Timing Induced Error Analysis for Wallace Tree Multipliers

A THESIS
SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

Vaishnavi Santhapuram

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

Prof. Keshab K. Parhi

September 2015

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Keshab. K. Parhi for guiding me through my research here at University of Minnesota. Due to his patience and constant motivation I have been able to complete this research work.

I would like to also thank the faculty members on my thesis committee, Prof. Antonia Zhai and Prof. Marc Riedel, for taking their valuable time in reviewing my thesis and providing useful feedback and suggestions.

I would like to also thank my friends Krishna Kiran Talamadupula, Nimish Agashiwala, Harini Suresh and Satya Prakash Upadhyay for their continued support and their time. And last but not least, I would like to express deepest gratitude to my parents, my father, and my mother, for working hard and giving me the opportunity to undertake a Master's degree so far away from home. They always encouraged me to put in my best effort.

Dedication

I dedicate this Thesis to my parents who have inspired and nurtured me throughout my life. Their love and support has enabled me to strive for more in academic life.

Abstract

With the increasing demand for embedded and mobile systems to support a wide breadth of applications with tight power budgets as well as increased heat dissipation in processors due to increased operating frequencies and processing capacity per chip with technology scaling, energy efficiency in VLSI systems has become a critical constraint. On the other hand with technology scaling into sub nm, energy efficiency due to scaling is diminished due to increased process variability. Process variations result in delay deviations. Voltage over scaling is considered as an effective technique to reduce energy consumption. Process variability and voltage over scaling result in timing errors. This thesis focuses on understanding the effects of timing error on different multiplier arithmetic units, since multipliers are one of the key hardware blocks in signal processing systems as well as general purpose processors and they consume considerable amount of power. It is observed that different hardware implementations of the same multiplier function may respond very differently to timing errors. Hence few architectures are inherently more error resilient to timing errors and selection of an appropriate architecture will result in better energy efficiency under voltage over scaling or over clocking. The second part of the thesis presents that same multiplier architecture will have different error statistics for different input distributions. Since most of the real signals used in signal processing systems and communication systems are Gaussian distributed, multiplier architectures are tested for Gaussian inputs and observations show that performance under timing induced error is worse for Gaussian distributed inputs than uniformly distributed inputs.

Table of Contents

Acknowledgements i
Dedication ii
Abstract iii
List of Tables v
List of Figures vi
1 Introduction 1
2 Background and Motivation 3
3 Multiplier Architecture 6
3.1 Partial Product Matrix Generation 6
  3.1.1 Modified Booth (Radix-4) Encoding 7
3.2 Partial Product Accumulation 9
  3.2.1 Compressor Architectures 10
  3.2.2 Wallace Tree Using 3:2 Compressors 12
  3.2.3 Wallace tree Using 4:2 Compressors 13
3.3 Final Stage Carry propagate Adder 14
  3.3.1 Kogge-Stone Adder 15
  3.3.2 Carry-Select Adder 17
3.4 Multiplier Architectures Implemented 19
4 Experiment and Result Analysis 20
4.1 Experimental Setup 20
  4.1.1 Verilog Code 20
  4.1.2 Input Generation 20
  4.1.3 Test Bench and Output Files for Analysis 20
4.2 Result Analysis 21
  4.2.1 Error metrics 22
  4.2.2 Gaussian Distributed Inputs Vs Uniformly Distributed Inputs 23
  4.2.3 Gaussian Distributed Inputs with different variances 26
  4.2.4 Wallace tree with Carry Select Adder 30
  4.2.5 Wallace tree with 4to2 Compressors 35
  4.2.6 Comparison of all Multiplier Modules for Gaussian Distributed Inputs 36
  4.2.7 Comparison of all Multiplier Modules for Uniformly Distributed Inputs 37
  4.2.8 Timing Error Characteristic for One Constant Input 38
  4.2.9 Power Estimation 41
5 Conclusion and Future Ideas 43
5.1 Conclusions 43
5.2 Future Ideas 44
References 45

List of Tables

Table 3.1 Radix-4 Booth Encoding Scheme 7
Table 4.1 Success rate for Wallace3to2KSA at different Clock Periods 26
Table 4.2 Success rate for Wallace3to2KSA for Gaussian inputs of different variances. 29
Table 4.3 Success rate for different Multiplier Modules at different Clock Periods 32
Table 4.4 Positive and negative constant inputs for which Wallace3to2KSA is tested ... 38
Table 4.5 Success rate for Wallace3to2KSA module for different positive constants 40
Table 4.6 Success rate for Wallace3to2KSA module for different negative constants 40
Table 4.7 Power values for different multiplier modules 41
Table 4.8 Area for different multiplier modules 42
Table 4.9 Power values at reduced voltage 42

List of Figures

Figure 3.1 Grouping of bits for Radix-4 Booth recoding 7
Figure 3.2 Radix-4 Booth Encoded Partial products with simplified sign extension [3] ... 8
Figure 3.3 Block diagram and dot notation of (a) 3-2 compressor (b) 2-2 compressor ... 10
Figure 3.4 N-bit CSA.. 10
Figure 3.5 Alternative implementation of 4:2 compressor 11
Figure 3.6 (a) 4:2 compressor carry chain (b) dot notation of 4:2 compressor 11
Figure 3.7 Block diagram of 2-bit half CLA and dot notation 12
Figure 3.8 Wallace Tree reduction of partial product matrix using 3:2 Compressors...... 13
Figure 3.9 Wallace Tree reduction of partial product matrix using 4:2 Compressors...... 14
Figure 3.10 General stages of logic in Addition with generate and propagate logic [3] .. 15
Figure 3.11 Kogge-Stone adder PG network[3] 16
Figure 3.12 Black and Grey cell logic block diagram [3]... 16
Figure 3.13 Carry-Select adder Module1 (CSA1) 17
Figure 3.14 Carry-Select adder Module2 (CSA2) 18
Figure 3.15 Carry-Select adder Module3 (CSA3) 18
Figure 4.1 SNR vs Clock period for Gaussian and Uniform Inputs for Wallace3to2KSA
.... 23
Figure 4.2 SNR vs Clock period for Gaussian and Uniform Inputs for Wallace3to2KSA
.... 25
Figure 4.3 SNR plot with Different input variance for Wallace3to2KSA........................ 27
Figure 4.4 SNR Vs Clock Period for Gaussian inputs with variance (a) 7500 (b) 5000 .. 28
Figure 4.5 SNR Vs Clock Period for Gaussian inputs with variance 2500 29
Figure 4.6 SNR Vs Clock Period for Wallace3to2 with different Final Stage CPA 30
Figure 4.7 Error probability for output bit positions at Clock period at (a) 1.9ns (b) 1.8ns
.... 31
Figure 4.8 Typical Wallace tree timeline [10] 33

Figure 4.8 28-bit Kogge-Stone Adder 34

Figure 4.9 SNR Vs Clock Period for Wallace3to2KSA, Wallace4to2KSA for Gaussian Inputs... 35

Figure 4.10 SNR Vs Clock Period for Eight Multiplier Modules 36

Figure 4.11 SNR Vs Clock Period for Eight Multiplier Modules for Uniform Inputs..... 37

Figure 4.12 SNR Vs Clock period for Wallace3to2KSA module for one constant input 39

Chapter 1

Introduction

As Integrated Circuit (IC) technology continues to scale to sub-nm and beyond, variability has been recognized as a major challenge. Due to increased variation in process, voltage and temperature (PVT), traditional worst-case design methodology leads to power consumption waste or reduced operating frequency. Because of the variability, energy efficiency gains due to technology scaling are diminished. On the other hand, with technology scaling as operating frequency and processing capacity per chip increases, large currents are required which results in a need for methods for removal of heat generated by large power consumption. At the same time, demand for high levels of performance and supporting wide range of applications for embedded systems and mobile systems is increasing. But the battery life for portable devices is not advanced at the same scale leading to diminished power budgets. Hence, energy and power consumption have become critical design constraints leading to the need of low-power VLSI systems. Variability is becoming a major challenge in designing low-power and high performance systems. Also with increase in variability, delay deviations are imposed which might result in timing errors [1] [2].

Moreover, recently there has been significant progress in approximate computing where accuracy is traded for energy efficiency or power consumption. Approximate computing in hardware is the implementation of circuit that does not exactly match the specification due to functional approximation (implementing slightly different Boolean equation for energy gains) or because of timing-induced errors (over-scaled supply voltage or over-clocking). Voltage over scaling can result in timing errors.

Multiplier is one of the basic hardware blocks in digital signal processing systems and an indispensable part of general purpose processors as well. Multipliers are used in arithmetic logic units, media processing units etc., in microprocessors. Signal processing systems require multipliers for implementing filters, convolution, FFTs etc. Multipliers generally have large areas and also consume considerable amount of power. Deeper understanding of statistical characteristics of timing error in multiplier architectures will help us choose an architecture more suitable for over clocking or voltage scaling so that power consumption of multiplier can be reduced.

The first part of this thesis is focused on comparing few multiplier architectures and understanding which architecture is more error-resilient to timing-induced error by observing different error metrics as well as understanding the reasons behind it. The second part of the thesis focuses on the analysis of the effects of different input distribution characteristics on timing-induced error characteristics. As multiplication by a constant is frequently present in DSP hardware, multiplier architectures are analyzed when one of the inputs remains constant.

The remaining chapters are organized as follows:

- Chapter 2 briefly explains the need of timing induced error analysis and the motivation behind it.
- Chapter 3 explains the multiplication process and the implementation of multiplier architectures analyzed for timing induced errors.
- Chapter 4 starts with outlining the experimental setup. Then it goes on to the analysis and reasoning of different timing induced error statistics on different multiplier architectures with different distributed inputs.
- Chapter 5 presents the observations and conclusions drawn from the experiments and analysis.

Chapter 2

Background and Motivation

One of the most important constraints of electronic devices especially for battery powered hand-held devices is power consumption. Along with power dissipation, process parameter variations are major challenges in sub-nm technology regime [3] [4]. With continuous technology scaling, leakage power has increased to nearly 20-40% of the total power in deep sub-micron modern microprocessors [5]. Due to the limited cooling capacity increased power dissipation results in increased junction temperature. Limitations of battery-life in portable devices with ever increasing demand of processing capability and supporting wide breadth of application, and temperature induced reliability issues have led to a lot of research and effort in low-power VLSI. Voltage scaling has become a popular technique to reduce dynamic power due to the quadratic dependence of dynamic power on supply voltage.

Process imperfections due to sub-wavelength lithography result in device level variations in small-geometry devices [5]. Variations in device parameters like width, length, oxide thickness, random dopant fluctuations, flat band voltage etc., result in large variation in circuit parameters like threshold voltage. The speed of the circuit strongly depends on V_{th}. High-V_{th} corresponds to lower currents and vice versa. Hence higher-Vth circuits are typically slower than lower-V_{th} ones. Therefore, statistical variations in device parameters result in statistical variations in delay of a circuit [6] [7]. Traditional pessimistic worst-case design approach leads to wastage of power consumption and reducing the probability of meeting desired performance or frequency. Hence, lot of research effort has been put to explore alternative design methodologies to achieve low power while meeting required specifications.

Recently, there has been lot of research to understand or estimate timing under process parameter variations. Process variation effects on the delays of the logic gates and timing errors are analyzed in [1] [2] [8]. In [2] comprehensive model VARIUS, that produces detailed statistics of timing errors as a function of different process parameters and operating conditions was developed. They developed a microarchitecture-aware model for parametric variation and developed a framework to model timing errors under variation. In [8], analysis of the effect of process variations on the delay of static and dynamic CMOS logic gates as a function of circuit-level parameters such as number of stacked transistors (fan-in), their size, the load capacitance (fan-out), and the circuit topology (static versus dynamic logic) is carried out. Estimation of timing response of nanometer digital circuits under the impact of process variations by the application of proposed delay variation models is described in [1].

To reduce the gap between worst case design and typical case, on-chip timing error detection and correction has been proposed in [9] [10] [11]. In [9], a new approach is proposed called Razor, which is a dynamic voltage scaling (DVS) technique based on dynamic error detection and correction of timing errors using razor flip-flop. Razor flip-flop double samples pipeline stage values, one with faster clock and one with time-borrowed delayed clock. In [10], Adaptive Voltage Scaling (AVS) system is proposed which is based on monitoring the timing of a few tunable replica critical paths designed with double sampling monitors to generate error prediction signals.

To reduce the effects of process variation with scaling on performance and power, there is a branch of research focusing on timing speculation under process variations for reliable models for static timing analysis so that yield and reliability are improved, and many techniques for on-chip timing error detection and correction coupled with voltage scaling are being developed to save power consumption and overhead in performance by designing for worst-case.

Another way to achieve energy efficiency is by trading accuracy with power. Many applications are inherently error resilient. Example of such applications are some DSP, multimedia (images/audio/video), neural networks, wireless communications, date recognition, data mining algorithms [12]. Approximation of a circuit could be achieved either by allowing some timing violations by voltage over-scaling or over-clocking, or by function approximation.

Voltage over-scaling (VOS) is demonstrated as an efficient method to reduce energy consumption in signal processing systems where satisfactory signal processing performance is sufficient. Multipliers are one of the key components used in processors as well as error resilient applications that consume considerable amount of power. All the above reasons motivates us to try to understand the effects of timing error in different multiplier architectures. In practice timing error can occur due to process parameter variations due to environment changes such as temperature, voltage over-scaling or over clocking. In this thesis, timing error is induced by over-clocking and how different architectures result in different computation error statistics are analyzed. In [13] it is shown that propagation delay is reduced linearly with voltage over-scaling factor. Hence, for the purpose of analyzing which architecture is better or to understand which input distribution is better, the effect of timing-induced error by over-clocking on error statistics would be almost the same as voltage over-scaling.

Many multiplier designs have been proposed in order to achieve various design requirements such as low energy consumption, high speed, smaller area etc. Wallace tree multiplication is often used for efficient parallel multiplication. Hence, in this thesis, timing-induced error statistics on Wallace tree multipliers is analyzed. Different architectures are analyzed for uniformly distributed inputs and Gaussian distributed inputs. Most of the analysis in literature for timing error or voltage over scaling analysis has been based on random inputs. But in reality many of the signal processing applications and communication systems have Gaussian distributed signals. This motivates us to analyze how error statistics of a given architecture would change with input distribution.

Wallace tree is analyzed with different adders at the final stage. The adder used in final CPA stage could play a major role in error statistics. This is due to the fact that, when multiplier is over-clocked or over-scaled, the paths that fail mostly will have sufficient time to finish computation so that final CPA has correct inputs unless scaling factor is really high. High scaling factor is unlikely because if scaled by that high factor, most of the paths will fail.

The following chapter explains different implementations of multiplier architectures for which timing-induced error statistics are analyzed.

Chapter 3

Multiplier Architecture

In this Chapter, Multiplication process and Multiplier Architectures which were implemented are described. Multiplier Architecture essentially can be divided into 3 phases.

