multipliers using all combinations of 3-2 compressor,5-2 compressor,4-bit CLA,Carry select adder and Brent-kung adder are shown below :

- Wallace

	3-2 Compressor			5-2 Compressor		
	Freq (MHz)	LUTs	Slices	Freq (MHz)	LUTs	Slices
Carry-lookahead	50.070	7099	3709	46.589	7360	3826
Brent-Kung	48.393	8676	4532	49.039	9955	5257
Carry Select	45.201	8697	4582	42.167	9562	5046

Figure 6.7: Results of Wallace Tree Multiplier

- **Booth**

	3-2 Compressor			5-2 Compressor		
	Freq (MHz)	LUTs	Slices	Freq (MHz)	LUTs	Slices
Carry-lookahead	44.050	6651	3447	47.303	7687	3983
Brent-Kung	40.708	9377	4912	45.198	8730	4532
Carry Select	44.592	7879	4109	44.273	8760	4617

Figure 6.8: Results of Booth Multiplier

Chapter 7

CONCLUSION

7.1 Conclusion

As CMOS technology moves deep in nanometer range, reliability poses a serious concern in the system design. The probability of soft and hard errors increases, even for ground based system. The processor which needs to work reliably in space or in harsh environment should have techniques to tolerate or mitigate the faults.

This thesis proposed a fault tolerant approach for a 4 stage pipelined multiplier. The proposed technique was developed with the intention of offering a reasonable fault tolerance with minimal area and power penalty. First we have discussed about various types of errors and their sources. The effects of SEUs in memory and combinational logic have been explored. Later the fault avoidance and fault tolerance techniques have been explained. Chapter 4 mainly described the complete design implemented in this project.

From the synthesis results it is observed that Wallace Tree occupies more space and runs at more clock frequency and Booth occupies comparitively less space but runs at less clock frequency.

Bibliography

[1] Bluespec,Inc.Bluespec System Verilog Reference Guide,Revision 30 July 2014.

[2] A.H.Johnston,G.M.Swift and D.C.Shaw, Impact of CMOS Scaling on Single Event Hard Errors in Space Systems,paper published by Jet Propulsion Laboratory,California Institute of Technology.

[3] Nguyen Minh Huu,Bruno Ribisson,Michel Agoyan and Nathalie Drach, Low-cost fault tolerance on the ALU in simple pipelined processors,IEEE 2010.

[4] Bluespec Inc, Bluespec SystemVerilog Reference Guide Revision: 30 January 2012.

[5] Shekher Borkar, "Design challenges of technology scaling," Micro, IEEE ,vol.19, no.4, pp.23,29,Jul-Aug 1999

[6] RISE Lab,Shakti Series Processors.ppt

[7] Kai Hwang "Computer Arithmetic: Principles, Architecture, and Design" John Wiley Sons 1979

[8] Chia-Hsiang Chen, David Blaauw,Dennis Sylvester and Zhengya Zhang, Design and Evaluation of Confidence-Driven Error-Resilient Systems,IEEE Transactions on Very Large Scale Integration (VLSI) Systems,vol 22,No 8,August 2014.