LOW-POWER APPROXIMATE UNSIGNED AND SIGNED MULTIPLIERS WITH CONGIGURABLE ERROR RECOVERY

A Major Project Report Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTHAPURAMU(JNTUA)

In partial fulfilment of the Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

K. VINAY (20KB1A0464)

Under the esteemed Guidance of
Dr. K. Nagi Reddy, Professor
Department of E.C.E,
N.B.K.R.I.S.T, Vidyanagar.

(DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING)
N.B.K.R. INSTITUTE OF SCIENCE AND TECHNOLOGY
(Approved by AICTE: Accredited by NBA: Affiliated to JNTUA, Ananthapuramu)
An ISO 9001-2008 Certified Institution
Vidyanagar-524413, Tirupati District, Andhra Pradesh,
India. (2020-2024)

Website: www.nbkrist.org
Email: ist@nbkrist.org
Ph: 08624-228 247
Fax: 08624-228 257

N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY
(AUTONOMOUS)
COLLEGE WITH POTENTIAL FOR EXCELLENCE (CPE)
Affiliated to JNTUA, Ananthapuramu Accredited by NAAC with 'A' Grade
An ISO 9001-2008 Certified Institution
Vidyanagar, Tirupati District, Andhra Pradesh, India – 524413

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **"LOW-POWER APPROXIMATE UNSIGNED AND SIGNED MULTIPLIERS WITH CONGIGURABLE ERROR RECOVERY"** is a Bonafide work done by K.Vinay(20KB1A0464) in the department of Electronics & Communication Engineering, **N.B.K.R Institute of Science and Technology, Vidyanagar** and is submitted to JNTUA, Ananthapuramu, in the partial fulfilment for the award of BTech degree in Electronics & Communication Engineering. This work has been carried out under my supervision.

Dr. K. NAGI REDDY
Professor
Department of ECE
NBKRIST, Vidyanagar

Dr. G. HARINATHA REDDY
Professor & HOD,
Department of ECE
NBKRIST, Vidyanagar

Submitted for the Viva-voce Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of a project would be incomplete without the people who made it possible. Their constant guidance and encouragement crowned our efforts with success.

We would like to express our profound sense of gratitude to our project guide **Dr. K. NAGI REDDY, Professor in the Department of Electronics & Communication Engineering, N.B.K.R.I.S.T (Affiliated to JNTUA, Ananthapuramu), Vidyanagar,** for his masterful guidance and the constant encouragement throughout the project. Our sincere appreciations for her suggestions and unmatched services without, which this work would have been an unfulfilled dream.

We express our gratitude and sincere thanks to **Dr. G. HARINATHA REDDY, Head of the Department, Electronics & Communication Engineering** for his significant suggestions and help in every aspect to accomplish the project work successfully.

We are grateful to **Dr. V. VIJAYA KUMAR REDDY, DIRECTOR, of N.B.K.R Institute of Science and Technology** for allowing us to utilize all the facilities in the college.

We would like to convey our special thanks to **Sri. N. RAM KUMAR REDDY, Respectable Correspondent of N.B.K.R Institute of Science and Technology,** for providing excellent infrastructure in our campus for the completion of the project.

We would like to convey our heartful thanks to Staff members, Lab Technicians, and our friends, who extended their cooperation in making this project as a successful one.

We Would like to thank one and all who have helped us directly and indirectly to complete this project successfully.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In the field of Digital Signal Processing and similar applications, Approximate circuits are being explored as a means to enhance performance and energy efficiency by sacrificing a degree of accuracy. Within these circuits, Multipliers play a crucial role and are being investigated for their potential impact on overall system optimization. In this paper, a novel approximate multiplier with a low power consumption and a short critical path is proposed for high-performance DSP applications. This multiplier leverages a newly designed approximate adder that limits its carry propagation to the nearest neighbors for fast partial product accumulation. Different levels of accuracy can be achieved by using either OR gates or the proposed approximate adder in a configurable error recovery. The multipliers using these two error reduction strategies are referred to as Approximate Multiplier 1 (AM1) and Approximate Multiplier 2 (AM2), respectively. Both AM1 and AM2 have a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared with a Unsigned multiplication multiplier, with the signed multiplication. The signed multiplication tends to have lower power consumption is observed. By utilizing an appropriate error recovery, the proposed approximate multipliers achieve similar processing accuracy as traditional exact multipliers, but with significant improvements in power.

Keywords: Multiplier, Digital Signal Processing, Digital Systems, Digital Systems, Carry Propagation Adder

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Overview

In digital systems where human interpretation or inherent error tolerance is present, such as multimedia, recognition, and data mining applications, strict precision in computation may not be essential, opening avenues for approximate computing to significantly reduce area, power, and delay while maintaining acceptable performance levels and energy efficiency. As one of the key components in arithmetic circuits, adders have been extensively studied for approximate implementation. The so-called speculative adders operate by using a reduced number of less significant input bits to calculate the sum, because the typical carry propagation chain is usually shorter than the width (in bits) of an adder. An error detection and recovery scheme has been proposed to extend the scheme of for a reliable adder with variable latency. A reliable variable-latency adder based on carry select addition has been presented. As a number of approximate adders have been proposed, new methodologies to model, analyze and evaluate them have been discussed. One area of focus that has received comparatively less attention is the optimization of approximate multipliers, despite their fundamental role in various computational tasks. A multiplier usually consists of 3 stages: Partial product generation, Partial product accumulation and a Carry Propagation Adder (CPA) at the final stage. Partial product generation is a crucial stage in the multiplication process, particularly in digital circuits like multipliers. It involves breaking down the multiplication of two numbers into smaller, simpler operations. For example, in binary multiplication, each digit of the multiplicand is multiplied with each digit of the multiplier. These individual multiplications result in partial products, which are then added together to produce the final product. In partial product generation, each multiplication operation between a digit of the multiplicand and a digit of the multiplier is performed separately. Partial product accumulation is the stage in the multiplication process where the partial products, generated in the previous stage, are combined or accumulated to obtain the final result of the multiplication operation. In binary multiplication, after generating the partial products by multiplying each digit of the multiplicand with each digit of the multiplier, these partial products are aligned according to their positions and added together to produce intermediate sums. These intermediate sums

represent partial results of the overall multiplication operation. A carry propagation adder (CPA) is a type of digital circuit used for addition in computer arithmetic. It's designed to add two binary numbers together, producing a sum along with any carry bits that result from the addition of corresponding bits in the numbers. This project focuses on the investigation of low power consumption in both unsigned and signed multiplication operations. The findings will contribute to a deeper understanding of power-efficient arithmetic operations and may have implications for the design and optimization of digital systems. The multiplier design is proposed using a simple, yet fast approximate adder. This newly designed adder can process data in parallel by cutting the carry propagation chain. It has a critical path delay that is even shorter than a conventional one-bit full adder. Although with a high error rate, this adder simultaneously computes the sum and generates an error signal; this feature is employed to reduce the error in the final result of the multiplier.

1.2 OVERVIEW OF KEY COMPONENTS

1.2.1Multiplier

Increment is a key movement in most banner dealing with computations. Multipliers have significant district, long inaction and eat up amazing force. In this way low-control multiplier design has an essential part in low-control VLSI system plan. A system is all around directed by the execution of the multiplier in light of the way that the multiplier is all things considered the slowest segment and more region using in the structure. From this time forward streamlining the speed and area of the multiplier is one of the huge diagram issues. Regardless, domain and speed are regularly conflicting objectives with the objective that upgrades in speed results in greater locales. Duplication is a logical action that consolidate methodology of adding an entire number to itself a predefined number of times. A number (multiplicand) is incorporated itself different events as shown by another number (multiplier) to shape a result (thing). Multipliers expect a basic part in the present modernized banner taking care of and distinctive applications. Multiplier arrangement should offer fast, low power use. Increase incorporates in a general sense 3 phases

1. Fragmentary thing age
2. Fragmentary thing diminish
3. Last development

Dadda Multiplier:

The Dadda multiplier was arranged by the specialist Luigi Dadda in 1965. It's is by all accounts like Wallace multiplier yet to some degree speedier and required less doors.

Dadda Multiplier was portrayed in three phases

- Multiply the each bit of one conflict with the each and every bit of other dispute and continue until the point that all conflicts are copied
- Reduce the amount of deficient things to two layers of full and half adders.
- Group the wires in two numbers, and incorporate them with a normal snake.

1.2.2 4:2 Compressor layout:

The 4:2 Compressor has 5 inputs A, B, C, D and Cin to create 3 yields Sum, Carry and Cout as showed up in Figure. The 4 inputs A, B, C and D and the yield Sum have a comparable weight. The information Cin is the yield from a past lower basic blower and the Cout yield is for the blower in the accompanying vital stage. The general method to manage realize 4:2 blowers is with 2 full adders related serially as showed up in figure.

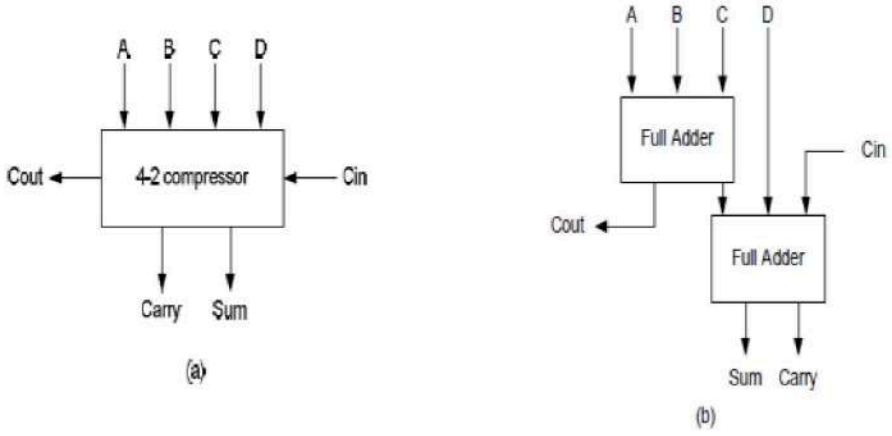


Fig.1.1 (a) 4-2 adder compressor. (b) 4-2 adder compressor implemented with full adders.

1.3 Proposed objectives:

In this project, a novel approximate multiplier design is proposed using a simple, yet fast approximate adder. This newly designed adder can process data in parallel by cutting the carry propagation chain. It has a critical path delay that is shorter than a conventional one-bit full

adder. Albeit with a high error rate, this adder simultaneously computes the sum and generates an error signal; this feature is employed to reduce the error in the final result of the multiplier. In the proposed approximate multiplier, a simple tree of the approximate adders is used for partial product accumulation and the error signals are used to compensate errors for obtaining a better accuracy.

The proposed multiplier can be configured into two designs by using OR gates and the proposed approximate adders for error reduction, referred to as approximate multiplier 1 (AM1) and approximate multiplier 2 (AM2), respectively. Different levels of error recovery can also be achieved by using a different number of MSBs for error recovery in both AM1 and AM2. As per the analysis, the proposed multipliers have significantly shorter critical paths and lower power dissipation than the traditional Wallace multiplier. Functional and circuit simulations are performed to evaluate the performance of the multipliers. Image sharpening and smoothing are considered as approximate multiplication-based DSP applications. Experimental results indicate that the proposed approximate multipliers perform well in these error-tolerant applications. The proposed designs can be used as effective library cells for the synthesis of approximate circuits.

CHAPTER 2
LITERATURE SURVEY

CHAPTER 2
LITERATURE SURVEY

2.1. Approximate Adders for approximate multiplication

Record The opening between capacities of CMOS development scaling and requirements of future application outstanding tasks at hand is growing rapidly. There are a couple of promising arrangement approaches that together can lessen this opening inside and out. Evaluated figuring is one of them and starting late, has pulled in the most grounded thought of standard scientists. Induced figuring abuses natural bumble quality of employments and features prevalent imperativeness beneficial programming and hardware utilization by trading off computational quality (e.g., precision) for computational undertakings (e.g., execution and essentialness). Consistently, a couple of research tries have explored estimated preparing all through each one of the layers of enrolling stack, regardless, most by far of the work at hardware level of thought has been proposed on adders. In [1], a relative audit of forefront unpleasant adders is given. Additionally, it in like manner gives examination in light of both standard arrangement estimations and what's more evaluated enrolling layout estimations.

2.2. Harsh Compressors for Multiplication

Harsh figuring is an engaging perspective for cutting edge getting ready at nanometric scales. Erroneous figuring is particularly entrancing for PC calculating plans. The examination and plan of two new induced 4-2 blowers are cleared up in [2] for use in a multiplier. These designs rely upon different features of weight, to such a degree, to the point that imprecision in computation (as evaluated by the screw up rate and the affirmed institutionalized misstep expel) can meet concerning circuit-based figures of estimation of a blueprint (number of transistors, deferral and power use). Four one of a kind gets ready for utilizing the proposed deduced blowers are proposed and separated for a Dadda multiplier [2]. Expansive entertainment results are given and a use of the gathered multipliers to picture planning is presented.

The results show that the proposed traces accomplish important abatements in control diffusing, deferral and transistor check stood out from a right arrangement; furthermore, two

of the proposed multiplier designs give awesome capacities to picture duplication with respect to ordinary institutionalized botch division and zenith movement to-uproar extent (more than 50dB for the considered picture outlines).

2.3. Unpleasant Wallace-Booth Multiplier

Unpleasant or obscure enrolling has starting late pulled in noteworthy thought in light of its potential purposes of enthusiasm with respect to unrivaled and low power usage. This induced multiplier [3] includes an expected Booth encoder, a vague 4-2 blower and an unpleasant tree structure. The unpleasant arrangement is executed and affirmed for 8x8, 16x16 and 32x32-piece checked increment designs concentrating on applications in embedded systems. Reenactment results at 45 nm advancement are given and analyzed. Differentiated and a right Wallace-Booth multiplier and furthermore other deduced multipliers found in the particular written work, the proposed assessed scheme achieves basic improvements in control use, delay and united estimations. These results show the common sense of the proposed plan.

2.4. Two varieties of harsh multipliers

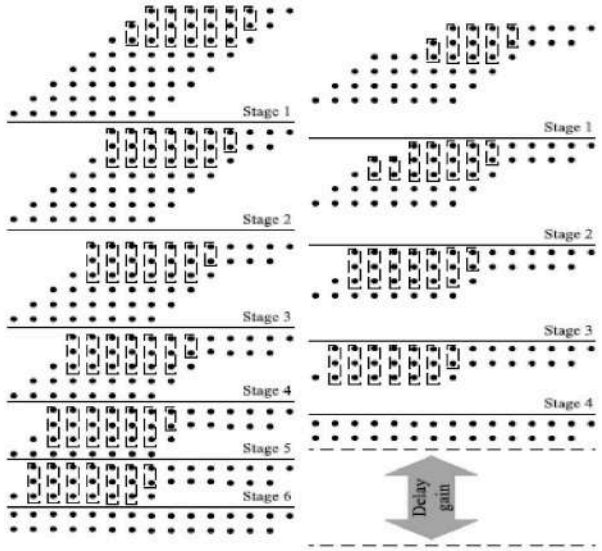
Harsh figuring can reduce the blueprint multifaceted nature with an extension in execution and power capability for screw up adaptable applications. Another arrangement approach for gauge of multipliers is discussed in [4]. The inadequate aftereffects of the multiplier are changed to display fluctuating probability terms. Basis multifaceted nature of estimation is vacillated for the total of changed midway things in perspective of their probability. The proposed estimation is utilized in two varieties of 16-bit multipliers. Amalgamation results reveal that two proposed multipliers achieve control hold assets of 72% and 38%, independently, appeared differently in relation to a right multiplier. They have better precision when appeared differently in relation to existing unpleasant multipliers.

Execution of the proposed multipliers is surveyed with a photo dealing with application, where one of the proposed models achieves the most hoisted zenith banner to upheaval extent.

The need to help distinctive electronic banner dealing with (DSP) and gathering applications on essentialness obliged contraptions has reliably created. Such applications routinely generally perform cross section enlargements using settled point calculating while in the meantime showing flexibility for some computational oversights. Hence, improving the essentialness capability of increments is fundamental. Finally, the showed computational bungle [5] does not make any unmistakable impact on the idea of DSP and the precision of request applications.

