A Floating-point MAC Architecture for IEEE-754 half-precision format

DISSERTATION-III

Submitted in partial fulfillment of the requirement for the

award of the degree of

MASTER'S OF TECHNOLOGY

In

VLSI Design

By

K. JAYA RAMESH

(Registration Number: 11906079)

Under the Guidance of

Dr. Rajkumar Sarma

(Assistant Professor)

School of Electrical and Electronics Engineering

Lovely Professional University, Jalandhar-Delhi G.T Road (NH-1)

Phagwara, Punjab (India)-144411

DECLARATION

I hereby declare that the Report of the Dissertation-III work entitled "A Floating-point MAC Architecture for IEEE-754 for half Precision format" which is being submitted to the LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, in partial fulfillment of the requirements for the award of the Degree of Master of Technology in VLSI, in School of Electronics and Electrical Engineering under the guidance of Dr. Rajkumar Sarma , during January to May 2021 is a bonafide report of the work carried out by me. The material contained in this Report has not been submitted to any University or Institution for the award of any degree. All the information furnished in this Thesis report is based on my own intensive work.

K. Jaya Ramesh (Reg. No. 11906079)

(Signature of Student)

School of Electronics and Electrical Engineering

CERTIFICATE

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. He has completed the M. tech Dissertation- III under my guidance and supervision. The present work is the result of his original investigation, effort and study. No part of the work has ever been submitted for any other degree at any university. The Dissertation- III report is fit for the submission and partial fulfillment of the conditions for the award of M. tech degree in VLSI from Lovely Professional University.

Dr. Rajkumar Sarma

Asst. Professor

School of Electronics and Electrical Engineering,

Lovely Professional University,

Phagwara, Punjab.

Date: 20-12-2020

ACKNOWLEDGEMENT

I have bestowed my time and efforts for this work. However, it would not have been possible without the kind support and help of teachers. I would like to thank my guide Mr. Rajkumar Sarma for his undeterred support he offered me throughout my work. I express my gratitude for many insightful discussions and his friendly advices. I feel grateful for working under him. My thanks and appreciation also goes to my HOD “Dr. Cherry Bhargava” who gave me the opportunity for experiencing such knowledge and people who have willingly helped me out with their abilities. I express my thanks to all my M. tech classmates for their extended support in my course works. I would like to extend special thanks to Lovely Professional University for providing an excellent and ideal environment for learning. Finally, I would like to thank my parents and family members for their love and continuous support during my academic studies.

K. Jaya Ramesh (Reg. No. 11906079)

(Signature of Student)

ABSTRACT

The MAC (Multiply and Accumulate Unit) is the basic building block in the Digital signal processing and digital image processing systems. For efficient systems, the MAC unit should be fast with high precision and consuming low power. A MAC unit can be designed in Fixed-point arithmetic and Floating-point arithmetic. The use of Floating-point arithmetic gives high precision but it consumes more power and occupies more silicon area. To achieve high performance in a MAC unit, standard arithmetic can be implanted in its design. The IEEE-754 is floating-point arithmetic which can be used in the design of MAC. The use of the IEEE-754 standard improves the precision of the MAC unit. The conventional MAC units are designed by using HDL languages like the Verilog and the VHDL. By using these languages, the MAC unit can be designed in less time but designing the MAC unit at transistor level will increase the performance of the overall MAC unit. The cadence virtuoso can be used for designing the MAC at the transistor level. The design can be completed by using different technologies like 90nm gpdk or the tsmc 130nm etc. Finally, designing a floating-point MAC unit with half-precision using IEEE-754 in the transistor level will lead to better performance and high precision.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In the Digital signal processing system, MAC (Multiplier and Accumulator) is the basic building block. The MAC consists of a multiplier and an accumulator, the multiplier multiplies the input samples and sends them to the accumulator which will add the present input and the past input and produce a new result that will be stored in a register [1][2][3].

1.1 Conventional MAC

Digital signal processing applications need high speed and low power MAC unit because the main operations in DSP like filtering and convolution are repetitively used [4]. So, A low power and highly efficient MAC is always recommended in Digital signal processing applications [5]. A conventional MAC unit is shown in figure 1.1.

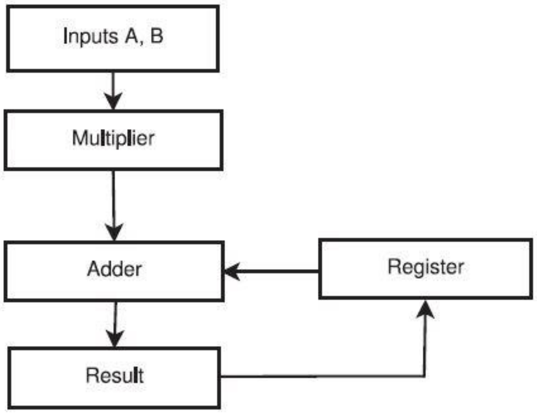


Fig 1.1: Block diagram of Conventional MAC

MAC is mainly used in applications like filtering and convolution. In many DSP applications, a MAC with high precision is needed for better accuracy [6].

The overall performance of the DSP system can be enhanced by MAC with a floating-point architecture [7]. The use of a floating-point number of IEEE 754 format in a MAC results in a wider range of values at the cost of more storage and accuracy. More storage is needed in the case of floating-point because the position of the radix point needs to be encoded [8][9]. In the case of CPUs without the floating-point architecture, a series of simple fixed-point architecture is used [10].

1.2 Types of MAC

A MAC can be implemented in fixed-point arithmetic and floating-point arithmetic [11][12].

1.2.1 Fixed-point arithmetic MAC:

In a fixed-point arithmetic MAC, the multiplier and accumulator used are of the fixed point. The use of fixed-point arithmetic results in less accuracy of output [13]. The block diagram of the fixed-point MAC is shown in figure 1.2.

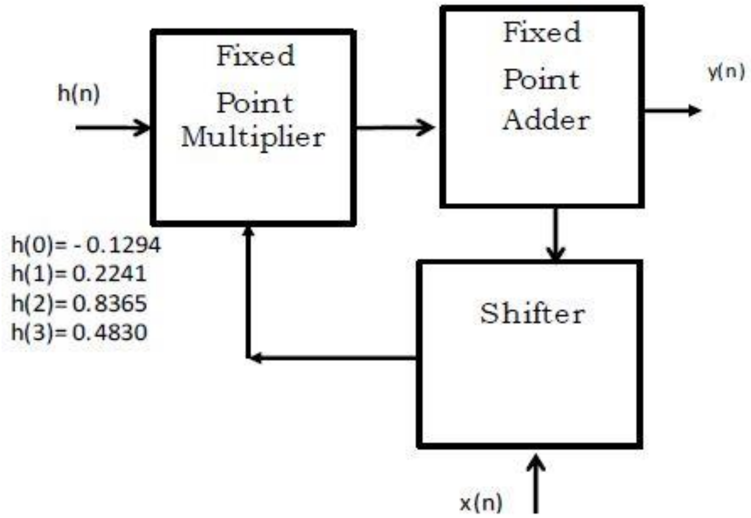


Fig 1.2: Block diagram of Fixed-point MAC.

