Design of High Performance Floating Point Multiply and Add Unit

A Thesis Submitted in Fulfillment of the Requirement for the Award of the Degree of

MASTER OF TECHNOLOGY

in

VLSI DESIGN

Submitted By

Deepti Sharma

601562009

Under Supervision of

Ms. Harpreet Vohra

Assistant Professor, ECED

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY, PATIALA, PUNJAB

JUNE, 2017

DECLARATION

I, Deepti Sharma hereby declare that the work presented in this thesis entitled " **Design of High Performance Floating Point Multiply and Add Unit**" in partial fulfillment of the requirement for the award of degree of Master of Engineering in VLSI Design submitted at Electronics & Communication Department, Thapar University, Patiala is an authentic record of work carried out under supervision of **Ms. Harpreet Vohra** (Assistant Professor, ECED, Thapar University, Patiala) The matter presented in this thesis has not been submitted either in part or full to any other university or institute for the award of any other degree.

Date: 16 Aug., 17

Deepti Sharma

Deepti Sharma

601562009

It is certified that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 16 Aug., 17

Ms. Harpreet Vohra

Assistant Professor, ECED

Thapar University, Patiala

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **Ms. Harpreet Vohra**, **Assistant Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala for her patient guidance and support throughout this report. I am truly very fortunate to have the opportunity to work with her. I found this guidance to be extremely valuable. I am also thankful to our **Head of the Department, Dr. Alpana Aggarwal** as well as **PG Coordinator, Dr. Hemdutt Joshi, Assistant Professor and Program Coordinator, Dr. Anil Arora, Assistant Professor, ECED,** entire faculty and staff of Electronics and Communication Engineering Department for providing us adequate environment in carrying the work.

I am also thankful to all my friends for giving their valuable time and sharing ideas with me. Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.

The study has indeed helped me to explore knowledge and avenues related to my topic and I am sure it will help me in my future.

Deepti Sharma

Place: Patiala

Date: 16-Aug-17

Deepti Sharma

M.Tech Final Year

Thapar University

ABSTRACT

Floating point Arithmetic is mainly used in biometrics, medical imaging, applications of audio, motion capture, conferencing, broadcasting etc. Multiplier is considered as the main element of the high performance systems so,it should be designed in such a way that it has high speed, less area and less power consumption.

Due to advancement in technology and increasing needs for high speed calculations, the design of such multipliers which offer high speed, low power consumption, regularity of layout and less area was mandatory. However area and speed are two conflicting constraints. So improving speed results always in larger areas.

In this thesis a 64-bit high performance Floating Point Multiply and Add (MAC) unit is presented that can perform both fixed and floating point addition and multiplication. An attempt has been made to reduce multiplication time. This thesis explores the possibility of combining fixed and floating point units thereby saving area without sacrificing too much speed. This design has been simulated on Xilinx ISE to analyse delay and power consumption.

TABLE OF CONTENTS

LIST OF FIGURES

Sr. No **Figure Details** **Page No**

LIST OF TABLES

CHAPTER-1

INTRODUCTION

This chapter gives a brief introduction to the floating point unit. Background and motivation of floating point operations and the necessity of combining the fixed and floating MAC unit is explained. A brief overview of the floating point operations along with the concepts that form the base of the proposed work are also presented. It is finally concluded by the organisation and flow of the thesis.

BACKGROUND AND MOTIVATON

As the name suggests, the word float means something that is not stationary. Floating point in mathematics means no fixed number of digits after and before the decimal point. On the other hand, in fixed point representation, there are fixed number of digits and decimal point in that number of digits is set. The representation of floating point numbers are less accurate and slower than the representation of fixed point numbers however, they can hold a much larger range of numbers than the fixed digit representations. Some applications like DSP processors govern its performance and accuracy.

In 1990, IBM implemented a floating point fused multiply-add arithmetic unit based on the RISC System [1], to increase the performance of the previously proposed design. This new design incorporated a floating-point multiplication fused with floating point addition into a single hardware block which was later termed as the floating point fused multiplier-adder. In 2011, floating point microarchitecture was proposed that increased the performance of this multiply and add unit while decreasing its latency [1]. This design was further improved in 2015 with the aspire of reducing power consumption and area [34] that consisted of high performance 64-bit MAC unit based on modified Wallace multiplier.

IEEE STANDARD 754 FOR FLOATING POINT ARITHMETIC BINARY

Only one encoding is specified by IEEE Standard for the representation of binary [35]. However, two encodings are specified for the representation of decimal digits. These encodings are also called interchanged formats as described by the implementers. This IEEE formats are shown by Table1.1. In this *Emax* is the maximum exponent, the base is two for binary and ten for decimal, and number of bits are 32 and 64-bits for single and double precision respectively.

The advantages of IEEE standard 754 are enumerated below:

- Extended and basic formats of floating point numbers
- Operations like Subtraction, Addition, Division, Multiplication, remainder, comparison and square root can be performed
- Conversions among formats of floating point and integer numbers
- Conversion among various floating point number format
- Comparison among formats of decimal strings and floating point numbers

Table 1.1 Basic IEEE754 Standard Format

Format	Base	Bits	Emax
Binary32	2	23+1	+127
Decimal64	10	16	+384
Binary64	2	52+1	+1023
Decimal128	10	34	+6144

FLOATING POINT ARITHMETIC UNIT

An arithmetic logic unit performs both logical and arithmetic operations on binary numbers. But floating point unit performs various operations on floating point numbers. A floating point arithmetic unit are proposed to perform different operations on floating point numbers such as addition (as shown in fig. 1.1), subtraction (as shown in fig. 1.2), multiplication (as shown in fig. 1.3), division(as shown in fig. 1.4) etc.

- **Addition Algorithm**

  Fig1.1 shows the addition algorithm of floating point numbers.

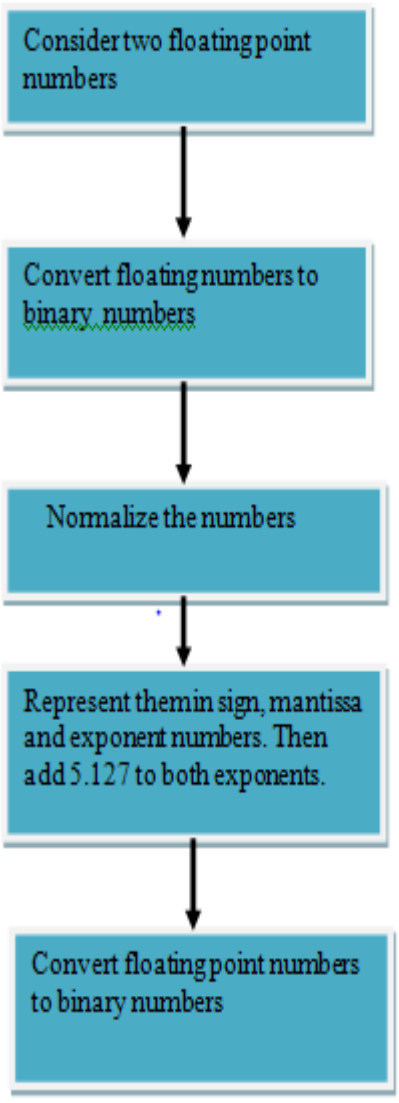


Fig1.1 Addition Algorithm

- **Subtraction Algorithm**

  Fig1.2 shows the subtraction algorithm of floating point numbers.

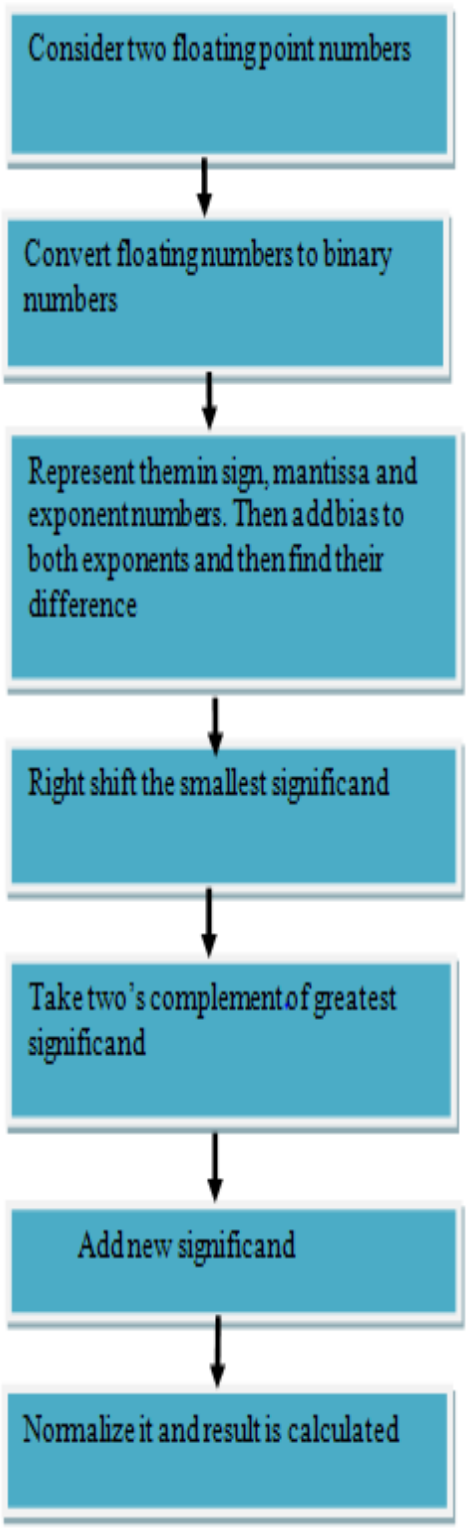


Fig1.2 Subtraction Algorithm

- **Multiplication Algorithm**

  Fig1.3 shows the multiplication algorithm of floating point numbers.

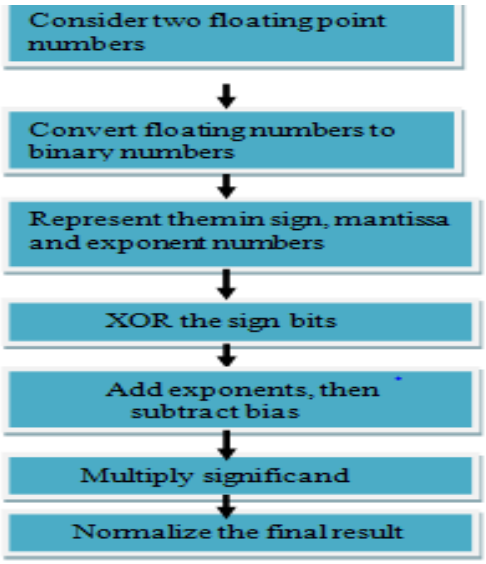


Fig1.3 Multiplication Algorithm

- **Division Algorithm**

  Fig1.4 shows the division algorithm of floating point numbers.

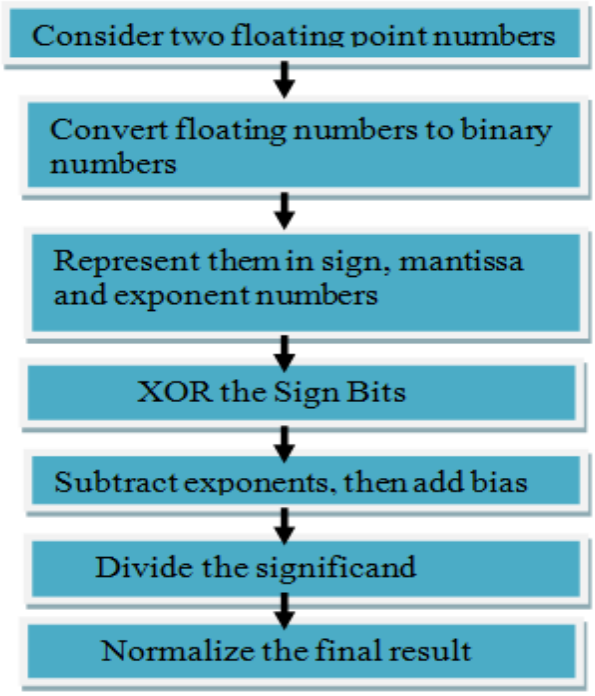


Fig1.4 Division Algorithm

FLOATING POINT MULTIPLIER

Fig1.5 shows the floating point multiplier. Multiplying two floating point numbers is a difficult task which is performed in steps. It consists of three parts i.e. sign, mantissa and exponent. But the calculations for these parts are carried out separately in series of steps.