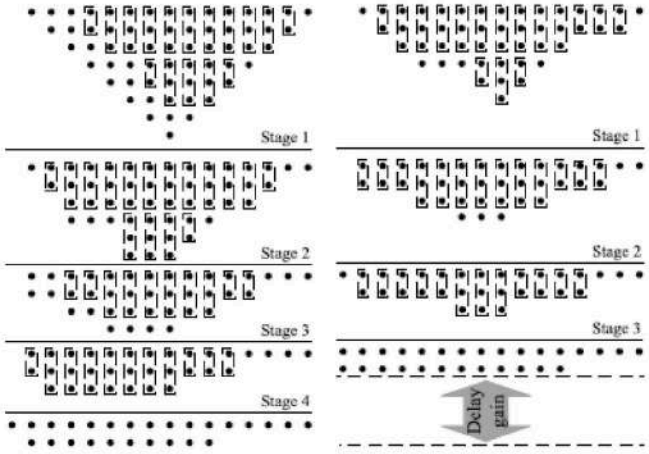
2.5. Induced Multiplier by Partial Product Perforation Technique

The diverse gathered multipliers (Array, Wallace and Dadda multipliers) are arranged by the deficient thing opening technique. The fragmented thing puncturing methodology is just to cut any two segments from the primary midway things made by the regular multipliers.
In the first place, we discussed the evaluated display multiplier. Show multiplier is prominent because of its essential structure. The enlargement of the multiplicand with one multiplier bit makes each partial thing. The made midway things are incorporated into the wake of moving based their bit orders. Pass on cause snake is used as the snake. N-1 adders are required where N is the multiplier length. The figure of show multiplier is made by fragmented thing puncturing technique. Fig. 2.1(b) shows the approximate array multiplier. In light of the result examination among correct and gauge, the figure multiplier delivers the better results with respect to deferment, district and power.



2.1(a) 2.1(b)

Fig-2.1: (a) accurate array multiplier (b)approximate array multiplier

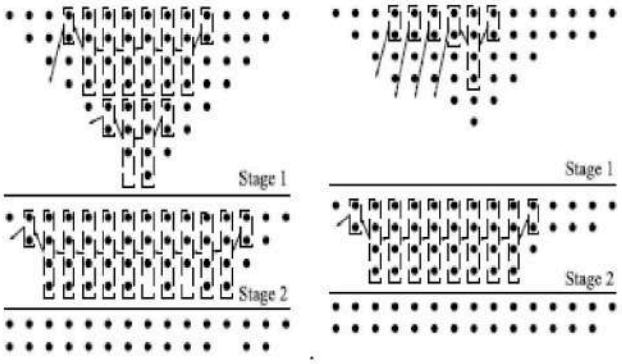


2.2(a) 2.2(b)

Fig-2.2: (a) Accurate Wallace multiplier (b) Approximate Wallace multiplier.

Second, the surmised Wallace multiplier is clarified quickly. The activity behind this multiplier is like cluster multiplier, yet the main contrast is multiplier structure. Precise and Approximate Wallace multiplier is appeared in Fig.2.2(a) and 2.2(b) individually. From

figure, blend of three focuses speaks to the full snake and mix of two focuses speaks to the half viper activities.



2.3(a) 2.3(b)

Fig-2.3: (a) Accurate Dadda multiplier (b) Approximate Dadda multiplier.

Finally, the correct and unpleasant dadda multiplier by using 4:2 blower spot traces is showed up in Fig. 2.3(a) and 2.3(b). The mix of four spots addresses the 4:2 blower assignments

In this project, we discussed the distinctive figure frameworks used in the Image planning application. This estimation system engages the parameters like high locale and power venture reserves while holding high precision. We explored thing puncturing on an extensive plan of multiplier models, surveying its impact on different outlines and mix-up limits. The diagram is incorporated between various front line estimation strategies; we exhibited that the systems achieve basic gains in power, district, and quality estimations of picture taking care of and data examination computations. Finally, these frameworks are flexible, offering better results as the multiplier's bit width increases.

CHAPTER 3
EXISTING METHOD

CHAPTER 3
EXISTING METHOD

3.1. Introduction:

In applications like sight and sound banner getting ready and data mining which can persevere through error, remedy figuring units are not continually major. They can be supplanted with their construed accomplices. Research on deduced enlisting for screw up tolerant applications is on the rising. Adders and multipliers outline the key fragments in these applications. In [1], inaccurate full adders are proposed at transistor level and they are utilized in cutting edge signal taking care of utilizations. Their proposed full adders are used in social affair of deficient things in multipliers. To diminish hardware multifaceted design of multipliers, truncation is comprehensively used in settled width multiplier designs.

By then a reliable or variable review term is added to compensate for the quantization botch displayed by the truncated part [2], [3]. Gauge techniques in multipliers revolve around gathering of partial things, which is noteworthy to the extent control usage. Broken display multiplier is completed in [4], where the smallest basic bits of wellsprings of data are truncated, while forming fragmented things to reduce gear multifaceted nature.

Disadvantages of existing system are given underneath

- More Logic multifaceted nature
- More power and more deferral

3.2. EXISTING METHOD

Execution of multiplier contains three phases:

• Generation of partial things,

• Partial things diminish tree, ultimately,

• A vector mix development to make last thing from the aggregate and pass on segments created from the diminishing tree.

Second step uses more power. In this short, estimation is associated in diminish tree organize. A 8-bit unsigned1 multiplier is used for layout to portray the proposed system in

gauge of multipliers. Consider two 8-bitunsigned input operands α = _7m=0 αm2m and β = _7n=0 βn2n.

The deficient thing am,n = αm ▪ βn in Fig. 1 is the outcome of AND assignment between the bits of αm and βn. The proposed inaccurate methodology can be associated with checked duplication including Booth multipliers as well, except for it isn't associated with sign extension bits.

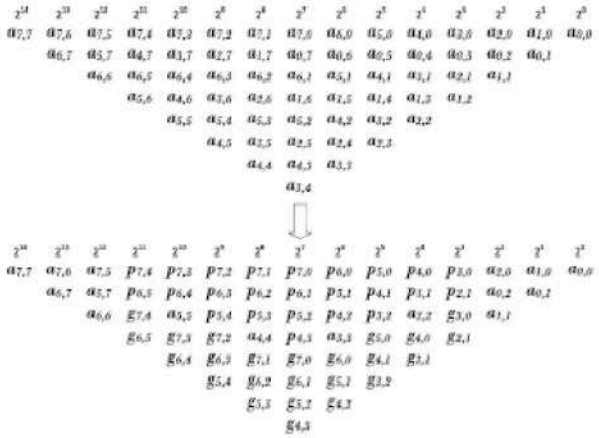


Fig. 3.1. Transformation of generated partial products into altered partial products.

3.3PROBABILITY STATISTICS OF GENERATE SIGNALS:

TABLE 3.1: Probability of the generate elements.

m	Probability of the *generate* elements being				P_{err}
	all zero	one 1	two 1's	three 1's and more	
2	0.8789	0.1172	0.0039	-	0.00390
3	0.8240	0.1648	0.0110	0.00024	0.01124
4	0.7725	0.2060	0.0206	0.00093	0.02153

From accurate viewpoint, the inadequate thing am, n has a probability of 1/4 of being 1. In the portions containing more than three midway things, the inadequate things am, n and an, m are united to outline propagate and make movements as given in (1). The resulting propagates and make signals outline changed inadequate things pm, n and gm, n. From area 3 with weight 23 to section 11 with weight 211, the partial things am, n and an, m are supplanted by changed

inadequate things pm, n and gm, n. The first and changed fragmentary thing structures are showed up in Fig. 1

$$pm,n = am,n + an,m$$

$$gm,n = am,n \cdot an,m. \quad (1)$$

The probability of the changed fragmented thing gm,n being one is 1/16, which is out and out lower than 1/4 of am,n. The probability of changed fragmentary thing pm,n being one is 1/16 + 3/16 + 3/16 = 7/16, which is higher than gm,n. These parts are considered, while applying supposition to the changed fragmentary thing cross section.

3.4. Figure of Altered Partial Products gm,n:

The social affair of make signals is done portion insightful. As each part has a probability of 1/16 of being one, two segments being 1 out of a comparative area even decreases. For example, in a fragment with 4 make signals, probability of all numbers being 0 is (1 − pr)4, only a solitary segment being one is 4pr(1 − pr)3, the probability of two segments being one in the segment is 6pr2(1 − pr)2, three ones is 4pr3(1−pr) and probability of all parts being 1 is pr4, where pr is 1/16. The probability bits of knowledge for different make segments m in each portion are given in Table I. In perspective of Table I, using OR entryway in the accumulation of portion keen make segments in the balanced fragmentary thing system gives adjust result in most of the cases. The probability of mix-up (Perr) while using OR entryway for lessening of create movements in each segment is furthermore recorded in Table I. As can be seen, the probability of misprediction is low. As the amount of make signals grows, the bumble probability augments straightly. Regardless, the estimation of screw up also rises. To keep this, the most extraordinary number of create signs to be amassed by OR door is kept at 4. For a portion having m deliver signals, m/4 OR entryways are used.

3.5. Approximation of Other Partial Products:

The storing up of other midway things with probability ¼ for am,n and 7/16 for pm,n uses deduced circuits. Assessed half-snake, full-snake, and 4-2 blower are proposed for their social event. Carr y and Sum are two yields of these assessed circuits. Since Carr y has higher weight of matched piece, bungle in Carry bit will contribute more by making botch qualification of two in the yield. Figure is managed with the goal that the aggregate difference between real

yield and induced yield is always kept up as one. In this way Carr y yields are approximated only for the cases, where Sum is approximated. In adders and blowers, XOR entryways tend to add to high district and delay. For approximating half-wind, XOR passage of Sum is supplanted with OR entryway as given in (2). This results in a solitary bumble in the Sum figuring as found as a general rule table of estimated half-snake in Table II. A tick check implies that inferred yield matches with reconsider yield and cross stamp implies jumble

$$\text{Entire} = x1 + x2$$

$$\text{Carry} = x1 \cdot x2. \quad (2)$$

In the gauge of full-snake, one of the two XOR gateways is supplanted with OR entryway in Sum calculation. This results in botch in last two cases out of eight cases. Carry is balanced as in (3) displaying one goof. This gives more revisions, while keeping up the qualification among exceptional and evaluated a motivator as one. Reality table of harsh full-wind is given in Table III

$$W = (x1 + x2)$$

$$\text{Total} = W \oplus x3$$

$$\text{Carry} = W \cdot x3. \quad (3)$$

Two construed 4-2 blowers in [5] convey nonzero yield despite for the circumstances where all information sources are zero. This results in high ED and abnormal state of precision incident especially in examples of zeros in all bits or in most important parts of the lessening tree. The proposed 4-2 blower vanquishes this drawback. In 4-2 blower, three bits are required for the yield exactly when all the four wellsprings of information are 1, which happens only once out of 16 cases. This property is taken to forgo one of the three yield bits in 4-2 blower. To keep up irrelevant screw up differentiate as one, the yield "100" (the estimation of 4) for four data sources being one needs to b supplanted with yields "11" (the estimation of 3). For Sum figuring, one out of three XOR entryways is supplanted with OR door. In like manner, to make the Sum identifying with the circumstance where all information sources are ones as one, an additional circuit $x1 \cdot x2 \cdot x3 \cdot x4$ is added to the Sum verbalization. This results in goof in five out of 16 cases. Carr y is unraveled as in (4). The looking at truth table is given in Table IV

TABLE 3.2 Truth Table of Approximate Half Adder

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	1	0	1	0 ✔	1 ✔	0
1	0	0	1	0 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✗	1

TABLE 3.3 Truth Table of Approximate Full Adder

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$x3$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0	0 ✔	0 ✔	0
0	0	1	0	1	0 ✔	1 ✔	0
0	1	0	0	1	0 ✔	1 ✔	0
0	1	1	1	0	1 ✔	0 ✔	0
1	0	0	0	1	0 ✔	1 ✔	0
1	0	1	1	0	1 ✔	0 ✔	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✔	0 ✗	1

$$W1 = x1 \cdot x2$$

$$W2 = x3 \cdot x4 \; Sum = (x1 \oplus x2) + (x3 \oplus x4) + W1 \cdot W2$$

$$Carr\,y = W1 + W2. \quad (4)$$

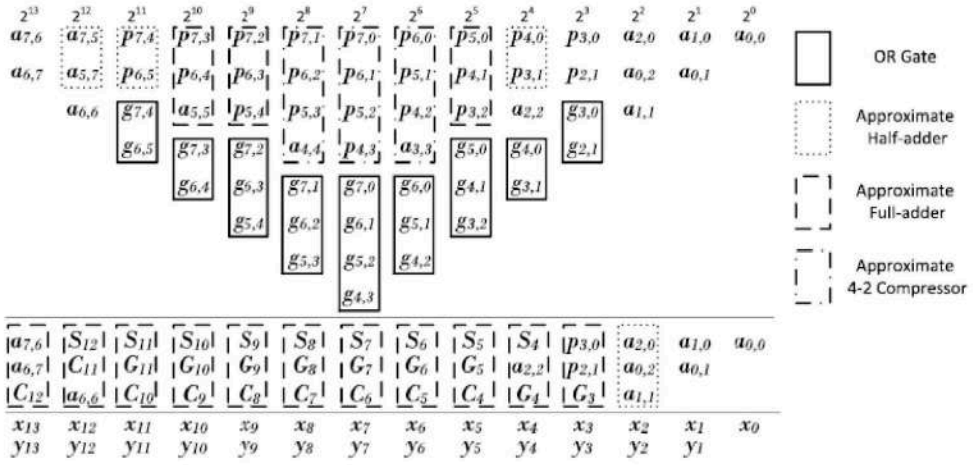


Fig. 3.2. Reduction of altered partial products.

Fig. 3.2 shows the diminishing of changed midway thing system of 8*8 harsh multiplier. It requires two stages to make sum and pass on yields for vector mix development step. Four 2-data OR gateways, four 3-data OR entryways, and one 4-information OR entryways are required for the lessening of create signals from areas 3 to 11. The resultant indications of OR entryways are set apart as Gi contrasting with the portion I with weight 2i.

For diminishing other fragmentary things, 3 inaccurate half-adders, 3 harsh full-adders, and 3 inferred blowers are required in the primary stage to make Sum and Carr y signs, Si and Ci identifying with segment I. The parts in the second stage are diminished using 1 inaccurate half-snake and 11 evaluated full-adders making last two operands Xi and Yi to be sustained to swell pass on snake for the last figuring of the result.

3.6. Two Variants of Multipliers

Two varieties of multipliers are proposed. In the essential case (Multiplier1), estimation is associated in all fragments of partial aftereffects of n-bit multiplier, however in Multiplier2, vague circuits are used in n − 1 smallest enormous segments.

TABLE 3.4 Truth Table of Approximate 4:2 Compressor

Inputs				Approximate outputs		Absolute Difference
$x1$	$x2$	$x3$	$x4$	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	0	0	1	0 ✔	1 ✔	0
0	0	1	0	0 ✔	1 ✔	0
0	0	1	1	1 ✔	0 ✔	0
0	1	0	0	0 ✔	1 ✔	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✔	1 ✔	0
1	0	0	0	0 ✔	1 ✔	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✔	1 ✔	0
1	1	0	0	1 ✔	0 ✔	0
1	1	0	1	1 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✔	0
1	1	1	1	1 ✗	1 ✗	1

CHAPTER 4
PROPOSED METHOD

CHAPTER 4
PROPOSED METHOD

4.1 INTRODUCTION

Approximate computing has emerged as a potential solution for the design of energy-efficient digital systems [1]. Applications such as multimedia, recognition and data mining are inherently error-tolerant and do not require a perfect accuracy in computation. For digital signal processing (DSP) applications, the result is often left to interpretation by human perception. Therefore, strict exactness may not be required and an imprecise result may suffice due to the limitation of human perception. For these applications, approximate circuits play an important role as a promising alternative for reducing area, power and delay, thereby achieving better performance in energy efficiency. As one of the key components in arithmetic circuits, adders have been extensively studied for approximate implementation [2]–[8]. As the typical carry propagation chain is usually shorter than the width of an adder, the speculative adders use a reduced number of less significant input bits to calculate the sum bits [2]. An error detection and recovery scheme has been proposed in [3] to extend the scheme of [2] for a reliable adder with variable latency. A reliable variable-latency adder based on carry select addition has been presented in [8]. As a number of approximate adders have been proposed, new methodologies to model, analyze and evaluate them have been discussed in [9]–[12]. A multiplier usually consists of three stages: partial product generation, partial product accumulation and a Carry Propagation Adder (CPA) at the final stage [13].

In the Under Designed Multiplier (UDM), approximate partial products are computed using inaccurate 2 × 2 multiplier blocks, while accurate adders are used in an adder tree to accumulate the approximate partial products [14]. In [15], approximate 4 × 4 and 8 × 8bit Wallace multipliers are designed by using a carry-in prediction method. Then, they are used in the design of approximate 16 × 16 Wallace multipliers, referred to as AWTM. The AWTM is configured into four different modes by using a different number of approximate 4 × 4 and 8 × 8 multipliers. The use of approximate speculative adders has been discussed in [10] for the final stage addition in a multiplier. The Error Tolerant Multiplier (ETM) of [16] is based on the partition of a multiplier into an accurate multiplication part for Most Significant Bits

(MSBs) and a non-multiplication part for Least Significant Bits (LSBs). The Static Segment Multiplier (SSM) utilizes a similar partition scheme [17]. In an n × n SSM, an m × m accurate multiplier (m n/2) is used to multiply the m consecutive bits from the two input operands. Whether the (n − m) MSBs of each input operand are all zero determines the selection of the inputs for the accurate multiplier (m MSBs or m LSBs). These approximate multipliers are designed for unsigned operation. Signed multiplication is usually implemented by using a Booth algorithm. Approximate designs have been proposed for fixed width Booth multipliers [18]–[20].