1. Partial Product Matrix generation
2. Partial Product Matrix reduction to two rows
3. Final Carry propagation addition

3.1 Partial Product Matrix Generation

In the first phase of Multiplication process, partial products are generated by multiplying each digit of the multiplier with the entire multiplicand. In binary multiplication, each digit of multiplier is either '0' or '1', so each partial product is either a multiplicand or all 0's. In case of N x N-bit multiplication, P = Y x X consists of forming N partial products of N-bits each. Binary multiplication of 2 bits is nothing but a logical AND operation. So, AND gates can be used to generate partial products. This is a simple and less time consuming method. But this results in high number of partial products which will lead to significant delay in the second phase of partial product accumulation.

One of the most common methods to reduce the number of partial products is using Booth encoder [14]. Booth encoding scheme could be implemented with various radices such as radix-4, radix-8 and so on. If radix 2^r is applied for encoding, then the number of partial products are reduced to N/r. Though radix-8, radix-16 encoding schemes reduce the number of partial products by higher amounts, the circuitry for encoding is more complex and takes more area as well. Hence, Radix-4 Booth encoding is implemented. Parallel Multipliers which use Booth encoding have an advantage of processing time at the cost of little increase in area due to extra circuitry for encoding based on [15]. The following section explains the scheme of Modified Booth (Radix-4) encoding.

3.1.1 Modified Booth (Radix-4) Encoding

The number of partial products is reduced by half using radix-4 Booth recoding scheme. Multiplicand is encoded based on the multiplier bits. Multiplier term is Booth recoded by considering blocks of 3-bits at a time, such that each block overlaps the previous block by one bit as shown in Figure 3.1 for a 16-bit multiplier.

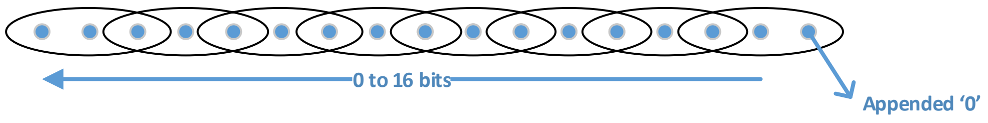


Figure 3.1 Grouping of bits for Radix-4 Booth recoding

First block is appended with 0, because there is no previous block. If X, Y are inputs to the 16-bit Multiplier, there will be 8=16/2, Partial Products (PP_i) which can be 0,Y,-Y,2Y,-2Y based on multiplier blocks $x_{2i+1}x_{2i}x_{2i-1}$ for i=0 to 7, and $x_{-1} = 0$. Table 3.1 illustrates the radix-4 encoding scheme.

Table 3.1 Radix-4 Booth Encoding Scheme

x_{2i+1}	x_{2i}	x_{2i-1}	Partial Product(PP_i)
0	0	0	0
0	0	1	Y
0	1	0	Y
0	1	1	2Y
1	0	0	-2Y
1	0	1	-Y
1	1	0	-Y
1	1	1	0

As can be seen from the table, partial product generation involves negation and complementing operations. In 2's complement representation, negation is performed by inverting all bits and adding 1 to the LSB. This addition of 1 will result in long carry propagation and is time consuming. So to avoid this extra time required to add 1 to the LSB, addition of 1 is postponed to next stage, i.e., it is added in the second phase of partial product accumulation.

Negative partial products need to be sign extended for correct summation. The problem with this in hardware is that all partial products need to be extended all the way till the last partial product ends. Although this is achievable, it needs a lot of extra logic gates. All the sign extension bits are either ‘1’ or ‘0’ if the partial product is negative or positive, respectively. This can be optimized as explained in [16]. If a single 1 is added to the LSB in a string of 1s, the result is a string of 0s plus a carry-out which may be discarded. Hence, the large number of sign bits in each partial product can be replaced by an equal number of constant 1s plus the inverse of the sign bit added to the LSB position. These constants can be precomputed and the simplified result is shown below in a dot diagram.

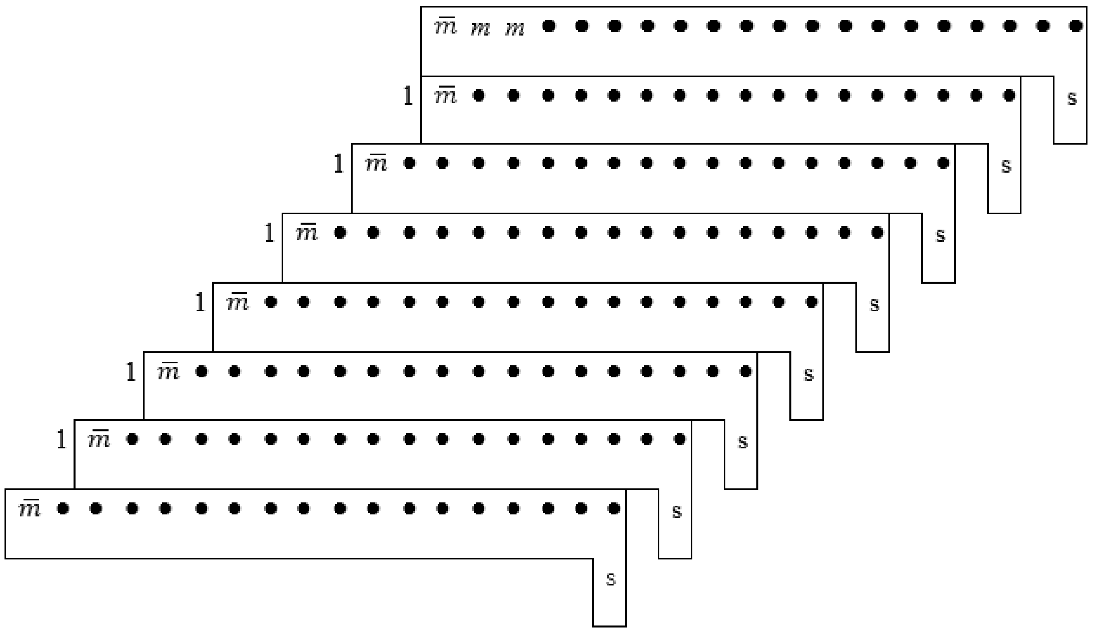


Figure 3.2 Radix-4 Booth Encoded Partial products with simplified sign extension [16]

where 'm' is the MSB or sign bit of the respective 17-bit Partial Product. Entry 's' for a particular partial product is 1 if 2's complement was required in Booth encoding and 0 otherwise.

3.2 Partial Product Accumulation

In the second phase, generally a tree adder structure is used for accumulating partial products. The dot diagram shown in Figure 3.2 should be reduced to two rows for final CPA. This phase is generally crucial in deciding delay, area and power of the multiplier.

Addition of dots in each column is generally done by using Carry Save Adders. Carry Save Adder is a 1-bit Full Adder or (3,2) counter which counts number of 1s on three input bits of column w and gives a 2-bit output, i.e., Sum and Carry bits. The sum bit has same weight as the current inputs, while the carry bit is passed on the next column w+1 and Carry bit is received from previous column's CSA. Essentially using CSAs at a time, 3 input vectors are compressed to 2 inputs vectors. These CSAs can be used in an array multiplier or in tree structures. However, array multipliers take a longer time to reduce the N input vectors to final 2 output vectors for CPA. Hence, tree structures are generally deployed using compressors to reduce the delay. Two common approaches are Wallace Tree Adder (WTA) [18] and Dadda Tree Adder (DTA) [19].

A Wallace tree is an implementation of adder tree that tries to achieve minimum propagation delay. So WTA based methods have fewer stages compared to DTA based methods. In DTA based designs, each row-addition uses lesser resources but the final length of the adder required for DTA is longer than the WTA based architectures. Another disadvantage of DTA is that it is less regular than WTA and hence more difficult to layout [20].

Compressors are basic building blocks for accumulating partial products using Wallace tree method. Conventional Wallace tree structures use 3:2 compressors or Full Adders and 2:2 compressors or Half Adder. To further reduce the number of stages in a Wallace tree higher order compressors such as 4:2 compressors can be used. In this thesis, implementations of a Wallace tree using 3:2 and 2:2 compressors with a few modifications as well as a Modified Wallace tree using 4:2 compressors are considered.

Section 3.2.1 explains different compressor architectures used for implementation followed by Sections 3.2.2 and 3.2.3 which briefly describe the Wallace tree structures implemented using these compressors.

3.2.1 Compressor Architectures

- 3:2 Compressor: It has three input bits of equal weight and two output bits, sum bit and carry bit. Carry bit has higher weight as explained above for CSA. Block diagram of 3:2 compressor or 3:2 counter or Full Adder is shown below in Figure 3.3. Figure 3.4 shows N-bit CSA where 3 N-bit vectors are merged to N-bit sum vector and N-bit carry vector. N-bit CSA is a parallel set of N 3:2 compressors.

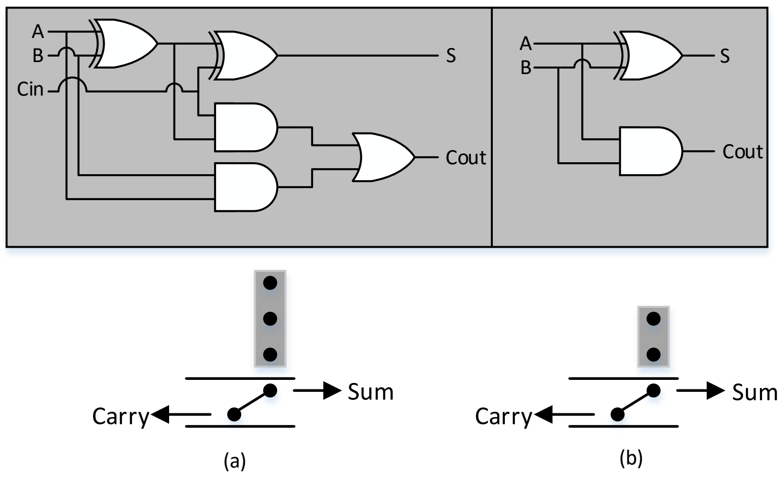


Figure 3.3 Block diagram and dot notation of (a) 3-2 compressor (b) 2-2 compressor

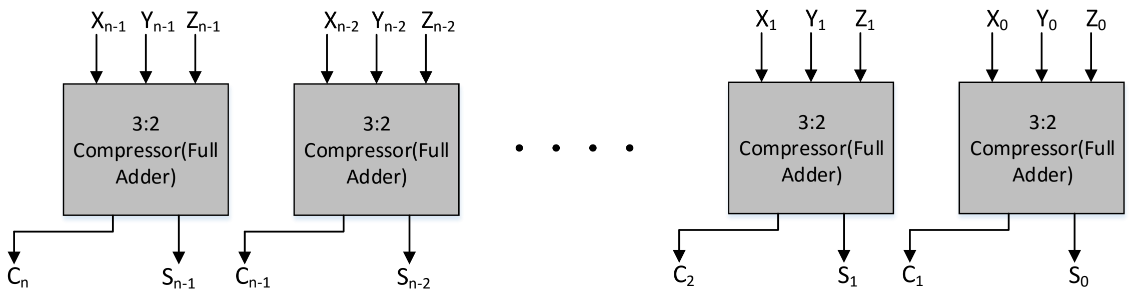

X_{n-1} Y_{n-1} Z_{n-1}
3:2 Compressor(Full Adder)
C_n S_{n-1}
X_{n-2} Y_{n-2} Z_{n-2}
3:2 Compressor(Full Adder)
C_{n-1} S_{n-2}
X_1 Y_1 Z_1
3:2 Compressor(Full Adder)
C_2 S_1
X_0 Y_0 Z_0
3:2 Compressor(Full Adder)
C_1 S_0

Figure 3.4 N-bit CSA

- 2:2 Compressor: It has two inputs of same weight and 2 output bits, sum bit and carry bit. 2:2 compressor is nothing but a half adder. Block diagram of 2:2 compressors is shown in Figure 3.3 above. When a column in Wallace tree has only 2 inputs, then half adder or 2:2 compressor is used.
- 4:2 compressor: Single 4:2 compressor actually has five inputs (one being Cin), and three outputs (sum and two carry bits). One of the carry bits is Cout which is given as Cin to the next column 4:2 compressor. Sum and carry bits are passed onto next stage of reduction. 4:2 compressors can be built using two 3:2 compressors. But this will have a critical path of 4 Xors. An alternative implementation of 4:2 compressor [21] is shown in Figure 3.5.

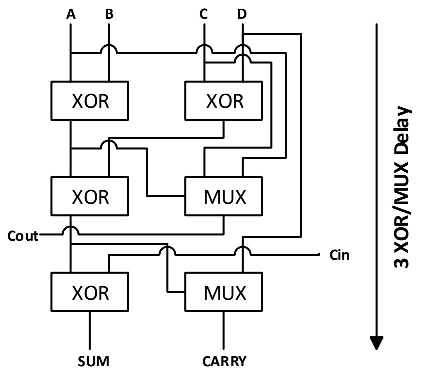


Figure 3.5 Alternative implementation of 4:2 compressor

It can be seen from the block diagram in Figure 3.6 that in this implementation of 4:2 compressor, Cout for the next cell in the chain doesn't depend on Cin which prevents long carry propagation path when 4:2 compressor is used in chain as shown in Figure 3.6.

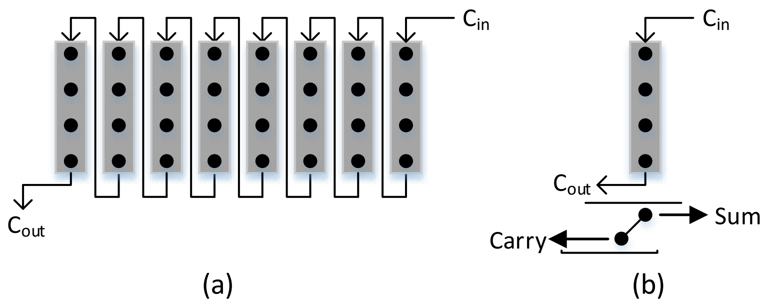


Figure 3.6 (a) 4:2 compressor carry chain (b) dot notation of 4:2 compressor

- Half CLA: Half CLA is a 2-bit carry look ahead adder without carry in. Half CLA was used in modified Wallace tree multiplier in [22]. This half CLA block can be used to modify Wallace tree slightly to get 2 final outputs in each stage rather than 1, thereby reducing the length of the final carry propagation adder when carry select adder is used to reduce propagation delay. The half CLA block diagram is shown in Figure 3.7.

  Critical path of half CLA is about the same as that of a 1-bit Full Adder.

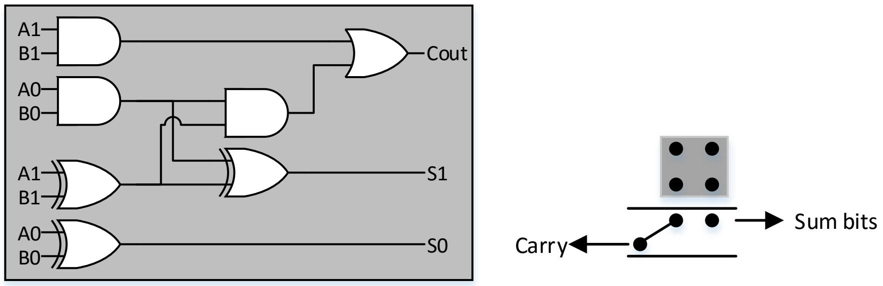


Figure 3.7 Block diagram of 2-bit half CLA and dot notation

3.2.2 Wallace Tree Using 3:2 Compressors

The Wallace tree reduction stages that are implemented using 3:2 and 2:2 compressors are shown in the form of a dot diagram in Figure 3.8.