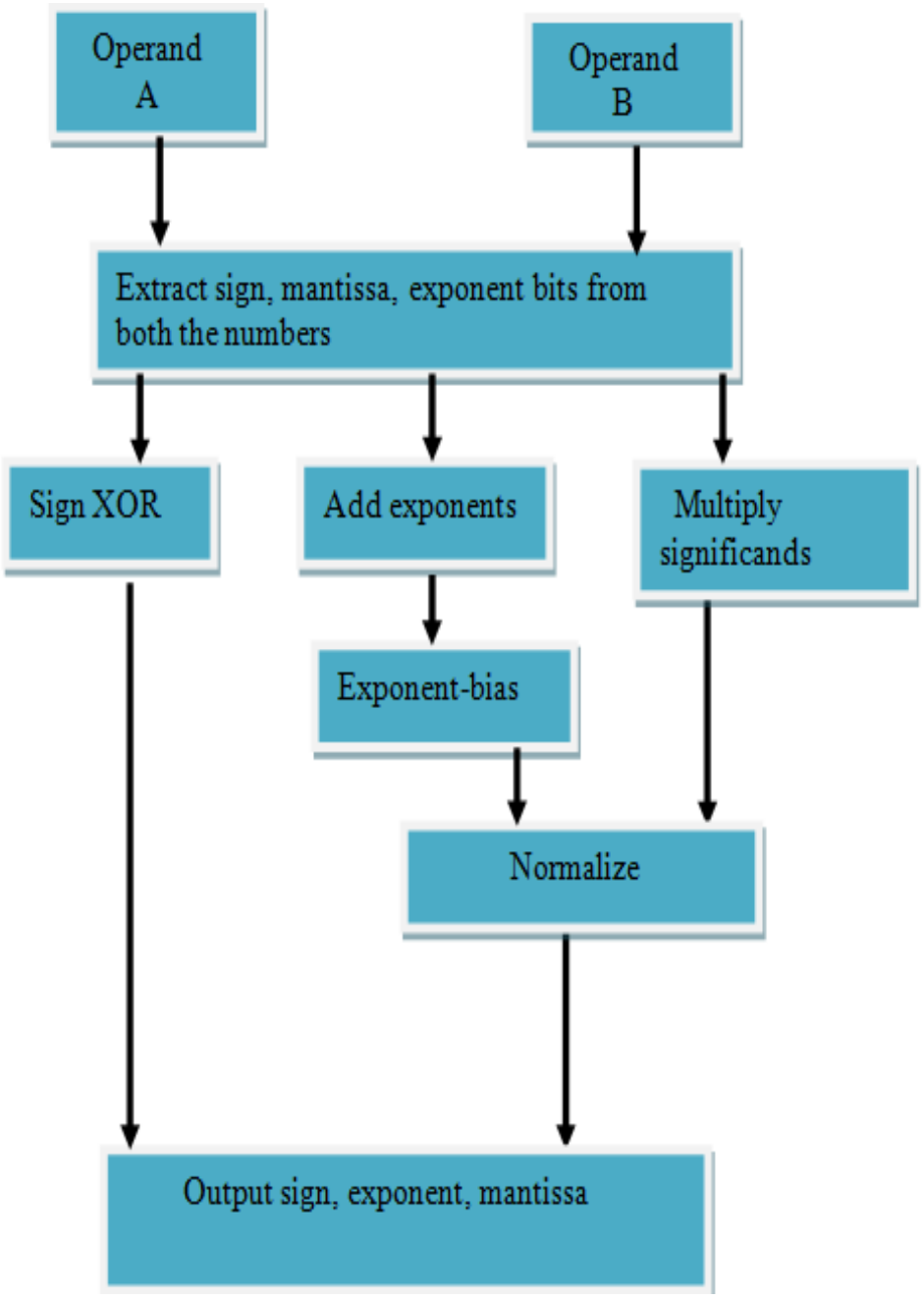


Fig.1.5 Basic Floating Point Multiplier

Calculation of Sign bits

Fig1.6 shows the calculation of sign bits. There are two operands A and B, then XOR the sign bits of both the operands. With the XOR operation, sign bit is obtained. Sign will be indicated as 1 for positive and 0 for negative values.

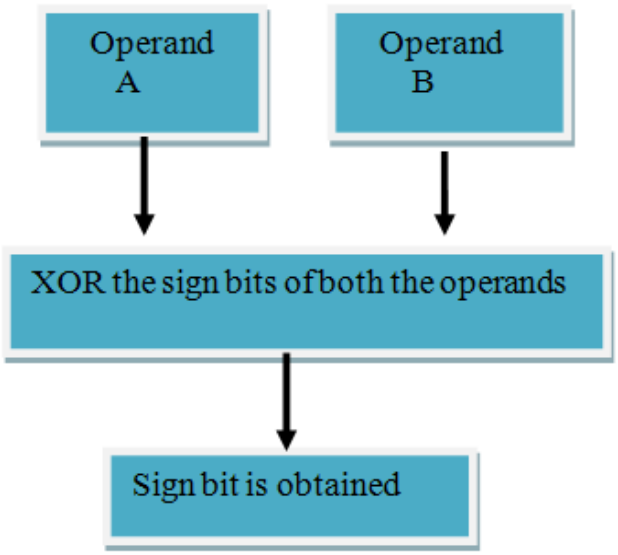


Fig1.6 Calculation of Sign Bit

.

Calculation of Mantissa Bits

Fig1.7 shows the calculation of mantissa bits. In double precision floating multiplier there are 52 bits of mantissa, the subsequent mantissa acquired by multiplication of both operands mantissa part, before that the operands must be standardized in order to guarantee that any of the numbers to be increased is not 0. But if anyone number is 0 then its result would be 0. If both numbers are zero then result would be not a number or undefined.

Normalization means adding 1 as MSB of mantissa with this the possibility of number being 0 is eliminated. After this, number of bits are increased by 1, so normalized mantissa contains 53 bits. Now 53 bits of mantissa are multiply and 106 bits result is obtained. But this result cannot be stored directly in output. MSB of a number must be 1 in normalized form therefore all bits that are 0 before 1 must be discarded. Then final 52 bits need to be extracted so the denormalization is carried out because mantissa is not in normalized form in IEEE-754 Standard such that by default the integer part of output is 1. Because of this discarded the first bit and as the mantissa output the next 52 bits are stored also discarding remaining LSB.

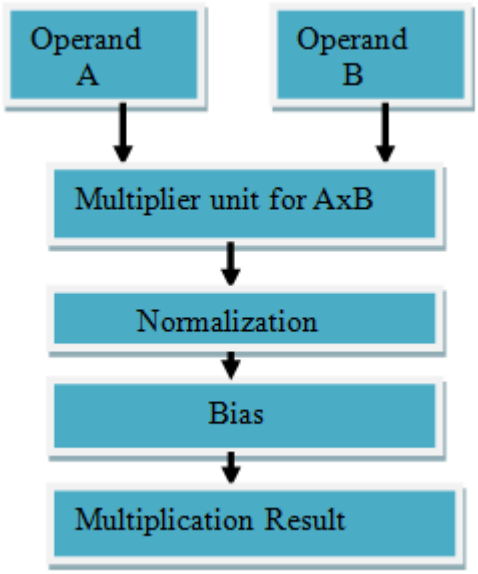


Fig.1.7 Calculation of Mantissa

Calculation of Exponent

Fig1.8 shows the calculation of exponent. The exponents are in IEEE-754 format, then to both the exponents a bias of 1023 is added. A bias of 1023 should be deducted from them so as to compute the resulting exponent. After the deduction both are added to provide the value of resulting exponent, but result is in unbiased form then again 1023 should be added to it to make it biased.

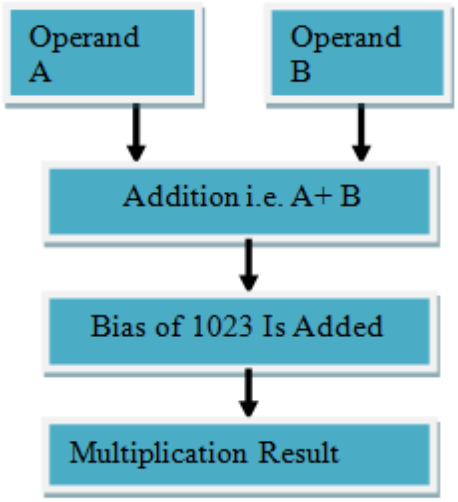


Fig.1.8 Calculation of Exponent

OBJECTIVE

The proposed work focuses on the designing of a fast MAC unit. The main objective of this work is as follows:

- Study and analysis of the performance and power of previously proposed adders.
- Study and analysis of the performance and power of previously proposed multipliers.
- Design of the high performance floating point MAC unit while not sacrificing the power and area much.

ORGANISATION OF THE THESIS

Chapter1 Introduction- This chapter gives a brief introduction to the floating point unit. Background and motivation of the thesis and the necessity of combining the floating and fixed MAC unit is explained. Overview of the proposed design along with the concepts that form the base of it are also presented. It finally concluded by presenting the organisation and flow of the thesis.

Chapter2 Literature Review- This chapter reviews the previous work done in the development of floating point MAC unit. An analysis of various techniques in adders and multiplier's has been done.

Chapter3 Proposed Floating Point MAC Unit - This chapter give the description about the proposed unit. Initially different adders are analyzed in terms of speed, power and area which motivated for the use of Koggestone adder.

Chapter4 Analysis and Implementation of proposed floating point fused multiply and add unit- This chapter presents the simulation of various adders. It also presents the result of proposed high speed multiplier and MAC unit.

Chapter5 Conclusion and Future Scope- This chapter presents a summary of work done. It also gives an introduction other techniques that can be used for further enhancement of proposed work.

CHAPTER-2
LITERATURE REVIEW

Before the designing of any unit, previous research on it needs to be studied so as to analyze the methodology and their results. With the results of previous research, proposed unit can be improved. Now in this thesis a floating point unit is designed but before that some literature review was done to compare different methodologies.

2.1 VARIOUS DESIGNS OF FLOATING POINT MAC UNIT

In 1990, IBM implemented a floating point fused multiply and add arithmetic unit based on the RISC System [1], to increase the performance of the previously proposed design. This new design incorporated a floating point multiplication fused with floating point addition into a single hardware block which was later termed as the fused multiplier adder. This paper describes the benefits of combining the adder and multiplier into a single unit. Latency for the unit was reduced by combined addition with multiplication operation in the hardware. Also, the precision of end result was increased, as the operands go through only the single stage of rounding. There was a decrease in number of ports for input and output to the register file and their sub-units. This will reduced the area of combined unit because adder is wired only to the multiplier's output connections.

Multiplier with the reduced complexity of Wallace Multiplier was developed by Kakde, Sandeep, Mithilesh Mahindra [2], so as to reduce the latency of the unit. The delay of proposed unit was 37.673ns. Alignment shifter and Normalization has also designed by the use of barrel shifter. To obtain the lower latency and higher precision, this barrel shifter was designed. The total delay for this shifter was 5.845ns.

The floating point division algorithm was presented by Pande, Kuldeep [3]. The main objective of was to enhance the precision and presentation of the application, where this process was applied. This works indicates that presented combined architecture of divide and unit has improved precision as compared to single adder unit and divider unit. This design was suitable for precision, less latency, and its price.