4.2 PROPOSED APPROXIMATE MULTIPLIER

A. The Approximate Adder

In this section, the design of a new approximate adder is presented. This adder operates on a set of pre-processed inputs. The Input Pre-Processing (IPP) is based on the commutativity of bits with the same weights in different addends. For example, Consider two sets of inputs to a 4-bit adder: i) A = 1010, B = 0101 and ii) A = 1111, B = 0000. Clearly, the additions in i) and ii) produce the same result. In this process, the two input bits Ai Bi = 01 are equivalent to Ai Bi = 10 (with i being the bit index) due to the commutativity of the corresponding bits in the two operands. The basic rule for the IPP is to switch Ai and Bi if Ai = 0 and Bi = 1 (for any i), while keeping the other combinations (i.e., Ai Bi = 00, 10 and 11) unchanged. By doing so, more 1's are expected in A and more 0's are expected in B. If A˙i B˙ i are the ith bits in the pre-processed inputs, the IPP functions are given by:

A˙i = Ai + Bi, (1)

B˙i = Ai Bi, (2)

TABLE 4.1: Truth Table of the Approximate Adder Cell

'X' REPRESENTS THAT NO SUCH A COMBINATION OCCURS DUE TO THE IPP

S_i/E_i		$\dot{B}_i\dot{B}_{i-1}$			
		00	01	11	10
$\dot{A}_i\dot{A}_{i-1}$	00	0/0	X	X	X
	01	0/0	1/0	X	X
	11	1/0	1/1	1/0	0/0
	10	1/0	X	X	0/0

Equations (1) and (2) compute the propagate and generate signals used in a parallel adder such as the Carry Lookahead Adder (CLA). The proposed adder can process data in parallel by cutting the carry propagation chain. Let A and B denote the two input binary operands of an adder, S be the sum result, and E represent the error vector. Ai, Bi, Si and Ei are the ith least significant bits of A, B, S and E, respectively. A carry propagation chain starts at the ith bit when B˙i = 1, A˙i+1 = 1, B˙i+1 = 0. In an accurate adder, Si+1 is 0 and the carry propagates to the higher bit. However, In the proposed approximate adder, Si+1 is set to 1 and an error signal is generated as Ei+1 = 1. This prevents the carry signal from propagating to the higher bits. Hence, a carry signal is produced only by the generate signal, i.e., Ci = 1 only when B˙i = 1, and it only propagates to the next higher bit, i.e., the (i +1)th position. Table 4.1 shows the truth table of the approximate adder, where A˙i, B˙i and B˙i−1 are the inputs after IPP. The error signal is utilized for error compensation purposes as discussed in a later section. In this case, the approximate adder is similar to a redundant number system [24] and the logical functions of Table 4.1 are given by

Si = B˙i−1 + B˙i A˙i, (3)

Ei = B˙i B˙i−1 A˙i . (4)

By replacing A˙i and B˙i using (1) and (2) respectively, the logic functions with respect to the original inputs are given by

Si = (Ai ⊕ Bi) + Ai−1Bi−1, (5)

Ei = (Ai ⊕ Bi)Ai−1Bi−1, (6)

where i is the bit index, i.e., i = 0, 1,⋯ , n for an n-bit adder. Let A−1 = B−1 = 0 when i is 0, thus, S0 = A0 ⊕ B0 and E0 = 0. Also, Ei = 0 when Ai−1 or Bi−1 is 0. Consider an n-bit adder, the inputs are given by A = An−1 ⋯ A1A0 and B = Bn−1⋯ B1B0, the exact sum is S = Sn−1 ⋯S1S0. Then, Si can be computed as Si + Ei and thus, the exact sum of A and B is given by

S = S + E…… (7)

In (7) '+' means the addition of two binary numbers rather than the 'OR' function. The error E is always non-negative and the approximate sum is always equal to or smaller than the accurate sum. This is an important feature of this adder because an additional adder can be used to add the error to the approximate sum as a compensation step. While this is intuitive in an adder design, it is a particularly useful feature in a multiplier design as only one additional adder is needed to reduce the error in the final product.

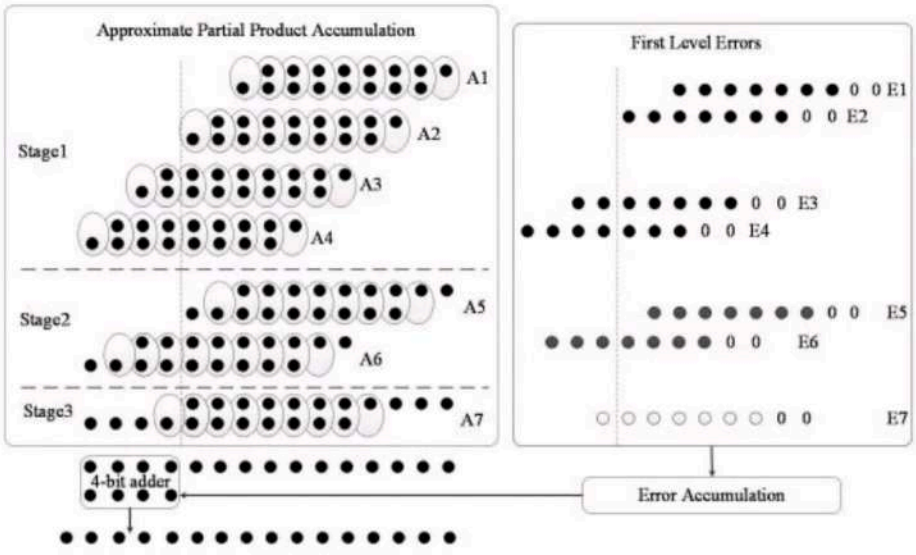


Fig. 4.1 An approximate multiplier with partial error recovery using 5 MSBs of the error vector: a partial product, sum or an error bit generated at the first stage; an error bit generated at the second stage; an error bit generated at the last stage.

B. Proposed Approximate Multiplier

A distinguishing feature of the proposed approximate multiplier is the simplicity to use approximate adders in the partial product accumulation. It has been shown that this may lead to low accuracy, because errors may accumulate and it is difficult to correct errors using existing approximate adders. However, the use of the newly proposed approximate adder overcomes this problem by utilizing the error signal. The resulting design has a critical path delay that is shorter than a conventional one-bit full adder, because the new n-bit adder can process data in parallel. The approximate adder has a rather high error rate, but the feature of generating both the sum and error signals at the same time reduces errors in the final product. An adder tree is utilized for partial product accumulation; the error signals in the tree are then used to compensate the error in the output to generate a product with a better accuracy. The architecture of the proposed approximate multiplier is shown in Fig. 4.1. In the proposed design, the simplification of the partial product accumulation stage is accomplished by using an adder tree, in which the number of partial products is reduced by a factor of 2 at each stage

of the tree. This adder tree is usually not implemented using accurate multi-bit adders due to the long latency.

4.3 ERROR REDUCTION

The approximate adder generates two signals: the approximate sum S and the error E; the use of the error signal is considered next to reduce the inaccuracy of the multiplier. As (7) is applicable to the sum of every single approximate adder in the tree, an error reduction circuit is applied to the final multiplication result rather than to the output of each adder. Two steps are required to reduce errors: i) error accumulation and ii) error recovery by the addition of the accumulated errors to the adder tree output using a CPA. In the error accumulation step, error signals are accumulated to a single error vector, which is then added to the output vector of the partial product accumulation tree. Two approximate error accumulation methods are proposed, yielding the Approximate Multiplier 1 (AM1) and Approximate Multiplier 2 (AM2).

A. Error Accumulation for Approximate Multiplier 1

As shown in Fig. 4.1, each approximate adder Ai generates a sum vector Si and an error vector Ei, where i = 1, 2, ⋯, 7. If the error signals are added using accurate adders, the accumulated error can fully compensate the inaccurate product; however, to reduce complexity, an approximate error accumulation is introduced.

Consider the observation that the error vector of each approximate adder tends to have more 0's than 1's. Therefore, the probability that the error vectors have an error bit '1'at the same position, is quite small. Hence, an OR gate is used to approximately compute the sum of the errors for a single bit. If m error vectors (denoted by E1, E2, ..., Em) have to be accumulated, then the sum of these vectors is obtained as Ei = E1i OR E2i OR ... OR Emi. (8)

To reduce errors, an accumulated error vector is added to the adder tree output using a conventional CPA (e.g. a carry lookahead adder). However, only several (e.g. k) MSBs of the error signals are used to compensate the outputs to further reduce the overall complexity. The number of MSBs is selected according to the extent that errors must be compensated. For example in an 8 × 8 adder tree, there are a total of 7 error vectors, generated by the 7 approximate adders in the tree. However, not all the bits in the 7 vectors need to be added, because the MSBs of some vectors are less significant than the least significant bits of the k MSBs. In the example of Fig. 4.1, 5 MSBs (i.e. the (11 − 14)th bits, no error is generated at

the 15th bit position) are considered for error recovery and therefore, 4 error vectors are considered (i.e., the error vectors E3, E4, E6 and E7). The error vectors of the other three adders are less significant than the 11th bit, so they are not considered. The accumulated error E is obtained using (8); then, the final result is found by adding E to S using a fast accurate CPA. The error accumulation scheme is shown in Fig. 3. As no error is generated at the least significant two bits of each approximate adder Ai (i = 1, 2,⋯ , 7), the least significant two bits of each error vector Ei are not accumulated.

B. Error Accumulation for Approximate Multiplier 2

The error accumulation scheme for AM2 is shown in Fig.4.3. To introduce the design of AM2, an 8 × 8 multiplier with two inputs X and Y is considered. For example, consider the first two partial product vectors X0Y7, X0Y6, ..., X0Y0 and X1Y7, X1Y6, ..., X1Y0 accumulated by the first approximate adder (A1 in Fig. 4.1), where Xi and Yi are the ith least significant bits of X and Y , respectively. Recall from (6) for the approximate adder, the condition for Ei = 1 is Ai−1 = Bi−1 = 1 and Ai = Bi. (9) For the first approximate adder in the partial product accumulation tree, its inputs are A = X0Y7, X0Y6, ..., X0Y0 and B = X1Y7, X1Y6, ..., X1Y0. Thus, the ith least significant bits for A and B are Ai = X0Yi and Bi = X1Yi−1, respectively.

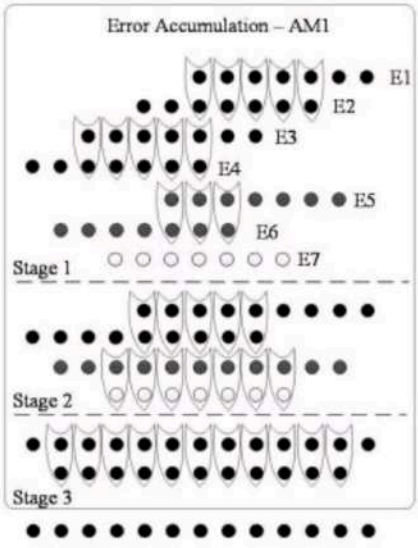


Fig. 4.2. Error accumulation tree for AM1.

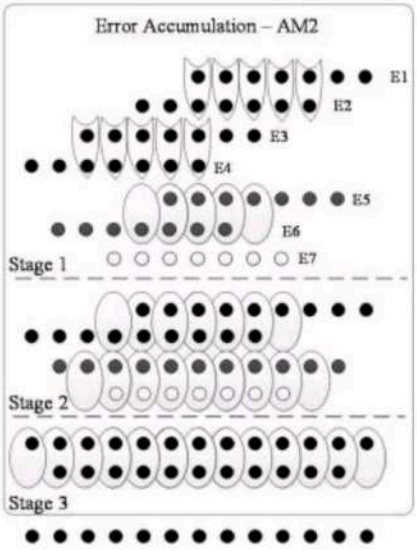


Fig. 4. 3 Error accumulation tree for AM2.

If X0 or X1 is 0, there will be no error in this approximate adder because either A or B is zero. Therefore, no error occurs unless X0X1 = 11. When X0X1 = 11, Ai and Bi are simplified to Yi and Yi−1, respectively. Then to calculate Ei, Ai−1, Bi−1, Ai and Bi are replaced by Yi−1, Yi−2, Yi and Yi−1, respectively. For Ei to be 1, YiYi−1Yi−2 = 011 according to (9). Therefore, an error only occurs when the input has "011" as a bit sequence. Based on this observation, the "distance" between two errors in an approximate multiplier is at least 3 bits. Thus, two neighboring approximate adders in the first stage of the partial product tree cannot have errors at the same column, because the errors in a lower approximate adder are those in the upper adder shifted by 2 bits when both errors exist. The errors in two neighboring approximate adders can then be accurately accumulated by OR gates, e.g., an OR gate can be used to accumulate the two bits in the error vectors E1 and E2. After applying the OR gates to accumulate E1 and E2 as well as E3 and E4, the four error vectors are compressed into two. For E5, E6 and E7, they are generated from the approximate sum of the partial products rather than the partial products. Therefore, they cannot be accurately accumulated by OR gates.

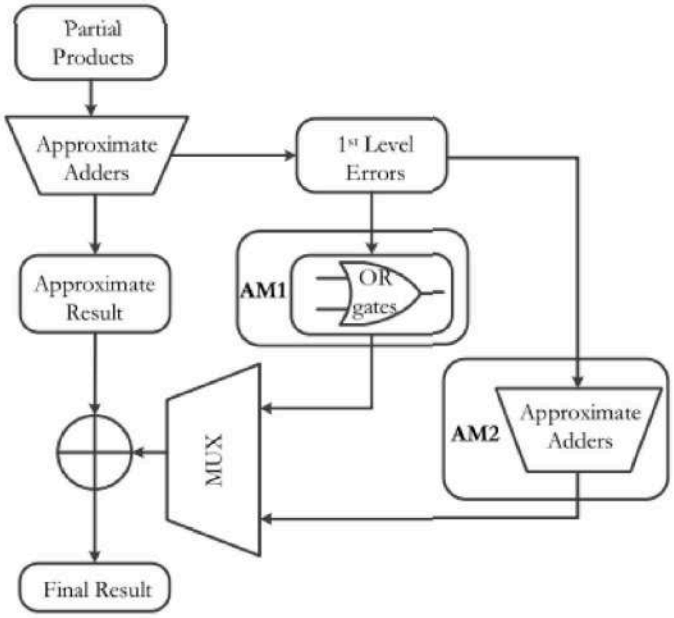


Fig. 4.4. Block diagram of the Proposed Multipliers.

Another interesting feature of the proposed approximate adder is as follows. Assume Ei = 1 in (6), then Ai−1 = Bi−1 = 1 and Ai = Bi. Since Ai−1 = Bi−1 = 1, i.e., Ai−1 ⊕ Bi−1 = 0, it is easy to show that Ei−1 = 0. Moreover, as Ai = Bi, i.e., Ai Bi = 0, then Ei+1 = 0. Thus, once there is an error in one bit, its neighboring bits are error free, i.e., there are no consecutive error bits in one row. Therefore, there is no carry propagation path longer than two bits when two error vectors are accumulated, and the error vectors are accurately accumulated by the proposed approximate adder. Based on the above analysis, E5 and E6 are accurately accumulated by one approximate adder in the first stage of the error accumulation. After the first stage of error accumulation, three vectors are generated, and another two approximate adders are then used to accumulate these three vectors as well as the error vector remaining from the previous stage (E7). Simulation results (found in later sections) show that the modified error accumulation outperforms the OR-gate error accumulation with little overhead on delay and power. Here after, the proposed n × n approximate multiplier with k-MSB OR-gate based error reduction is referred to as an n/k AM1, while an n × n approximate multiplier with k-MSB approximate adder based error reduction is referred to as an n/k AM2. The Block diagram of the Proposed Multipliers is shown in Fig.4.4

C. 16 × 16 Approximate Multipliers

In both AM1 and AM2, all the error vectors are compressed to one error vector, which is then added back to the approximate output of the partial product tree. Compared to 8 × 8 designs, 16 × 16 multipliers generate more error vectors, and too much information would be ignored if the same error reduction strategies are used. That is, using only one compressed error vector does not make a good estimate of the overall error. In the modified designs, the error vectors generated by the approximate adders are compressed to two final error vectors. Take a 16 × 16 AM1 as an example, the eight error vectors generated at the first stage of the partial product accumulation tree are compressed to one error vector, EV1, using OR gates. The remaining seven error vectors from the second, third and fourth stages are compressed to another error vector EV2. Then both EV1 and EV2 are added back to the output of the partial product at the fourth stage. Similarly, the proposed approximate adders are used in a 16×16 AM2 to compress the eight error vectors from the first stage to one error vector and the remaining error vectors to another error vector. Truncation can also be applied to the proposed designs for large input operands. Therefore, 16 LSBs of the partial products are truncated in 16×16 AM1 and AM2, resulting in truncated AM1 (TAM1) and truncated AM2 (TAM2).