In this, the input samples are given to the fixed-point multiplier. The filter coefficients are specified by the user and the multiplier is used to multiply filter coefficients and the input samples and the result is driven to the fixed-point adder. The fixed-point adder will add the present input to the previous result and the output is obtained as y(n) [10][14].

Limitations of fixed-point MAC:

- ADC Quantization error.
- Coefficient quantization error.
- Overflow errors.
- Round-off errors and truncation errors.

1.2.2 Floating-point arithmetic MAC:

In a Floating-point arithmetic MAC, the multiplier and accumulator used are of the Floating point. The use of Floating-point arithmetic results in more accuracy of output. The block diagram of floating-point MAC is shown in figure 1.3.

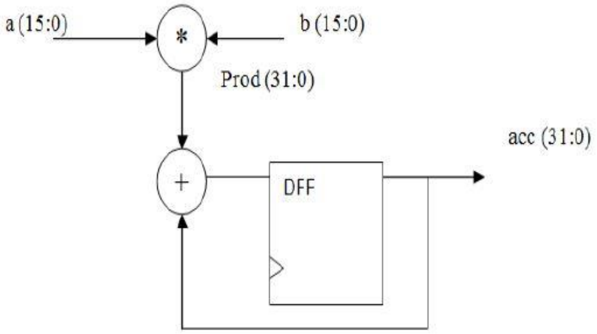


Fig 1.3: Block diagram of Floating-point MAC.

In Floating-point MAC, the input samples are given to the multiplier. The multiplier will multiply the input samples and the result is passed to the floating-point adder. The adder will add the present input and the past result and produce a new output y(n) which is stored in a register [15]. The Floating-point MAC consumes more silicon area when compared to Fixed-point MAC but it will result in higher precision [16].

1.3 Floating-point Adder:

The operation in a Floating-point adder is carried out in four steps.

1.Sorting.

2.Alignment.

3.Addition or Subtraction.

4.Normalization.

1.Sorting: In this step, the numbers are sorted in the decrement order i.e., from the largest number to the smallest number.

2.Alignment: In this step, the alignment of numbers is done to have the same exponent. This process is carried out by adjusting the exponent of a small number until the exponents of both numbers are matched.

3.Addition or Subtraction: In this step, the significands of the aligned numbers are added or subtracted.

4.Normalization: In this step, the result obtained will be normalized [17].

The block diagram of the Floating-point addition is given in figure 1.4.

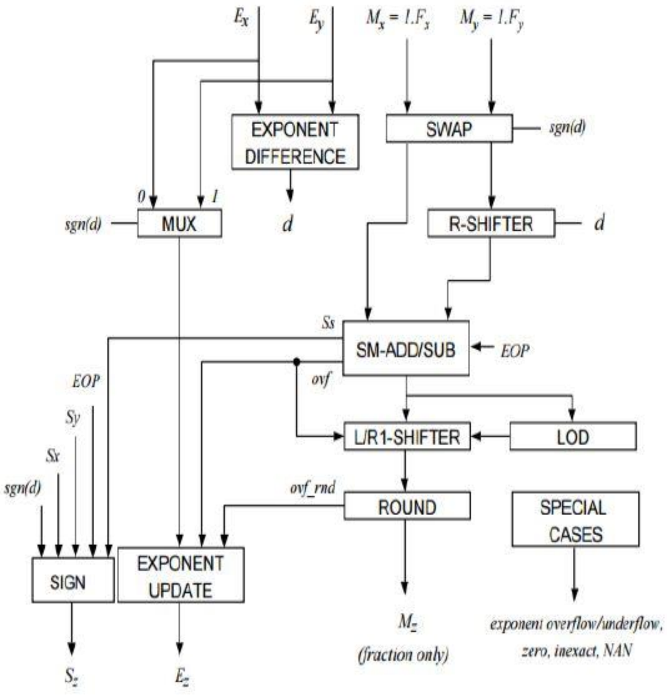


Fig 1.4: Block diagram of floating-point addition.

1.4 Floating-point Multiplier

Floating-point multipliers play an important role in Digital signal processing and digital image processing systems [18]. The block diagram of the Floating-point multiplier is given in figure 1.5.

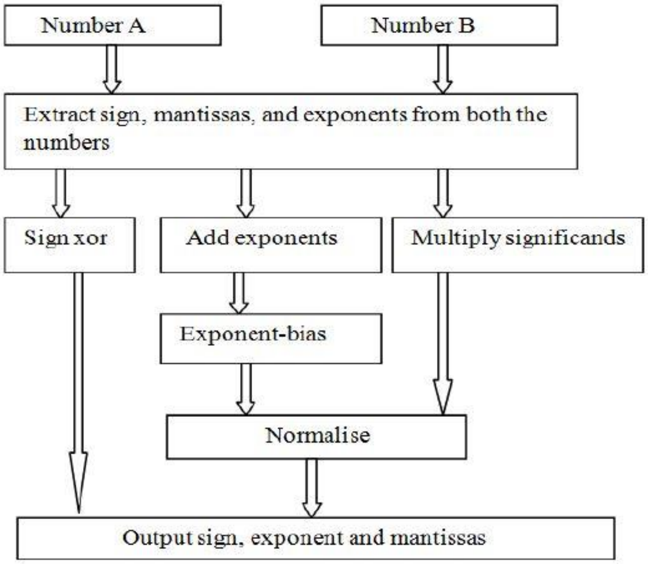

Number A
Number B
Extract sign, mantissas, and exponents from both the numbers
Sign xor
Add exponents
Multiply significands
Exponent-bias
Normalise
Output sign, exponent and mantissas

Fig 1.5: Floating-point multiplier.

Let us assume the two floating-point numbers are m1 and m1 and the result obtained after multiplication is m, then

m = m1*m2

= $(-1)^{s1}.m1.2^{e1} * (-1)^{s2}.m2.2^{e2}$

= $(-1)^{s1+s2}.p1.p2.2^{e1+e2}$

1.5 Standard IEEE 754 format

In 1985, the IEEE has released the standard binary Floating-point format. It includes different types of floating-point formats which include single precision and double precision, round mechanisms, arithmetic operations, etc [19]. The floating-point number can be represented by the given equation.

$$Z= (-1s) *2 \ (exp-bias)*(1*M)$$

According to the IEEE 754 format, a 32-bit floating-point number consists of 8 bits which represent the exponential part, 23 bits represent the significand and the sign is represented by one bit [20].

Sign	Exponent	Mantissa

32 31 23 0

Fig 1.6: IEEE 754 single-precision Floating-point format.

The standards which are defined in the IEEE 754 format are shown in table 1.1

Table 1.1: Different standards in IEEE-754 format

Name	Common name	Base	Digits	Emin	Emax
Binary32	Single Precision	2	23+1	-126	+127
Binary64	Double Precision	2	52+1	-1022	+1023
Binary128	Quadruple Precision	2	112+1	-16382	+16383
Decimal64		10	16	-383	+384
Decimal128		10	34	-6143	+6144

The first three formats are used for Binary floating-point numbers and use 32,64 and 128 bits respectively. The last two formats are used for Decimal Binary floating-point numbers and use 64 and 128 bits respectively [8].