Multiple precision and multi functional, floating point Multiply Add Fused unit was presented by Manolopoulos, K.D Reis [4]. The proposed unit was capable for quadruple accuracy. Implementation of the proposed work was done on 65 nm silicon process that attains maximum operating frequency i.e. of 293.5 MHz at 381 mW power.

Design for double precision floating point fused multiply and add unit was presented by Bruguera, Javier D [5]. This process was based on FMA operation i.e. A + (B × C) which

allow the calculation for floating point addition with less latency than floating point multiplication. This allows skipping pipeline stage related with the multiplication of B*C.

128-Bit Fused Multiply Add Unit for a crypto processor was proposed by Atish Khobragade [6]. This unit was executed on FPGA and this design worked to decrease the latency. It was found that latency decreased up to 25%. To obtain the accuracy, alignment shifter and normalization was also been developed in this unit.

An improved design for Floating Point fused unit was proposed by Dhanabal R [7]. This design was used for condensed area and low power applications. In this, floating-point $(A \times B) + C$ was computed in single instruction. Bridge unit was also used in this design. The Bridge was considered as a link between floating point multiplier and also adds the round unit.

A framework for fused multiply and add unit was proposed by Huang, Libo, Li Shenet [8]. The proposed unit executes both the single and double precision which occupies 18% of hardware and 9% delay was increased. The proposed was compared with the other double precision units and it was found that the proposed unit was better in terms of delay and area. Various modules of double precision unit were redesigned so that it can contain the computation of two single-precision MAF operations. The proposed unit was pipelined.

Two floating-point fused multiply and add unit was proposed by Quinnell, Eric, Earl E [9] for the execution of (A times B) + C. The three path frameworks in this design utilize the parallel path which was same as in dual path floating adders. Every collection and new framework of floating-point arithmetic units was proposed using 65 nm CMOS technology.

An in-memory Booth multiplier based on racetrack memory was proposed by Tao Luo [10]. As a building block of multiplier, a racetrack memory based adder is proposed, which saves 56.3% power when compared with the state-of-the-art magnetic adder.

Radix-4 and Radix-8 algorithms are compared and also design a approximate booth multiplier was proposed by Honglan Jiang [11] for better performance. Radix-4 algorithm is considered as efficient due to the easiness of generation of partial products, on other hand radix-8 algorithm is considered as time-consuming because of the complication of generation of multiplicand's odd multiples. In this paper, the applications of approximate designs are used to alleviate this issue. Designing of an approximate two-bit adder done for calculate the sum of binary numbers. But requirements of adders used are less area, short critical path delay and low power. On the way to implement less significant section of the recoding adder, the two-bit adder is employed to generate triple multiplicand having no carry propagation.

Also two signed 16x16 bit, radix 8 approximate booth multiplier are intended with help of approximate adder that is with or without number's truncation of least significant bits that are in products.

Approximated multipliers are considered to be quick and use less power when compared to accurate multiplier, also the best performance in parameters of complexity &hardware is being achieved by the multiplier with 15-bit truncation when it was compared with other approximate designs of Booth multiplier. To low pass FIR filter design, the approximate multipliers are applied which shows better performance when comparison is done with other Booth multipliers. Finally, the approximate multipliers are applied to the design of a low-pass FIR filter and they show better performance than other approximate booth multipliers.

A fixed-width modified booth multiplier was proposed by Jiun-Ping Wang [12] proposed. For the reduction of truncation error, slightly modifications of the partial products generated by booth multiplier was done and then compensation error functions was derived so that distribution of error became more centralized and symmetric in the error that was equal to 0 that lead to very small mean-square and mean errors of this multiplier. Also, a compensation circuitry that consists of the sorting network was also designed. Experimental results on applications based on real life, demonstrate that the designed fixed-width multiplier could get better average peak signal to noise ratio having output images by at least 2.0 and1.1 db.

The design of an efficient multiplier unit was proposed by Razaidi Hussinet [13]. This architecture of multiplier is dependent on Radix 4 multiplier. In this architecture, the encoder of modified booth multiplier is modified as it is the fastest method for the generation of the partial products. When this multiplier was implemented with the Simplify Sign extension methods, then output results are incorrect. Second was to improve the delay in 4:2 compressor. Compressor is redesigned which reduces the delay of Carry. This modification was done through rearrangement of the Boolean equations of the Carry has been proposed using QuartusII. To estimate delay and size of circuit, rule of Gajski has adopted in the circuit. The total transistor count is slightly bigger in this case because of the new modified booth that uses large number of transistors. Propagation delay was reduced by 2– 7% when compared with other designs.

A fixed-width two's complement multiplier-a probabilistic estimation bias circuit was proposed by Chung-Yi Li [14]. From theoretical computation, instead of heuristic compensation strategies and exhaustive simulations which tends introduction of errors of curve-fitting and time of exponential grown simulation, probabilistic estimation bias circuit was derived. When compared with existing designs, this circuit provides a lower truncation

error and a smaller area. 8 × 8, 2-d discrete cosine transform DCT core was implemented, then the peak signal to noise ratio was improved with 17 dB with 2% area penalty when comparison was made to direct-truncated method.

A.S.Prabhuet et al. [15] speaks to the proposed plans of booth multipliers that are helpful to reduce power in the circuit. The multiplier circuit is then composed by utilizing regular full viper. At that point schematics are drawn and recreated. For the distinctive sources of info, power comes about are analyzed that demonstrates that average power devoured by the multiplier circuit. At that point booth multiplier procedure is contrasted with cluster multipliers and segment sidesteps strategy then booth multiplier expends nearly less power and consequently for low power consumption, multiplier circuit with booth recoding unit was outlined.

A novel radix-4 multiplier was proposed by Hsin-Lei Lin [16]. To improve the circuit speed and area, different schemes were designed. Modified Booth decoder was designed and summation column was compressed by this. With Synopsys and Apollo the proposed design is simulated which results in 17& -24% power reduction, 20% area reduction and 15% reduction in delay.

Hwang-Cherng Chow et al. [17] depicts the proposed designs of a new MBE recoder, also a new MBE decoder was designed in the CMOS transistor level' so that the performance analysis of traditional multipliers could be improved . With the designed pipelined multiplier, delay time of the critical path can be reduced by means of levelizing complex gate in modified booth decoder. This will results in the improvement of 66.3 % in speed. At l Ghz a low voltage, Booth multiplier having glitch free architecture of high speed pipelined was presented in the TSMC 0.35um that has only 100.52mW power consumption.

Justin Hensley et al. [18] made following commitments that was the presentation of a novel counter flow organization, where data bits flow in one way, and then Booth commands piggyback on the affirmations that was flowing in other way. Likewise the shifter and arithmetic units were consolidated in order to get upgrades in zone, vitality and speed. This outline could perform covered execution of various emphasis of the Booth algorithm. At last the outline was very isolated which could enable scaling to arbitrary operand widths, without gate resizing.

An efficient fixed-width modified Booth multiplier was designed by K. J. Cho [19]. Outputs of booth encoder were used to generate the compensation bias so as to capably compensate the quantization error with reduced complexity of hardware. Depending upon their effects on

the quantization error, further the truncated bits are divided into minor and major groups. To each group, different error compensation methods were applied. Simulation results shows that with the proposed method the major reduction in truncation error can be achieved when comparison was done with fixed width Booth multiplier.

Shyh-JyeJouet et al. [20] used a proper pay vector a low-blunder diminished width Booth multiplier. On the data there was dependency of compensation vector. Exactly when data was exacting then compensation regard will adaptively adjusts. The outcomes of the layout from a 16 x16 to 16 Booth multiplier exhibit that the critical path delay and entryway checks of the new diminished width multipliers were 66.04% and 50.94%. A module generator of this design was made that deliver Verilog code and C code for each diminished width multiplier. As a piece of CATV handsets, Pulse-shaping filter-system applications were used that exhibits the promising execution with 50.04% equipment reduction and 33.82% critical path delay reduction.

A power-efficient booth multiplier was designed by Shiann-Rong Kuanget [21] which reinforces single8-b, single16-b or twin parallel 8-b duplicated operation. A novel dynamic-range detector was created to diminish the power consumption. The identification result was not quite recently used to pick the operand with humbler dynamic range for booth. By giving up a bit of yield precision, the yield result of this multiplier can be truncated to additionally diminish in power consumption. Some extra segments that incorporates an adjustor, a modified error compensation circuit, correcting-vector generator, sign-bit generator, etc were likewise grown in order to efficiently and effectively join these methods. Likewise to assess the power productivity and error execution of the proposed multiplier, three genuine applications were embraced. The outcome demonstrates that the proposed multiplier was more unpredictable, yet it can spare power and vitality. Likewise, it keeps up a satisfactory nature of the yield required for these applications when truncation was performed.

Kyung-Ju Cho et al. [22] works on Error compensation strategy in favour of an adjusted booth fixed-width multiplier that gets a W-bit input and also produces a W-bit item as yield. The encoder output produces an error compensation inclination to proficiently add up to for the quantization error. Contingent on their impacts on the quantization error the truncated bits were additionally isolated into two gatherings that are minor and significant gatherings. At that point, to each gathering diverse error compensation strategies are connected. By reproductions, it was demonstrated that by the proposed error compensation technique, quantization error can be lessened up to half when it was contrasted and the current strategy with roughly a similar equipment overhead in the inclination era circuit. Likewise this

strategy prompts 35% decrease in region and power consumption of a multiplier when it was contrasted and the perfect multiplier.

The plan of a rough multiplier was proposed by Liangyu Qian [23], this multiplier comprises of an estimated Booth encoder, an inexact tree structure and a surmised 4-2 compressor. The outline was actualized and confirmed for 8, 16 and 32-bit marked duplication plans. Additionally at 45 nm innovation the re-enactment result were given and examined. Contrasted and Wallace-Booth multiplier and additionally other rough multiplier, the proposed inexact multiplier accomplishes upgrades in power consumption, combined metrics and delay.

CHAPTER-3

PROPOSED FLOATING POINT MAC UNIT

The Floating point unit actualizes Double Precision and Single Precision unit. It additionally executes floating point subtraction, division, addition and multiplication. It likewise actualizes 4 adjusting modes that were round towards positive infinity, round to closest, round towards negative infinity and round to zero.

ANALYSIS OF DIFFERENT ADDERS USED IN MAC UNIT

Different adders are used in the proposed design so as to make comparative analysis between them on different parameters.

- **Ripple Carry Adder**

In Fig3.1 shows a adder circuit in which the *Cout* i.e. carry output of every full adder will be the *Cin* i.e. carry input for the progressive next full adder. In this each carry bit is being rippled to next stage that is the reason it is called as ripple carry adder. In this adder carry out and entirety bits of half adder is not legitimate till the carry in for that stage will happen as a result of the propagation delay. It is the delay passed amongst info and event of the relating yield.

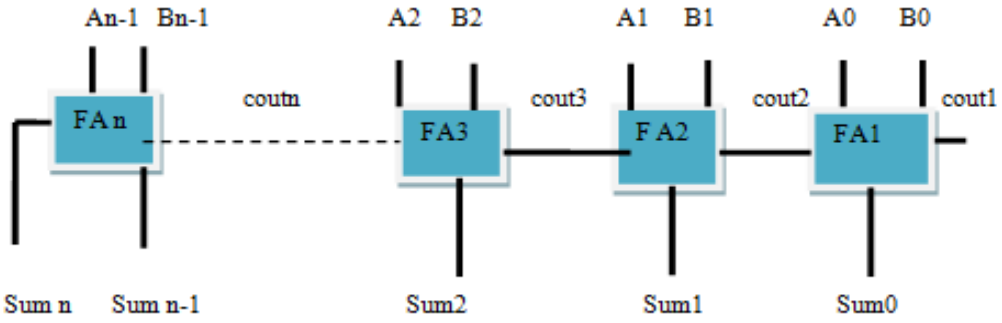


Fig.3.1 Schematic of N-bit Ripple Carry Adder

In this adder carry out and entirety bits of half adder is not legitimate till the carry in for that stage will happen as a result of the propagation delay. It is the delay passed amongst info and event of the relating yield. This critical path of ripple carry adder is being shown by Fig3.2.