4.4 SIGNED NUMBERS

Binary number systems use signed representations to encode negative integers. This is known as a signed number representation in computing.

When representing negative integers in mathematics, a minus sign ("-") is used before the numbers. Computer hardware, on the other hand, only uses bits to represent numbers. It is well known that there are four ways to express signed integers using the binary numeral system: sign and magnitude, ones' complement, twos complement, and offset binary Negative binary, which uses the base 2, is an alternative approach that makes use of implicit signs rather than explicit ones. This may also be done for various bases such as fractional and fractional elaborations as well as positive and negative.

There isn't a single criteria that can be used to determine which representation is the best. Most modern computer systems utilize two's complement representation for integers, with the exception of mainframes from the Unisys Clear Path Dorado series, which use ones'

complement.

History

Many conflicting views regarding hardware and mathematical technology characterized the early days of digital computing (numbering systems). When it came to the format of negative numbers, several of the most respected specialists of the time were divided on their views. [reference required] One group was in favour of the system that is now in use: two's complement. Another group favoured ones' complement, which inverts all the bits in a word to make any positive value into a negative one. As for "sign and magnitude," it was supported by the third group, where a value could be altered from positive to negative by merely toggling the word's sign bit.

Every one of the systems has its supporters and detractors. With sign & magnitude, memory dump tracing (a popular practise in the 1960s) was simplified since smaller numeric quantities used fewer 1 bit values. As a consequence, when integers were sent from a register to the math unit, they had to be transformed to ones' complement values before they could be converted back to sign-magnitude and transmitted back to the register. Because discrete transistors are so expensive and difficult to package, the electronics system needed more gates than the other systems. There are many well-known IBM systems that used sign-magnitude, including the 704, 709, and 709x series.

When using Ones' complement, hardware designs may be simplified since values didn't have to be converted to and from and from the arithmetic unit The only drawback was that it could also express negative zero (0) like sign-magnitude. Positive zero acts precisely like negative zero; the outcome will be the same regardless of whether an operand is positive or negative zero when used as an operand in a computation. Having two different forms of the same value, on the other hand, requires doing two comparisons rather than one when looking for equality with zero. The removal of one's complement may also lead to an end-around loan (described below). One might argue that this complicates the logic of addition and subtraction, or that it simplifies it since the second operand's bits must be inverted when it is sent to the adder. Some computers, such the PDP-1, CDC 160 series and CDC 3000 series utilize one's complement representation whereas others, like UNIVAC 1100 series, use the same as the LINC computer.

Since two's complement is the simplest to implement in hardware, it's no surprise that it's so widely used.

There were many thousands of transistors in early mainframe processors, thus getting rid of a large number of them saved a lot of money. For example, the IBM System/360 uses two's complement as does the GE-600 series, along with minicomputers like the PDPS-5 and PDPS-8. Many early integrated circuit CPUs (such as the Intel 8080) used two's complement arithmetic since it was more efficient at the time. As IC technology progressed, two's complement technology became the norm. There are a lot of processors that are two's complement, such as x86, m68k, Power ISA, MIPS, SPARC, ARM, Itanium, PA-RISC, and DEC Alpha.

Magnitude with a signed signature

Another name for this kind of representation is "sign–magnitude" or "sign and magnitude" Sign bits are used to denote the positive and negative aspects of a number. For example, a positive number or positive zero would have a sign bit of 0, whereas an opposite sign bit of 1 would have a sign bit of 1. Even if there are only a few bits left, they imply magnitude (or absolute value). To provide an example, in an eight-bit byte the magnitude is represented by just seven bits and may thus vary from 0 to 1111111 (127). Once the sign bit (the eighth bit) is included, the range of numbers that may be represented expands to include 12710 to the power of 2. This means that 4310 is 10101111, whereas 4310 encoded in an eight-bit byte is 00101111. Because of the many implications of using signed magnitude representation, they are more difficult to implement:

00000000 (0) and 10000000 (0) are both valid methods to represent zero.

There are two ways to add and subtract: one uses an end-around carry and the other uses overflow behaviour. The behaviour required for addition is dependent on the sign bit, and the overflow behaviour may be ignored for addition.

It's necessary to verify the sign bit in comparison to two's complement since subtracting two integers and seeing whether the result is positive or negative is simpler.

Instead of 128 in two's complement, the minimum negative number is 127.

For comparison, you may place a "+" or a "" next to the magnitude of the number in this method. It's possible that early binary computers (such the IBM 7090) used this format since it's so close to everyday use. The most common method to express the significand in floating-point numbers is as a signed magnitude.

Complementary one

To represent negative numbers, an alternative method called ones' complement may be employed. If a negative binary number is expressed as a one's complement, then the bitwise NOT operation is performed to it, resulting in the "complement" of the positive number. 00000000 (+0) and 11111111 (-0) are the two representations of 0 in ones' complement, same as in sign-and-magnitude representation.

For instance, the ones' complement of 00101011 (4310) becomes 11010100 (4310). If you want to express the range of signed numbers using the ones' complement, you may use the notation. There are two possible values for zero in an eight-bit conventional byte: 00000000 (+0) or 11111111 (-0).

Two integers represented in this system must be added using the usual binary addition method, but this must be followed by an end-around carry, which means adding the resultant sum's carry back in.

Think about the following example, which illustrates the situation of adding 1 (11111110) to +2

```
  Binary   decimal
  11111110   −1
+ 00000010   +2
──────────────────  ──
1 00000000    0  ← Not the correct answer
      1    +1  ← Add carry
──────────────────  ──
  00000001    1  ← correct answer
```

In the previous example, the first binary addition gives 00000000, which is incorrect. The correct result (00000001) only appears when the carry is added back in.

To clarify nomenclature, the system is called "ones' complement" because the negation of a positive value x (expressed as the bitwise NOT of x) may also be created by subtracting x from the ones' complement representation of zero, which is a lengthy series of ones (0). Three-way arithmetic on the other hand uses a big power of two that is equivalent to +0 to create an equation for negating an arbitrary number (x). [9] Due to the fact that one's complement and twos complement representations vary by one, the same negative number will be represented by both.

Be aware that bitwise addition of the magnitude yields the ones' complement representation of a negative number from the sign-magnitude format (inverting all the bits after the first). Examples are the decimal 125 and its ones' complement version of 10000010, both of which have sign-magnitude representations of 11.111101.

Complementary numbers two and three

Multiple 0 representation issues as well as the need for an end-around carry may be solved by using a technique known as "two's complement". Negative numbers are represented in two's complement by the bit pattern that is one larger (in an unsigned sense) than that of the ones' complements of the positive value.

There is just one zero in two's complement, which is 00000000. To make a number negative, invert all of the bits and then add one to the result. [10] This is the ring structure of all numbers modulo 2N. The addition of two two's complement integers is the same as the addition of two unsigned numbers (save for the detection of excess, if that is done); same is true for the subtraction and even for N lowest relevant ones of a product (value of multiplication). Which can be seen in the 8-bit two's complement table, a two's complement addition of 127 and 128 produces the same binary bit sequence as an unsigned addition of 127 and 128, for example

An easier method to get the negation of a number in two's complement is as follows:

	Example 1	**Example 2**
1. Starting from the right, find the first "1"	0010100**1**	00101**1**00
2. Invert all of the bits to the left of that "1"	**1101011**1	**1101**0100

Method two:

1. Invert all the bits through the number
2. Add one

Example: for +2, which is 00000010 in binary (the ~ character is the C bitwise NOT operator, so ~X means "invert all the bits in X"):

1. ~00000010 → 11111101
2. 11111101 + 1 → 11111110 (−2 in two's complement)

CHAPTER 5
INTRODUCTION TO VLSI

CHAPTER 5

INTRODUCTION TO VLSI

Extensive scale compromise (VLSI) is the route toward making fused circuits by joining an enormous number of transistor-based circuits into a singular chip. VLSI began in the 1970s when complex semiconductor and correspondence progressions were being made. The microchip is a VLSI contraption. The term is no longer as essential as it once may have been, as chips have extended in unconventionality into the an enormous number of transistors.

5.1 Overview:

The essential semiconductor chips held one transistor each. Following advances incorporated a consistently expanding number of transistors, and, accordingly, more individual limits or structures were consolidated after some time. The fundamental facilitated circuits held only several contraptions, perhaps upwards of ten diodes, transistors, resistors and capacitors, making it possible to make no less than one justification entryways on a single device. By and by alluded to brilliantly as "little scale joining" (SSI), improvements in methodology incited devices with a few reason entryways, known as huge scale mix (LSI), i.e. systems with no not as much as a thousand reason entryways. Current advancement has moved far past this stamp and the present microchips have an extensive number of entryways and countless transistors.

At one time, there was a push to name and modify diverse levels of immense scale mix above VLSI. Terms like Ultra-significant scale Integration were used. Regardless, the gigantic number of passages and transistors open on standard contraptions has rendered such fine capabilities begging to be proven wrong.

Terms proposing more important than VLSI levels of blend are no longer in expansive use. Without a doubt, even VLSI is as of now genuinely intriguing, given the consistent assumption that all chip are VLSI or better.

Beginning at mid 2008, billion-transistor processors are fiscally available, an instance of which is Intel's Montecito Itanium chip. This is depended upon to wind up more run of the mill as semiconductor fabricate moves from the present age of 65 nm systems to the accompanying 45 nm ages (while experiencing new challenges, for instance, extended

assortment across over process corners). Another noticeable case is NVIDIA's 280 game plan GPU.

This microchip is exceptional in the manner in which that its 1.4 Billion transistor count, prepared for a teraflop of execution, is when in doubt focused on reason (Itanium's transistor check is, as it were, due to the 24MB L3 hold). Current designs, as opposed to the soonest contraptions, use wide layout automation and robotized method of reasoning amalgamation to spread out the transistors, engaging more lifted measures of capriciousness in the resulting justification value. Certain prevalent reason squares like the SRAM cell, regardless, are so far created by hand to ensure the most significant viability (a portion of the time by contorting or disturbing set up plan standards to gain the last bit of execution by trading trustworthiness).

5.2 What is VLSI?

VLSI stays for "Gigantic Scale Integration". This is the field which incorporates squeezing progressively method of reasoning contraptions into more diminutive and tinier domains.

1. Simply we say Integrated circuit is various transistors on one chip.
2. Design/gathering of to an awesome degree little, complex equipment using changed semiconductor material
3. Integrated circuit (IC) may contain an extensive number of transistors, each two or three mm in measure
4. Applications unfathomable: most electronic reason contraptions

5.3 Advantages of ICs over discrete portions:

While we will center around facilitated circuits, the properties of joined circuits-what we can and can't gainfully put in an organized circuit, all things considered, choose the building of the entire system. Joined circuits improve structure properties in a couple of essential ways. ICs have three key positive conditions over cutting edge circuits worked from discrete parts:

Size: Fused circuits are extensively smaller the two transistors and wires are contracted to micrometer sizes, appeared differently in relation to the millimeter or centimeter sizes of discrete sections. Minimal size prompts positive conditions in speed and power usage, since more diminutive fragments have humbler parasitic securities, capacitances, and inductances.

Speed: Signs can be traded between justification 0 and reason 1 impressively quicker inside

a chip than they can between chips. Correspondence inside a chip can happen numerous events speedier than correspondence between chips on a printed circuit board. The quick of circuits on-chip is a result of their little size-tinier sections and wires have smaller parasitic capacitances to back off the banner.

Power consumption: Logic exercises inside a chip furthermore take fundamentally less power. Eventually, cut down power use is, all things considered, in light of the little size of circuits on the chip-smaller parasitic capacitances and assurances require less ability to drive them.

5.4 VLSI and systems:

These central purposes of joined circuits convert into positive conditions at the system level:

Smaller physical size: Modesty is as often as possible great position in itself-consider minimized TVs or handheld mobile phones.

Lower control usage: Supplanting a pack of standard parts with a singular chip reduces mean control usage. Decreasing force use impacts whatever is left of the system: a smaller, more affordable power supply can be used; since less power use suggests less warmth, a fan may never again be vital; a less intricate authority with less ensuring for electromagnetic securing may be feasible, also.

Reduced cost: Reducing the amount of parts, the power supply requirements, agency costs, and whatnot, will unavoidably diminish system cost. The logically outstretching impact of coordination is to such a degree, to the point that the expense of a system worked from custom ICs can be less, notwithstanding the way that the individual ICs cost more than the standard parts they supplant.

Understanding why consolidated circuit development has such critical effect on the blueprint of cutting-edge structures requires understanding both the advancement of IC manufacturing and the money related issues of ICs and electronic systems.

5.5 Applications of VLSI:

Electronic systems at present play out a wide variety of endeavors in step by step life. Electronic systems from time to time have supplanted segments that worked mechanically, intensely, or by various means; equipment are commonly more diminutive, more versatile, and less requesting to profit. In various cases electronic structures have made totally new applications. Electronic structures play out a combination of endeavors, some of them indisputable, some more concealed:

1. Personal redirection systems, for instance, minimized MP3 players and DVD players perform refined counts with incredibly little essentialness.

2. Electronic structures in automobiles work stereo systems and introductions; they moreover control fuel mixture systems, adjust suspensions to evolving scene, and play out the control limits required for antilock braking (ABS) systems.

3. Digital equipment pack and decompress video, even at unrivaled quality data rates, on-the-fly in customer devices.

4. Low-cost terminals for Web examining still require complex devices, paying little mind to their committed limit.

5. Personal PCs and workstations give word-planning, fiscal examination, and diversions. PCs fuse both central getting ready units (CPUs) and remarkable reason gear for plate get to, speedier screen appear.

6. Medical electronic structures measure considerable limits and perform complex getting ready estimations to alert about phenomenal conditions. The availability of these unpredictable structures, far from overwhelming purchasers, just makes enthusiasm for extensively more staggering systems.

The creating progression of usages steadily pushes the blueprint and gathering of fused circuits and electronic structures higher than any time in recent memory of multifaceted nature.

CHAPTER 6
SOFTWARE ENVIRONMENT

CHAPTER 6

SOFTWARE ENVIRONMENT

6.1 Software Environment

Create a Vivado Project

Vivado "projects" are directory structures that contain all the files needed by a particular design. Some of these files are user-created source files that describe and constrain the design, but many others are system files created by Vivado to manage the design, simulation, and implementation of projects. In a typical design, you will only be concerned with the user-created source files. But, in the future, if you need more information about your design, or if you need more precise control over certain implementation details, you can access the other files as well.

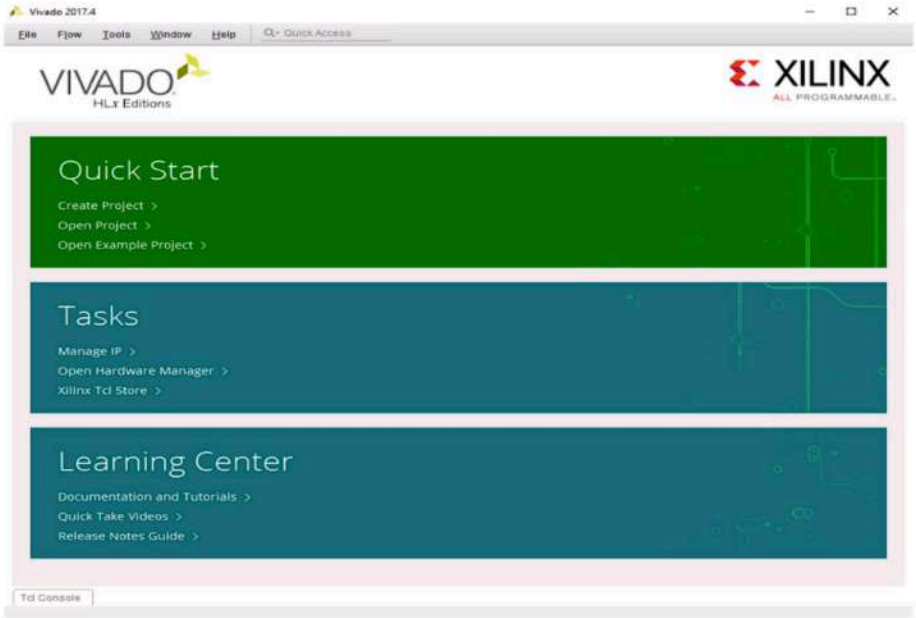


Figure:6.1 Vivado Start-Up Window

When setting up a project in Vivado, you must give the project a unique name, choose a location to store all the project files, specify the type of project you are creating, add any pre-

existing source files or constraints files (you might add existing sources if you are modifying an earlier design, but if you are creating a new design from scratch, you won't add any existing files you haven't written them yet), and finally, select which physical chip you are designing for. These steps are illustrated below.

Start Vivado

In Windows, you can start Vivado by clicking the shortcut on the desktop. After Vivado is started, the window should look similar to the picture in figure 1.