As can be seen from the Figure 3.8, there are 4 stages. Each stage gives 1 final output multiplication bit, thereby reducing the final carry propagation adder length to 28-bit wide. Few modifications have been made to conventional Wallace tree multiplier to reduce the redundant half adders in the last columns and a few initial columns of the partial product matrix. This is done so that they will be available for final CPA sooner, thereby reducing the path delays of few paths to the output even though critical path won't reduce. Compressor symbols of 3:2 and 2:2 compressors used in Figure 3.8 are described in Figure 3.3.

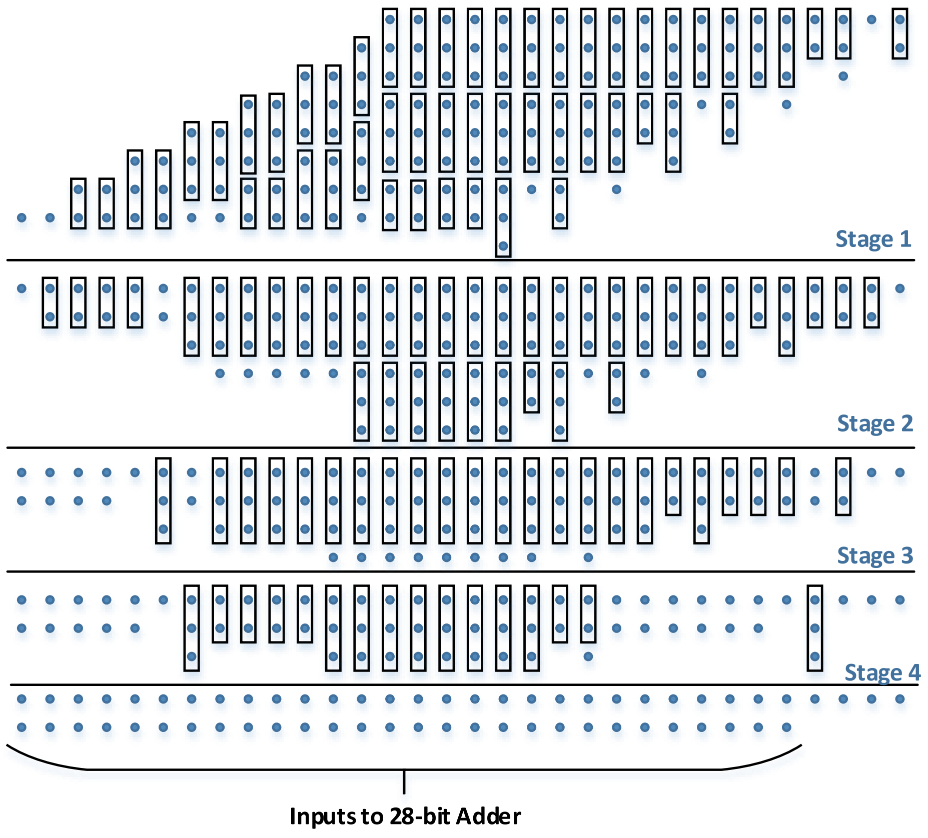


Figure 3.8 Wallace Tree reduction of partial product matrix using 3:2 Compressors

3.2.3 Wallace tree Using 4:2 Compressors

The Wallace tree implementation that is implemented using 4:2 compressors combined with 3:2 and 2:2 compressors is shown in the form of a dot diagram in Figure 3.9.

The number of stages of this Wallace tree is reduced by 1 stage because of the use of 4:2 compressors. Few modifications have been made to the typical Wallace tree multiplier to reduce the redundant half adders in the last columns and a few initial columns of the partial product matrix. This is done so that they will be available for final CPA sooner there by reducing path delays of few paths to the output though critical path won't reduce because of this. A 2-bit half CLA is used in 3rd stage of Wallace tree reduction to reduce the length of final CPA to 28-bit as in the case of the Wallace tree using 3:2 compressors with 4 stages. The dot notation of 3:2 and 2:2 compressors used in

Figure 3.9 are described in Figure 3.3. Compressor symbol for 2:2 compressor used is described in Figure 3.5.

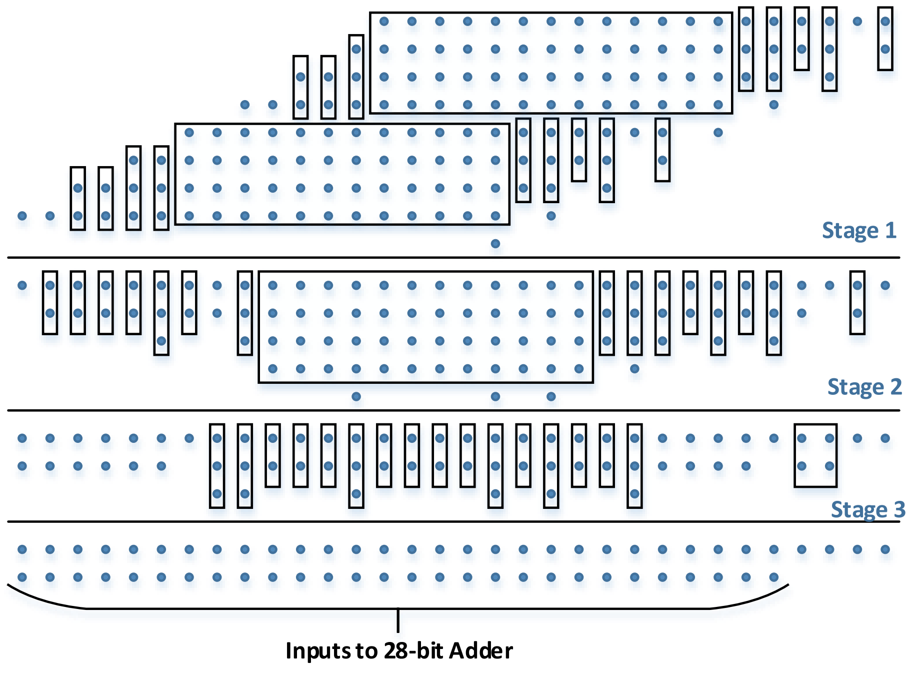


Figure 3.9 Wallace Tree reduction of partial product matrix using 4:2 Compressors

3.3 Final Stage Carry propagate Adder

In the third phase, the last two rows are added using carry propagate adder to compute the final multiplier result. CPA can be implemented using a ripple carry adder (RCA) or a carry look-ahead adder (like Kogge-Stone Adder) or a carry select adder etc. Ripple carry adder has long latency due to long critical path of carry propagation. Parallel prefix form carry look-ahead Kogge-Stone tree adder is widely used in high performance 32-bit and 64-bit adders [3]. A Carry-select adder achieves performance due to parallel computation of different segments and hence is fast. In this thesis both Wallace trees are implemented in combination with Kogge-Stone Adder and Carry-select adder.

3.3.1 Kogge-Stone Adder

Kogge-Stone Adder is a fast carry propagation adder. The Kogge–Stone adder generates the carry signals in O (log n) time. Kogge-Stone tree adder has $\log_2 N$ stages and each stage has a fanout of 2.

In look-ahead tree adders, propagate and generate signals are generated using PG logic for all the inputs and then group PG logic is used to calculate carry signals. The number of stages required to generate carry signals depends on the tree structure. The calculated carry signals are combined with propagate signals of respective input bits to generate final sum output of the adder. Figure 3.10 shows 3 stages of addition using generate and propagate logic for 4-bit adder.

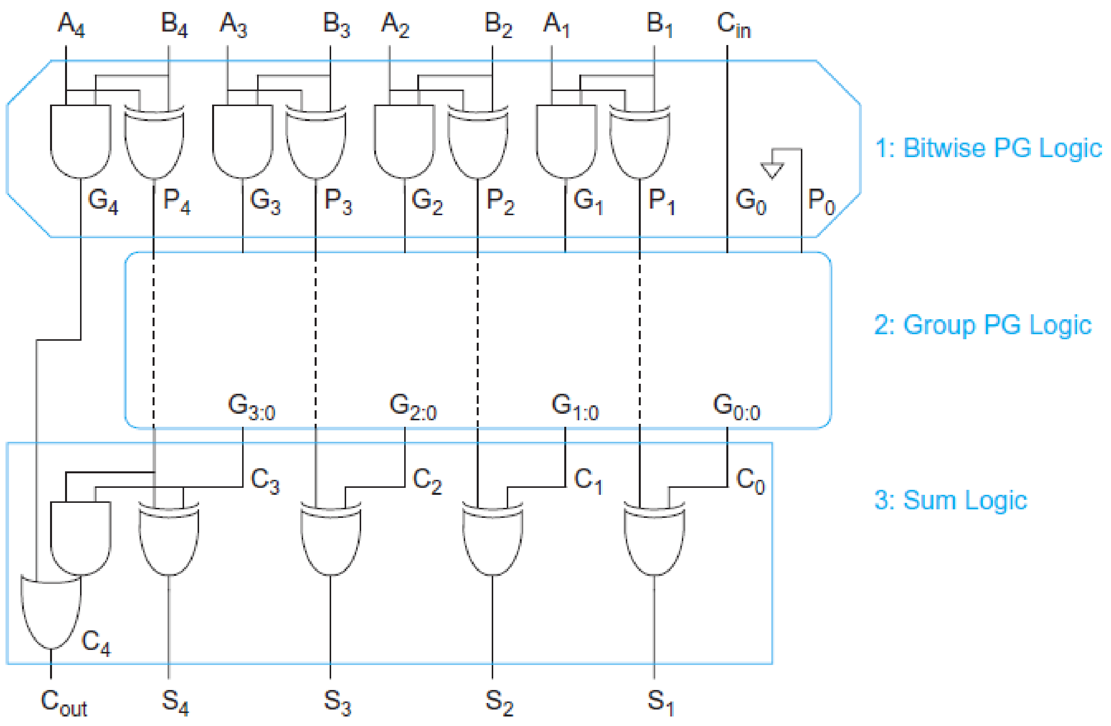


Figure 3.10 General stages of logic in Addition with generate and propagate logic [16]

For Kogge-Stone adder group PG logic is shown in Figure 3.11. The example shown in Figure 3.11 is for a 16-bit adder. So the Kogge-Stone tree has 4 stages. After each stage the number of final outputs available are 2^m where m is the stage number. Building blocks of black and grey cells used in PG logic are described in Figure 3.12.

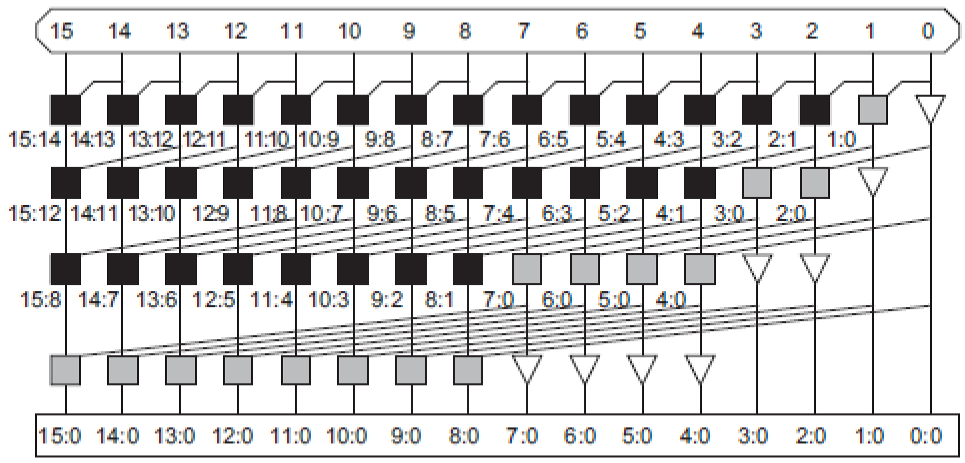


Figure 3.11 Kogge-Stone adder PG network[16]

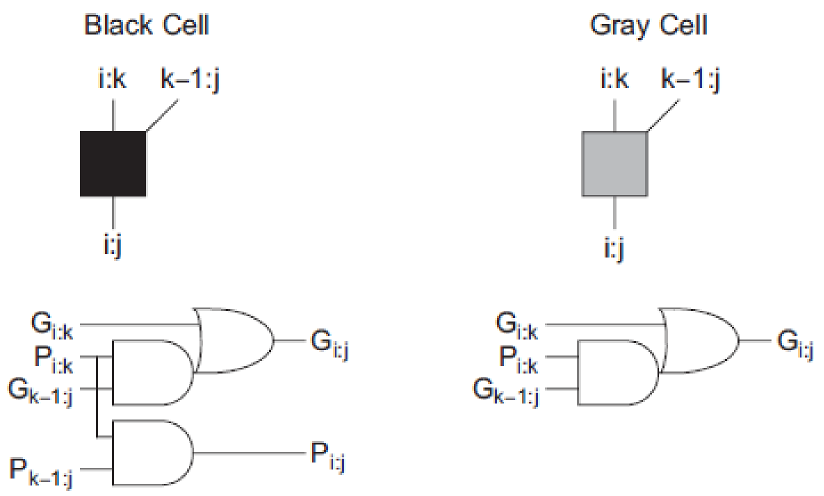


Figure 3.12 Black and Grey cell logic block diagram [16]

Black cells are used to generate group propagate and generate logic where as grey cells are used to generate group generate logic alone in final cell in each column of the tree. Only generate signals are required to compute the final sums. Equations for group generate and propagate logic and final sum computation using generate signals are given by:

$$G_{i:j} = G_{i:k} + P_{i:k}\, G_{k-1:j}$$

$$P_{i:j} = P_{i:k}\, P_{k-1:j} \qquad \text{where } i \geq k > j$$

$$S_i = P_i \oplus G_{i-1:0}$$

3.3.2 Carry-Select Adder

In carry-select adder outputs for both possibilities of carry in '0' and carry in '1' are precomputed and a multiplexer is used in the last stage to choose between the two sums based on actual carry in. Hence, the critical path of carry propagation is accelerated by the benefit of parallel computation. Performance of carry-select adder in terms of speed, area or power depends on the block sizes used. Due to parallel computation using duplicate hardware, area of carry-select adder is generally more. To gain better performance with less expense of area, individual blocks or segments could be implemented with a fast adder such as Kogge-Stone Adder.