CHAPTER 2

Review of Literature

Dr. R. Prakash Rao et al., 2018: A Floating-point MAC using IEEE 754 floating-point adder has been proposed to improve the performance than the fixed floating-point adder, a Floating-point MAC using IEEE 754 floating-point adder has been proposed which can be used in the Digital signal processors. A floating-point MAC of 16 bit was designed. The block diagram of the proposed floating-point MAC is given in figure 2.1.

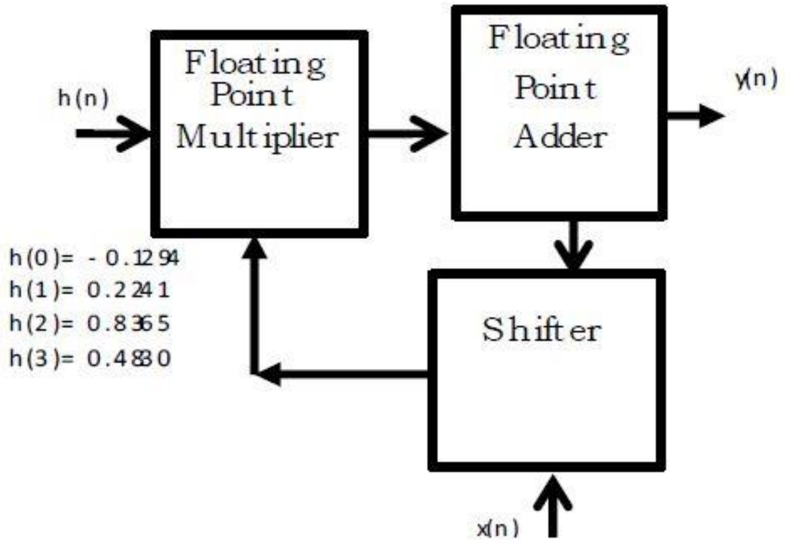


Fig 2.1: Standard Floating-Point MAC Block diagram.

The Floating-point MAC includes a 16-bit Floating-point Adder and a 16-bit Floating-point multiplier. The Floating-point adder used in this design is given in figure 2.2.

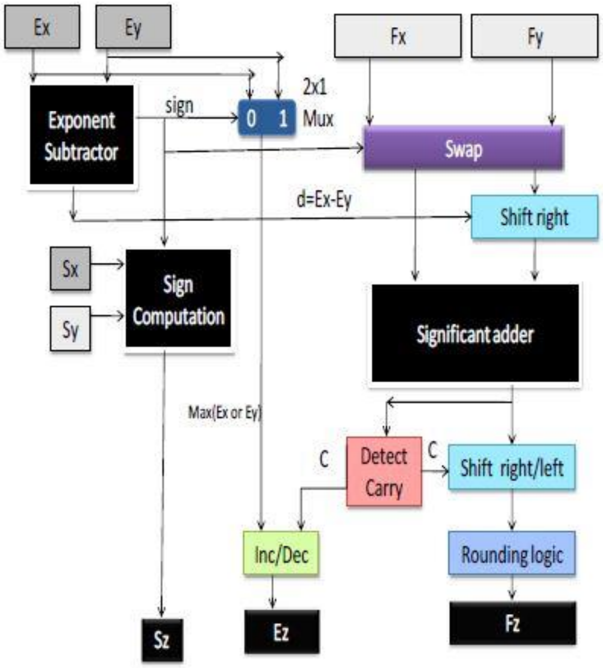


Fig 2.2: 16-bit Floating-point adder.

The 16-bit floating-point multiplier used in this design is given in figure 2.3.

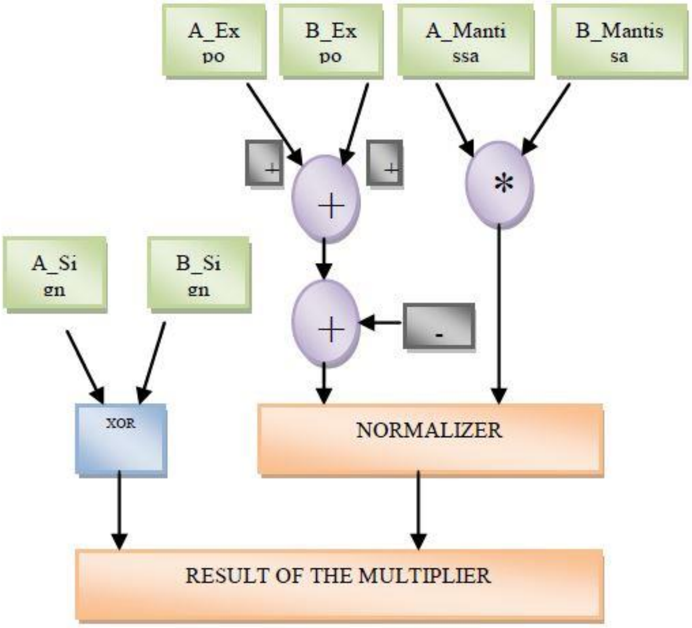


Fig 2.3: 16-bit floating-point multiplier.

The simulation of the work is performed in the Modelsim 10.3c tool and the hardware device selected for the synthesis is Xc2s50e-ft256-6. The results of the designed floating-point MAX achieved a maximum frequency of 793.65MHz with a minimum period of 1.286ns and power consumed was observed as 12.493mw. The designed MAC with IEEE 754 standard achieved a higher precision than the fixed-point MAC [15].

Yadagiri Karri et al., 2015: A MAC unit was designed in the IEEE 754 standard of Single precision. The block diagram of the 16-bit MAC unit is given in figure 2.4.

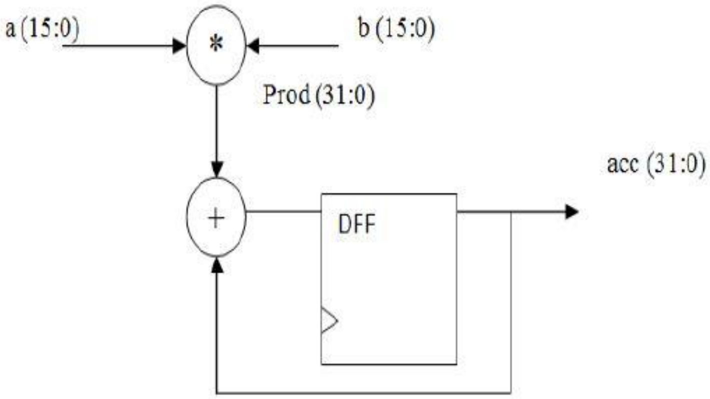


Fig 2.4: Block diagram for floating-point MAC.

The artificial neural networks require a wide range for the representation of data which can be achieved by a Floating-point MAC using the IEEE 754 standard. The implemented design is used to feed weighted inputs in artificial neural networks [20].