Open Create Project Dialog

Click on "Create Project" in the Quick Start panel. This will open the New Project dialog as shown in Figure 2. Click Next to continue.

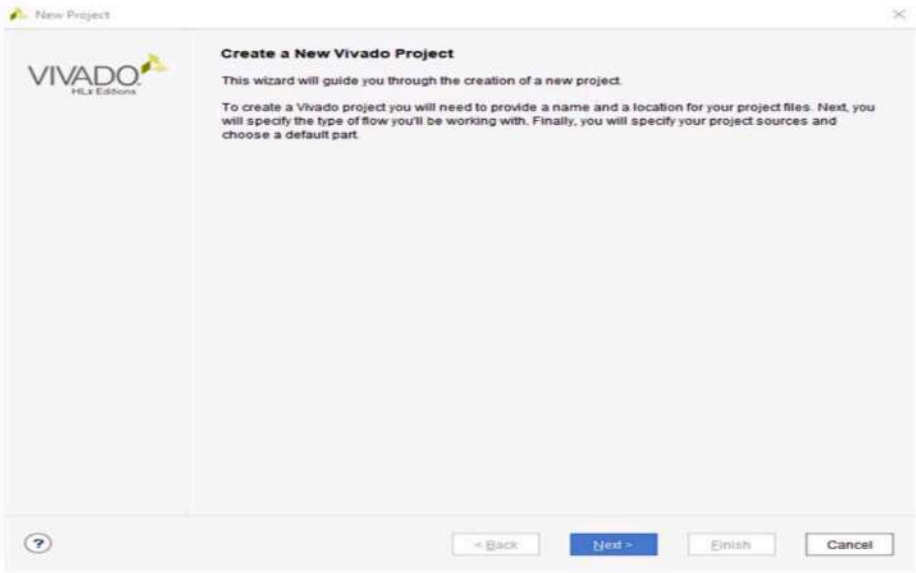


Figure:6.2 Create Project Dialog

Set Project Name and Location: Enter a name for the project. In the figure, the project name is "project_1", which isn't a particularly useful name. It's usually a good idea to make the project name more descriptive, so you can more readily identify your designs in the future.

You should avoid having spaces in the project name or location, because spaces can cause certain tools to fail.

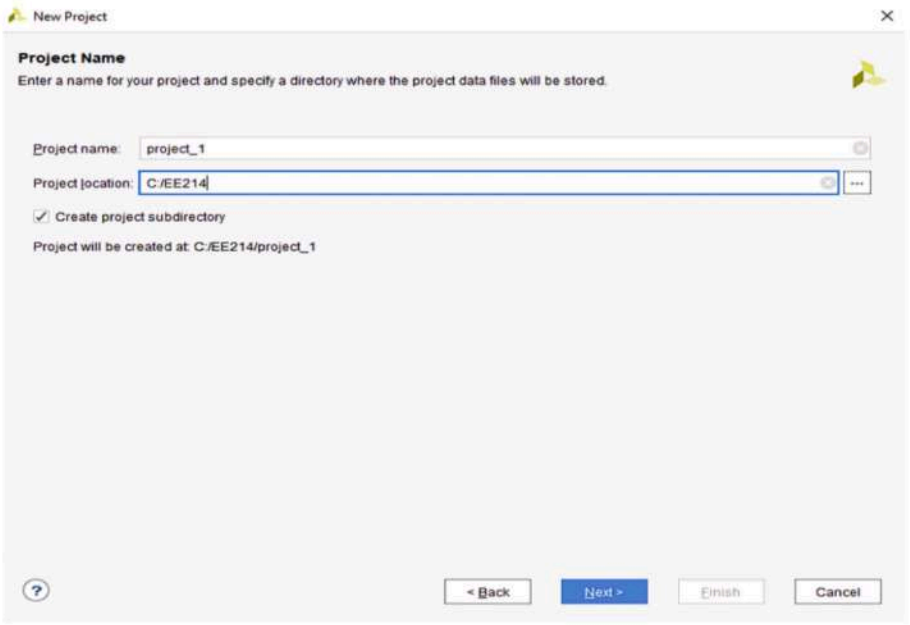


Figure:6.3 Enter Project Name

Select Project Type: The "project type" configures certain design tools and the IDE appearance based on the type of project you are intending to create. Most of the time, and for all Real Digital courses, you will choose "RTL Project" to configure the tools for the creation of a new design. (RTL stands for Register Transfer Language, which is a term sometimes used to mean a hardware design language like Verilog).

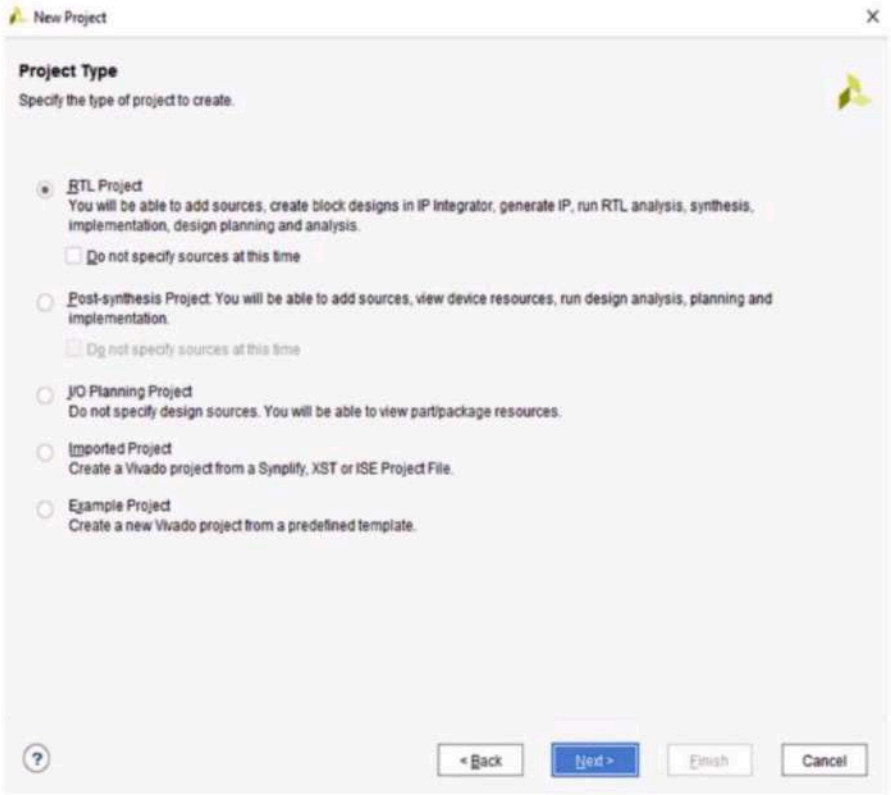


Figure:6.4 Select Project Type

Add Existing Sources: In a typical new or early-stage design, you won't add any existing sources because you haven't created them yet. But as you complete more designs and build up a library of previously completed and known good designs, you may elect to add sources and them use them in a new design. For now, there are no existing sources to add, so just click Next.

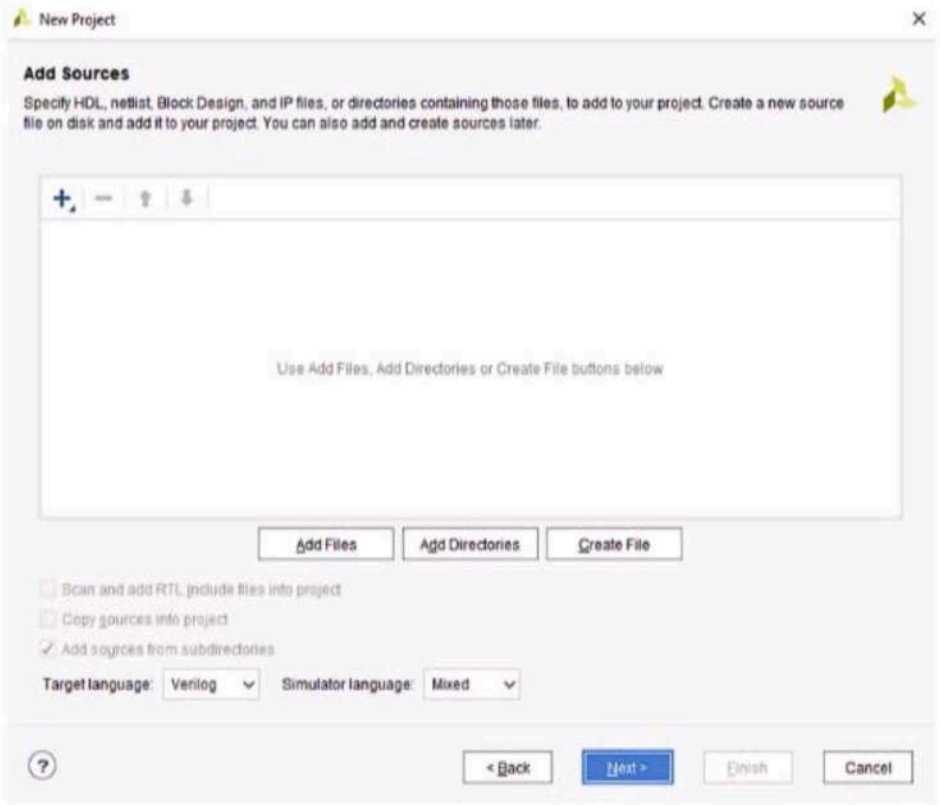


Figure:6.5 Add Sources

Select Parts: Xilinx produces many different parts, and the synthesizer needs to know exactly what part you are using so it can produce the correct programming file. To specify the correct part, you need to know the device family and package, and less critically, the speed and temperature grades (the speed and temperature grades only affect special-purpose simulation results, and they have no effect on the synthesizer's ability to produce accurate circuits). You must choose the appropriate part for the device installed on your board.

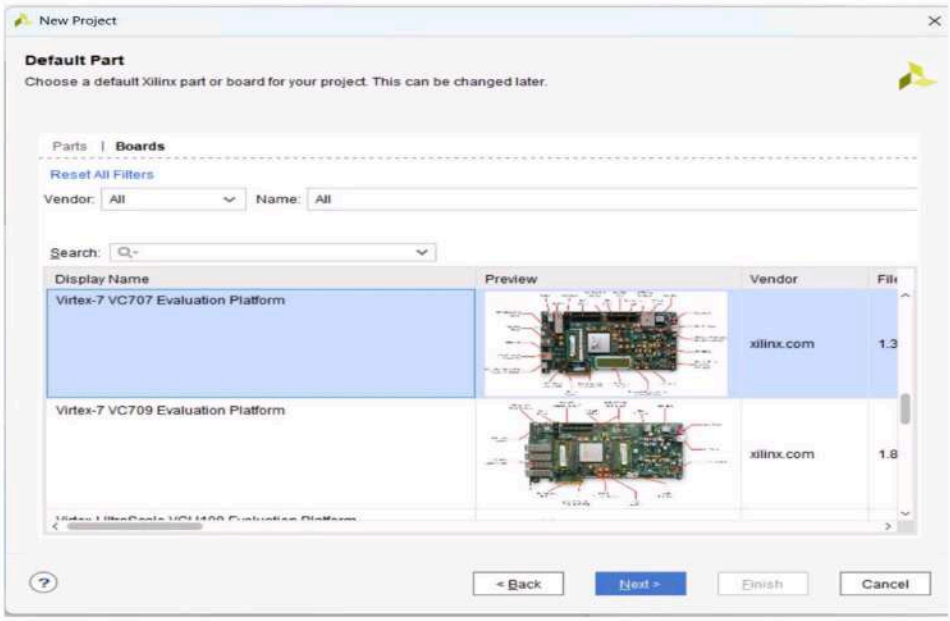


Figure:6.6 Select Board Part

Check Project Configuration Summary: On the last page of the Create Project Wizard a summary of the project configuration is shown. Verify all the information in the summary is correct, and in particular make sure the correct part is selected. If anything is incorrect, click back and fix it; otherwise, click Finish to finish creating an empty project.

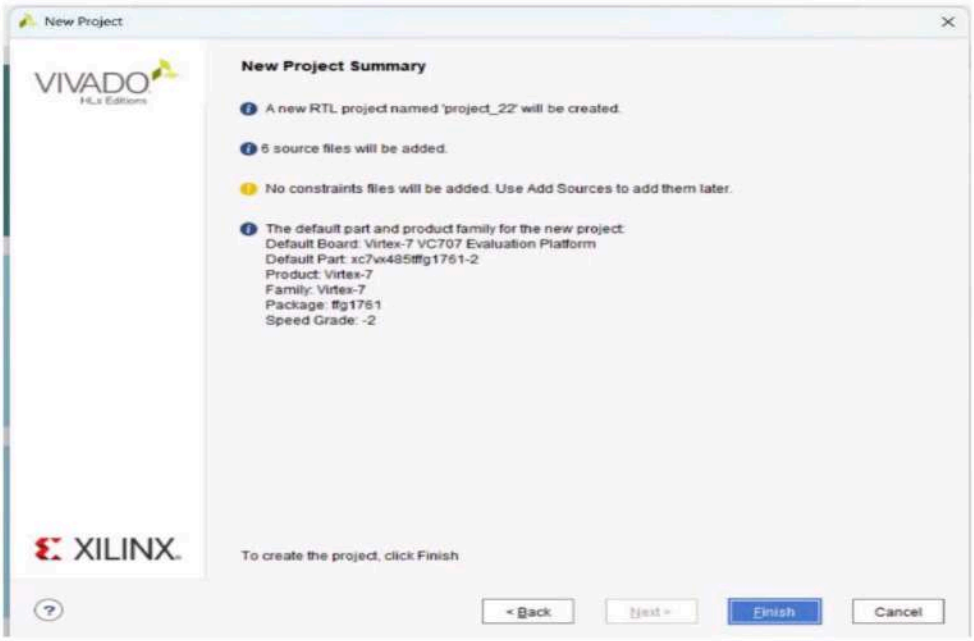


Figure:6.7 Create a project Summary

Vivado Project Window

After you have finished with the Create Project Wizard, the main IDE window will be displayed. This is the main "working" window where you enter and simulate your Verilog code, launch the synthesizer, and program your board. The left-most pane is the flow navigator that shows all the current files in the project, and the processes you can run on those files. To the right of the flow navigator is the project manager window where you enter source code, view simulation data, and interact with your design. The console window across the bottom shows a running status log. Over the next few projects, you will interact with all of the panels.

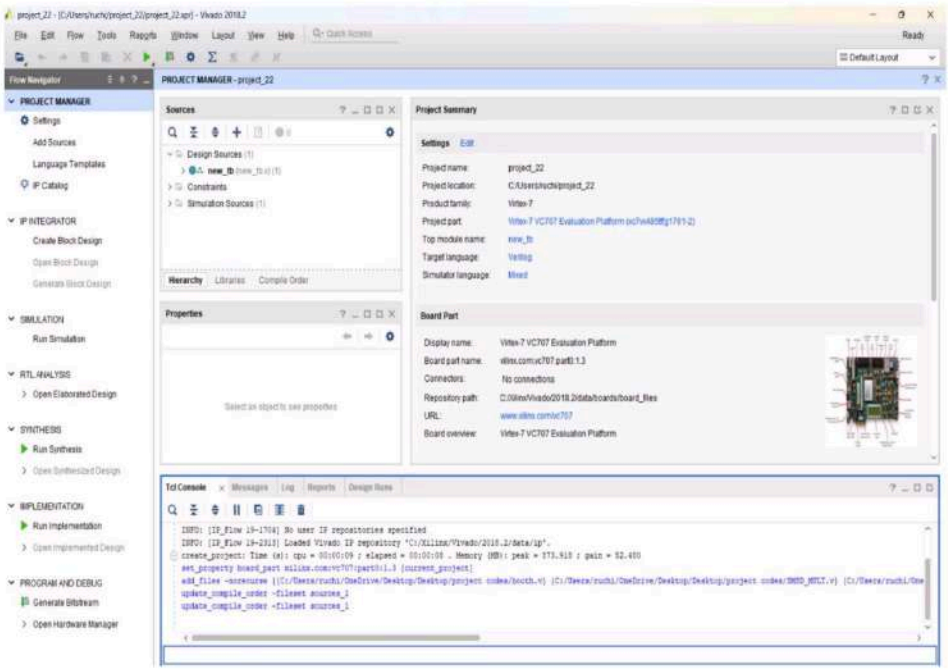


Figure:6.8 Vivado Project Window

6.2 INTRODUCTION TO VERILOG

In the semiconductor and electronic arrangement industry, Verilog is a hardware delineation language (HDL) used to show electronic structures. Verilog HDL, not to be mixed up for VHDL (a battling tongue), is most conventionally used in the arrangement, affirmation, and utilization of digital method of reasoning chips at the enlist trade level of consultation. It is furthermore used in the affirmation of analog and mixed banner circuits.