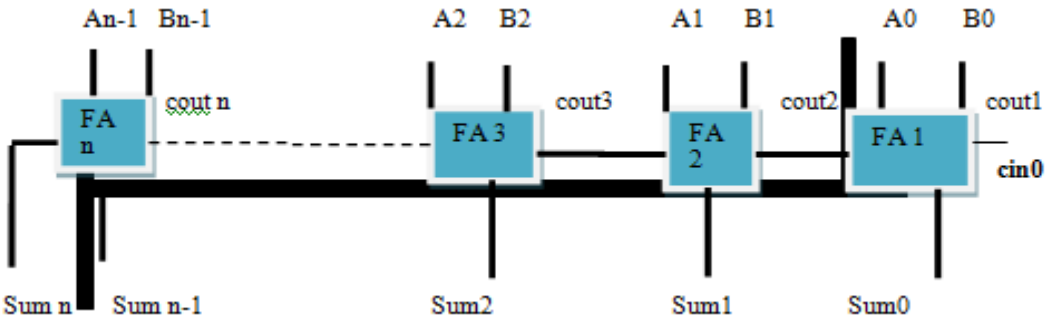


Fig3.2 Critical path for an N-bit Ripple Carry Adder

Sum bit and Carry bit of ripple carry adder will be calculated as

Sn i.e. Sum bit = An ^Bn^Cn-1

Cout .i.e. Carry bit = ((An . Bn)+ (Bn . Cn-1)+(An . Cn-1))

- **Carry Save Adder**

Fig3.3 shows the schematic of Carry Save Adder. It is an arrangement of one bit full adders, a n-bit CSA gets three n-bit operands and produces two n-bit result esteems. It is likewise used to ascertain partial products in whole number augmentation. Finally arranged partial products should be included this adder is utilized as this is quick in order to acquire the ideal execution.

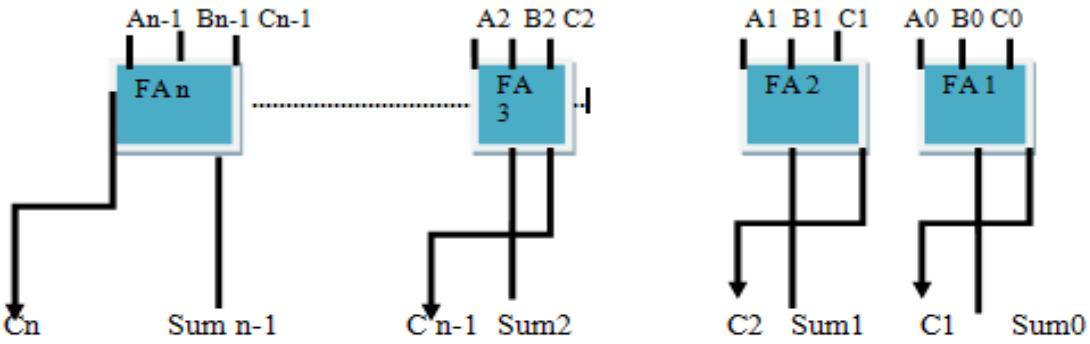


Fig3.3 Schematic of N-bit Carry Save Adder

Fig3.4 shows the idea behind a Carry Save Adder (CSA) to reduce length of the critical path by giving a shortcut to carry path, if all the bits in the block would generate a carry. A block-wide propagate signal is easy to calculate, and each block compute their own propagate signal.

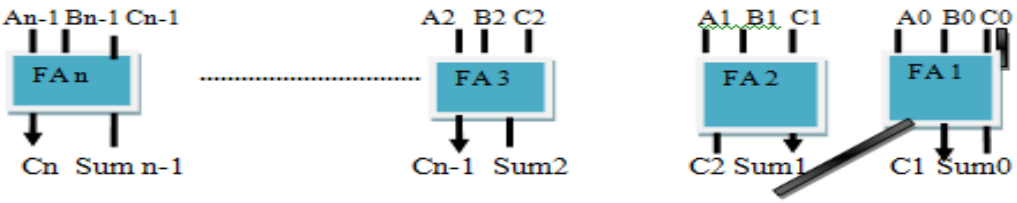


Fig3.4 Critical Path of n-bit Carry Save Adder

- **Carry Look Ahead Adder**

Fig3.5 shows the carry look ahead adder. For the calculation of propagate and generate a signal the CLA adder uses the partial full adder i.e. needed for carryout equations. By lessening the time that is fundamental to decide resultant bits will enhances the speed. It compute at least one convey bits before the total which lessen the holdup time to ascertain the consequence of greater esteem bits.

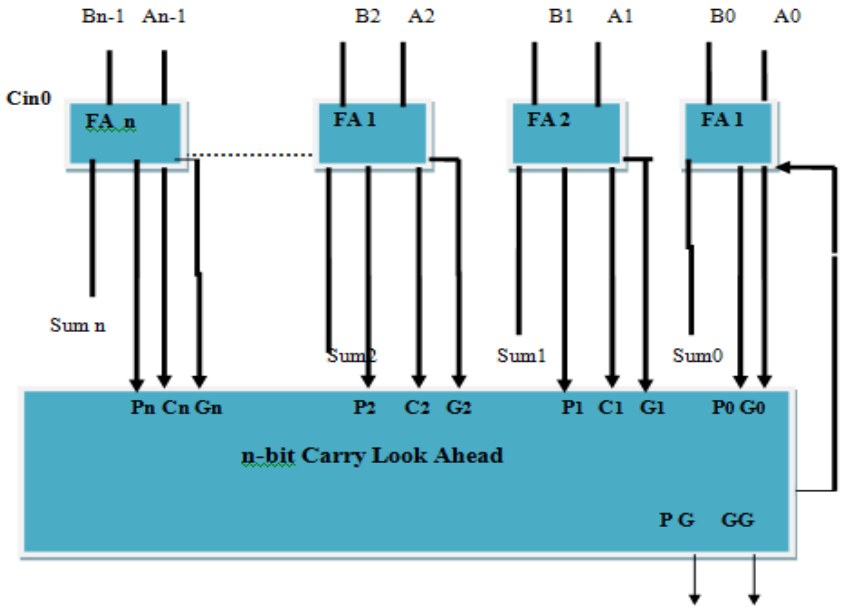


Fig.3.5 Schematic of n-bit Carry Look Ahead Carry Adder

- **Prefix Adder**

They are binary adders that exhibit various space and time characteristics. They are considered as the fastest adders. In prefix adders carries are computed which is based upon the prefix equation and those adders in which multiple terms are computed in parallel by exploits the property of associativity are known as Parallel Prefix adders.

Operations of Parallel Prefix Adders

Fig3.6 shows the stages of the prefix adders that consist of prefix, operation and parallel stages. Prefix means the result of operation is dependent on inputs. Parallel involves the parallel implementation of an operation. Implementation is done by segmentation into small pieces that are then computed in parallel. Operation**:** Any arbitrary primitive operator **"o"** that is associative is parallelizable. *Gi* and *Pi* are the carry generate and carry propagate.

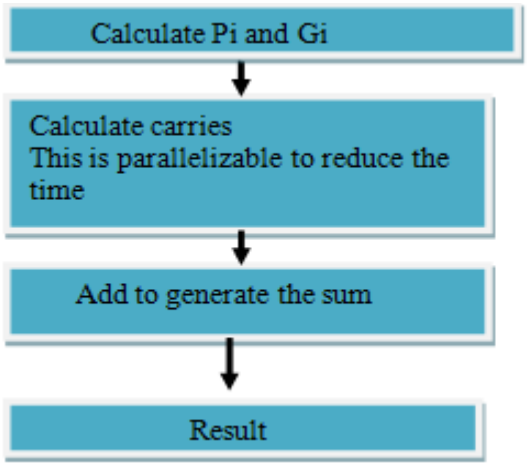


Fig.3.6 Parallel Prefix Adder Stages

Fig.3.7 shows the associative operator. In Parallel prefix adder a prefix operator is used, whose symbol is "o" that is also known as associative operator. *Gin1*, *Gin2*, *Gout* are the carry generate inputs & outputs respectively and *Pin1*,*Pin2* are carry propagate inputs.

Fig.3.7 "o" Used in Prefix Network

- **Koggestone Adder**

In this floating point unit koggestone adder is used for partial products summation in less time therefore it is considered as the one of the fastest adder, carry generation is much faster in this because of parallel computation.

Stages of Koggestone Adder

There are three stages involved in this process that are as follows:
Consider *Xi* and *Yi* as inputs of the adder, *Pi* and *Gi* are the carry propagate and carry generate respectively and *Ci* is the carry input.

Pre-processing stage: In this stage carry generate and carry propagate signals i.e. *Gi* and *Pi* respectively are being generated.

$$Pi = Xi \text{ XOR } Yi$$

$$Gi = Xi \text{ AND } Yi$$

*Carry Look Ahead Network***:** It contains grey and black cells that are used to generate carry for next stages. In this *Gi* and *Pi* are taken as and carry generate and carry propagate. *Giprev*, *Piprev* are inputs and it computes *(G,P)* as output signal.

$$G = Gi \text{ OR } (Pi \text{ AND } Giprev)$$
$$P = Pi \text{ AND } Piprev$$
$$Ci = Gi$$

Post Processing: This is the final stage which is used to compute the carry out and sum bit.

$$Si = Pi \text{ XOR } Ciprev$$

ANALYSIS OF MULTIPLIERS USED IN MAC UNIT

Booth multiplier and double precision floating point multiplier is used in the MAC unit for the partial product generations. Analysis of that multiplier is shown in 3.2.1 and 3.2.2.

Booth Multiplier

Booth algorithm is the algorithm in which two signed binary numbers are being multiplied in two's complement notation. In general multiplication is the important unit in high performance systems, but designing the multiplier with less power consumption and high speed are major challenges.

Fig3.8 shows the booth multiplier .This multiplier scans 3-bits at any given moment such that the quantity of partial products will be lesser and the 3-bits are taken in the way, such that two bits are taken from present pair and third bit from the higher order bit of adjacent lower order pair, after this, they are changed into the arrangement of 5 control signals by booth logic that are utilized by the adder cells. This technique allows smaller and faster multiplication circuit, by recording that numbers which are been multiplied.

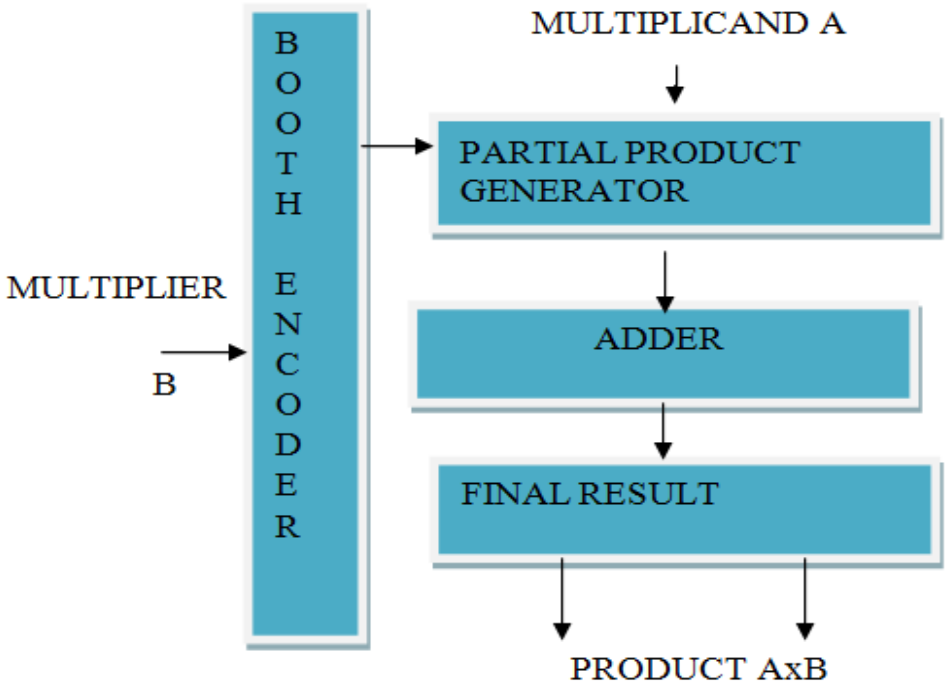


Fig.3.8 Booth Multiplier

Booth Algorithm

Consider A and B as operands multiplier and let X and Y represents bits in A and B. Booth algorithm is the simplest method to reduce the number of partial products as recoding in booth is done with that only half of partial product are generated. In this algorithm two bits of multiplier are examined as shown in Table3.1 and then it determines the generation of partial products. For both signed and unsigned numbers this algorithm is works.