Mohamed Asan Basiri M et al., 2014: A floating-point adder for the accumulation operation in MAC can be avoided by directing the mantissa of the past MAC result to the partial product input of the multiplier unit in the MAC unit. 48.54% worst path delay improvement was observed in radix-2 Wallace multiplier-based MAC unit. Several other MAC units with different proposed structures were analyzed by Mohamed Asan Basiri M and Noor Mahammad SK. The proposed MAC unit is given in the below figure 2.5 [22].

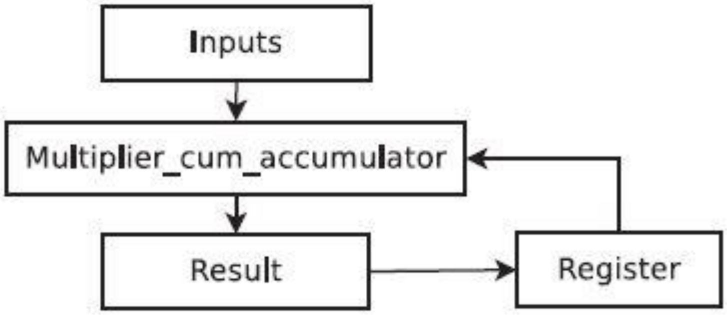


Fig 2.5: MAC with Multiplier cum accumulator unit.

Dhananjaya A et al., 2013: An ancient Indian technique Urdhvatiryegbhyam sutra is used in decimal multiplications. This technique was implemented in the MAC unit and tested in different FPGA boards using the Xilinx ISE tool and achieved high speed. The Vedic multiplier used in the design is shown in figure 2.6 [23].

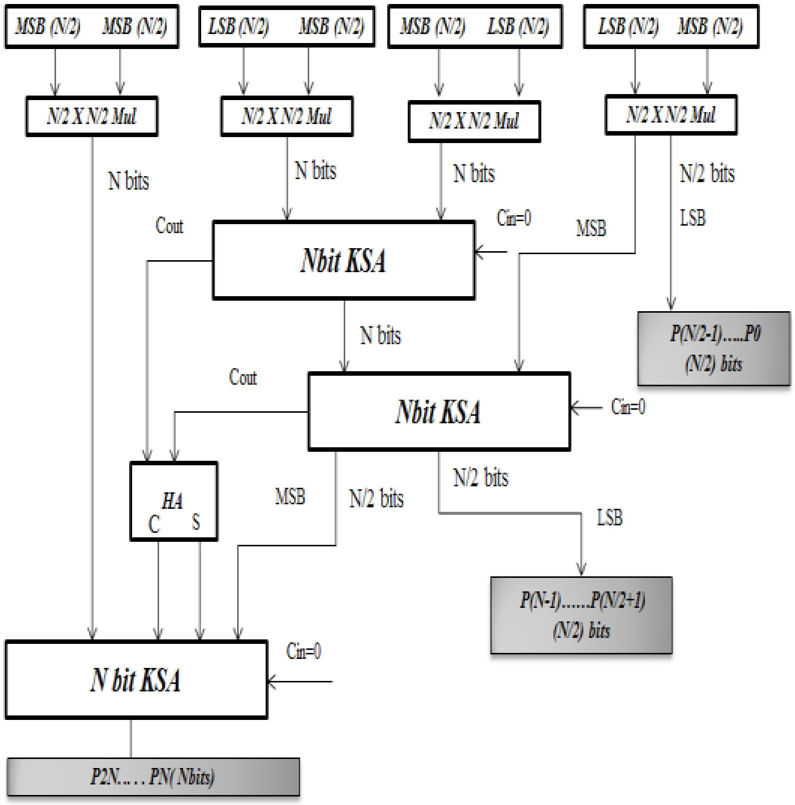


Fig 2.6: N*N Vedic Multiplier.

Hao Zang et al., 2018: A fixed or floating-point accumulator has been proposed for a deep learning processor that has a half-precision multiplier and single precision accumulator as shown in the figure. Two parallel 8-bit multiplications can be carried out by this design followed by a 32-bit accumulation operation. The karatsuba algorithm is used in the half-precision multiplier in this design.

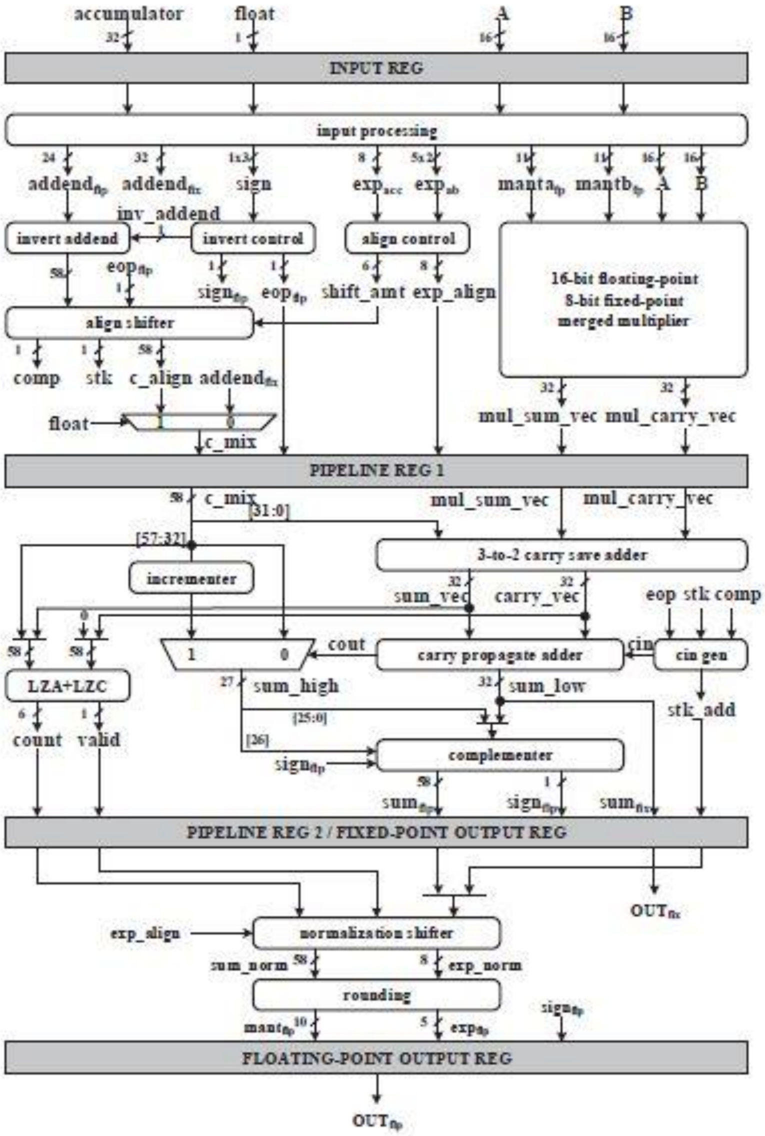


Fig 2.7: Fixed/floating-point merged mixed-precision multiply-accumulate unit

Dhanabal R et al., 2014: The Residue number system (RNS) has the properties of parallelism and carry free computation which is very useful in the implementation of fast arithmetic applications as well as in fault-tolerant computing applications. A 16-bit floating-point MAC was implemented by using this number system and the design was synthesized by using the cadence RTL compiler and achieved a speed of 25% higher than the conventional floating-point MAC unit. This design is proved to be efficient due to its parallelism which performs additions and multiplications parallelly on the residues. The block diagram of the design is shown in the below figure 2.8 [25].