Audit Hardware delineation tongues, for instance, Verilog differentiate from programming vernaculars since they consolidate strategies for depicting the spread of time and banner conditions (affectability). There are two undertaking overseers, a blocking assignment (=), and a non-blocking (<=) errand. The non-blocking errand empowers makers to portray a state-machine revive without hoping to broadcast and use brief accumulating factors (in any wide programming lingo we need to describe some passing storage spaces for the operands to be

taken a shot at in like manner; those are temporary limit factors). Since these thoughts are a bit of Verilog's tongue semantics, originators could quickly make delineations out of far reaching circuits in a by and large insignificant and brief shape. At the period of Verilog's introduction (1984), Verilog addressed a huge gainfulness change for circuit organizers who were by then using graphical schematic capture software and particularly formed programming activities to file and reproduce electronic circuits.

The makers of Verilog required a vernacular with accentuation like the C programming tongue, which was by then comprehensively used in outlining programming progression. Verilog is case-fragile, has a fundamental preprocessor (anyway less present day than that of ANSI C/C++), and indistinguishable control stream watchwords (if/else, for, while, case, et cetera.), and great chairman need. Syntactic complexities join variable disclosure (Verilog requires bit-widths on net/reg types[clarification needed]), framework of procedural squares (begin/end as opposed to wavy props {}), and various other minor differences.

A Verilog setup includes a dynamic arrangement of modules. Modules exemplify layout levels of leadership, and talk with various modules through a course of action of announced data, yield, and bidirectional ports. Inside, a module can contain any mix of the going with: net/variable declarations (wire, reg, entire number, et cetera.), synchronous and progressive decree squares, and instances of various modules (sub-levels of leadership). Sequential decrees are set inside a begin/end square and executed in progressive demand inside the square. In any case, the squares themselves are executed all the while, qualifying Verilog as a dataflow lingo.

Verilog's concept of 'wire' contains both banner regards (4-state: "1, 0, skimming, vague") and characteristics (strong, weak, et cetera.). This system grants calculated exhibiting of shared banner lines, where different sources drive a regular net. Right when a wire has various drivers, the wire's (discernable) regard is settled by a component of the source drivers and their characteristics.

A subset of clarifications in the Verilog vernacular is synthesizable. Verilog modules that fit in with a synthesizable coding style, known as RTL (enroll trade level), can be physically recognized by mix programming. Association programming algorithmically changes the

(dynamic) Verilog source into a net summary, a really indistinguishable depiction involving just of fundamental method of reasoning locals (AND, OR, NOT, flip-flops, et cetera.) that are available in a specific FPGA or VLSI advancement. Elevate controls to the net summary finally incite a circuit creation plan, (for instance, a photo cover set for an ASIC or a bitstream appeal to for a FPGA).

History

Beginning

Verilog was the principle current hardware depiction vernacular to be composed. It was made by Phil Moorby and Prabhu Goel in the midst of the winter of 1983/1984. The wording for this method was "Electronic Integrated Design Systems" (later renamed to Gateway Design Automation in 1985) as a hardware showing lingo. Entry Design Automation was obtained by Cadence Design Systems in 1990. Beat by and by has full selective rights to Gateway's Verilog and the Verilog-XL, the HDL-test framework that would transform into the genuine standard (of Verilog logic test frameworks) for the next decade. At first, Verilog was relied upon to depict and allow generation; just in this manner was help for amalgamation included.

Verilog-95

With the extending achievement of VHDL at the time, Cadence made the lingo available for open systematization. Musicality moved Verilog into the overall public space under the Open Verilog International (OVI) (now known as Accellera) affiliation. Verilog was later submitted to IEEE and advanced toward getting to be IEEE Standard 1364-1995, ordinarily insinuated as Verilog-95.

In a comparable day and age Cadence began the arrangement of Verilog-A to put standards support behind its straightforward test framework Specter. Verilog-A was never intended to be an autonomous lingo and is a subset of Verilog-AMS which wrapped Verilog-95.

Verilog 2001

Developments to Verilog-95 were submitted back to IEEE to cover the insufficiencies that customers had found in the primary Verilog standard. These developments advanced toward getting to be IEEE Standard 1364-2001 known as Verilog-2001.

Verilog-2001 is a basic overhaul from Verilog-95. In the first place, it incorporates unequivocal help for (2's supplement) checked nets and factors. As of now, code journalists expected to perform stamped exercises using lopsided piece level controls (for example, the total bit of a clear 8-bit development required an express delineation of the Boolean polynomial math to choose its correct regard). A comparative limit under Verilog-2001 can be simply more briefly portrayed by one of the intrinsic overseers: +, - ,/, *, >>>. A make/endgenerate manufacture (like VHDL's create/endgenerate) licenses Verilog-2001 to control event and verbalization instantiation through standard decision directors (case/if/else). Using make/endgenerate, Verilog-2001 can instantiate an assortment of events, with power over the system of the individual cases. Record I/O has been improved by a couple of new structure errands. Ultimately, two or three accentuation augmentations were familiar with upgrade code clarity (e.g. constantly @*, named parameter repeal, C-style work/errand/module header introduction).

Verilog-2001 is the mind-boggling sort of Verilog reinforced by the predominant piece of business EDA programming packs.

Verilog 2005

Not to be confused for System Verilog, Verilog 2005 (IEEE Standard 1364-2005) contains minor corrections, spec enlightenments, and a few new lingo features, (for instance, the uwire catchphrase).

An alternate bit of the Verilog standard, Verilog-AMS, attempts to organize basic and mixed banner showing with standard Verilog.

System Verilog

System Verilog is a superset of Verilog-2005, with various new features and capacities to enable blueprint to check and arrangement illustrating. Beginning at 2009, the System Verilog and Verilog vernacular rules were merged into SystemVerilog 2009 (IEEE Standard 1800-2009).

The methodology of gear affirmation lingos, for instance, Open Vera, and Verisity's e vernacular bolstered the change of Super log by Co-Design Automation Inc. Co-Design Automation Inc was later purchased by Synopsys. The foundations of Super log and Vera were given to Accellera, which later transformed into the IEEE standard P1800-2005:

In the late 1990s, the Verilog Hardware Description Language (HDL) transformed into the most comprehensively used vernacular for depicting gear for propagation and association. Regardless, the underlying two versions standardized by the IEEE (1364-1995 and 1364-2001) had quite recently fundamental forms for making tests. As setup sizes surpassed the affirmation limits of the tongue, business Hardware Verification Languages (HVL, for instance, Open Vera and ewere made. Associations that did not want to pay for these mechanical assemblies somewhat consumed many man-years making their own custom instruments. This benefit crisis (close by a practically identical one on the arrangement side) provoked the making of Accellera, a consortium of EDA associations and customers who expected to make the best in class time of Verilog. The endowment of the Open-Vera lingo surrounded the purpose behind the HVL features of System Verilog Accellera's goal was met in November 2005 with the gathering of the IEEE standard P1800-2005 for System Verilog, IEEE (2005). The most beneficial favorable position of System Verilog is that it empowers the customer to fabricate strong, repeatable check circumstances, in a dependable punctuation, that can be used over various exercises

There are various other supportive features, yet these empower you to make test seats at a bigger measure of reflection than you can achieve with a HDL or a programming vernacular, for instance, C.

System Verilog gives the best structure to achieve scope driven affirmation (CDV). CDV merges modified test age, self-checking testbenches, and degree estimations to in a general sense diminish the time spent affirming a diagram. The purpose behind CDV is to:

- Eliminate the effort and time spent making a few tests.
- Ensure cautious check spending front target setting.
- Receive early bungle admonitions and send run-time checking and botch examination to enhance investigating.

Delineations

Ex1: An appreciated world program looks like this:

```
module guideline;
starting
begin
$display ("Hello world!");
$finish;
end
endmodule
```

Ex2: A fundamental instance of two flip-flops takes after:

```
moduletoplevel (clock, reset);
input clock;
input reset;
reg flop1;
reg flop2;
```

```
always@ (posedge reset orposedge clock)
if(reset)
begin
flop1 <=0;
flop2 <=1;
end
else
begin
flop1 <= flop2;
flop2 <= flop1;
end
endmodule
```

The "<=" director in Verilog is another piece of its being a hardware delineation tongue rather than a run of the mill procedural lingo. This is known as a "non-blocking" errand. Its action doesn't select until the point that the accompanying clock cycle. This infers the demand of the assignments are irrelevant and will make a comparative result: flop1 and flop2 will swap regards each clock.

The other assignment head, "=", is implied as a blocking undertaking. At whatever point "=" undertaking is used, for the inspirations driving basis, the target variable is revived rapidly. In the above case, had the declarations used the "=" blocking manager as opposed to "<=", flop1 and flop2 would not have been swapped. Or maybe, as in regular programming, the compiler would grasp to simply set flop1 identical to flop2 (and henceforth ignore the abundance method of reasoning to set flop2 proportionate to flop1.)

6.3 Constants

The importance of constants in Verilog supports the development of a width parameter. The basic phonetic structure is:

<Width in bits>'<base letter><number>

Cases:

1. 12'h123 - Hexadecimal 123 (using 12 bits)

2. 20'd44 - Decimal 44 (using 20 bits - 0 development is customized)

3. 4'b1010 - Binary 1010 (using 4 bits)

4. 6'o77 - Octal 77 (using 6 bits)

6.4 Synthesizable Constructs

There are a couple of declarations in Verilog that have no basic in real hardware, e.g. $display. Thusly, an incredible piece of the tongue cannot be used to depict hardware. The cases presented here are the commendable subset of the vernacular that has a quick mapping to bona fide entryways.

/Mux cases - Three distinctive approaches to complete a comparative thing.

/The vital outline uses relentless undertaking

wire out;

dole out =sel? a: b;

/the second case uses a system

/to accomplish a comparative thing.

reg out;

always@(a or b orsel)

begin

case(sel)

```
1'b0: out = b;

1'b1: out = a;

endcase

end

/Finally - you can use if/else in a

/procedural structure.

reg out;

always@(a or b orsel)

if(sel)

out= a;

else

out= b;
```

The accompanying charming structure is a direct bolt; it will pass the commitment to the yield when the entryway hail is set for "experience", and gets the data and stores unending supply of the portal banner to "hold". The yield will remain stable paying little personality to the data signal while the portal is set to "hold". For the situation underneath the "experience" level of the entryway would be the time when the estimation of the if explanation is substantial, i.e. entryway = 1. This is scrutinized "if entryway is legitimate, the commotion is reinforced to latch_out reliably." Once the if condition is false, the last an impetus at latch_out will remain and is self-governing of the estimation of racket.

6.5 Initial Vs Always:

There are two separate strategies for announcing a Verilog strategy. These are the constantly and the hidden watchwords. The constantly catchphrase exhibits a free-running strategy. The basic watchword shows a strategy executes accurately once. The two forms begin execution at test framework time 0, and both execute until the complete of the square. Once a constantly square has accomplished its end, it is rescheduled (yet again). It is a run of the mill confused

judgment to assume that a fundamental square will execute before a constantly square. Honestly, it is more astute to consider the basic square a remarkable occasion of the reliably square, one which closes after it completes all of a sudden.

/Examples:

```
starting
begin
a =1;//Assign a motivating force to reg an at time 0
#1;//Wait 1 time unit
b = a;//Assign the estimation of reg a to reg b
end
always@(a or b)//Any time an or b CHANGE, run the strategy
begin
if(a)
c = b;
else
d =~b;
end//Done with this square, now return to the best (i.e. the @ event control)
always@(posedge a)//Run at whatever point reg a has a low to high change
a <= b;
```

These are the considerable uses for these two watchwords, anyway there are two vital additional occupations. The most generally perceived of these is an always keyword without the @(...) affectability list. It is possible to use reliably as showed as takes after:

```
persistently
begin//Always begins executing at time 0 and NEVER stops
clk=0;//Set clk to 0
#1;//Wait for 1 time unit
clk=1;//Set clk to 1
#1;//Wait 1 time unit
end//Keeps executing - so continue back at the most astounding purpose of the begin
```

The constantly catchphrase acts like the "C" manufacture while(1) {..} as in it will execute until the finish of time.

The other charming extraordinary case is the use of the basic catchphrase with the development of the unending length of time watchword.

6.6 Race Condition

The ask for of execution isn't always guaranteed inside Verilog. This can best be sketched out by a commendable case. Consider the code bit underneath:

```
starting
a =0;
starting
b = a;
starting
begin
#1;
$display ("Value a=%b Value of b=%b",a,b);
end
```

What will be printed out for the estimations of an and b? Dependent upon the demand of execution of the fundamental squares, it could be zero.

6.7 Operators

Note: These executives are not seemed orchestrated by need.

Head type	Operator symbols	Operation performed
Bitwise	~	Bitwise NOT (1's supplement)
	&	Bitwise AND
	|	Bitwise OR
	^	Bitwise XOR
	~^ or ^~	Bitwise XNOR
Logical	!	NOT
	&&	AND
	||	OR
Reduction	&	Reduction AND
	~&	Reduction NAND

| | Reduction OR
~| Reduction NOR
^ Reduction XOR
~^ or ^~ Reduction XNOR
Arithmetic + Addition
- Subtraction
- 2's supplement
* Multiplication
/ Division
** Exponentiation (*Verilog-2001)
Relational > Greater than
< Less than
>= Greater than or proportionate to
<= Less than or identical to
== Logical consistency (bit-regard 1'bX is ousted from examination)
!= Logical dissimilarity (bit-regard 1'bX is ousted from examination)
=== 4-state reasonable value (bit-regard 1'bX is taken as strict)
!== 4-state authentic difference (bit-regard 1'bX is taken as demanding)
Shift >> Logical right move
<< Logical left move
>>> Arithmetic right move (*Verilog-2001)
<<< Arithmetic left move (*Verilog-2001)
Concatenation {,} Concatenation
Replication {n{m}} Replicate regard m for n times
Conditional ?: Conditional

CHAPTER 7
SIMULATION RESULTS

CHAPTER 7
SIMULATION RESULTS

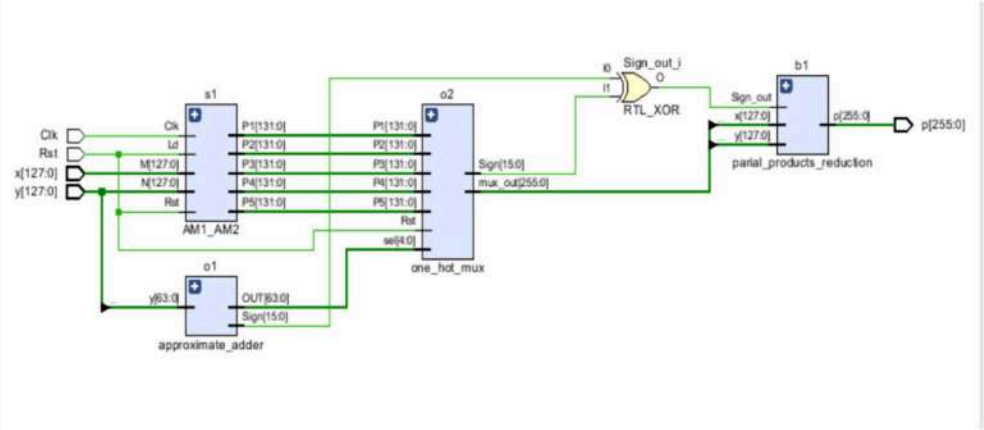


Fig. 7.1 Schematic diagram for Unsigned Multiplication

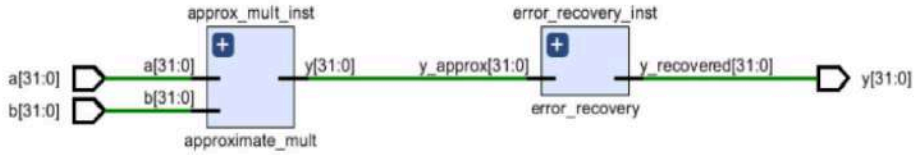


Fig. 7.2 Schematic diagram for Signed Multiplication

Fig 7.1 and 7.2 represents the Schematic diagram for Signed and Unsigned Multiplication

Fig. 7.3 Power Summary Unsigned Multiplications

Fig.7.3 represents the power consuming by using Vivado software. The consumed power is 9.288 w.