Table3.1 Radix 2 Booth Multiplier Encoding

X	X-1	ACTION
0	0	No Action
0	1	Addition
1	0	Subtraction
1	1	No Action

Algorithm for Unsigned Numbers

Fig3.9 shows the booth algorithm for unsigned numbers.

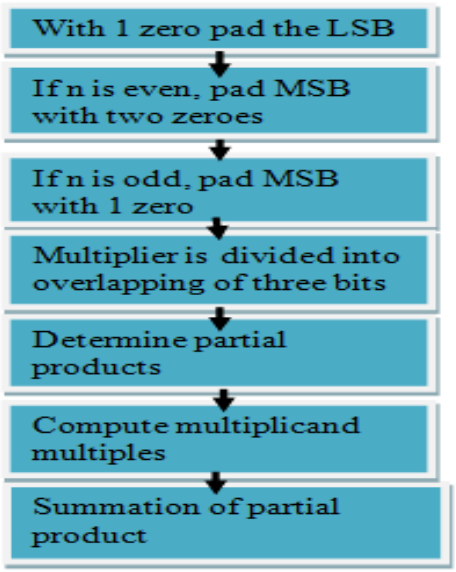


Fig.3.9 Booth algorithm for unsigned number

Algorithm for Signed Numbers

Fig3.10 shows the booth algorithm for signed numbers.

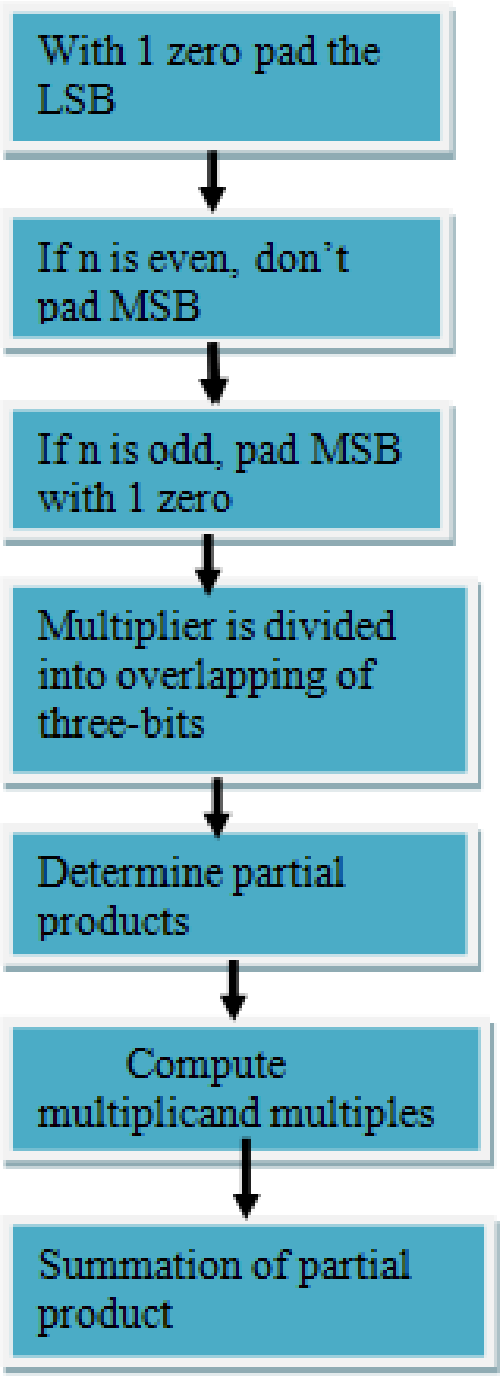


Fig.3.10 Booth algorithm for unsigned numbers

Double Precision Multiplier

Following Fig3.11 shows the hierarchy of double precision floating point unit as in this floating point operations are being performed to the inputs given and then rounding is performed according to the result of that operations , also exception handling is there to check the special cases so that appropriate output will achieve.

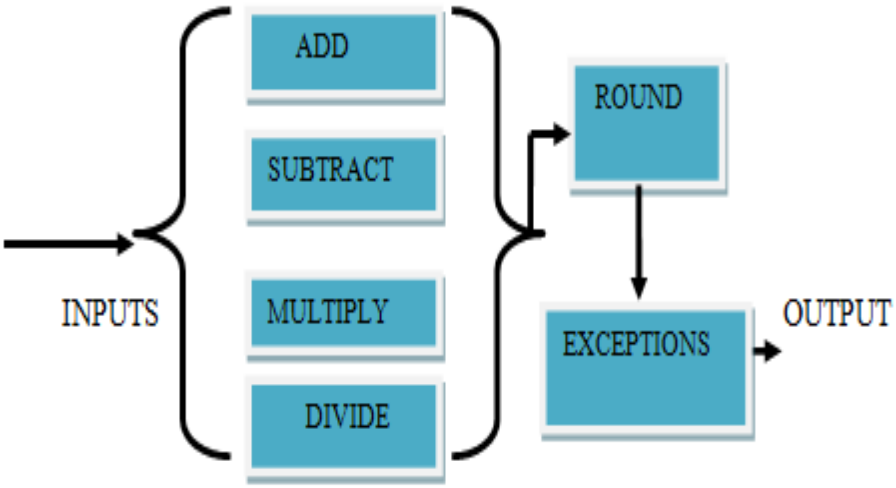


Fig.3.11 Hierarchy of Double Precision Floating Point Multiplier

- **FPU Operations**

  Different floating point operations such as addition, multiplication, subtraction and division are described.

Addition

The two sources of info are partitioned into their exponent components and mantissa components, and then the larger operand goes to exponent_large and mantissa_large while littler operand at that point goes to exponent_small and mantissa_small.

Assessment can be made to discover which is operand is large by making examination between their exponents however in the event that exponent components are equivalent then littler operand goes to exponent_large and mantissa_large.

Subtraction

The two information sources are isolated into their exponent parts and mantissa segments, the large operand goes to exponent_large and mantissa_large though the smaller operand go to exponent_small and mantissa_small . Presently we need to compute the distinction between their exponents and then before doing subtraction shift the mantissas of smaller exponent to right.

Multiplication

For normalized numbers the leading 1 is the mantissa of one operand and are stored in 53 bit register and leading 1 and the mantissa of other operand and is stored in another 53 bit register, but 53x53 bit multipliers are not presented in Xilinx so it must discussed into smaller bits and then result must be added to get 106- bit product as a result. This product is store in product register and then output is left shift if there will be no 1 in the MSB and number of leading 0s in the product register is detected by some signals and then output of the exponent component will be reduce by that signal. The exponents components of both the operands are add together and then from the sum of the operands, the value of 1022 is subtracted. At that point if consequence of example is under zero then the product reg is should be moved ideal by the sum however that esteem is not being spared in the register and henceforth the last type part of the yield that will be appeared as 0 and denormalized number will be an outcome.

By and by if its regard is more than 52 than the mantissa will shift out of the item enlist and its yield be shown 0 and an undercurrent signal will be state. The mantissa yield is in 56-bit enrol and the MSB will be driving zero to consider an overflow signal in modifying module and the principle piece that is 0 is being trailed by 0 for denormalized numbers and 1 for institutionalized numbers and 52 bits of the mantissa takes after however 2 extra bits that are takes after mantissa used for altering module. The additional piece that starts things out is convey by the following piece when mantissa is in consequence of 106-piece product and the additional piece that comes next is the or operation of the 52 LSB of after effect of 106-piece product.

Division

The mantissa and leading 1 of first operand is considered as the dividend and mantissa of second operand leading 1 is considered as the divisor. The division is executed with the 1-bit of quotient that calculate each block cycle which is depend upon the comparison between the divisor and dividend register. The quotient will be 0 and the divisor bit if dividend is greater than divisor.

Rounding

The inputs to module are from previous stage of addition, subtraction, multiplication, division are sign bits i.e. one-bit, mantissa i.e. 56-bits and the exponent i.e. 12 bits.

The mantissa incorporates 0 as extra piece as a two extra remainder bits as LSB and MSB and in mid there are 52 mantissa bit and driving 1 and the example has extra 0-bit as MSB with the end goal that flood from the most extreme type that is 2047 would be recognized but if there are only eleven bits in the register then it could result as 0 value in the exponent and the final result would be incorrect.

Exception Module

In the exception section special cases were check and suitable output is created accordingly. The signals at the output like overflow, underflow, invalid, inexact and exception will be asserted where they are needed in following special cases.

Table.3.2 Special Cases In Exception Module

Special Cases	Results
Divide by 0	Result will be infinity and the sign that is positive or negative is depend upon the sign of the operand
Divide 0/0	In this case the result is SNAN therefore invalid signal is asserted in this case
Divide ∞/∞	The result will be SNAN in this case and therefore the invalid signal is declared.
Divide by ∞	In this case result will be zero and sign that is positive or negative ,this will depend upon the sign of operand and Underflow signal is asserted.
Multiply zero by ∞	In this case result will be SNAN therefore invalid signal is asserted in this case

Addition, Subtraction , Multiplication, Division Underflow	In this case result will be 0 and Underflow signal is asserted.
Addition, Subtraction , Multiplication, Division Overflow	In this case result will be ∞ and Overflow signal is asserted.
Addition of positive infinity with negative infinity	In this case the result will be SNAN therefore invalid signal is asserted in this case
Subtraction of positive infinity from positive infinity	In this case the result will be SNAN therefore invalid signal is asserted in this case
Subtraction of negative infinity from negative infinity	In this case the result will be SNAN therefore invalid signal is asserted in this case.
One or both the inputs be QNAN	In this case output will be QNAN
One or both the inputs are SNAN	Output will be QNAN in this case

DESIGN OF HIGH PERFORMANCE FLOATING POINT MAC UNIT

Fig3.12 shows the design of high performance floating point multiplier. This unit contains sign, exponent and mantissa bits that are the inputs to the booth multiplier. Then decimal and binary compressors are used to reduce partial products.