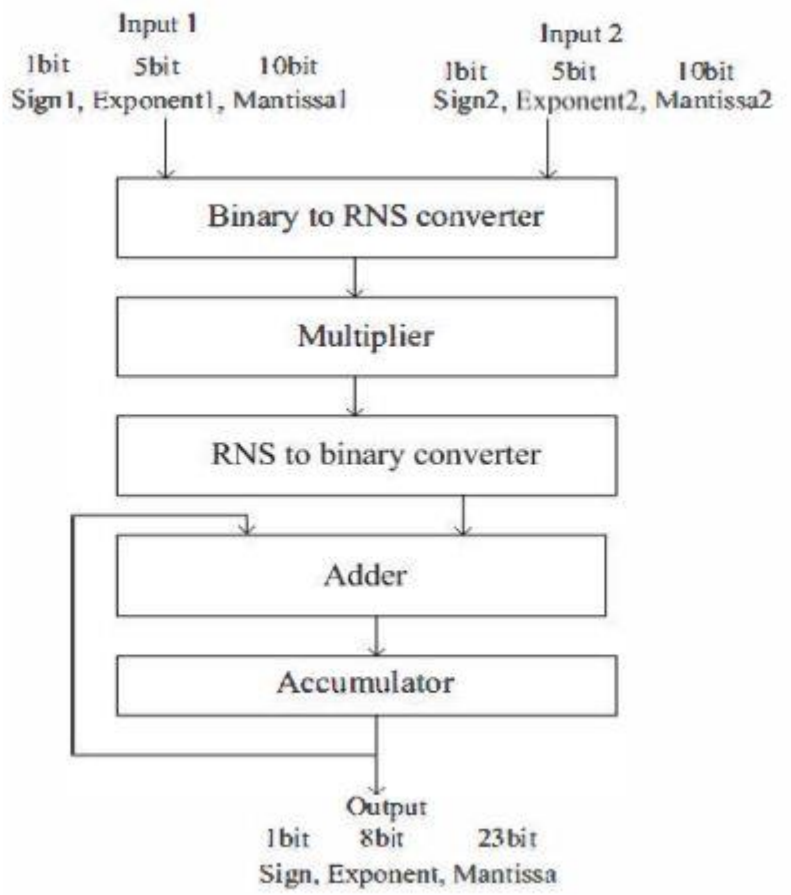


Fig 2.8: Floating-point RNS MAC unit.

M Karthik Kumar et al., 2014: Digital signal processors and digital image processors need efficient MAC units in their design to provide high speed and less power dissipation. This can be achieved by employing a Modified Booth multiplier in the MAC. M.Karthikkumar, D.Manoranjitham, and K.Praveenkumar proposed a MAC unit with a modified booth algorithm and SPST adder. The multiplier helps in reducing the number of partial products resulting in higher speed and the SPST adder removes the glitches in reduces the power dissipation [26].

Deepika Setia et al., 2013: A hybrid carry-save adder was designed by merging the multiplication with the addition which can be used to increase the performance of the MAC unit. The carry-save adder uses a modified booth algorithm. The design is simulated using the Xilinx ISE10.1 simulator and the performance was improved twice. This was possible by removing the accumulation part that has more delay [11].

Shanthala S et al., 2009: Speed is one of the important parameters to be considered in present-day technology. In the DSP applications, high speed is required and it can be achieved by pipelining. So, a pipelined MAC was designed to increase the performance of the DSP. Different architectures were analyzed by using the spice simulations which showed the best results in power and speed. The power dissipation observed in the design is 50.26 mw and the latency is observed to be 6 clock cycles [27].

Paolo Zicari et al., 2005: Speed is an important parameter in Digital signal processing systems. The delay of the final adder in the MAC unit can be reduced by merging the adder with the accumulator register. This hybrid adder is called an adder accumulator. With this adder accumulator, the delay is reduced at the cost of an increase in the area of the architecture. The experimental results show an increase in the speed of the system and the problem of carry propagation delay in the MAC was resolved [28].

Arun paidimarri et al., 2009: A floating-point multiplier accumulator was designed that is based on intel’s design will perform the accumulation operation in a single clock cycle at high clock frequencies. The floating-point multiplier accumulator is synthesized using the Stratix FPGA board using Altera’s Quartus software. The booth encoding is used in the design that will reduce the number of pipelined stages in the design but the clock frequency is reduced [29].

Li-Hsun Chen et al., 2003: Farag’s, Kwon's, and Modified Yu’s architectures are implemented in the design of the MAC unit for low power consumption. The MAC unit is designed by using tsmc 0.35um technology. The booth encoding is used to reduce the switching activities. The proposed MAC unit with the Farag’s, Kwon's and Modified Yu’s architectures achieve a power reduction of up to 35.3%, 21.6%, and 36.3% respectively. Out of these architectures, the Modified Yu’s architecture best suits for low power applications and also adds the best performance. So, this design can be best used for multimedia applications [30].

Shishir Kumar das et al., 2013: As speed is the important parameter in the Digital signal processing applications, the MAC unit must be designed with optimum speed. The MAC unit is designed using IEEE 754 Floating-point multiplier and Vedic multiplication techniques. The IEEE 754 standard enhances the design and the Vedic multiplication technique increases the speed by decreasing the delays in the design. The design is written in VHDL language and tested using the Spartan FPGA board. A comparative analysis between different multipliers like Array multiplier, Booth multiplier, and Vedic multiplier is reported. Out of these multipliers, the Vedic multiplier has less delay and power dissipation. The Vedic multiplier used in the design uses less hardware compared to other multipliers [31].

CHAPTER 3

Scope and Objectives of The Study

The Scope and research objectives covered under this work are listed below

1. Design and analysis of Half-precision MAC unit with IEEE 754 format to obtain better efficiency and performance.
2. The MAC unit will be designed in the Cadence Virtuoso so that each component in the MAC unit like an adder and the multiplier can be designed with optimum efficiency by using the latest trends.
3. A detailed analysis of the IEEE 754 will be obtained in the process of designing the Half-precision MAC architecture.
4. The design flow and the algorithm will be developed which will be helpful for further improvements in the design for optimum efficiency.
5. A comparison with the existing MAC architectures will be done that will help the scientific society to choose the best architectures for different applications.
6. A comparison will be made for the designed MAC architectures with different technology files like gpdk 45nm, gpdk 90nm, tsmc 130nm and aby other finfet technolgies.