As explained in Section 3.2, 2-bit half CLA blocks are used in the Wallace tree used to reduce the width of carry-select adder required to 24 from 28. 24-bit carry-select adder is implemented in 3 different structures. These are shown in Figures 3.13, 3.14 and 3.15. The three segmentation of 24-bits implemented using carry-select adder are

- 8-16
- 16-8
- 8-8-8

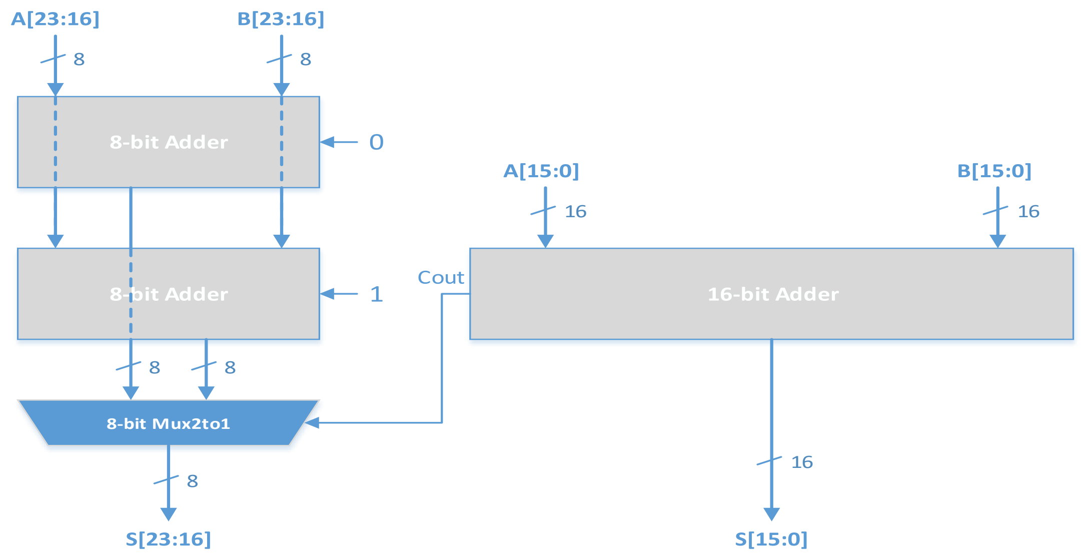


Figure 3.13 Carry-Select adder Module1 (CSA1)

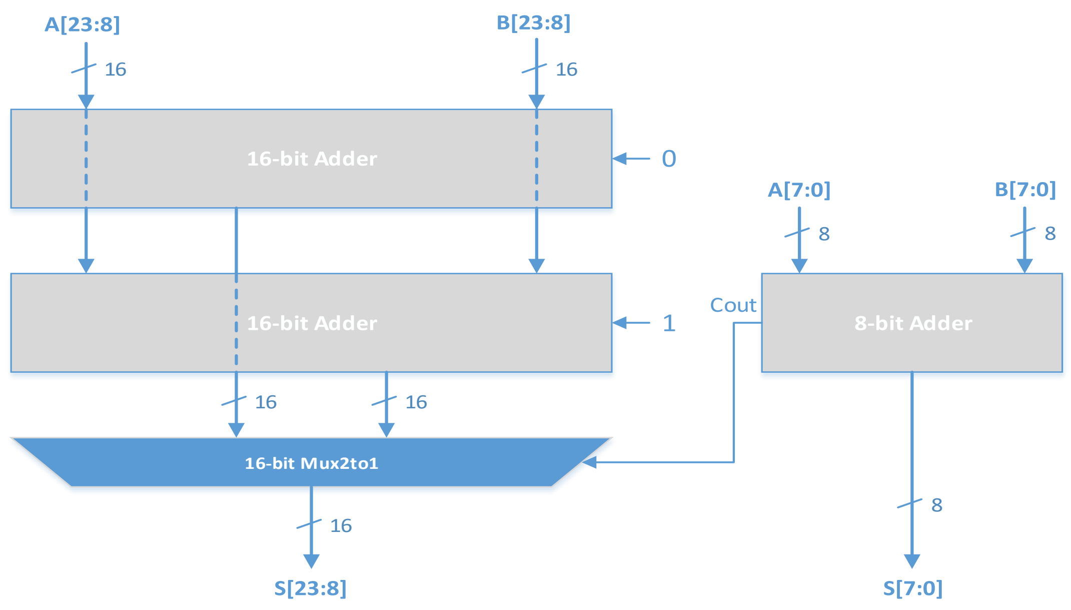


Figure 3.14 Carry-Select adder Module2 (CSA2)

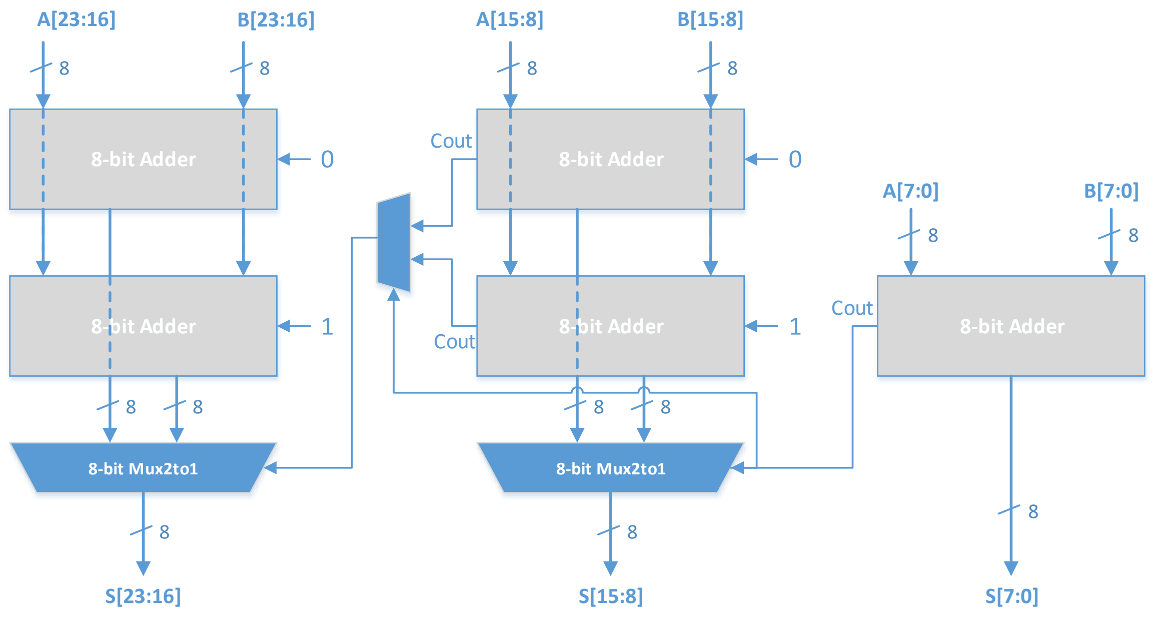


Figure 3.15 Carry-Select adder Module3 (CSA3)

16-bit and 8-bit adders in above three CSAs are implemented using Kogge-Stone Adder.

3.4 Multiplier Architectures Implemented

Two Wallace tree structures described in Section 3.2 are implemented deploying 4 adders described in Section 3.3 as the final stage CPA. So 8 different multiplier modules are implemented. Timing induced error analysis is done on each of the 8 modules. In all the multiplier modules partial product matrix is generated using a radix-4 Booth encoder.

Eight modules implemented are referred to by names given below in further chapters.

- Wallace3to2KSA: The Wallace Tree described in Section 3.2.2 is implemented with the 28-bit Kogge-Stone Adder as final CPA.
- Wallace3to2CSA1: The Wallace Tree described in Section 3.2.2 is implemented with the 24-bit carry-select adder (8-16 structure) described in Figure 3.13 as final CPA.
- Wallace3to2CSA2: The Wallace Tree described in Section 3.2.2 is implemented with the 24-bit carry-select adder (16-8 structure) described in Figure 3.14 as final CPA.
- Wallace3to2CSA3: The Wallace Tree described in Section 3.2.2 is implemented with the 24-bit carry-select adder (8-8-8 structure) described in Figure 3.15 as final CPA.
- Wallace4to2KSA: The Wallace Tree described in Section 3.2.3 is implemented with the 28-bit Kogge-Stone Adder as final CPA.
- Wallace4to2CSA1: The Wallace Tree described in Section 3.2.3 is implemented with the 24-bit carry-select adder (8-16 structure) described in Figure 3.13 as final CPA.
- Wallace4to2CSA2: The Wallace Tree described in Section 3.2.3 is combined with the 24-bit carry-select adder (16-8 structure) described in Figure 3.14 as final CPA.
- Wallace4to2CSA3: The Wallace Tree described in Section 3.2.3 is combined with the 24-bit carry-select adder (8-8-8 structure) described in Figure 3.15 as final CPA.

Chapter 4

Experiment and Result Analysis

4.1 Experimental Setup

4.1.1 Verilog Code

All the eight different multiplier modules described in Section 3.4 are implemented in System Verilog at the gate level. To perform basic time induced error analysis on different structures we need to model the delay. Delay is modeled as scaled to inverter delay for each of the basic logic gates. Delay is modeled either using # construct for basic gates or using specify construct for basic blocks like 2:1 mux.

4.1.2 Input Generation

Multiplier modules are tested for the following set of inputs:

1. Gaussian distributed inputs
2. Uniformly distributed inputs

These inputs are generated using a code written in perl. Sets of 10^6 inputs are generated. Real inputs are rounded off to the nearest integer and converted to 2's complement binary which are inputs to a Verilog testbench.

Multiplier modules are tested for 5 following sets of 10^6 inputs.

- Uniformly Distributed inputs
- Gaussian Distributed inputs with 0 mean and 10000 variance.
- Gaussian Distributed inputs with 0 mean and 7500 variance.
- Gaussian Distributed inputs with 0 mean and 5000 variance.
- Gaussian Distributed inputs with 0 mean and 2500 variance.

4.1.3 Test Bench and Output Files for Analysis

The testbench is written in System Verilog to run tests for these inputs at different clock periods. Error values in each output bit as well as the error magnitude for 10^6 inputs are written into an output file. These files are processed by MATLAB to calculate different error metrics such as SNR (Signal-to-noise ratio), BER (Bit error rate) for different multiplier modules at different clock periods for all 5 sets of inputs.

Partial products are generated parallelly using radix-4 Booth recoder. Hence, all the partial products of the partial product matrix could be assumed to be available simultaneously at inputs of Wallace tree structure. Thus the following analysis shows the effects of timing-induced error on different Wallace tree implementations and different final phase adders having different critical paths.

4.2 Result Analysis

It's important to understand the timing error before analyzing the results and drawing conclusions. Timing error occurs when delay of a path from any input bit to any output bit is less than the clock period applied. Hence when an input bit through a particular path to output doesn't have sufficient time to reach the output bit, then the current output is influenced by current as well as previous inputs. Current inputs change output value through the paths that have delay less than the clock period applied, and previous inputs influence the output through the critical paths which have longer delay than the clock period. If clock period applied is T and delay of the longest path of the circuit is $D > T$, then current output depends on $n = \lceil D/T \rceil$ inputs, i.e., current inputs as well as n-1 previous inputs. So there are three cases from a particular input bit position to a particular output bit position.

Case I: All the paths from that input bit to respective output bit are shorter than clock period applied.

Case II: All the paths from that input bit to respective output bit are longer than clock period applied.

Case III: Few paths are shorter and few are longer than the clock period applied.

The output bit value depends on only current input bit value, on only previous input bit values, on current and previous input bit values for case I, case II, case III respectively. Hence, if input bit positions which have longer paths switch more frequently, error frequency in output will increase.

Number of critical paths that fail with over clocking or scaling depends on the architecture. The frequency of these critical paths being activated or invoked depends on the nature of inputs and the architecture. So even when insufficient time is available for the multiplier to finish, a fair number of correct results are obtained for a given set of inputs. This number depends on number of paths that are failing and how often these paths are activated.

Error magnitude on the other hand not only depends on the paths failing, but also depends on the correlation between switching errors in output bits and the weights of the output bits. So there might be cases where average BER is less, but average error magnitude is higher. To understand how failing timing paths affect output error completely, different error metrics should be observed.

4.2.1 Error metrics

The error metrics used in the following analysis are:

- SNR: Signal to noise ratio is defined as signal to noise power ratio. SNR is calculated for all multiplier modules at different clock periods to understand the SNR pattern for different modules with clock period as well as which architecture is better in terms of SNR. SNR is one of the important criteria in signal processing systems.

  SNR = Signal power / Error power

- Success rate: Probability of error occurrence at multiplier module output for a given set of inputs. Success rate is observed for different multiplier modules with scaling or over clocking for all 5 sets of inputs.

  Success rate = no. of erroneous outputs / total number of inputs

- Error probability in Output bit positions: Probability of error occurrence in individual output bit positions are calculated for different clock periods. Error probability patterns give information about which bit positions are more prone to error in different architecture and also give information about which architecture might have higher error magnitude.

  SNR, success rate, individual bit error probabilities, error magnitude etc together are used to decide which architecture is better.

4.2.2 Gaussian Distributed Inputs Vs Uniformly Distributed Inputs

Booth Recoded Wallace tree Multiplier with 28-bit Kogge-Stone Adder in the final stage is tested with 10^6 Gaussian distributed inputs (mean = 0, variance = 10000) and 10^6 uniformly distributed inputs using the Verilog Testbench. The Following graph compares the SNR for the two given 2 sets of Inputs.

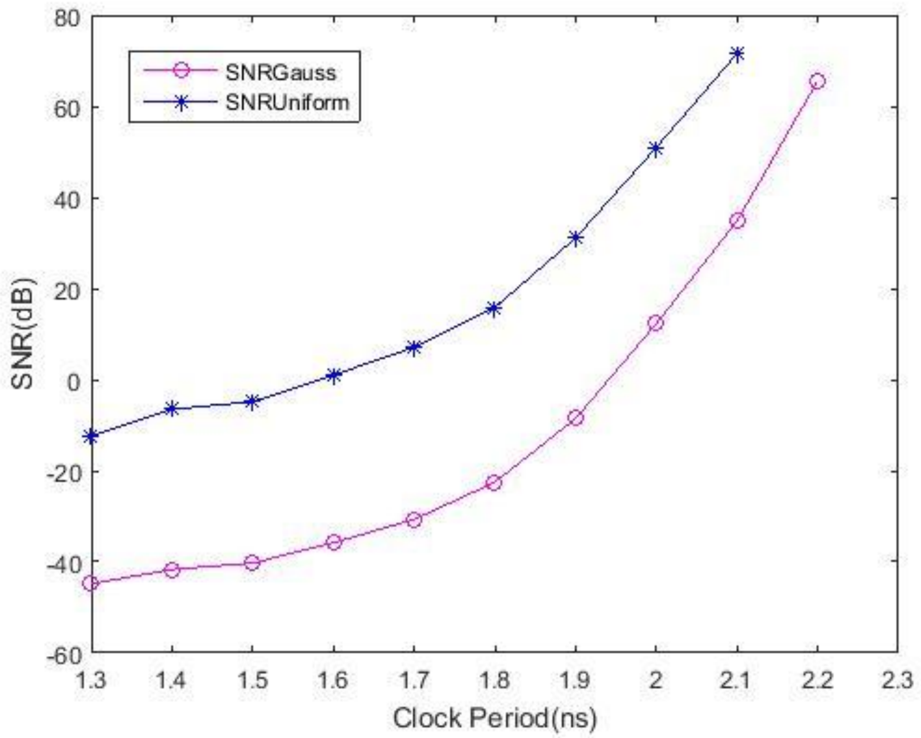


Figure 4.1 SNR vs Clock period for Gaussian and Uniform Inputs for Wallace3to2KSA