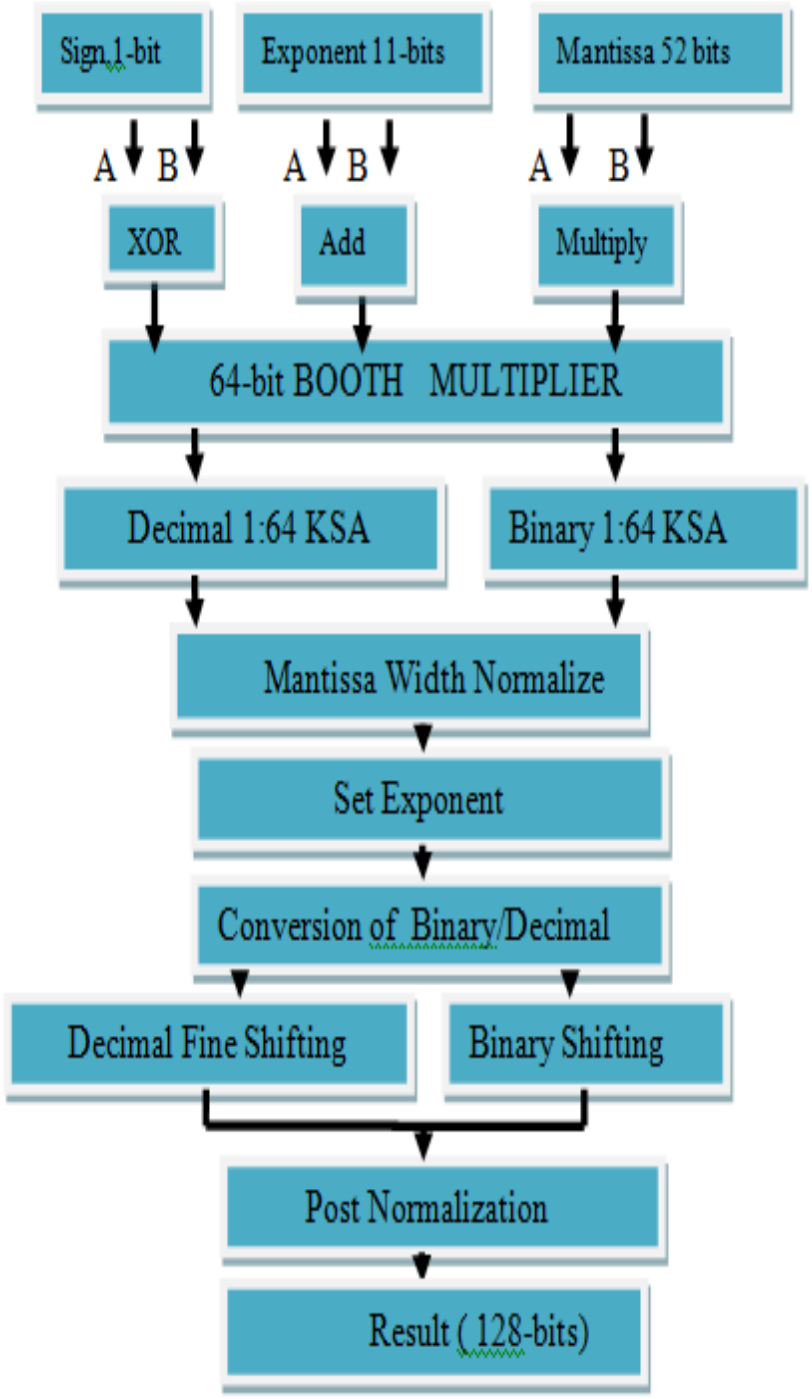


Fig.3.12 Proposed Floating Point Multiplier

In Fig.3.13 shows Operation of floating point MAC unit that consists of adder unit, multiplier unit and an accumulator unit .Inputs that are given to MAC are fetched from the memory location and are fed to its multiplier block that perform multiplication and also gives the result back to the adder that will accumulate the final result and that final result was store in a memory location. This Complete process is achieved in single cycle.

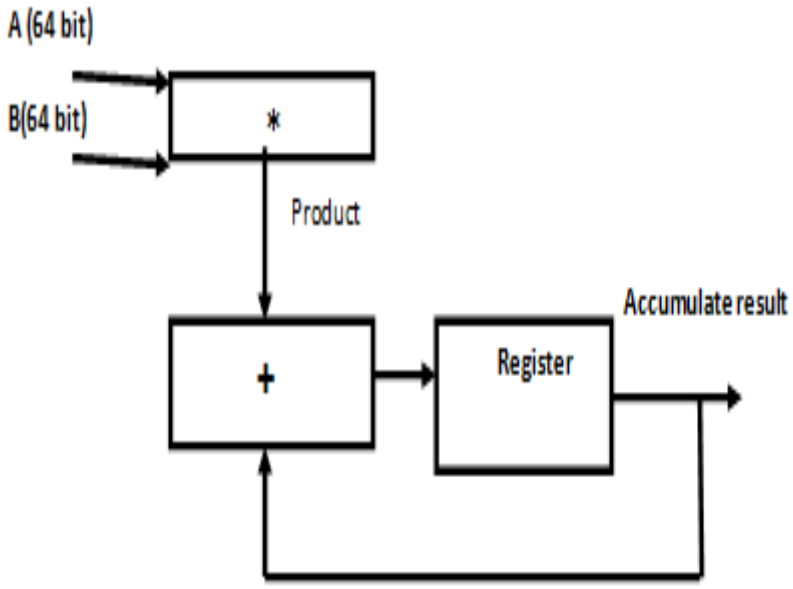


Fig.3.13 Operation Of Floating Point MAC Unit

CHAPTER-4

ANALYSIS AND IMPLEMENTATION OF PROPOSED FLOATING POINT MULTIPLY AND ADD UNIT

This chapter introduces the execution of double precision floating point by using different adders and then comparative analysis of these adders was done. The implementation was done by using Xilinx ISE.

ANALYSIS OF DIFFERENT ADDERS

Power and delay analysis of different adders are described below:

Carry Save Adder

Fig.4.1 shows the power results of 64- bit CSA adder. In this booth multiplier is used to generate the partial products and CSA adder is used for the summation of that partial products. But power consumed by this adder is 0.082W.

Device	
Family	Artix7
Part	xc7a100t
Package	csg324
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	75	63400	0
Signals	0.000	164	---	---
IOs	0.000	132	210	63
Leakage	0.082			
Total	0.082			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.6	25.4

Supply Summary		Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Source	Voltage			
Vccint	1.000	0.017	0.000	0.017
Vccaux	1.800	0.013	0.000	0.013
Vcco18	1.800	0.004	0.000	0.004
Vccbram	1.000	0.000	0.000	0.000
Vccadc	1.710	0.020	0.000	0.020

Supply Power (W)	Total	Dynamic	Quiescent
	0.082	0.000	0.082

Fig4.1 Simulation Results of Power of CSA adder

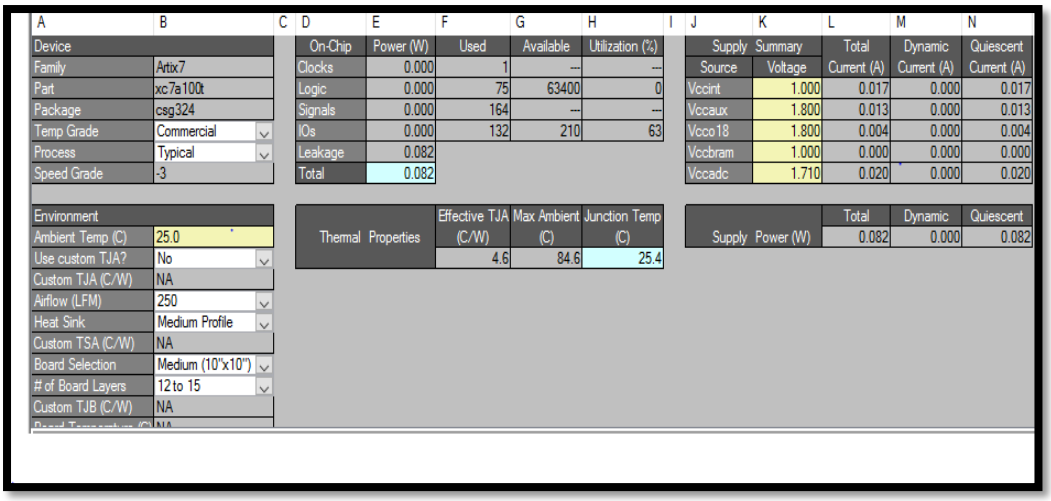
Fig4.2 shows the delay analysis of Carry save adder. Delay of CSA adder is 32.680 ns.

Source Clock	Src:Rise | Dest:Rise	Src:Fall | Dest:Rise	Src:Rise | Dest:Fall	Src:Fall | Dest:Fall
clk	32.680			

Fig4.2 Delay analysis of CSA

Koggestone Adder

Fig4.3 shows the power results of 64-bit KSA adder. In this booth multiplier is used to generate the partial products and KSA adder is used for the summation of that partial products. But power consumed by this adder is 0.082W.



Device	
Family	Artix7
Part	xc7a100t
Package	csg324
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	75	63400	0
Signals	0.000	164	--	--
IOs	0.000	132	210	63
Leakage	0.082			
Total	0.082			

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.017	0.000	0.017
Vccaux	1.800	0.013	0.000	0.013
Vcco18	1.800	0.004	0.000	0.004
Vccbram	1.000	0.000	0.000	0.000
Vccadc	1.710	0.020	0.000	0.020

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.6	25.4

Supply Power (W)	Total	Dynamic	Quiescent
	0.082	0.000	0.082

Fig4.3 Simulation Results of Power Of KSA

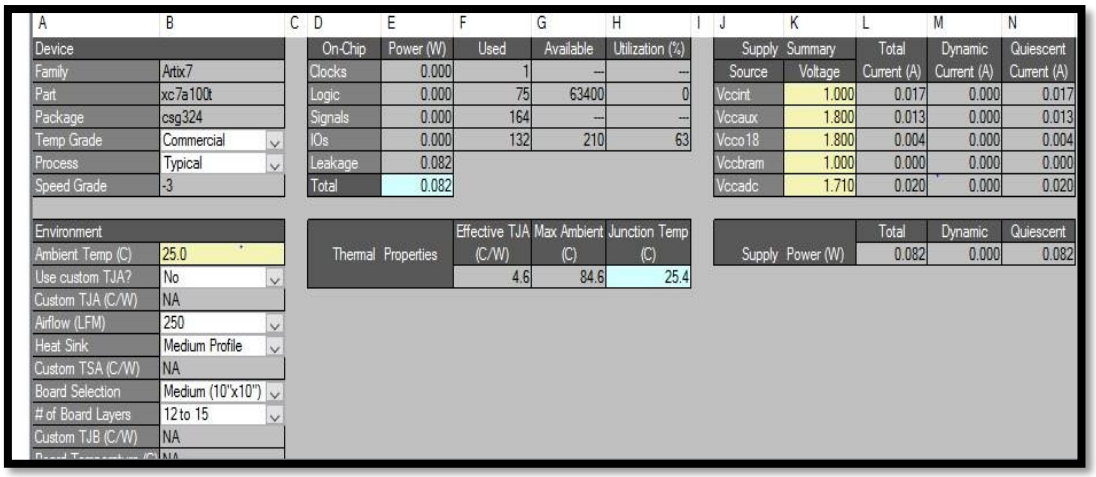
Fig4.4 delay analysis of KSA Fig.4.4 shows the delay analysis of koggestone adder. Delay of KSA is 1.104 ns.

Source Clock	Src:Rise|Dest:Rise	Src:Fall|Dest:Rise	Src:Rise|Dest:Fall	Src:Fall|Dest:Fall
clk	1.104			

Fig4.4 Delay analysis of KSA

Ripple Carry Adder

Fig4.5 shows the Power results of 64-bit RCA i.e. ripple carry adder. In this booth multiplier is used to generate the partial products and RCA adder is used for the summation of that partial products. But power consumed by this adder is 0.082W.