CHAPTER 4

Equipment, Materials, Experimental Setup, and Research Methodology

The equipment needed for the design of the half-precision MAC architecture would be a system loaded with Cadence Virtuoso software. Linux OS or a windows OS with a virtual machine is needed to run the Cadence software. The virtual machine software like VMware workstation player or Virtual Box should be installed in case of windows OS.

The Cadence software must be booted in the virtual machine and the required technology files should be loaded. Here, we have gpdk 90nm and tsmc 130nm technology files.

The design is made in Cadence virtuoso and then the ADEL tool is used to simulate the design. The required power, delay, and other calculations are made with the help of the ADEL tool.

Research Methodology

The MAC is the most important part of the digital signal processing systems must have high efficiency and accuracy. Designing the MAC with IEEE 754 will increase the performance of the MAC unit. In this thesis work, the MAC unit is designed in the Cadence virtuoso. Each component of the MAC like Adder and the multiplier are designed by using this tool. This design will increase the efficiency of the MAC unit.

A cadence is a software and hardware design company that supplies tools to its customers in the areas of 5G communications, aerospace, and medical applications. Cadence virtuoso software is used for our design.

There is a wide range of technologies that can be used for the design of the cadence tool. For this design, we can use the available 90nm gpdk technology or the tsmc 130nm technology.

CHAPTER 5

Experimental Work

The Basic building blocks of MAC architecture like Normalization block, adder, multiplier, etc., are designed in cadence virtuoso and the analysis on the individual components is done. The power and the delay calculations for the normalization block, adder and the multiplier are given in the table 5.1.

Table 5.1 Power and Delay Analysis of different blocks in SFMAC Architecture.

Block	Static Power	Average Power	Delay
Normalization block	139.7mW	146.4mW	121.6ps
Adder	0.54mW	4.09mW	455.7ps
Multiplier	13.6mW	17.77mW	4.53ns

The delay will be high in the case of circuits like exponential shifter and multiplier circuits which will result in faults when the circuits are run for longer periods. To avoid this, the outputs are latched with the help of a PED- latch. These latches will work with a clock signal, whenever the clock signal is high, all the outputs will arrive at a time which will reduce the errors and provide better synchronization between other circuits.

The Half-precision MAC unit uses 2’s complement block to represent negative numbers. the size of the architecture is 16 bits in which one bit is used for representing the sign bit of mantissa and 10 bits are used for the mantissa. 5 bits are used for exponent representation in which MSB is the sign bit and the 4th bit is ‘0’. This bit is reserved for representing 2’s complement negative numbers and the rest 3 bits are the exponents of inputs. The MAC architecture input format representation is shown in figure 5.1.

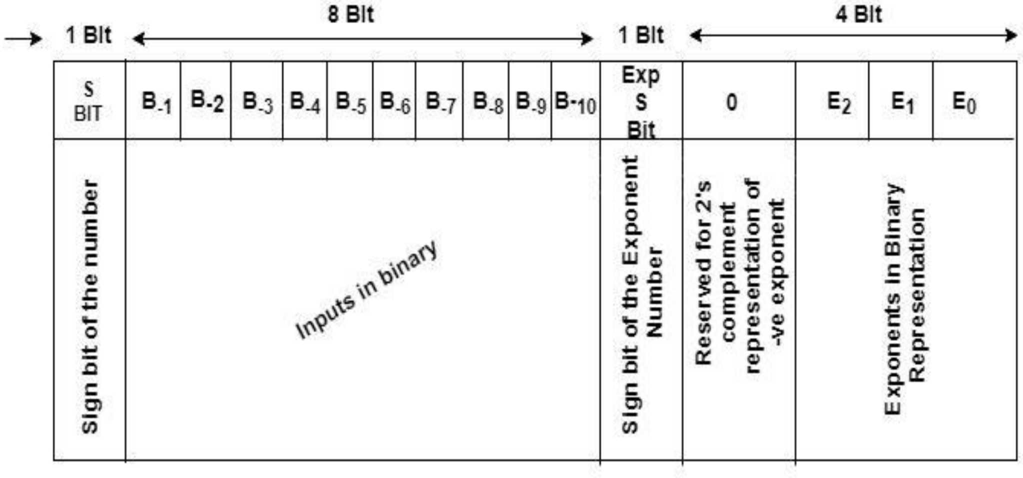

1 Bit
8 Bit
1 Bit
4 Bit
S BIT
B-1
B-2
B-3
B-4
B-5
B-6
B-7
B-8
B-9
B-10
Exp S Bit
0
E2
E1
E0
Sign bit of the number
Inputs in binary
Sign bit of the Exponent Number
Reserved for 2's complement representation of -ve exponent
Exponents in Binary Representation

Fig 5.1: Input Format Representation of MAC architecture.

The main blocks of the MAC architecture are

1. Exponential Adder
2. Exponential Comparator circuit
3. Exponential shifter circuit
4. 10-bit Multiplier
5. 20-bit Adder
6. 20-bit Register
7. Multiplexers of different sizes
8. 2’s complement blocks of different sizes
9. Half adders/ full adders.
10. PED latch block.

5.1 Exponential Adder

This EA circuit uses the 2’s complement block to ease the process of addition in case of any negative input exponents. The 3-bit input in the EA would produce an output of 4 bits and to represent 4 bits in 2’s complement form, it will take 5 bits. The 6th bit would be the sign bit.