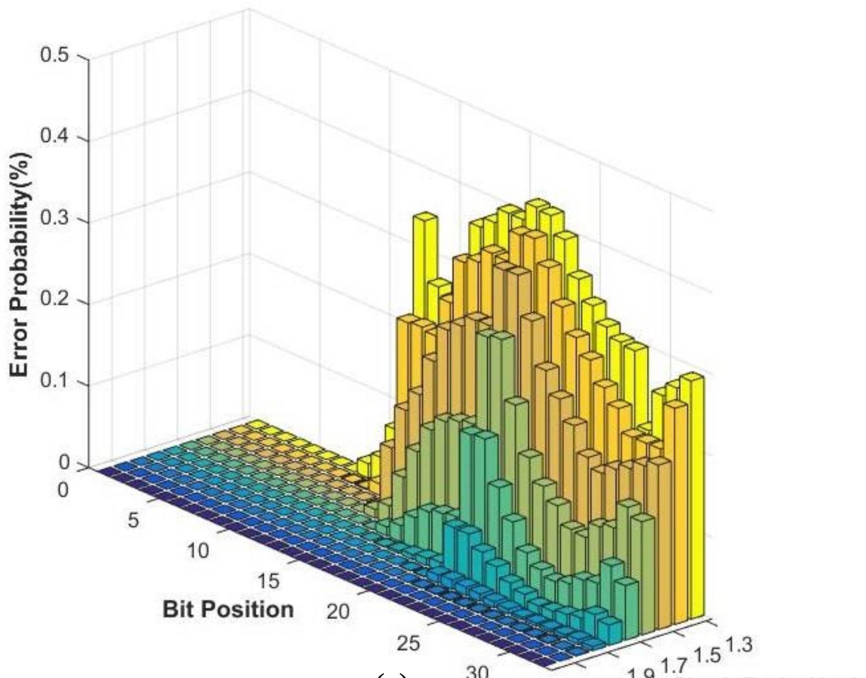
It can be observed from the graph that SNR is worse for Gaussian distributed inputs compared to uniformly distributed inputs for the same multiplier architecture. One reason for reduced SNR could be reduced signal power. Total signal power for Gaussian distributed inputs will be less than uniformly distributed inputs because more number of inputs are near 0 and hence of lesser magnitude. When signal power is reduced, unless error power is also reduced, SNR will become worse. Error magnitude reduces if error occurrence in MSB bits are reduced as when compared to the case of uniform inputs. Figure 4.2 shows the error probability for output bit positions for Gaussian and uniformly distributed inputs as clock period is reduced.

As can be seen from Figure 4.2, error probability for MSB bits 28-32 actually increase for Gaussian distributed inputs as compared to uniformly distributed inputs rather than decrease. This implies that average error magnitude is higher for Gaussian distributed inputs. This behavior collectively with the fact that signal power is less results in worse SNR for Gaussian distributed inputs.

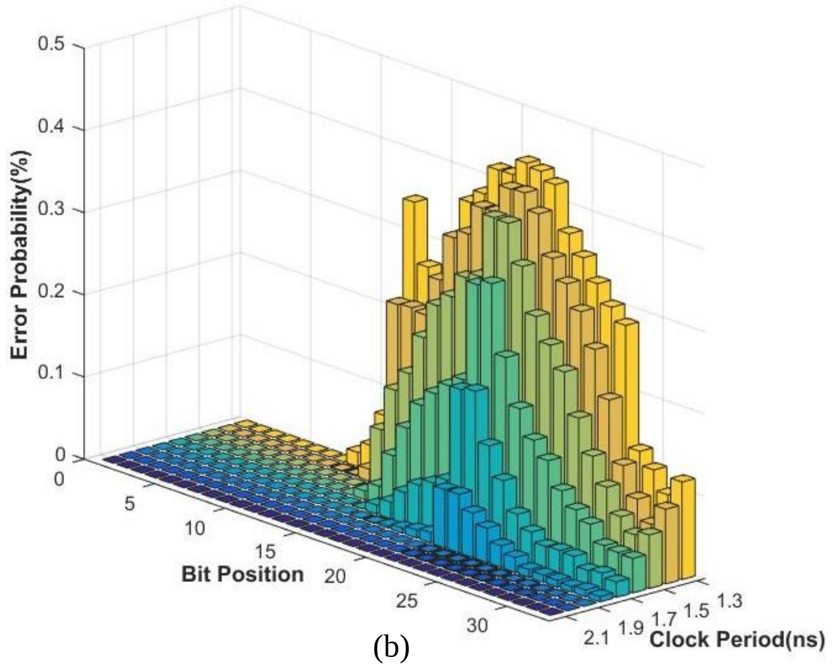
The reason for an increase in error occurrence in MSB positions when more number of inputs are of lesser magnitude could be the following. Timing induced error as explained in Section 4.2 occurs when a path's delay is less than the clock period applied. The critical paths or longer delay paths which fail are generally those paths through which signals propagate from the LSB position to the MSB position. In general these paths are activated less frequently. Consider a N-bit adder. Carry in to this adder will affect all the way from LSB to MSB (nth position output bit) when output without Cin is a string of 1s and input Cin is 1 as shown below.

```
   1 1 1------n times------1
   +                       1
  -----------------------------
= 1 0 0 0 ------n times ---- 0
```

So critical paths which propagate carry in final phase CPA of the multiplier are activated more when these kind of scenarios occur. Probability of long strings of same bit in MSB bit positions is more when inputs are of lesser magnitude. Hence these paths might be activated more number of times for Gaussian distributed inputs as compared to uniformly distributed inputs, resulting in more error occurrence or error probability for MSBs for Gaussian inputs.



(a)



(b)

Figure 4.2 SNR vs Clock period for Gaussian and Uniform Inputs for Wallace3to2KSA

Success rate comparison for Gaussian and uniformly distributed inputs for Wallace3to2KSA architecture is given in the following table.

Table 4.1 Success rate for Wallace3to2KSA at different Clock Periods

Inputs \ Clk(ns)	2.1	2.0	1.9	1.8	1.7
Gaussian Inputs	99.9623	99.3	94.4684	77.1478	43.8389
Uniform Inputs	99.9619	99.2600	94.4005	76.9430	43.1791

It can be observed from above table that the success rate remains almost the same for uniformly distributed inputs compared to Gaussian distributed Inputs. Similar success rate but lower SNR clearly show that the average computation error magnitude is higher for Gaussian distributed inputs. Gaussian distributed inputs have a worse SNR as well as a worse error magnitude.

4.2.3 Gaussian Distributed Inputs with different variances

In the previous section, performance with timing induced error for Gaussian distributed inputs with zero mean and 10000 variance was compared with uniformly distributed inputs. Variance was chosen as 10000 so that most samples will be within -30000 to 30000, since 16-bit 2's complement input range is [-32768, 32767). According to the reasoning of the above section for bad performance for Gaussian inputs, if variance of the Gaussian is reduced, number of errors and error magnitude should increase. SNR should also become worse with reducing variance due to a higher number of input samples with lesser magnitude. Figure 4.3 shows SNR plot is for Wallace3to2KSA compared for Gaussian inputs with zero mean and variances = 10000, 7500, 5000, 2500.

It can be observed from Figure 4.3 that, SNR does become worse as variance is reduced. The pattern in which SNR reduces with clock period remains same as variance decreases. The amount of reduction in SNR is increased as variance is reduced, i.e., reduction in SNR is not linearly related to reduction in variance. This strongly shows that as the amount of inputs with lesser magnitude increases, the computation error magnitude

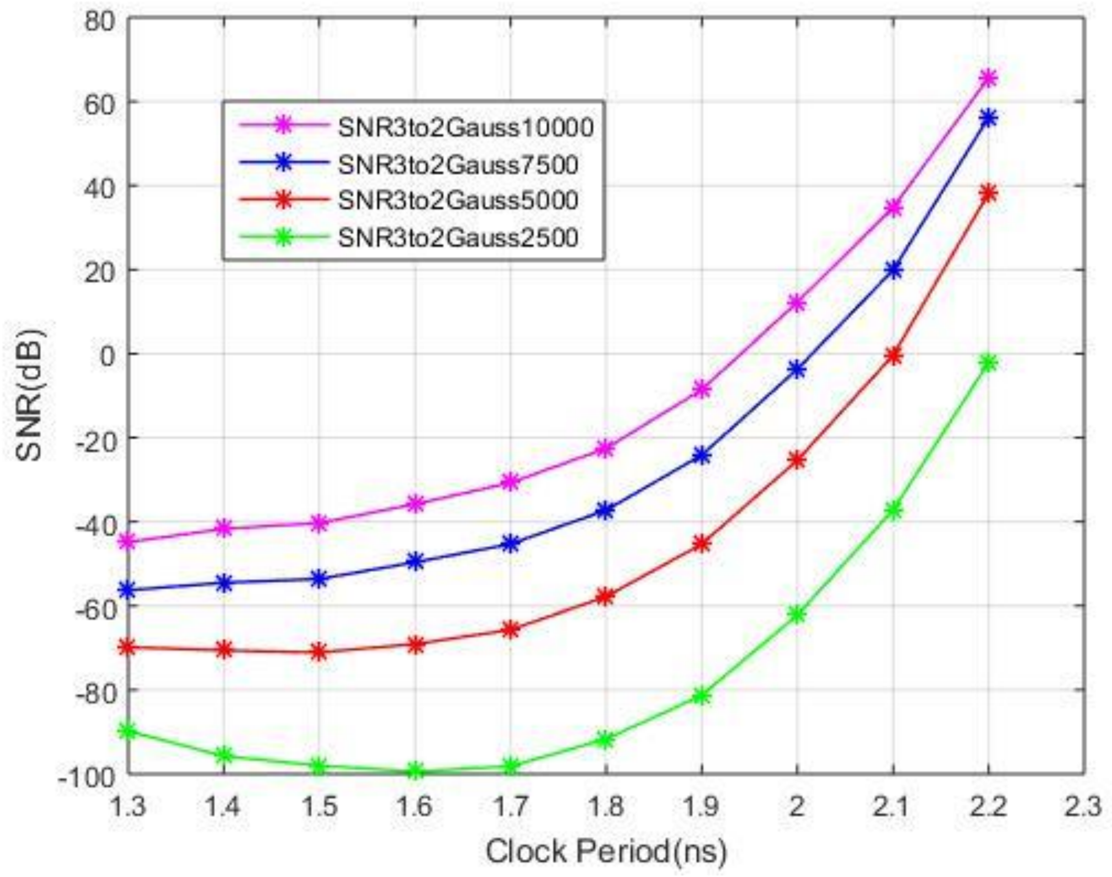
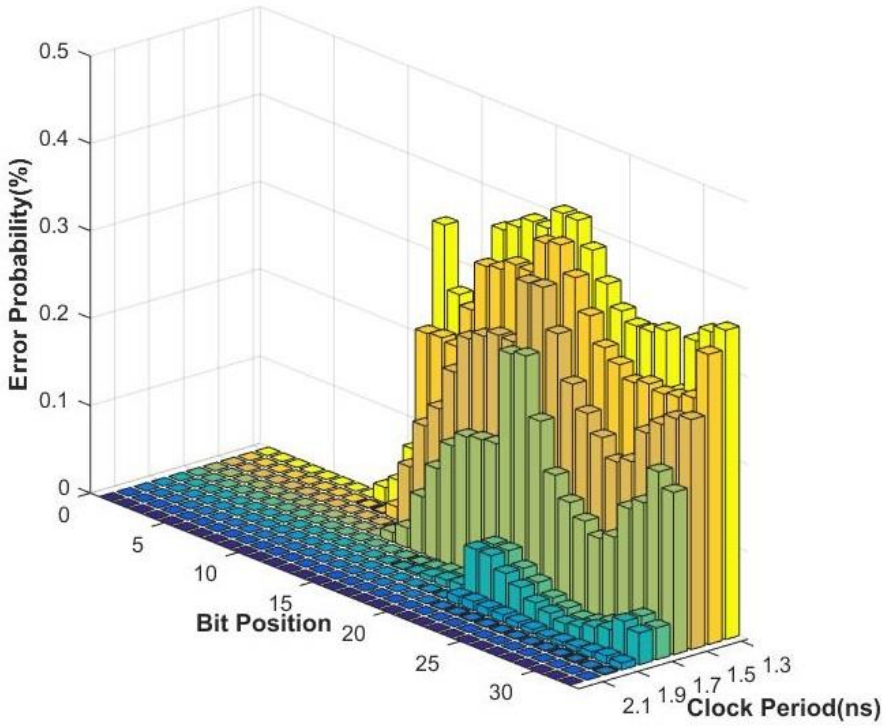


Figure 4.3 SNR plot with Different input variance for Wallace3to2KSA

increases, thereby proving that critical paths of MSB bits are activated more often as explained in Section 4.2.2. This can be confirmed by error probability plots for output bit positions for variances 7500, 5000, 2500 in Figure 4.4(a), Figure 4.4(b), and Figure 4.5, respectively. From these figures it can be observed that

- Probability of error occurrence in MSB bits 27-32 increases clearly as variance is reduced.
- Probability of error occurrence in other bit positions remains almost same.

This is consistent with our expectation of increase in computation error magnitude as well as a decrease in SNR.