Device	
Family	Artix7
Part	xc7a100t
Package	csg324
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	75	63400	0
Signals	0.000	164	--	--
IOs	0.000	132	210	63
Leakage	0.082			
Total	0.082			

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.017	0.000	0.017
Vccaux	1.800	0.013	0.000	0.013
Vcco18	1.800	0.004	0.000	0.004
Vccbram	1.000	0.000	0.000	0.000
Vccadc	1.710	0.020	0.000	0.020

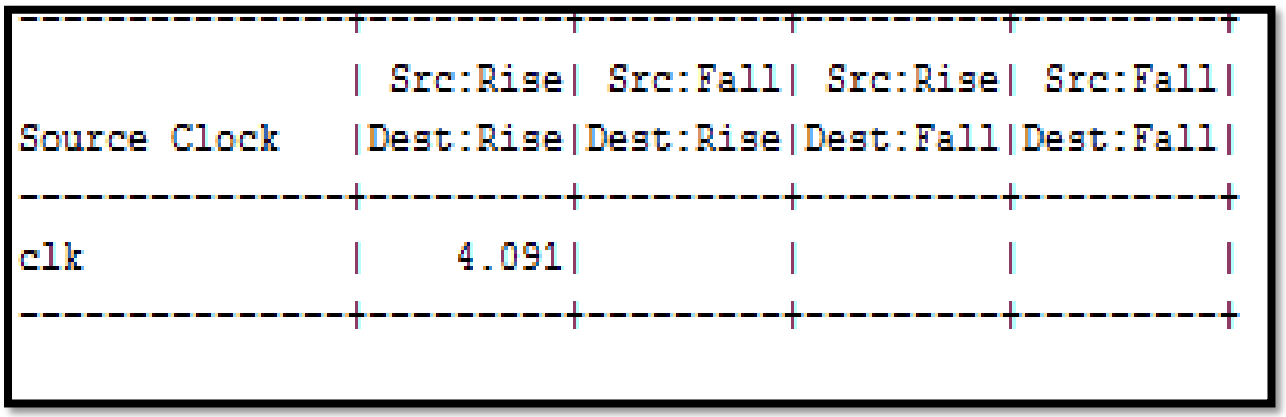
Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.6	25.4

Supply Power (W)	Total	Dynamic	Quiescent
	0.082	0.000	0.082

Fig.4.5 Simulation Results of Power of RCA

Fig4.6 shows the delay analysis of Ripple carry adder. Delay of RCA is 4.091ns.

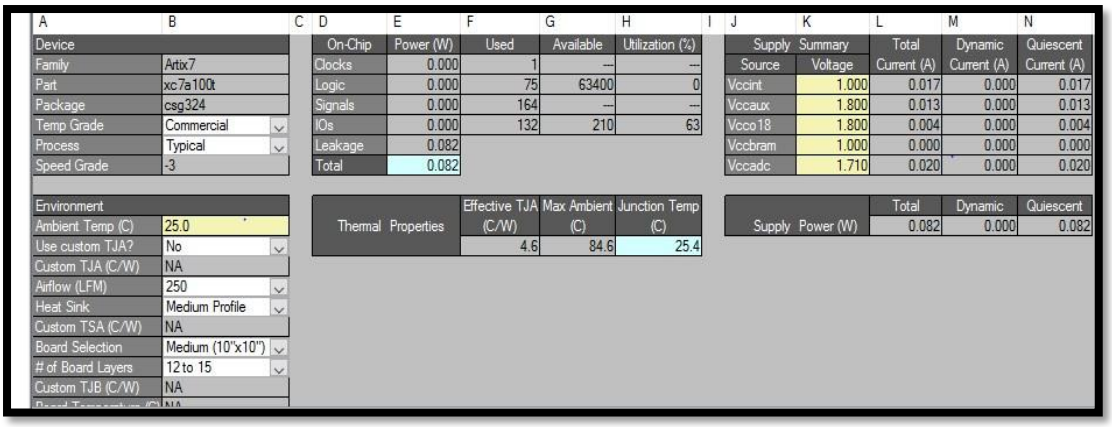


Source Clock	Src:Rise|Dest:Rise	Src:Fall|Dest:Rise	Src:Rise|Dest:Fall	Src:Fall|Dest:Fall
clk	4.091			

Fig.4.6 Delay Analysis of RCA

Carry Look Ahead Adder

Fig4.7 shows the Power results of 64- bit CLA i.e. carry look ahead adder. In this booth multiplier is used to generate the partial products and CLA adder is used for the summation of that partial products. But power consumed by this adder is 0.082W.



Device	
Family	Artix7
Part	xc7a100t
Package	csg324
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	75	63400	0
Signals	0.000	164	--	--
IOs	0.000	132	210	63
Leakage	0.082			
Total	0.082			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.6	25.4

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.017	0.000	0.017
Vccaux	1.800	0.013	0.000	0.013
Vcco18	1.800	0.004	0.000	0.004
Vccbram	1.000	0.000	0.000	0.000
Vccadc	1.710	0.020	0.000	0.020

Supply Power (W)	Total	Dynamic	Quiescent
	0.082	0.000	0.082

Fig.4.7 Simulation Results of Power of CLA

Fig.4.8 shows the delay analysis of Carry look ahead adder. Delay of CLA is 1.493 ns.

Source Clock	Src:Rise|Dest:Rise	Src:Fall|Dest:Rise	Src:Rise|Dest:Fall	Src:Fall|Dest:Fall
clk	1.493			

Fig.4.8 Delay Analysis of CLA

The proposed design had been simulated using the XILINX ISE. In the given Table4.1, the delay and power of different used adders like ripple carry, koggestone, carry save and carry look ahead are presented. Here 64-bit booth multiplier with these adders is being implemented and results are calculated as result of power is calculated by using X POWER analyzer in Xilinx and delay is calculated by the timing report of the design that is generated after the simulation of the design. This implies that the critical path delay is reduced that also

means the complexity of the path is reduced by using booth koggestone stone adder when it is being compared with the others.

Table 4.1 Comparative Analysis of Different Adders

ADDERS(64- BIT)	POWER(W)	DELAY (ns)
CSA	0.082	32.68
CLA	0.282	1.493
RCA	0.082	4.977
KSA	0.082	1.104

Fig4.9 shows the power consumption by different adders. The power of adders is almost same. Also power consumption is not reducing because of Double precision floating point modules. These modules are new leading zeros detector which produce the output in base-3 to simplify normalization shifting in binary data path and redundant adder is used to perform addition at the final stage. Using this, a rounding redundant delay and remove it from the critical path, and using a new simple conversion technique from redundant to binary/decimal.

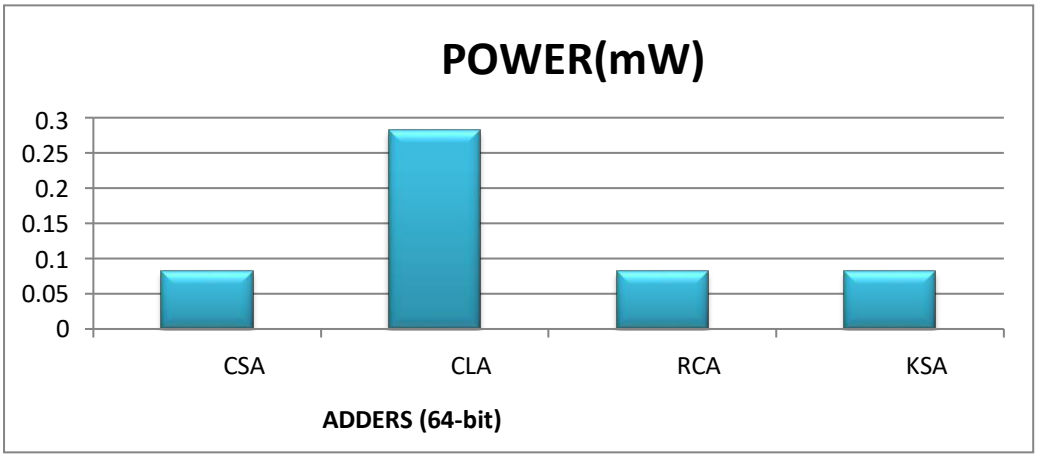


Figure 4.9 Graph of Power Used by different Adders

Fig4.10 shows the delay of different adders. In this thesis main objective to reduce multiplication time by comparing different adders on delay parameter and to find out which adder reduces the partial product and give multiplication value in less computation time.

Different adders of such as koggestone adder (KSA), carry save adder (CSA), carry look head adder (CLA) and ripple carry adder of 64-bit were used. But KSA adder has lesser computation time when compared to other adders. In this floating point unit koggestone adder is used for partial products summation in less time therefore it is considered as the one of the fastest adder, carry generation is much faster in this because of parallel computation.

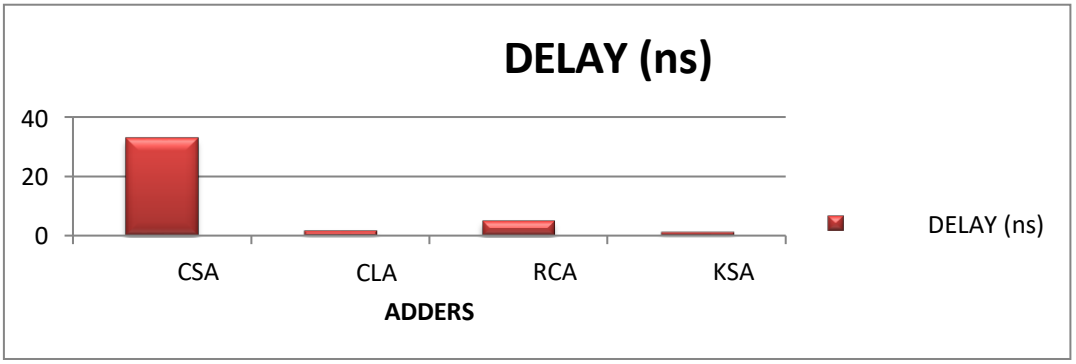


Fig 4.10 Graph of Delay of different Adders

Fig.4.11 shows the comparison between delay and power of different adders used in the floating point unit. As per the analysis, KSA adder has lesser computation time. But power consumption is not reducing because of Double precision floating point modules. These modules are new leading zeros detector which produce the output in base-3 to simplify normalization shifting in binary data path and redundant adder is used to perform addition at the final stage.

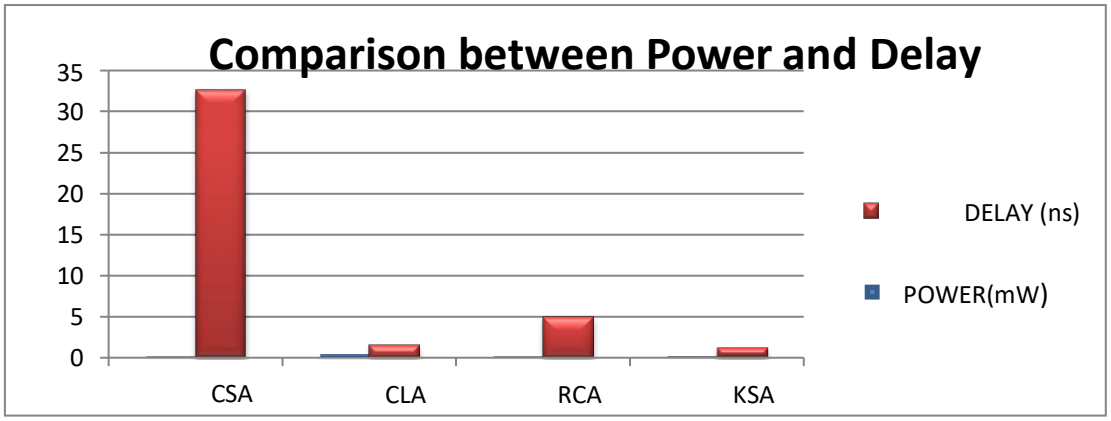


Fig.4.11 Comparison Graph of Power and Delay in different Adders

SIMULATION RESULTS OF FLOATING POINT MAC UNIT

Fig.4.9 shows the simulation result of Floating Point MAC Unit. The implementation is verified by performing a simulation with some random input vectors. To evaluate the design, parameters such as delay and power consumption are verified. The double precision floating point adder and multiplier designs were synthesized using Xilinx ISE.