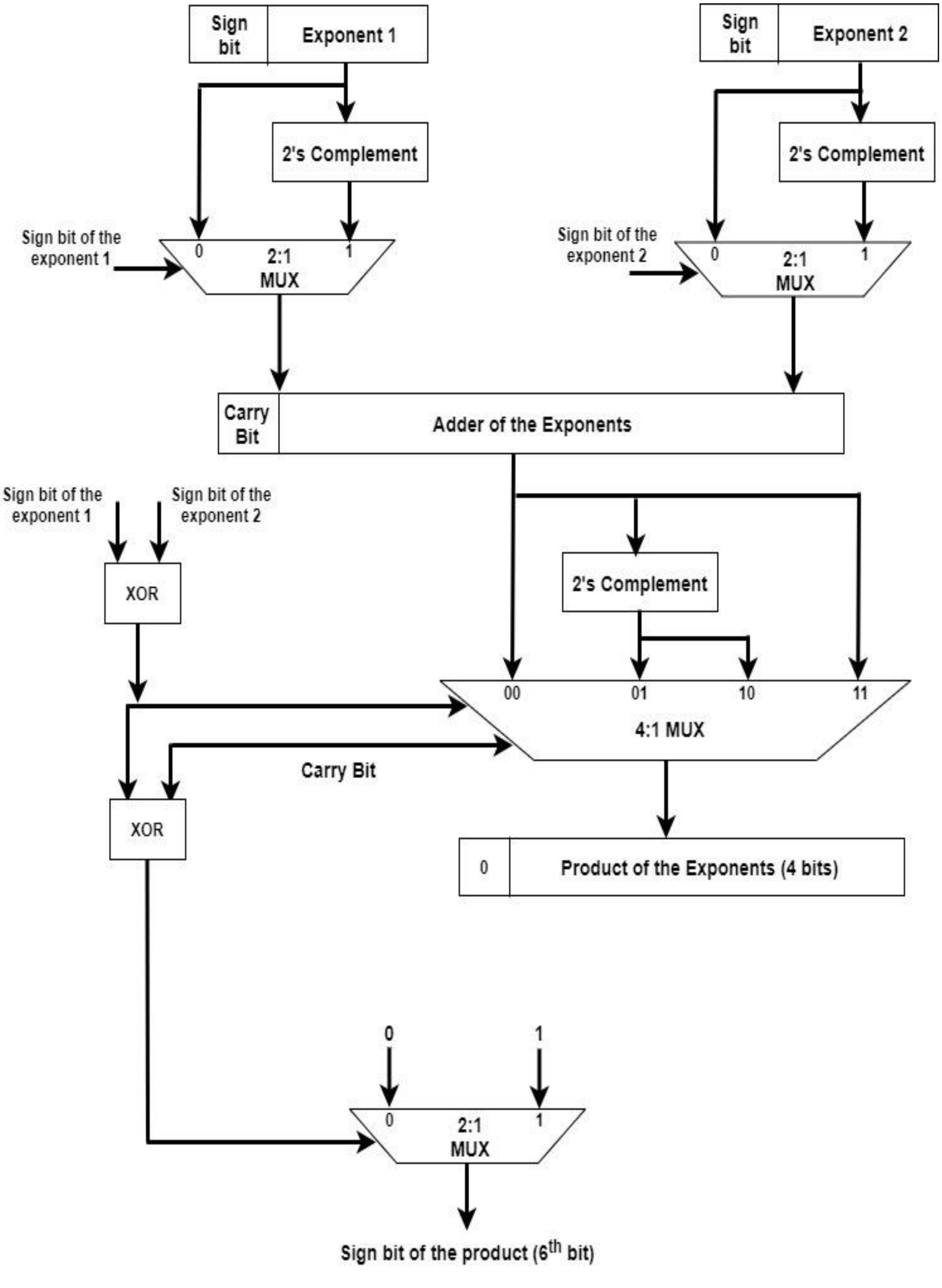

Sign bit
Exponent 1
Sign bit
Exponent 2
2's Complement
2's Complement
Sign bit of the exponent 1
0
2:1 MUX
1
Sign bit of the exponent 2
0
2:1 MUX
1
Carry Bit
Adder of the Exponents
Sign bit of the exponent 1
Sign bit of the exponent 2
XOR
2's Complement
00
01
10
11
4:1 MUX
Carry Bit
XOR
0
Product of the Exponents (4 bits)
0
1
0
2:1 MUX
1
Sign bit of the product (6th bit)

Fig 5.2: Exponent Adder

Based on the sign of the exponents, the inputs will be passed through a 2’s complement block if it is a negative number. Then these exponents are passed to the 5-bit adder through 2*1 multiplexers. A 4*1 multiplexer is used to pass the final output from the adder. An XOR gate whose inputs are the sign bits of the exponents and the carry bit of the adder is used as the selection lines for the 4*1 multiplexer. The sign bit is produced by a 2*1 multiplexer where the selection line would be the output of the XOR gate is shown in figure 5.2 [32]. The design is done cadence virtuoso as shown in figure 5.3.

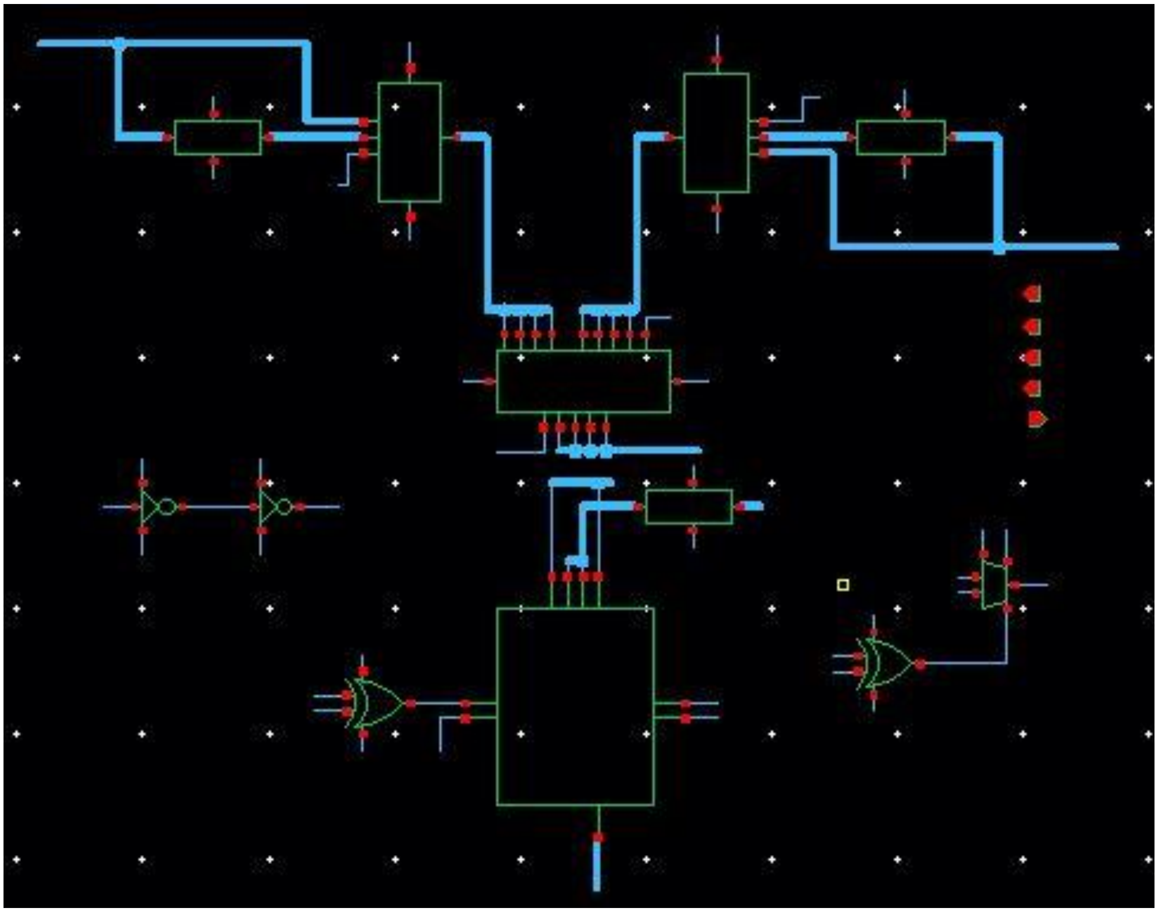


Fig 5.3: Exponent adder designed in Cadence Virtuoso

The simulated waveform is shown in the figure 5.4 where two inputs are given to the EA, Input1: 02(00010) and input2: 13(10011) and the produced results is 21(100001)

Input1: 1d(11101) and input2: 0c(01100) and the produced result is 21(100001)