Fig. 7.4 Power Summary Signed Multiplications

Fig.7.4 represents the power consuming by using Vivado software. The consumed power is 6.632

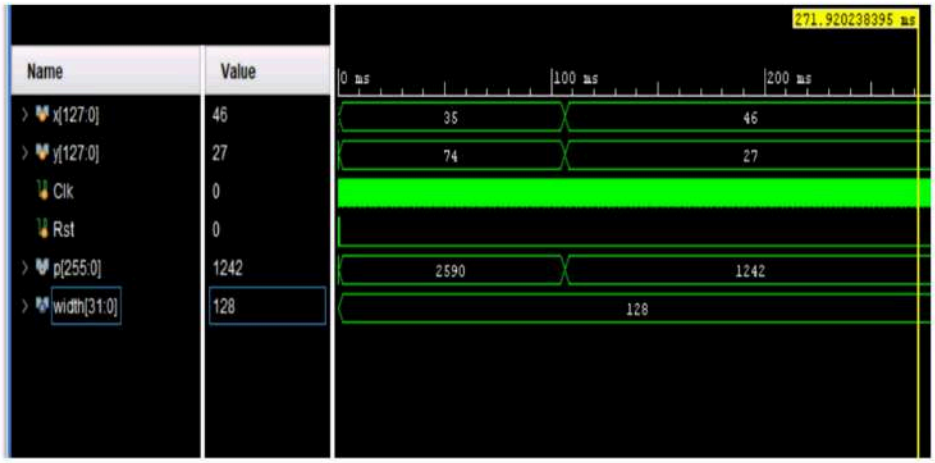


Fig. 7.5 Simulation Outcome for Unsigned Multiplication

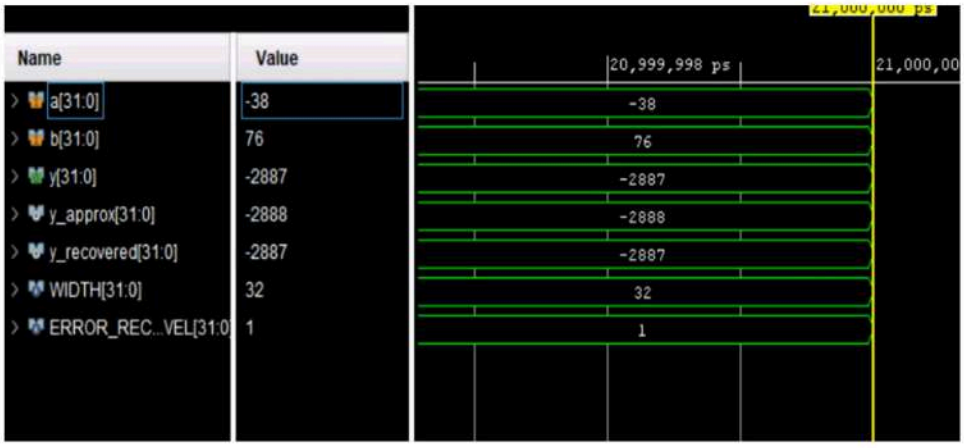


Fig. 7.6 Simulation Outcome for Signed Multiplication

TABLE 7.1 Performance Comparison

Parameter	**Unsigned Multiplication**	**Signed Multiplication**
Power	9.288w	0.933w
Time delay	34.196ns	14.156ns
LUTs	21541	306

The above table represents that the Signed multiplication consumes significantly less power compared to unsigned multiplication. This reduction in power consumption is likely due to the fact that signed multiplication typically involves fewer transitions and operations compared to unsigned multiplication.

Signed multiplication also exhibits a reduced time delay compared to unsigned multiplication. This improvement in speed is advantageous in applications where fast computation is critical. Signed multiplication utilizes far fewer resources (LUTs) compared to unsigned multiplication. This reduction in resource utilization can lead to more efficient use of FPGA or ASIC resources, allowing for more complex designs or additional functionalities within the same hardware constraints.

CHAPTER 8
CONCLUSION

CHAPTER 8
CONCLUSION

This Project proposes a high-performance and low-power approximate partial product accumulation tree for a multiplier using a newly designed approximate adder. The proposed approximate adder ignores the carry propagation by generating both an approximate sum and an error signal. OR gate and approximate adder based error reduction schemes are utilized, yielding two different approximate 8 × 8 multiplier designs: AM1 and AM2. Moreover, Compared to unsigned multiplication, Signed multiplication consumes less power. They are additionally found to have better accuracy when contrasted with existing surmised multiplier outlines. The proposed multiplier plans can be utilized in applications with negligible misfortune in yield quality while sparing huge influence and zone.

CHAPTER 9
REFERENCES

CHAPTER 9

REFERENCES

[1]. Honglan Jiang*, Student Member, IEEE, Cong Liu*, Fabrizio Lombardi, Fellow, IEEE and Jie Han, Senior Member, IEEE

[2] E. J. Ruler and E. E. Swartzlander, Jr., "Information subordinate truncation plot for parallel multipliers," in Proc. 31st Asilomar Conf. Signs, Circuits Syst., Nov. 1998, pp. 1178– 1182.

[3] K.- J. Cho, K.- C. Lee, J.- G. Chung, and K. K. Parhi, "Plan of low-blunder settled width adjusted corner multiplier," IEEE Trans. Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522– 531, May 2004.

[4] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-roused uncertain computational squares for effective VLSI execution of delicate figuring applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850– 862, Apr. 2010.

[5] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Plan and examination of inexact blowers for duplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984– 994, Apr. 2015.

[6] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Stop, and N. S. Kim,"Energy-effective inexact augmentation for computerized flag preparing and grouping applications," IEEE Trans. Large Scale Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180– 1184, Jun. 2015.

[7] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Outline proficient surmised increase circuits through halfway item aperture," IEEE Trans. Large Scale Integer. (VLSI) Syst., vol. 24, no. 10, pp. 3105– 3117, Oct. 2016.

[8] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Exchanging precision for control in a multiplier engineering," J. Low Power Electron., vol. 7, no. 4,pp. 490– 501, 2011.

[9] C.- H. Lin and C. Lin, "High precision estimated multiplier with blunder revision," in Proc. IEEE 31st Int. Conf. Comput. Plan, Sep. 2013,pp. 33– 38.

[10] C. Liu, J. Han, and F. Lombardi, "A low-control, elite surmised multiplier with configurable halfway mistake recuperation," in Proc. Conf. Display. (DATE), 2014, pp. 1– 4.

[11] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and examination of circuits for surmised registering," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Oct. 2011, pp. 667– 673.

[12] J. Liang, J. Han, and F. Lombardi, "New measurements for the dependability of surmised and probabilistic adders," IEEE Trans. Compute., vol. 63, no. 9, pp. 1760– 1771, Sep. 2013.

[13] S. Suman et al., "Picture improvement utilizing geometric mean channel and gamma revision for WCE images," in Proc. 21st Int. Conf., Neural Inf. Process. (ICONIP), 2014, pp. 276– 283

APPENDIX

International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307

OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/CSEIT2410311

Low-Power Approximate Unsigned and Signed Multipliers with Configurable Error Recovery

Dr. K. Nagi Reddy[1], K. Ruchitha[2], B. Sai Srinivas[2], D. Venu, K. Vinay[2]

[1]Professor of ECE, NBKR Institute of Science & Technology, Vidyanagar, Nellore, Andhra Pradesh, India

[2]NBKR Institute of Science & Technology, Vidyanagar, Nellore, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 25 April 2024

Published: 05 May 2024

Publication Issue

Volume 10, Issue 3

May-June-2024

Page Number

109-117

ABSTRACT

In the field of Digital Signal Processing and similar applications, Approximate circuits are being explored as a means to enhance performance and energy efficiency by sacrificing a degree of accuracy. Within these circuits, Multipliers play a crucial role and are being investigated for their potential impact on overall system optimization. In this paper, a novel approximate multiplier with a low power consumption and a short critical path is proposed for high-performance DSP applications. This multiplier leverages a newly designed approximate adder that limits its carry propagation to the nearest neighbours for fast partial product accumulation. Different levels of accuracy can be achieved by using either OR gates or the proposed approximate adder in a configurable error recovery. The multipliers using these two error reduction strategies are referred to as Approximate Multiplier 1 (AM1) and Approximate Multiplier 2 (AM2), respectively. Both AM1 and AM2 have a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared with a Unsigned multiplication multiplier, with the signed multiplication. The signed multiplication tends to have lower power consumption is observed. By utilizing an appropriate error recovery, the proposed approximate multipliers achieve similar processing accuracy as traditional exact multipliers, but with significant improvements in power.

Keywords : Multiplier, Digital Signal Processing, Digital Systems, Digital Systems, Carry Propagation Adder

I. INTRODUCTION

In digital systems where human interpretation or inherent error tolerance is present, such as multimedia, recognition, and data mining applications, strict precision in computation may not be essential, opening avenues for approximate computing to significantly reduce area, power, and delay while maintaining acceptable performance levels and energy efficiency [1].

As one of the key components in arithmetic circuits, adders have been extensively studied for approximate implementation [2]–[8]. The so-called speculative

adders operate by using a reduced number of less significant input bits to calculate the sum, because the typical carry propagation chain is usually shorter than the width (in bits) of an adder [2]. An error detection and recovery scheme has been proposed in [3] to extend the scheme of [2] for a reliable adder with variable latency. A reliable variable-latency adder based on carry select addition has been presented in [8]. As a number of approximate adders have been proposed, new methodologies to model, analyse and evaluate them have been discussed in [9]–[12].

One area of focus that has received comparatively less attention is the optimization of approximate multipliers, despite their fundamental role in various computational tasks. A multiplier usually consists of 3 stages: Partial product generation, Partial product accumulation and a Carry Propagation Adder (CPA) at the final stage [13]. Partial product generation is a crucial stage in the multiplication process, particularly in digital circuits like multipliers. It involves breaking down the multiplication of two numbers into smaller, simpler operations. For example, in binary multiplication, each digit of the multiplicand is multiplied with each digit of the multiplier. These individual multiplications result in partial products, which are then added together to produce the final product. In partial product generation, each multiplication operation between a digit of the multiplicand and a digit of the multiplier is performed separately. Partial product accumulation is the stage in the multiplication process where the partial products, generated in the previous stage, are combined or accumulated to obtain the final result of the multiplication operation.

In binary multiplication, after generating the partial products by multiplying each digit of the multiplicand with each digit of the multiplier, these partial products are aligned according to their positions and added together to produce intermediate sums. These intermediate sums represent partial results of the overall multiplication operation. A carry propagation adder (CPA) is a type of digital circuit used for addition in computer arithmetic. It's designed to add two binary numbers together, producing a sum along with any carry bits that result from the addition of corresponding bits in the numbers. This paper focuses on the investigation of low power consumption in both unsigned and signed multiplication operations. The findings will contribute to a deeper understanding of power-efficient arithmetic operations and may have implications for the design and optimization of digital systems. The multiplier design is proposed using a simple, yet fast approximate adder. This newly designed adder can process data in parallel by cutting the carry propagation chain. It has a critical path delay that is even shorter than a conventional one-bit full adder. Although with a high error rate, this adder simultaneously computes the sum and generates an error signal; this feature is employed to reduce the error in the final result of the multiplier.

II. DESIGN OF PROPOSED MULTIPLIER

A. The Approximate adder

In this section, A novel approximate adder designs introduced, which functions on a pre-processed set of inputs. This Input Pre-Processing (IPP) methodology capitalizes on the interchangeable nature of bits possessing identical weights across distinct addends. For example, consider two sets of inputs to a 4-bit adder: i) A = 1010, B = 0101 and ii) A = 1111, B = 0000. Clearly, the additions in i) and ii) produce the same result. In this process, the two input bits Ai Bi = 01 are equivalent to Ai Bi = 10 (with i being the bit index) because of the interchangeability of the corresponding bits in the two operands. The basic rule for the IPP is to switch Ai and Bi if Ai = 0 and Bi = 1 (for any i), while keeping the other combinations (i.e., Ai Bi = 00, 10 and 11) unchanged. By doing so, more 1's are expected in A and more 0's are expected in B. If A ' i B ' i are the ith

bits in the pre-processed inputs, the IPP functions are given by:

$$A' i = Ai + Bi \quad (1)$$

$$B' i = AiBi \quad (2)$$

TABLE 1 : Truth table of the inexact addition cell. 'x' addresses that no such a mix happens due to the IPP

S_i/E_i		$\dot{B}_i\dot{B}_{i-1}$			
		00	01	11	10
$\dot{A}_i\dot{A}_{i-1}$	00	0/0	X	X	X
	01	0/0	1/0	X	X
	11	1/0	1/1	1/0	0/0
	10	1/0	X	X	0/0

(1) and (2) compute the propagate and generate signals used in a parallel adder such as the Carry Look-Ahead (CLA). The proposed adder can process data in parallel by cutting the carry propagation chain. Let A and B denote the two input binary operands of an adder, S be the sum result, and E represent the error vector. Ai , Bi , Si and Ei are the ith least significant bits of A, B, S and E, respectively. A carry propagation chain starts at the i^{th} bit when $B' i = 1$, $A' i+1 = 1$, $A' i+1 = 0$. In an accurate adder, Si+1 is 0 and the carry propagates to the higher bit. However, in the proposed approximate adder, Si+1 is set to 1 and an error signal is generated as Ei+1 = 1. This prevents the carry signal from propagating to the higher bits. Hence, a carry signal is produced only by the generate signal, i.e., Ci = 1 only when B i = 1, and it only propagates to the next higher bit, i.e., the (i + 1)th position. Table I shows the truth table of the approximate adder, where $A' i$, $B' i$ and $B' i–1$ are the inputs after IPP. The error signal is utilized for error compensation purposes as discussed in a later section. In this case, the approximate adder is similar to a redundant number system [14] and the logical functions of Table I are given by

$$Si = B' i–1 + \overline{B'} i A' i \quad (3)$$

$$Ei = + \overline{B'} i B i–1 A' i \quad (4)$$

By replacing $A' i$ and $B' i$ using (1) and (2) respectively, the logic functions with respect to the original inputs are given by

$$Si = (Ai \oplus Bi) + Ai–1Bi–1 \quad (5)$$

$$Ei = (Ai \oplus Bi)Ai–1Bi–1 \quad (6)$$

where i is the bit index, i.e., i = 0, 1, … , n for an n-bit adder. Let A–1 = B–1 = 0 when i is 0, thus, S0 = A0 ⊕ B0 and E0 = 0. Also, Ei = 0 when Ai–1 or Bi–1 is 0.

Consider an n-bit adder, the inputs are given by A = An–1 … A1 A0 and B = Bn–1 · · · B1B0, the exact sum is S = Sn–1….S1S0. Then, Si can be computed as Si +Ei and thus, the exact sum of A and B is given by

$$S = S + E \quad (7)$$

In (7) '+' means the addition of two binary numbers rather than the 'OR' function. The error E is always non-negative and the approximate sum is always equal to or smaller than the accurate sum. This is an important feature of this adder because an additional adder can be used to add the error to the approximate sum as a compensation step. While this is intuitive in an adder design, it is a particularly useful feature in a multiplier design as only one additional adder is needed to reduce the error in the final product.

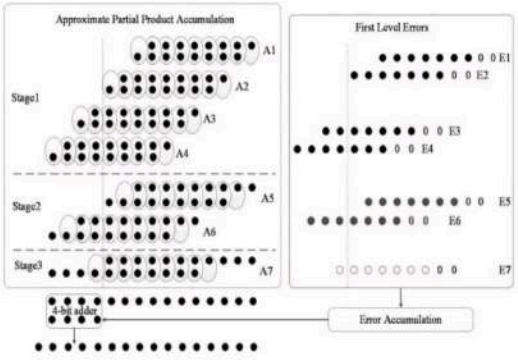


Fig. 1 illustrates an approximate multiplier featuring partial error recovery, utilizing the five most significant bits (MSBs) of the error vector.

The diagram depicts the generation of partial products, sums, and error bits across different stages: an error bit

is generated at the first stage, followed by additional error bits at subsequent stages, culminating in the last stage.

B. Proposed Approximate Multiplier

The incorporation of approximate adders in partial product accumulation streamlines the multiplier's architecture, offering a balance between simplicity and performance optimization. . It has been shown that this may lead to poor performance [15], because errors may accumulate and it is difficult to correct errors using existing approximate adders. However, the use of the newly proposed approximate adder overcomes this problem by utilizing the error signal. The resulting design has a critical path delay that is shorter than a conventional one-bit full adder, because the new n-bit adder can process data in parallel. The approximate adder has a rather high error rate, but the feature of generating both the sum and error signals at the same time reduces errors in the final product. An adder tree is utilized for partial product accumulation; the error signals in the tree are then used to compensate the error in the output to generate a product with a better accuracy. The architecture of the proposed approximate multiplier is shown in Fig. 1. In the proposed approximate multiplier, the simplification of the partial product accumulation stage is accomplished by using an adder tree, in which the number of partial products is reduced by a factor of 2 at each stage of the tree. This scheme is usually not implemented using accurate multi-bit adders, because either the hardware overhead or the delay is unacceptable. However, the newly proposed approximate adder is suitable for implementing an adder tree, because it is less complex than a conventional adder and has a much shorter critical path delay.