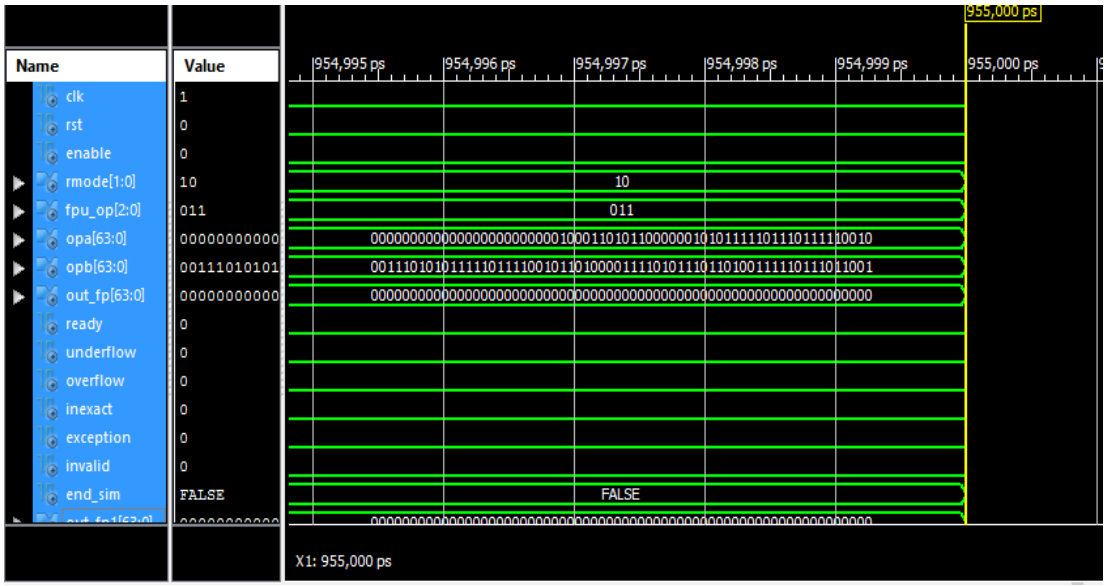


Fig4.12 Simulation results of Floating Point MAC unit

Delay and Power Analysis Of Floating Point MAC Unit

Table 4.2 presents the power and delay results as obtained for Fixed and floating point operations performed using the proposed design. Delay analysis of Fixed and floating point units are 1.43ns and 2.88 ns respectively. Power analysis of combined unit is 36.76mW.

Table 4.2 shows simulation results of floating point unit

Parameters	Combined Fixed and floating Unit	
Delay (ns)	Fixed unit =1.43	Floating unit =2.88
Power (mW)	36.76	

CHAPTER-5

CONCLUSION AND FUTURE SCOPE

CONCLUSION

In High performance systems multiplication of two binary numbers are mostly used arithmetic operations In this work, a 64-bit floating point Fused Multiply and Add (MAC) unit is presented that can work on both integer and floating pointy numbers. The work presented in this thesis tries to reduce multiplication time by usage of the Koggestone Adder at the partial product row addition level. The thesis report also presented a comprehensive analysis of the different adders on the basis of different parameters like area, power and delay. The adders that have been investigated include: koggestone adder (KSA) with carry save adder (CSA), carry look head adder (CLA) and ripple carry adder (RCA). The analysis reflected that the usage of KSA adder has very less computation time which motivated for its use in the MAC unit proposed herein. Result of power had been calculated using X POWER analyzer in XILINX and delay is calculated by the timing report of the design that is generated after the simulation of the design

FUTURE SCOPE

The work presented in this thesis can be further enhanced by the addition of other operations like division, square root calculation etc. which are normally required in various signal processing systems. Pipelining can be further utilized to improve the throughput of the system. Though as per the analysis done, Booth's algorithm was found to be the best preposition among various multipliers but, combination of Wallace with different parallel prefix adders such as Ladner fischer adder, Han Carlson adder, Brent-Kung adder etc. could also be explored to enhance the performance.

REFERENCES

[1] R.K. Montoye, E. Hokenek and S.L. Runyon(1990). Design of the IBM RISC System/6000 floating-point execution unit , *IBM Journal of Research &Development*.

[2] Kakde, Sandeep, Mithilesh Mahindra, Atish Khobragade, and Nikit Shah (2015.FPGA Implementation of 128-Bit Fused Multiply Add Unit for Crypto Processors, In Security in *Computing and Communications*, pp. 78-85.

[3] Pande, Kuldeep, AbhinavParkhi, ShashantJaykar, and AtishPeshattiwar et al.(2015). Design and Implementation of Floating Point Divide-Add Fused Architecture,In *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, pp. 797-800.

[4] Manolopoulos, K., D. Reisis, and V. A. Chouliaras (2016). An efficient multiple precision floating-point Multiply-Add Fused unit, *Microelectronics Journal* 49 ,10-18.

[5] Bruguera, Javier D., and Tomás Lang (2005). Floating-point fused multiply-add: reduced latency for floating-point addition. In *Computer Arithmetic, 2005. ARITH-17 2005. 17th IEEE Symposium on*, pp. 42-51.

[6] AtishKhobragade and Nikit Shah (2015). FPGA Implementation of 128-Bit Fused Multiply Add Unit for Crypto Processors, In *Security in Computing and Communications*, pp. 78-85.

[7] Dhanabal, R., Sarat Kumar Sahoo, and V. Bharathi (2016). Implementation of Low Power and Area Efficient Floating-Point Fused Multiply-Add Unit, In *Proceedings of the International Conference on Soft Computing Systems*, pp. 329-342.

[8] Huang, Libo, Li Shen, Kui Dai, and Zhiying Wang (2007). A new architecture for multiple-precision floating-point multiply-add fused unit design, In 18th IEEE Symposium on Computer Arithmetic (ARITH'07), pp. 69-76.

[9] Quinnell, Eric, Earl E. Swartzlander, and Carl Lemonds et al.(2007). Floating-point fused multiply-add architectures, *Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, pp. 331-337.

[10] Luo, Tao, et al.(2016),A racetrack memory based in-memory booth multiplier for cryptography application,*Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific* .

[11] Jiang, Honglan, et al.(2016).Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation, *IEEE Transactions on Computers* 65.8, 2638-2644.

[12] Wang, Jiun-Ping, Shiann-RongKuang, and Shish-Chang Lian(2011).High-accuracy fixed-width modified Booth multipliers for lossy applications, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19.1,52-60.

[13] Hussin,Razaidi, et al. (2008).An efficient modified booth multiplier architecture,*Electronic Design, 2008.ICED 2008.International Conference*.

[14] Li, Chung-Yi, et al.(2011).A probabilistic estimation bias circuit for fixed-width Booth multiplier and its DCT applications, *IEEE Transactions on Circuits and Systems II: Express Briefs* 58.4,215-219.

[15] Prabhu, A.S, and V. Elakya et al.(2012).Design of modified low power booth multiplier,*Computing, Communication and Applications (ICCCA), 2012 International Conference*.

[16] Lin, Hsin-Lei, Robert C. Chang, and Ming-Tsai Chan et al(2004). Design of a novel radix-4 booth multiplier,*Circuits and Systems, 2004.Proceedings.The 2004 IEEE Asia-Pacific Conference on*,Vol. 2.

[17] Chow, Hwang-Cherng, and I-Chyn Wey(2002) .A 3.3 V 1 GHz high speed pipelined Booth multiplier,*Circuits and Systems, 2002.ISCAS 2002.IEEE International Symposium on*, Vol. 1.

[18] Hensley, Justin, AnselmoLastra, and Montek Singh et al. (2004). An area-and energy-efficient asynchronous Booth multiplier for mobile devices, *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004.Proceedings.IEEE International Conference on*.

[19] Cho, K. J., et al. (2002).Low error fixed-width modified Booth multiplier,*Signal Processing Systems, 2002.(SIPS'02),IEEE Workshop on* .

[20] Jou, S-J, Meng-Hung Tsai, and Ya-Lan Tsao(2013). Low-error reduced-width Booth multipliers for DSP applications, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 50.11 , 1470-1474.

[21] Kuang, Shiann-Rong, and Jiun-Ping Wang(2010). Design of power-efficient configurable booth multiplier, *IEEE Transactions on Circuits and Systems I: Regular Papers* 57.3, 568-580.

[22] Cho, Kyung-Ju, et al.(2004).Design of low-error fixed-width modified booth multiplier, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12.5, 522-531.

[23] Qian, Liangyu, et al.(2016). Design and evaluation of an approximate Wallace-Booth multiplier,*Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*.

[24] Davis, Timothy Don. (1999).Booth multiplier with squaring operation accelerator.*U.S. Patent No*. 5,957,999.

[25] Yuan-Ho Chen,'An(2015). Accuracy Adjustment Fixed-Width Booth Multiplier Based on Multilevel Conditional Probability, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , vol. 23.

[26] Swapna M S and Vimala(2015). Delay Comparison of Various Multipliers With Adaptive Hold Logic(AHL), *International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)*,Vol. 2.

[27] Suresh R.Rijal, Sharda G.Mungale(2013).Design And Implementation Of 8*8 Truncated Multiplier On FPGA, *International Journal Of Scientific And Research Publications,* Vol 3.

[28] Jin U. Kang and Kang Zhang(2012). Speed Real-Time Fourier Domain Optical Coherence Tomography, *IEEE journal of selected topics in quantum electronics*, vol. 18.

[29] Andrea C. Bickerstaff, Earl E. Swartzlander, Michael J. Schulte(2001).Analysis of Column Compression Multipliers,15th *IEEE Symposium on Computer Arithmetic Proceedings.*

[30] Himanshu Thapliyal, Neela Gopi, K.K. Praveen Kumar, M.B. Srinivas et al.(2006). Low Power Hierarchical Multiplier and Carry Look-Ahead Architecture, *IEEE International Conference on Computer Systems and Applications*.

[31] Kiat Seng Yeo, Kaushik Roy(2005).Low-Voltage, Low Power VLSI Subsystems, Mc Graw Hill.

[32] Sumit Vaidya and Deepak Dandekar(2010). Delay-power Performance Comparison of Multipliersin VLSI circuit design, *International Journal of Computer Networks and Communications(IJCNC).*

[33] Suresh R.Rijal, Sharda G.Mungale(2013).Design And Implementation Of 8*8 Truncated Multiplier On FPGA, *International Journal Of Scientific And Research Publications.*

[34] David R.Lutz et al.(2011). Fused Multiply-Add Microarchitecture Comprising Separate Early-Normalizing Multiply and Add Pipeline, *20th IEEE Symposium on Computer Arithmetic.*

[35] Patterson, D. & Hennessy, J(2005).*Computer Organization and Design*: The Hardware/software Interface, Morgan Kaufmann.

601562009_DeeptiSharma_thesis_report.docx

by 241449 User

Submission dat e: 10- Aug- 2017 07:57AM (UT C- 0400)
Submission ID: 836280154
File name: 601562009_DeeptiSharma_thesis_report.docx (396.7K)
Word count : 6795
Charact er count : 35725

601562009_DeeptiSharma_thesis_report.docx

ORIGINALITY REPORT

9%	1%	8%	3%
SIMILARIT Y INDEX	INT ERNET SOURCES	PUBLICAT IONS	ST UDENT PAPERS

PRIMARY SOURCES

1	2%

Sachin Raghav, Rinkesh Mittal. "Implementation of Fast and Efficient Mac Unit on FPGA", International Journal of Mathematical Sciences and Computing, 2016

Publicat ion

2	1%

Wahba, Ahmed, and Hossam Fahmy. "Area Efficient and Fast Combined Binary/Decimal Floating Point Fused Multiply Add Unit", IEEE Transactions on Computers, 2016.

Publicat ion

3

1%

Submitted to Punjab Technical University

St udent Paper

4

<1%

www.eurojournals.com

Int ernet Source

5

<1%

Jiang, Honglan, Jie Han, Fei Qiao, and Fabrizio Lombardi. "Approximate Radix-8 Booth Multipliers for Low-Power and High-performance Operation", IEEE Transactions on Computers, 2015.

Publicat ion

6

<1%

Shiann-Rong Kuang, , and Jiun-Ping Wang. "Design of Power-Efficient Configurable Booth Multiplier", IEEE Transactions on Circuits and

Systems I Regular Papers, 2010.

Publicat ion

7

<1%

Advances in Intelligent Systems and Computing, 2016.

Publicat ion

8

<1%

www.amsv.umac.mo

Int ernet Source

9

<1%