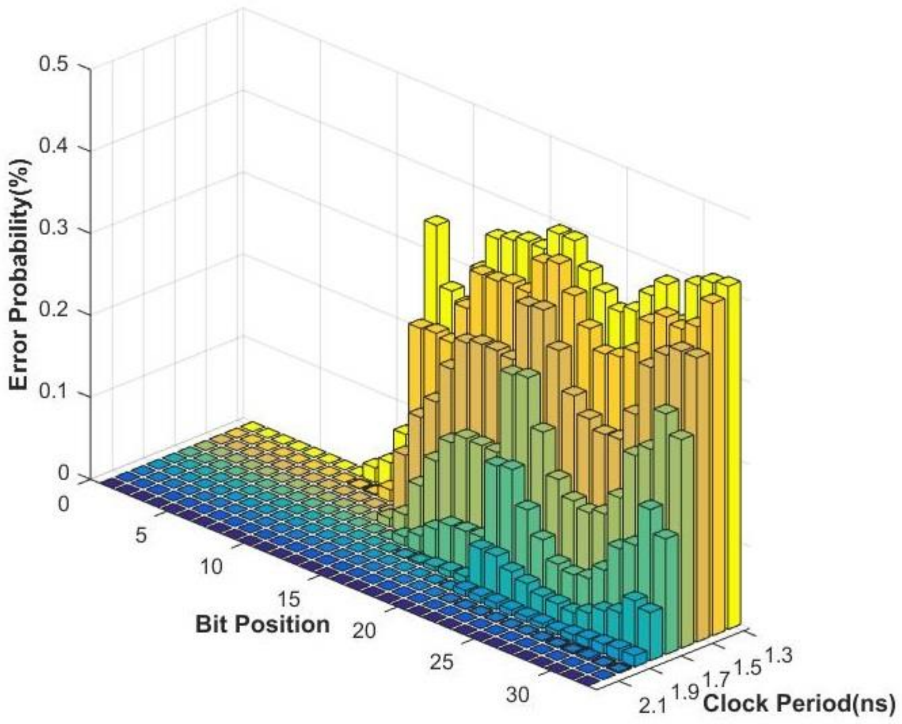


Figure 4.4 SNR Vs Clock Period for Gaussian inputs with variance (a) 7500 (b) 5000

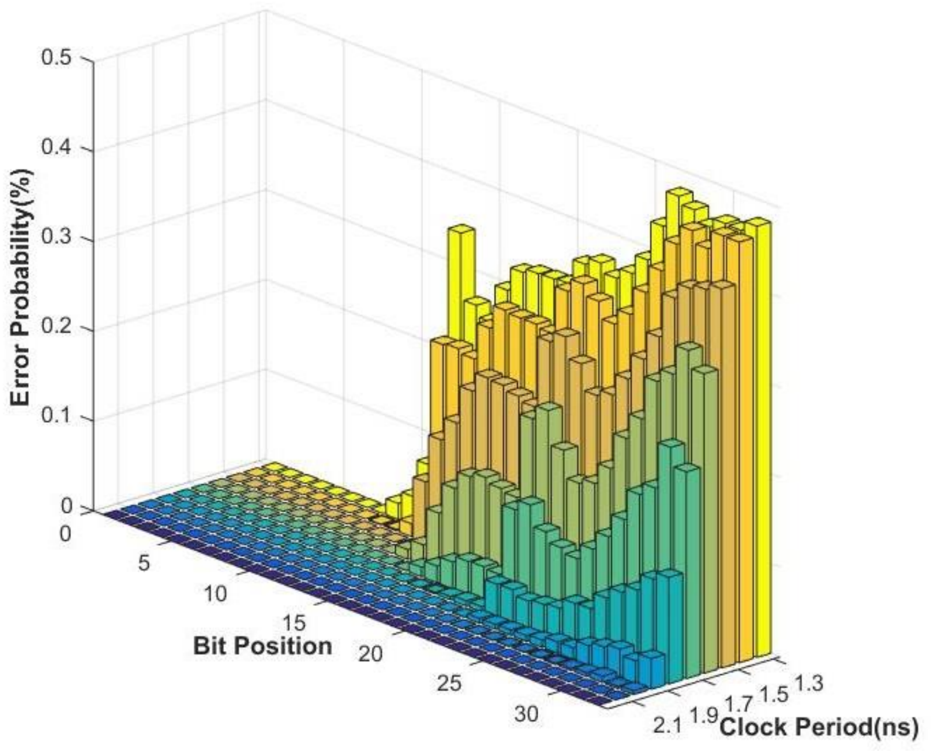


Figure 4.5 SNR Vs Clock Period for Gaussian inputs with variance 2500

Success rate variation as variance is reduced is shown in Table 4.2. This table shows that success rate does decrease as variance decreases but not as drastically as SNR. Hence, the computation error magnitude will be more as variance reduces.

Table 4.2 Success rate for Wallace3to2KSA for Gaussian inputs of different variances

Variance\Clk	2.1	2.0	1.9	1.8
10000	99.9623	99.3	94.4684	77.1478
7500	99.96	99.2350	94.1965	76.7172
5000	99.9542	99.0730	93.4540	75.1929
2500	99.9328	98.2759	89.8329	69.2394

From all observations it can be concluded that the probability of error occurrence as well as the probability of higher computation error magnitude increases as variance reduces, i.e., as higher number of inputs are of lower magnitude.

4.2.4 Wallace tree with Carry Select Adder

As explained in Section 3.4, modified Booth recoded Wallace tree with three different carry select adders is implemented. The effect of these carry select adders on the output error as compared to Kogge-Stone Adder is analyzed here. The following figure shows SNR comparison between 4 structures as clock period is reduced for Gaussian distributed inputs.

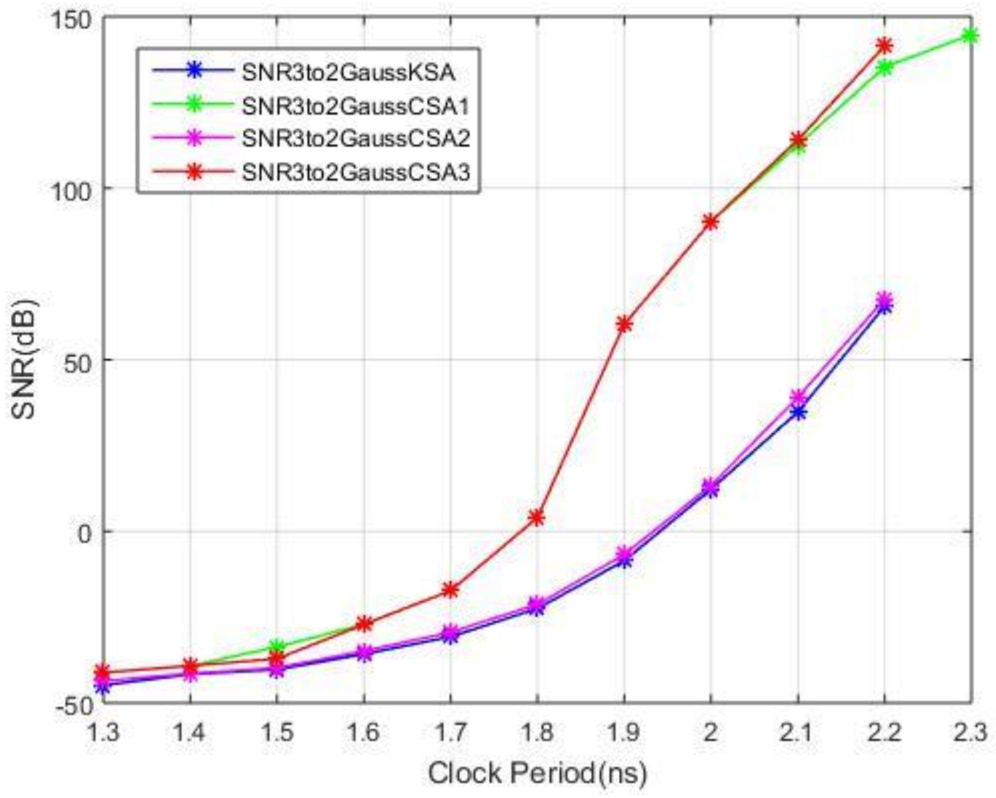
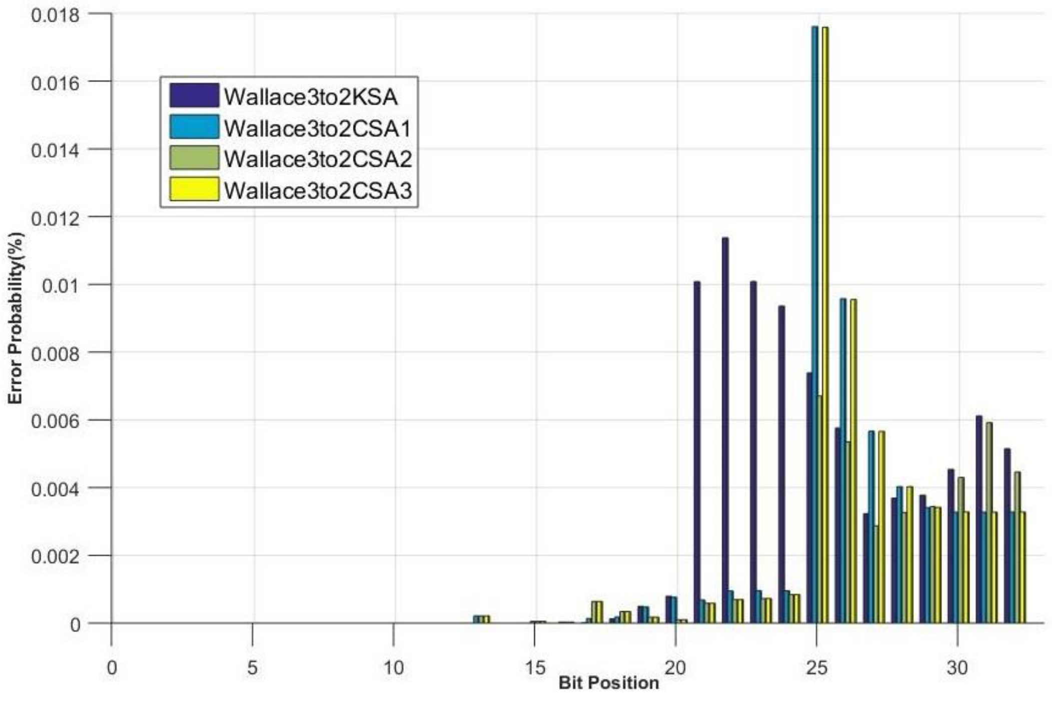
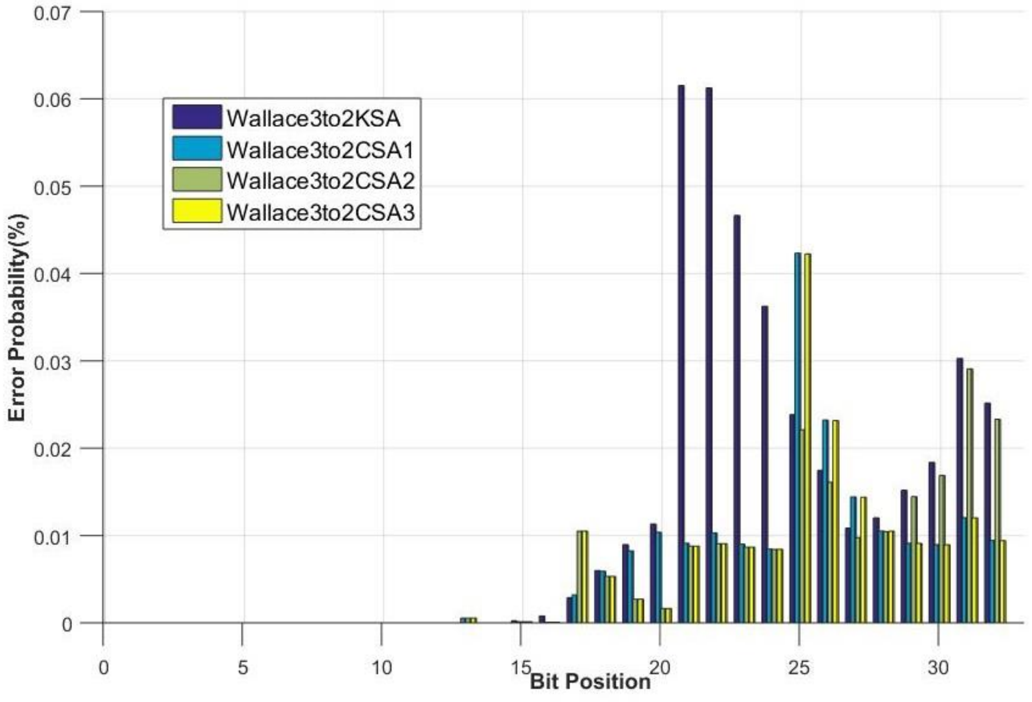


Figure 4.6 SNR Vs Clock Period for Wallace3to2 with different Final Stage CPA

It can be clearly seen from the Figure 4.6 that SNR is improved when CSA1 or CSA3 module is used over Kogge-Stone Adder where as CSA2 performs almost equivalent to Kogge-Stone Adder in terms of SNR. Figure 4.7 shows error probability for output bit position at Clock period 1.9ns and 1.8ns for all 4 multiplier modules.



(a)



(b)

Figure 4.7 Error probability for output bit positions at Clock period at (a) 1.9ns (b) 1.8ns

As can be seen from Figure 4.7 Multiplier module using Kogge-Stone Adder has peak probability of error in bits 21-24 output bit positons. Both CSA1 and CSA3 have peak probability of error in bits 25-26. Though from the graph it might seem CSA1 and CSA3 have higher BER, SNR is actually worst for KSA and CSA2. This is because error occurrence in MSB bit positions 29-32 is higher for Multiplier with Kogge-Stone and CSA2 than CSA1 and CSA3. SNR depends on the magnitude of the error and hence it is worst for Wallace3to2KSA and Wallace3to2CSA2. Since signal power is same for all 4 cases, it can be concluded that average error magnitude will be worst for Wallace3to2KSA and Wallace3to2CSA2 as well. Generally signal processing system's performance degradation depends on computation error magnitude. Hence, even though BER seems to be higher in Wallace3to2CSA1 and Wallace3to2CSA3 from Figure 4.7, they are better since they have better average error magnitude characteristics.

The following table compares success rate for the 4 multiplier modules under discussion at clock periods 2.2ns to 1.8ns.

Table 2.3 Success rate for different Multiplier Modules at different Clock Periods

Multiplier\Clk(ns)	2.2	2.1	2.0	1.9	1.8
Wallace3to2KSA	99.9995	99.9623	99.3	94.4864	77.1478
Wallace3to2CSA1	99.997	99.9534	99.5706	97.8418	90.9903
Wallace3to2CSA2	99.9999	99.9728	99.6123	97.3836	88.9488
Wallace3to2CSA3	99.9975	99.9596	99.5783	97.9497	91.9670

As can be seen from the above table, success rate reduces quickly for Wallace3to2KSA when compared to others. For Wallace3to2CSA2 though success rate is almost same as compared other two modules, it has higher average error and lower SNR due to more error probability in MSB bits. CSA2 is a 24-bit wide carry select adder with 16-8 structure as explained in Section 3.3. In this carry-select adder structure, two possible addition results for higher 16-bits will be precomputed and are simply selected by carry-in from lower 8 bit adder. So LSBs of the higher 16-bit segment should be prone to more errors than MSBs of that segment if all inputs were to arrive at the same time. But as can be seen from graphs, it can be observed that error occurrence is more in MSBs rather than LSBs of higher segment. This can be attributed to two factors. One is because 16-bit adder takes longer time to finish than 8-bit adder and hence carry-in for multiplexer is available earlier than 16-bit section outputs. Hence critical paths due to carry propagation within 16-bit adder fail even with little scaling or reduction in clock period. Secondly, due to unequal arrival times of inputs from Wallace tree, this effect is increased further. The following Figure 4.8 shows the typical delay distribution of output of Wallace tree section from [10].