III. ERROR REDUCTION

Subsequent to the approximate adder's production of both the approximate sum S and the error E, the exploration shifts towards harnessing the error signal to alleviate inaccuracies within the multiplier's functionality, enhancing its overall precision and reliability. As [7] is applicable to the sum of every single approximate adder in the tree, an error reduction circuit is applied to the final multiplication result rather than to the output of each adder. Two steps are required to reduce errors: i) Error accumulation and ii) Error recovery by the addition of the accumulated errors to the adder tree output using an adder. In the error accumulation step, error signals are accumulated to be a single error vector, which is then added to the output vector of the partial product accumulation tree. Two approximate error accumulation methods are proposed, yielding the approximate multiplier 1 (AM1) and approximate multiplier 2 (AM2). Fig. 2 shows the symbols for an OR gate, a full adder and half adder cell and an approximate adder cell used in the error accumulation tree.

A. Error Accumulation for Approximate Multiplier 1

As displayed in Fig 1, each approximate adder Ai generates a sum vector Si and an error vector Ei, where $i = 1, 2, \cdots, 7$.

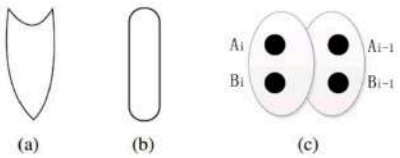


Fig. 2. Symbols for (a) an OR gate, (b) an full adder or a half adder and (c) an approximate adder cell

If the error signals are added using accurate adders, the accumulated error can fully compensate the inaccurate product; however to reduce complexity, an approximate error accumulation is introduced. Consider the observation that the error vector of each approximate adder tends to have more 0's than 1's. Therefore, the probability that the error vectors have

an error bit '1'at the same position, is quite small. Hence, an OR gate is used to approximately compute the sum of the errors for a single bitas shown in fig 2. If m error vectors (denoted by E1, E2, ..., Em) have to be accumulated, then the sum of these vectors is obtained as

$$Ei = E1i \text{ OR } E2i \text{ OR } ... \text{ OR } Emi \quad (8)$$

To reduce errors, an accumulated error vector is added to the adder tree output using a conventional adder (e.g. a carry look-ahead adder). However, only several (e.g. k) MSBs of the error signals are used to compensate the outputs and further reduce the overall complexity. The number of MSBs is selected according to the extent that errors must be compensated. For example, in an 8 × 8 adder tree, there are a total of 7 error vectors, generated by the 7 approximate adders in the tree. However, not all the bits in the 7 vectors need to be added, because the MSBs of some vectors are less significant than the least significant bits of the k MSBs. In the example of Fig. 1, 5 MSBs (i.e. the (11 – 14)th bits, no error is generated at the 15th bit position) are considered for error recovery and therefore, 4 error vectors are considered (i.e., the error vectors of adders E3, E4, E6 and E7). The error vectors of the other three adders are less significant than the 11th bit, so they are not considered. The accumulated error E is obtained using (8); then, the final result is found by adding E to S using a fast accurate adder. The error accumulation scheme is shown in Fig. 3. As no error is generated at the least significant two bits of each approximate adder Ai (i = 1, 2, · · · , 7), the least significant two bits of each error vector Ei are not accumulated.

B. Error Accumulation for Approximate Multiplier 2

Figure 4 illustrates the error accumulation scheme utilized for AM2, providing insight into how errors propagate and accumulate throughout the approximate multiplier's operation. To introduce the design of AM2, consider an 8 × 8 multiplier with two inputs X and Y . For example, consider the first two partial product vectors X0Y7, X0Y6, ..., X0Y0 and X1Y7, X1Y6, ..., X1Y0 accumulated by the first approximate adder (A1 in Fig. 1), where Xi and Yi are the i th least significant bits of X and Y , respectively. Recall from (6) for the approximate adder, the condition for Ei = 1 is

$$Ai-1 = Bi-1 = 1 \text{ and } Ai \neq Bi \quad (9)$$

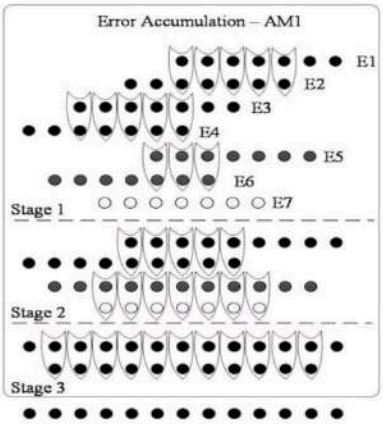


Fig. 3. Error accumulation tree for AM1. : an error bit generated at the first stage; : an error bit generated at the second stage; : an error bit generated at the last stage.

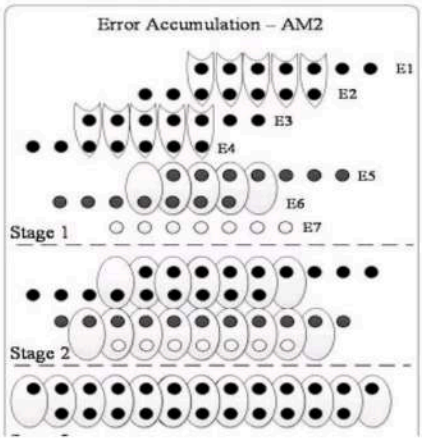


Fig. 4. Error accumulation tree for AM2. : an error bit generated at the first stage; : an error bit generated at the second stage; : an error bit generated at the last stage.

For the first approximate adder in the partial product accumulation tree, its inputs are A = X0Y7, X0Y6, ..., X0Y0 and B = X1Y7, X1Y6, ..., X1Y0. Thus, the i th least significant bits for A and B are Ai = X0Yi and Bi = X1Yi−1, respectively. If X0 or X1 is 0, there will be no error in this approximate adder because either A or B is zero. Therefore, no error occurs unless X0X1 = 11. When X0X1 = 11, Ai and Bi are simplified to Yi and Yi−1, respectively. Then to calculate Ei , Ai−1, Bi−1, Ai and Bi are replaced by Yi−1, Yi−2, Yi and Yi−1, respectively. For Ei to be 1, YiYi−2Yi−1 = 011 according to (9). Therefore, an error only occurs when the input has "011" as a bit sequence. Based on this observation, the "distance" between two errors in an approximate multiplier is at least 3 bits. Thus, two neighbouring approximate adders in the first stage of the partial product tree cannot have errors at the same column, because the errors in a lower approximate adder are those in the upper adder shifted by 2 bits when both errors exist. The errors in two neighbouring approximate adders can then be accurately accumulated by OR gates, e.g. an OR gate can be used to accumulate the two bits in the error vectors E1 and E2 in Fig. 1. After applying the OR gates to accumulate E1 and E2 as well as E3 and E4, the four error vectors are compressed into two. For E5, E6 and E7, they are generated from the approximate sum of the partial products rather than the partial products. Therefore, they cannot be accurately accumulated by OR gates.

Another interesting feature of the proposed approximate adder is as follows. Assume Ei = 1 in (6), then Ai−1 = Bi−1 = 1 and Ai 6= Bi . Since Ai−1 = Bi−1 = 1, i.e., Ai−1 ⊕Bi−1 = 0, it is easy to show that Ei−1 = 0. Moreover as Ai 6= Bi , i.e., AiBi = 0, then Ei+1 = 0. Thus, once there is an error in one bit, its neighbouring bits are error free, i.e., there are no consecutive error bits in one row. Therefore, there is no carry propagation path longer than two bits when two error vectors are accumulated, and the error vectors are accurately accumulated by the proposed approximate adder. Based on the above analysis, E5 and E6 are accurately accumulated by one approximate adder in the first stage of the error accumulation. After the first stage of error accumulation, three vectors are generated, and another two approximate adders are then used to accumulate these three vectors as well as the error vector remaining from the previous stage (E7). Simulation results (found in later sections) show that the modified error accumulation outperforms the OR-gate error accumulation with little overhead on delay and power as depicted in fig.5

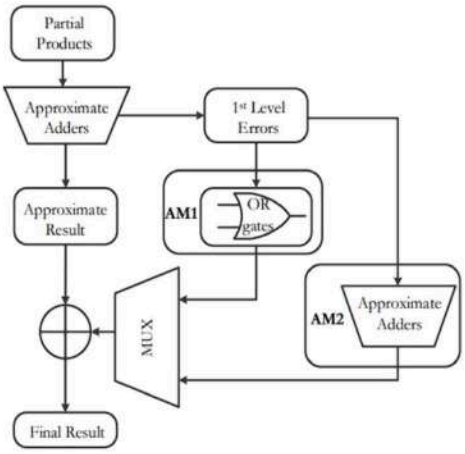


Fig 5. Block diagram of the Proposed Multipliers.

C. 16 x 16 Approximate Multipliers

In both AM1 and AM2, all the error vectors are compressed to one error vector, which is then added back to the approximate output of the partial product tree. Compared to 8 × 8 designs, 16 × 16 multipliers generate more error vectors, and too much information would be ignored if the same error reduction strategies are used. That is, using only one compressed error vector does not make a good estimation of the overall error. In the modified designs, the error vectors generated by the approximate adders are compressed to two final error vectors. Take a 16 × 16 AM1 as an example, the eight error vectors generated at the first stage of the partial product accumulation tree are compressed to one error vector, EV1, using OR gates. The remaining seven error vectors from the second,

third and fourth stages are compressed to another error vector EV2. Then both EV1 and EV2 are added back to the output of the partial product at the fourth stage. Similarly, the proposed approximate adders are used in a 16 × 16 AM2 to compress the eight error vectors from the first stage to one error vector and the remaining error vectors to another error vector. Truncation can also be applied to the proposed designs for large input operands. Therefore, 16 LSBs of the partial products are truncated in 16 × 16 AM1 and AM2, resulting in truncated AM1 (TAM1) and truncated AM2 (TAM2).

IV. SIMULATION RESULTS

For Unsigned Multiplication

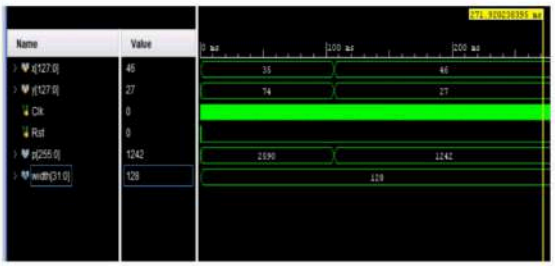


Fig 6 : Simulation outcome

Table 2 : Design Summary results

Resource	Estimation	Available	Utilization...
LUT	21541	433200	4.97
FF	645	866400	0.07
IO	514	850	60.47
BUFG	2	32	6.25

The Table 2 represents the synthesis implementation by using the Vivado software. From the above table, it is observed that only 21541 look up tables are used out of available of 433200. It indicates very less area (5%) was used for the design.

Table 3 : Power summary unsigned multiplications

Table 3 represents the power consuming by using Vivado software. The consumed power is 9.288 w.

For Signed Multiplication

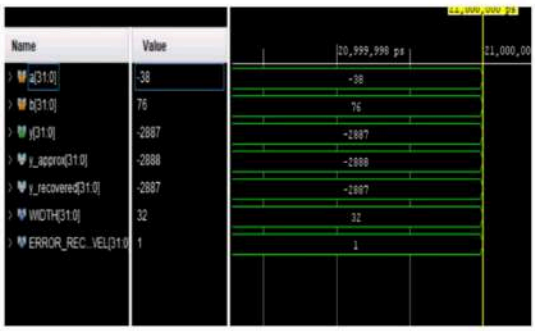


Fig 7 : Simulation Outcome

Table 4: Design summary for Signed Multiplication

Resource	Estimation	Available	Utilization %
LUT	306	303600	0.10
DSP	36	2800	1.29
IO	384	700	54.86

The Table4 represents the synthesis implementation by using the Vivado software. Compared to the Unsigned Multiplication, the area occupied for the signed multiplication is less i.e. 0.10 %

Table 5 : Power summary signed multiplications

Table 5 represents the power consuming by using Vivado software. The consumed power is 6.632.

Table 6. PERFOMANCE COMPARISON

Parameter	Unsigned Multiplication	Signed Multiplication
Power	9.288w	0.933w
Time delay	34.196ns	14.156ns
Luts	21541	306

V. CONCLUSION

This paper proposes a high-performance and low-power approximate partial product accumulation tree for a multiplier using a newly designed approximate adder. The proposed approximate adder ignores the carry propagation by generating both an approximate sum and an error signal. OR gate and approximate adder based error reduction schemes are utilized, yielding two different approximate 8 × 8 multiplier designs: AM1 and AM2. Moreover, Compared to unsigned multiplication, Signed multiplication consumes less power.

VI. REFERENCES

[1]. Honglan Jiang*, Student Member, IEEE, Cong Liu*, Fabrizio Lombardi, Fellow, IEEE and Jie Han, Senior Member, IEEE

[2]. S. L. Lu, "Speeding up processing with approximation circuits," Computer, vol. 37, no. 3, pp. 67–73, 2004.

[3]. A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in Proceedings of the conference on Design, automation and test in Europe. ACM, 2008, pp. 1250–1255.

[4]. N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power highspeed adder for error-tolerant application," in Proceedings of the 2009 12th International Symposium on Integrated Circuits. IEEE, 2009, pp. 69–72.

[5]. H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bioinspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 4, pp. 850–862, 2010.

[6]. V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: imprecise adders for low-power approximate computing," in International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2011, pp. 409–414.

[7]. A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in Proceedings of the 49th Annual Design Automation Conference. ACM, 2012, pp. 820–825.

[8]. K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in Design, Automation Test in Europe Conference Exhibition (DATE), 2012, pp. 1257–1262.

[9]. J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," Computers, IEEE Transactions on, vol. 62, no. 9, pp. 1760–1771, 2013.

[10]. J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality trade off using imprecise hardware," in Proceedings of the 49th Annual Design Automation Conference. ACM, 2012, pp. 504–509.

[11]. J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modelling and synthesis of quality-energy optimal approximate adders," in Proceedings of the International Conference on Computer-Aided Design. ACM, 2012, pp. 728–735.

[12]. R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modelling and analysis of circuits for approximate computing," in Proceedings of the International Conference on Computer-Aided Design. IEEE Press, 2010, pp. 667–673.

[13]. H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification and comparative evaluation of approximate arithmetic circuits," ACM Journal on Emerging Technologies in Computing Systems, vol. 13, no. 4, p. 60, 2017

[14]. B. Parhami, Computer arithmetic. Oxford university press, 2000.

[15]. P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," Journal of Low Power Electronics, vol. 7, no. 4, pp. 490–501, 2011.

International Journal of Scientific Research in Computer Science, Engineering and Information Technology

Peer Reviewed and Refereed International Scientific Research Journal

[UGC Journal No : 64718] **Scientific Journal Impact Factor : 8.154**

Certificate of Publication

Ref : Certificate/Volume 10/Issue 3/CSEIT2410311 06-May-2024

This is to certify that **Dr. K. Nagi Reddy, K. Ruchitha, B. Sai Srinivas, D. Venu, K. Vinay** have published a research paper entitled *'Low-Power Approximate Unsigned and Signed Multipliers with Configurable Error Recovery'* in the International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), Volume 10, Issue 3, May-June 2024.

This Paper can be downloaded from the following IJSRCSEIT website link
https://ijsrcseit.com/index.php/home/article/view/CSEIT2410311
DOI : https://doi.org/10.32628/CSEIT2410311

IJSRCSEIT Team wishes all the best for bright future

Editor in Chief
IJSRCSEIT

ISSN : 2456-3307

Website : https://ijsrcseit.com

International Journal of Scientific Research in Computer Science, Engineering and Information Technology

Peer Reviewed and Refereed International Scientific Research Journal

[UGC Journal No : 64718] **Scientific Journal Impact Factor : 8.154**

Certificate of Publication

Ref : Certificate/Volume 10/Issue 3/CSEIT2410311 06-May-2024

This is to certify that **Dr. K. Nagi Reddy** has published a research paper entitled 'Low-Power Approximate Unsigned and Signed Multipliers with Configurable Error Recovery' in the International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), Volume 10, Issue 3, May-June 2024.

This Paper can be downloaded from the following IJSRCSEIT website link

https://ijsrcseit.com/index.php/home/article/view/CSEIT2410311

DOI : https://doi.org/10.32628/CSEIT2410311

IJSRCSEIT Team wishes all the best for bright future

Editor in Chief
IJSRCSEIT

ISSN : 2456-3307

Website : https://ijsrcseit.com

International Journal of Scientific Research in Computer Science, Engineering and Information Technology

Peer Reviewed and Refereed International Scientific Research Journal

[UGC Journal No : 64718] **Scientific Journal Impact Factor : 8.154**

Certificate of Publication

Ref : Certificate/Volume 10/Issue 3/CSEIT2410311 06-May-2024

This is to certify that **K. Vinay** has published a research paper entitled 'Low-Power Approximate Unsigned and Signed Multipliers with Configurable Error Recovery' in the International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), Volume 10, Issue 3, May-June 2024.

This Paper can be downloaded from the following IJSRCSEIT website link
https://ijsrcseit.com/index.php/home/article/view/CSEIT2410311

DOI : https://doi.org/10.32628/CSEIT2410311

IJSRCSEIT Team wishes all the best for bright future

Editor in Chief
IJSRCSEIT

ISSN : 2456-3307

Website : https://ijsrcseit.com