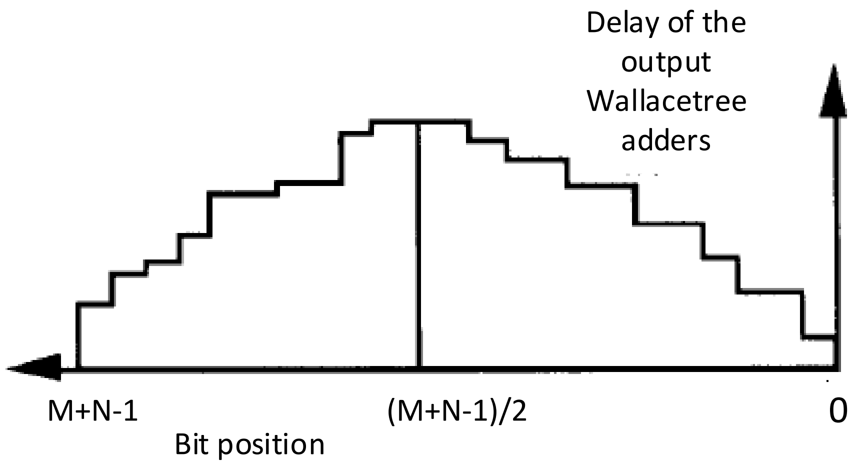


Figure 4.8 Typical Wallace tree timeline [10]

So the Wallace tree output bits which arrive last are 16-18 bits. So for CSA3, LSB inputs for higher 16-bit adder's segment arrive late, due to which 16-bit adder takes a longer time to finish and there by degrading the advantage of parallel computation of carry-select adder. For the same reason of arrival time of input signals, CSA1 and CSA2 have an advantage, since upper 8-bits of Wallace tree output are available earlier. So higher 8-bit adder segment finishes earlier than the arrival time of carry in. So critical path for these structures for higher 8 MSBs will be through carry-in or select line of the multiplexer. But signal through this critical path will change MSBs of that particular segment less frequently than the frequency with which it will change LSBs of that segment. This is because Cin affects all the way from LSB to MSB when the actual sum is all 1s and carry in is 1. The probability of occurrence of this scenario is less. So input arrival time explains that Wallace3to2CSA1, Wallace3to2CSA3 have an advantage over others because of parallel computation and CSA structure resulting in more errors in LSBs of each segment which explains the high error probability in bits 25-27 for Wallace3to2CSA1 and Wallace3to2CSA3.

Kogge-Stone Adder has a lower performance in terms of success rate or SNR as compared to others. Though Kogge-Stone Adder is a fast adder, it is not very efficient for voltage scaling or over clocking because it has many critical paths. Hence more number of paths fail simultaneously even with little scaling or reduction in clock period resulting in faster degradation of success rate. Since Kogge-Stone Adder has more number of longer critical paths for MSB positions unlike CSA, computation error magnitude is larger as compared to CSA. For the same reason, Wallace3to2CSA2 performs worse in terms of SNR, because Kogge-Stone Adder is used for 16-bit adder segment and it doesn't get the benefit of parallel computation of CSA structure due to input arrival times.

Following Figure 4.9 shows 28-bit Kogge-Stone Adder. In figure 4.9 paths to output $G_{27:0}$ from all inputs is shown. The paths colored red are the longest paths from inputs to $G_{27:0}$ having 4 black stages and 1 grey stage. Hence as explained MSBs of KSA have more number of longer paths than CSA. If the times inputs are available to Kogge-Stone from Wallace tree are considered, the longest paths are from 12-14 input bit positions which are 16-18 output bit positions of Wallace tree.

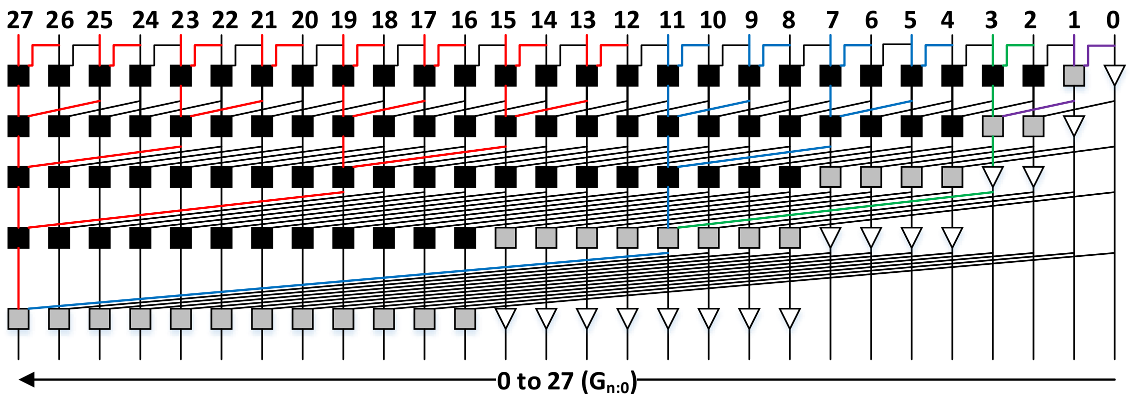


Figure 4.9 28-bit Kogge-Stone Adder

4.2.5 Wallace tree with 4to2 Compressors

When Wallace tree with 4:2 Compressors was implemented, SNR improved compared to Wallace Tree with only 3:2 Compressors.

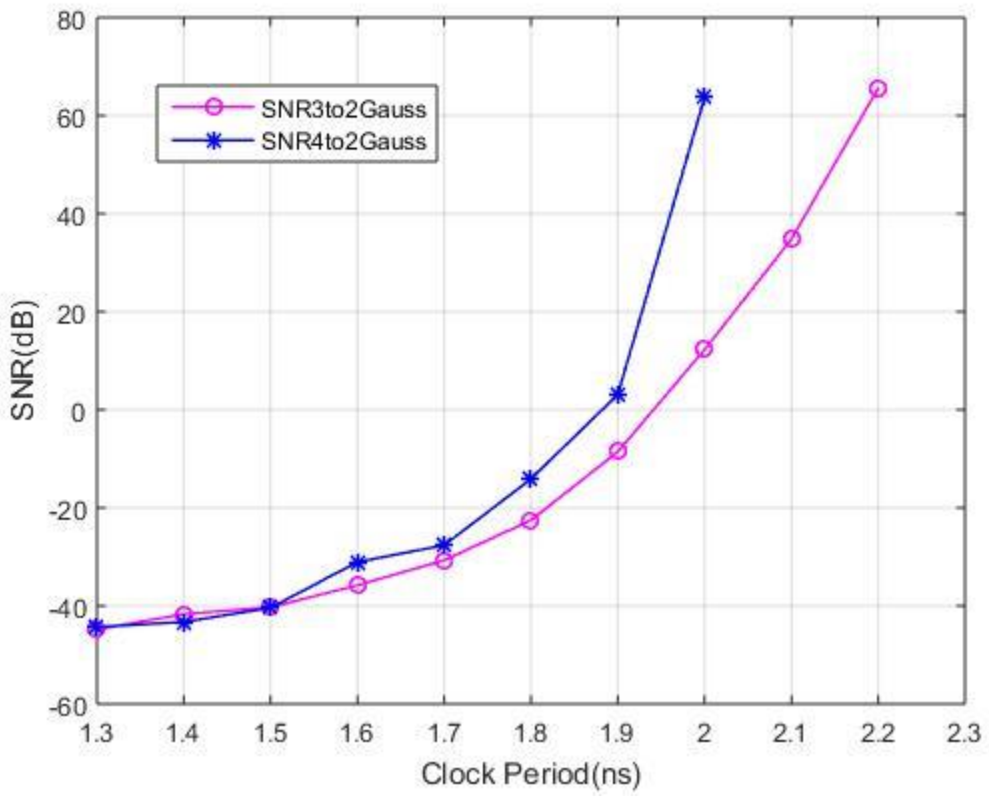


Figure 4.10 SNR Vs Clock Period for Wallace3to2KSA, Wallace4to2KSA for Gaussian Inputs

Minimum clock period for Wallace Tree Multiplier using 4:2 compressors is lesser than that of Wallace tree Multiplier using 3:2 compressors as expected. It can be observed from the above graph that SNR for Wallace4to2 at 2ns is almost equal to SNR for Wallace3to2 at 2.2ns. But if clock period is reduced further to 1.9ns (i.e., all the paths which need 1.9ns or less will finish), SNR drops by a large amount. This shows that in Wallace4to2 module there are a lot of paths which require 2ns. After this point, SNR reduces gradually as for Wallace3to2. Also the difference in SNR between the two modules is less when the clock period is reduced to less than 1.9ns.

4.2.6 Comparison of all Multiplier Modules for Gaussian Distributed Inputs

The following Figure 4.11 shows SNR plots for all eight multiplier modules for Gaussian distributed inputs.

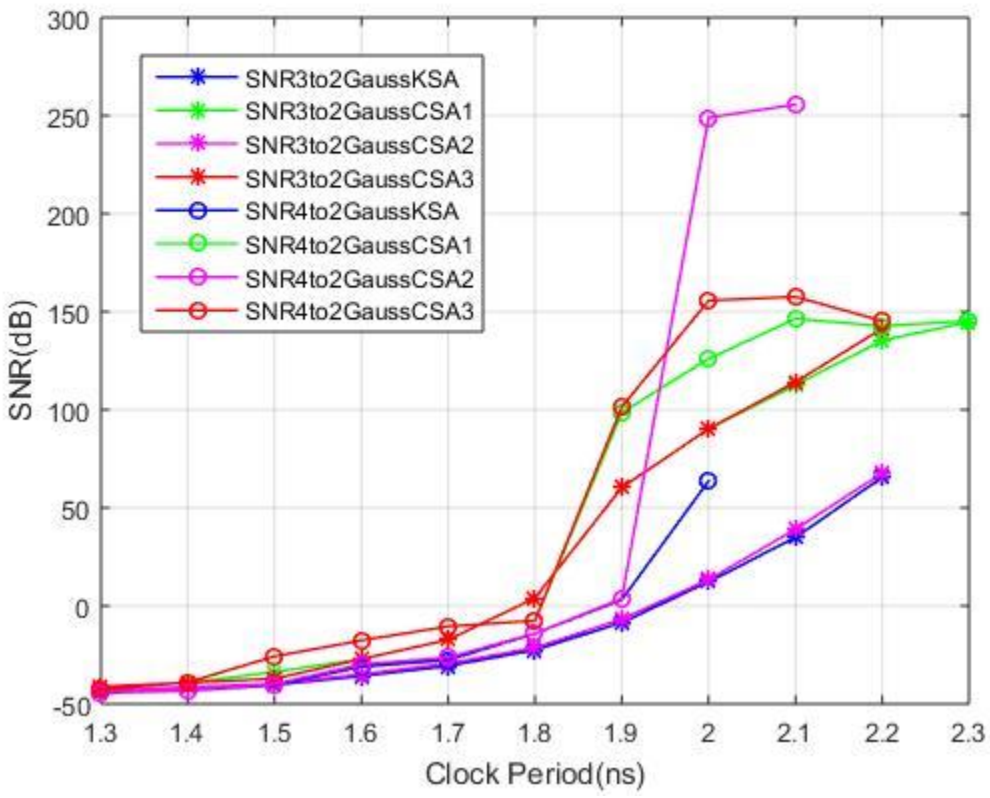


Figure 4.11 SNR Vs Clock Period for Eight Multiplier Modules

As can be seen in the Figure 4.11, Wallace4to2 has higher SNR than Wallace3to2. But, SNR for both Wallace4to2CSA1 and Wallace4to2CSA3, first increases and then decreases in SNR. This doesn't imply that error occurrence follows this pattern. Error occurrence increases with decrease in Clock period as does the bit error rate. But SNR depends on the magnitude of error. Since errors in few bits can compensate for each other, sometimes even if more bits are prone to error with over clocking, the magnitude of error might decrease. For example, consider the following 2 cases,

Case 1: Upper 8 MSB bits have positive error, i.e., correct output bit is 0, but the multiplier result is 1. Then magnitude of error for this case is = $-1 * 2^{31} + 2^{30} + \dots\dots + 2^{24}$

Case2: Upper 9 MSB bits have positive error. The magnitude of error for this case is $= -1 * 2^{31} + 2^{30} + \ldots\ldots. + 2^{24}+2^{23}$

Even though more number of bits are prone to error in case 2, error magnitude of case 1 is higher. This happens due to negative weight of the sign bit. Due to these kinds of errors, SNR for higher clock period becomes worse than lower clock period for Wallace4to2CSA1 and Wallace4to2CSA3. As clock period is further reduced, though these error patterns might occur, jump in error occurrence is huge diminishing these effects.

4.2.7 Comparison of all Multiplier Modules for Uniformly Distributed Inputs

Following Figure 4.12 shows SNR plots for all eight multiplier modules for uniformly distributed inputs. In terms of SNR the only difference between uniform inputs and Gaussian inputs is that it is higher for uniform inputs while retaining the same pattern.

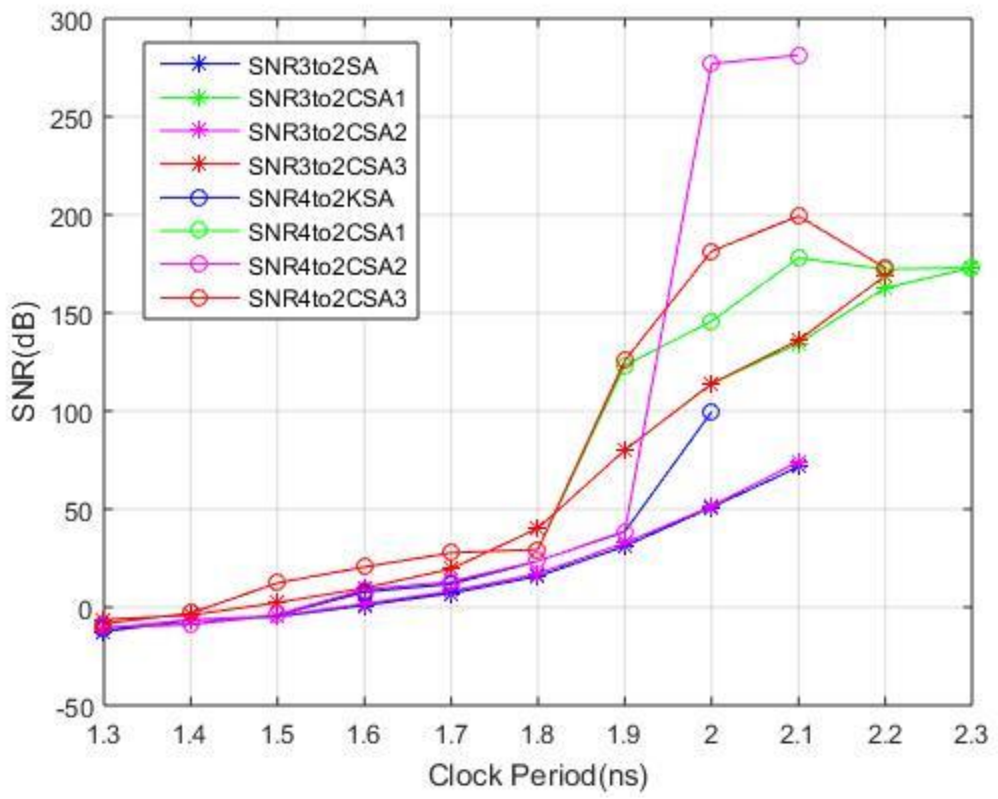


Figure 4.12 SNR Vs Clock Period for Eight Multiplier Modules for Uniform Inputs

4.2.8 Timing Error Characteristic for One Constant Input

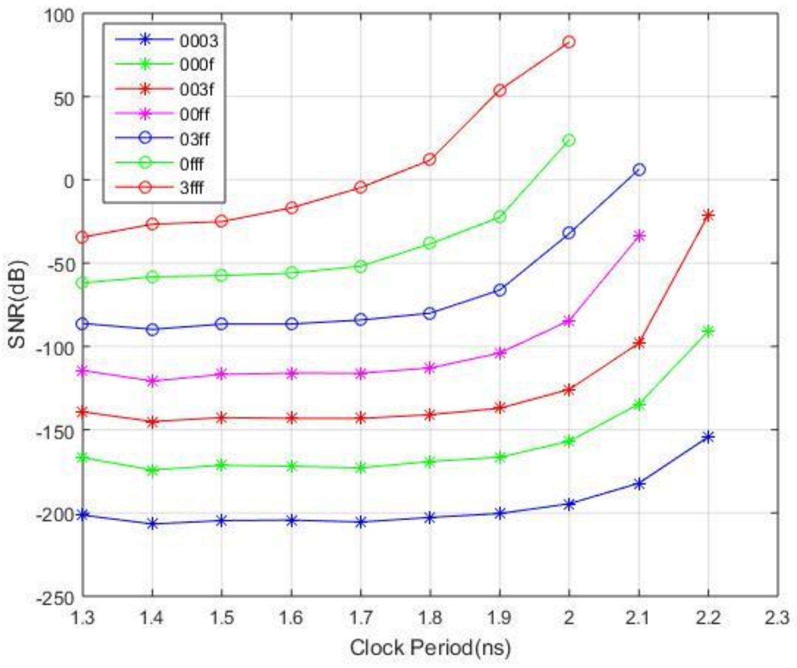
Signal processing systems many times involve having multiplications by a constant number when processing. For example, signal processing through a filter, have multiplier by constant filter co-efficients. So it will be interesting to know how constant value affects timing error characteristic.

From analysis in Sections 4.2.2 and 4.2.3, which show that error magnitude increases as inputs are of lesser magnitude, it can be expected that error magnitude and SNR will be worse when constant is of lesser magnitude. Wallace3to2KSA multiplier module is tested with one Gaussian distributed input and another constant. It is tested for varying values of constant from negative high to positive high numbers to understand the effects. Multiplier module is tested for the following constant inputs shown in Table 4.4.

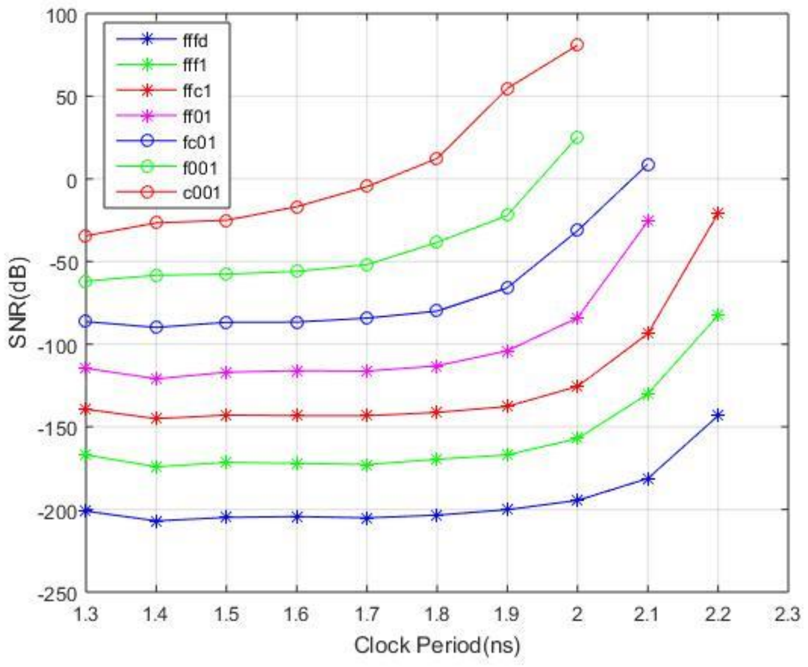
Table 4.4 Positive and negative constant inputs for which Wallace3to2KSA is tested

Constant value in Hex	**Integer value of the constant**	**Constant value in Hex**	**Integer value of the constant**
16’h0003	3	16’hfffd	-3
16’h000f	15	16’hfff1	-15
16’h003f	63	16’hffc1	-63
16’h00ff	255	16’hff01	-255
16’h03ff	1023	16’hfc01	-1023
16’h0fff	4095	16’hf001	-4095
16’h3fff	16383	16’hc001	-16383

Figure 4.13 shows SNR plots for different constant inputs shown in the above table. It can clearly be seen from the graphs that SNR is improved for positive as well as negative constants as magnitude increased. Another thing to note is that, for positive constant and negative constant of same magnitude, SNR is almost the same. As the magnitude of constant input decreases, signal power of output multiplier also decreases. But error power increases rather than decreasing. Table 4.5 and 4.6 show the success rate of output for different constant values.



(a)



(b)

Figure 4.13 SNR Vs Clock period for Wallace3to2KSA module for one constant input

Table 4.5 Success rate for Wallace3to2KSA module for different positive constants

Constant\Clk(ns)	2.2	2.1	2.0	1.9	1.8
16'h0003	99.894	97.5368	88.7018	69.899	51.0998
16'h000f	99.9957	99.4411	93.2396	74.8043	55.4601
16'h003f	99.9989	99.0524	92.8638	75.4591	56.1378
16'h00ff	100	99.9666	97.9296	87.2331	64.0063
16'h03ff	100	99.9808	99.5448	93.965	75.7755
16'h0fff	100	100	99.7799	96.0748	83.0776
16'h3fff	100	100	99.8691	97.5633	86.6347

Table 4.6 Success rate for Wallace3to2KSA module for different negative constants

Constant\Clk(ns)	2.2	2.1	2.0	1.9	1.8
16'hfffd	99.9654	97.9030	88.9947	68.9437	48.9562
16'hfff1	99.9981	99.6568	93.0757	72.9470	53.0610
16'hffc1	99.9989	99.2676	92.6737	73.5315	54.4353
16'hff01	100	99.9805	98.0460	86.9392	63.1925
16'hfc01	100	99.9853	99.5783	94.2668	75.7563
16'hf001	100	100	99.8000	96.2154	82.7881
16'hc001	100	100	99.8694	97.4968	86.3638

As can be seen from the above two tables, success rate is better and higher as magnitude of constant increases. Figure 4.14 shows error probability for output bit position for 4 different constant values. Probability of error in MSB bits is also observed to increase as magnitude of constant increases. Hence it can be concluded that as magnitude of constant increases, success rate and average computation error magnitude decrease and SNR increases.

4.2.9 Power Estimation

All eight multiplier modules are synthesized using Synopsys Design Compiler using CORE65GPSVT library. The power, area values obtained from synthesis are given in the Table 4.7 and Table 4.8 respectively.

Table 4.7 Power values for different multiplier modules

Multiplier\Power(μW)	Internal Power	Net Switching Power	Total Dynamic Power	Cell Leakage Power	Total Power
Wallace3to2KSA	266.86	249.77	516.63	34.50	551.13
Wallace3to2CSA1	251.19	239.43	490.62	32.15	522.77
Wallace3to2CSA2	299.26	286.78	586.04	36.47	622.50
Wallace3to2CSA3	276.99	260.87	537.86	33.70	571.56
Wallace4to2KSA	309.94	287.22	597.17	38.34	635.50
Wallace4to2CSA1	306.91	285.00	591.91	38.12	630.03
Wallace4to2CSA2	356.84	331.62	688.47	42.26	730.73
Wallace4to2CSA3	337.45	307.04	644.49	40.75	685.23

Total dynamic power = Internal power + Net switching power

Total power = Total dynamic power + Cell leakage power

In multiplier modules using 3:2 compressors as well as with 4:2 compressors multiplier module with CSA2 has highest power and highest area. Multiplier modules with 4:2 compressors have higher power and higher area compared to their respective multiplier modules with 3:2 compressors. Wallace3to2CSA1 is best in terms of power as well as area followed by Wallace3to2KSA. Power values obtained are at voltage 1 Volt. Approximate power estimation of power at same clock period and at lower voltage can be calculated by multiplying with square of reduced voltage. Table 4.9 gives approximate power estimation of total power for multiplier modules at lower voltages.

Table 4.8 Area for different multiplier modules

Multiplier	Area
Wallace3to2KSA	3205.80
Wallace3to2CSA1	3089.32
Wallace3to2CSA2	3405.48
Wallace3to2CSA3	3196.96
Wallace4to2KSA	3390.40
Wallace4to2CSA1	3354.52
Wallace4to2CSA2	3661.84
Wallace4to2CSA3	3450.20

Table 4.9 Power values at reduced voltage

Multiplier\Voltage(V)	0.9	0.8	0.7	0.6
Wallace3to2KSA	446.41	352.72	270.05	198.41
Wallace3to2CSA1	423.44	334.57	256.16	188.20
Wallace3to2CSA2	504.23	398.40	305.03	224.10
Wallace3to2CSA3	462.96	365.80	280.06	205.76
Wallace4to2KSA	514.76	406.72	311.40	228.78
Wallace4to2CSA1	510.32	403.22	308.71	226.81
Wallace4to2CSA2	591.89	467.67	358.06	263.06
Wallace4to2CSA3	555.04	438.55	335.76	246.68

Chapter 5

Conclusion and Future Ideas

5.1 Conclusions

Timing error statistics depend on the architecture as well as the input distribution. Architecture suitable for voltage over-scaling or over-clocking depends on the distribution timing paths or critical paths. If the number of paths that fail with reducing time period is gradual, then that architecture will be more suitable. Wallace tree multipliers with CSA1 or CSA3 are better suitable for VOS than compared to Wallace tree multipliers with KSA and CSA2 in terms of timing error statistics. Though Wallace tree multipliers with CSA3 have slightly higher power and occupy little more area as compared to Wallace tree multipliers with KSA, performance difference in terms of SNR as well as success rate is much better. In terms of timing error characteristics, power and area, multiplier modules with CSA1 are best. Wallace trees using 4:2 compressors are better than Wallace trees with 3:2 compressors in terms of SNR, but have higher power and more area. Depending on specifications of power, area and the extent to which error can be tolerated respective architecture suitable can be chosen.

Gaussian distributed inputs have significantly lower SNR than uniformly distributed inputs for the same architecture and clock period. Hence, in applications where signal distribution is expected to be Gaussian, it is necessary to test the architecture for voltage scaling with Gaussian inputs. Though error occurrence is not affected much with input distribution, error magnitude is affected. From comparison of multiplier architectures with Gaussian distributed inputs to uniformly distributed inputs, it can be learned that error magnitude is less when number of inputs in the distribution are of higher magnitude. Because of this observation, multiplier architectures were analyzed for error statistics when one of the input remains constant. As expected from above conclusion, as the magnitude of constant decreases, error magnitude is increased. In addition to error magnitude, error occurrence increases as well. Hence, in DSP applications such as FFTs or Filter, where multiplication with constant is a common, it can expected that voltage over-scaling performs better when magnitude of the constant is high.

5.2 Future Ideas

Combination of different multiplier architectures with different merge adders can be implemented to develop multiply-accumulate (MAC) units, which are the basic building blocks of digital signal processing systems. Implementation of FIR filters with these MAC units and performing timing-induced error analysis for real-time signals can be done.

Timing-induced error analysis on floating-point multipliers is an area that can be explored. Comparison of the performance of floating-point multipliers to fixed-point multipliers to understand which architectures perform better under voltage over-scaling or over-clocking can be done

References

[1] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Trans. VLSI Syst.*, vol. 18, no. 5, pp. 697–710, 2010.

[2] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," *IEEE Trans. Semiconductor Manufacturing*, vol. 21, no. 1, pp. 3–13, 2008.

[3] International Technology Roadmap for Semiconductors, http://public.itrs.net, 2004.

[4] S. Borkar et al., "Parameter Variations and Impact on Circuits and Micro-architecture", *DAC*, 2003, pp. 338-342.

[5] S. Bhunia and K. Roy, "Power Dissipation, Variation and Nanoscale CMOS Design: Test Challenges and Self-Calibration/Self-Repair Solutions", *ITC,* Oct 2007 , pp. 1-10.

[6] K. Kang et al., "Statistical Timing Analysis using Levelized Covariance Propagation", *DATE*, 2005, pp. 764-769.

[7] A. Agarwal et al., "Circuit Optimization using Statistical Timing Analysis", *DAC*, 2005, pp. 321-324.

[8] J. Freijedo, J. Semiao, J.J. Rodriguez-Andina, F. Vargas, I.C. Teixeira, J.P Teixeira, "Modeling the effect of process variations on the timing response of nanometer digital circuits", *LATW,* March 2011, pp. 1-5.

[9] D. Ernst, N. S. Kim, S. Das, S. Pant, R. R. Rao, T. Pham, C. H. Ziesler, D. Blaauw, T. M. Austin, K. Flautner, and T. N. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Int'l Symposium on Microarchitecture*, 2003, pp. 7–18.

[10] Weiwei Shan, Zhipeng Xu, "Timing error prediction based adaptive voltage scaling for dynamic variation tolerance", *APCCAS,* November 2014, pp 17-20.

[11] J. Tschanz, K. A. Bowman, C. Wilkerson, S.-L. Lu, and T. Karnik, "Resilient circuits - enabling energy-efficient performance and reliability," in *Int'l Conf. on CAD*, 2009, pp. 71–73.

[12] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, “MACACO: Modeling and analysis of circuits for approximate computing,” in *Int’l Conf. on CAD*, 2011, pp. 667–673.

[13] Y. Liu, T. Zhang, and K. Parhi, “Computation error analysis in digital signal processing systems with overscaled supply voltage,” *IEEE Trans. VLSI Syst.*, vol. 18, no. 4, pp. 517–526, Apr. 2010.

[14] D. Villeger and V. Oklobdzija. “Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partial Products”, in *Proc. 27th Annual Asilomar Conf. on Signals, Systems, and Computers*, volume 1, pages 781–784, 1993.

[15] S. Shah, A. J. Al-Khalili, and D. Al-Khalili, “Comparison of 32-bit multipliers for various performance measures,” in *Proc. Int. Conf. Microelectronics*, 2000, pp. 75–80.

[16] Neil H.E. Weste and K.Eshraghian, *Principles of CMOS VLSI Design: A systems Perspective*, Addison-Wesley Publishing Company, 1993.

[17] O. MacSorley, “High-Speed arithmetic in binary computers,” *Proc. IRE*, vol. 49, pt. 1, Jan. 1961, pp. 67–91.

[18] C.S. Wallace, “A suggestion for a fast multiplier,” *IEEE Trans. Electronic Computers*, Feb. 1964, pp. 14–17.

[19] L. Dadda, “Some schemes for parallel multipliers,” *Alta Frequenza*, vol. 34, no. 5, May 1965, pp. 349–356.

[20] Keshab K. Parhi, *VLSI Digital Signal Processing Systems*, John Wiley and Sons, Inc., 1999.

[21] K. Prasad and K. K. Parhi, “Low-power 4-2 and 5-2 compressors,” in *Proc. 35th Asilomar Conf. Signals, Syst. Comput.*, 2001, vol. 1, pp. 129–133.

[22] Xuan-Vy Luu, Trong-Thuc Hoang, Trong-Tu Bui, Anh-Vu Dinh-Duc, “A high-speed unsigned 32-bit multiplier based on booth-encoder and wallace-tree modifications”, in *International Conference on Advanced Tenchnologies for Communications (ATC)*, October 2014, pp. 739-744.

[23] J. Fadavi-Ardekani, “mxn booth encoded multiplier generator using optimized wallace trees,” in *Proc. 1992 IEEE Int. Conf. on Computer Design*, 1992, pp. 114-117.