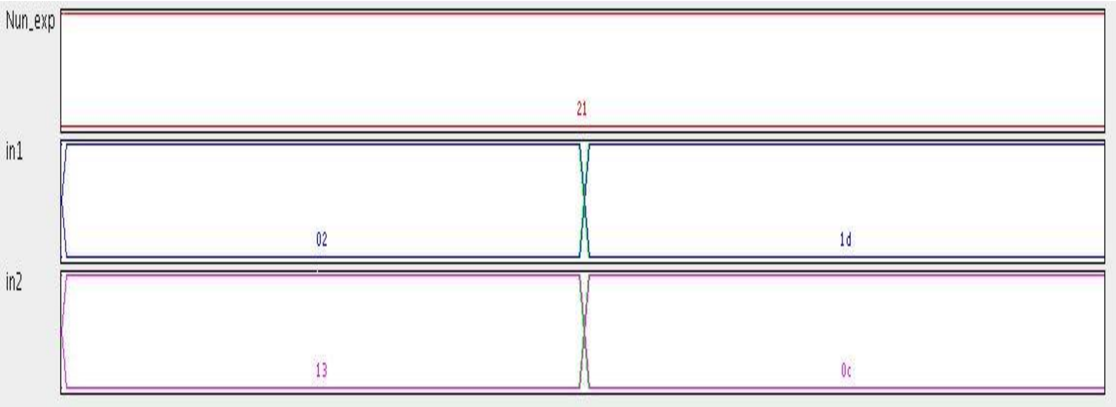


Fig 5.4: Simulated waveform of exponent adder

5.2 Exponential Comparator Circuit

The inputs to the Exponential circuit are the 6-bit output produced by the EA block. One of the main points in the ECC is that if both the inputs carry the same sign, then the comparator will produce a difference between the two inputs, or else the comparator will produce output by adding both the inputs.

For example, +x and +y, if given to the Comparator, the produced output will either x-y or y-x and if the inputs are -x and +y or +x and -y, then the output will be y+x or x+y. The ECC is given in figure 5.5 which uses multiplexers to compare the inputs.

If the inputs are of the same sign, then they are passed through an exponent comparator circuit with both positive or negative signs. In the other case, the output will be passed from an adder with the help of a multiplexer.

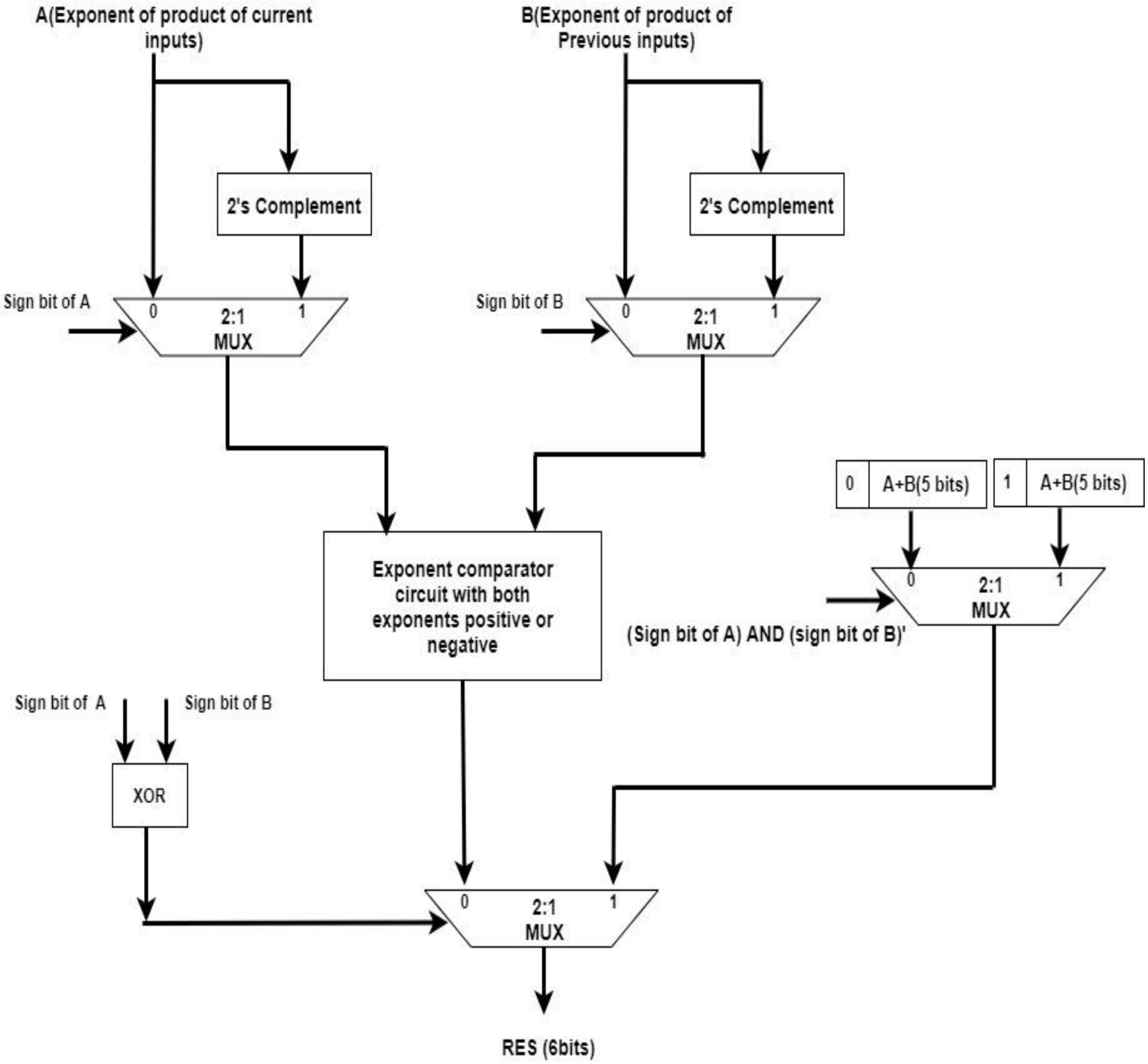

A(Exponent of product of current inputs)
B(Exponent of product of Previous inputs)
2's Complement
2's Complement
Sign bit of A
0
2:1 MUX
1
Sign bit of B
0
2:1 MUX
1
0
A+B(5 bits)
1
A+B(5 bits)
Exponent comparator circuit with both exponents positive or negative
0
2:1 MUX
1
(Sign bit of A) AND (sign bit of B)'
Sign bit of A
Sign bit of B
XOR
0
2:1 MUX
1
RES (6bits)

Fig 5.5: Exponential Comparator Circuit

If both the inputs are having the same sign, then they are compared bit by bit with the help of multiplexers as shown in figure 5.6.

The difference is found by 2’s complement approach. The difference produces a 5-bit output in which borrow is discarded but introduces the 6th bit as '0' if the product of the exponents of the inputs is higher than the previous cycle exponent. Make the 6th bit as ‘1' in the other cases.

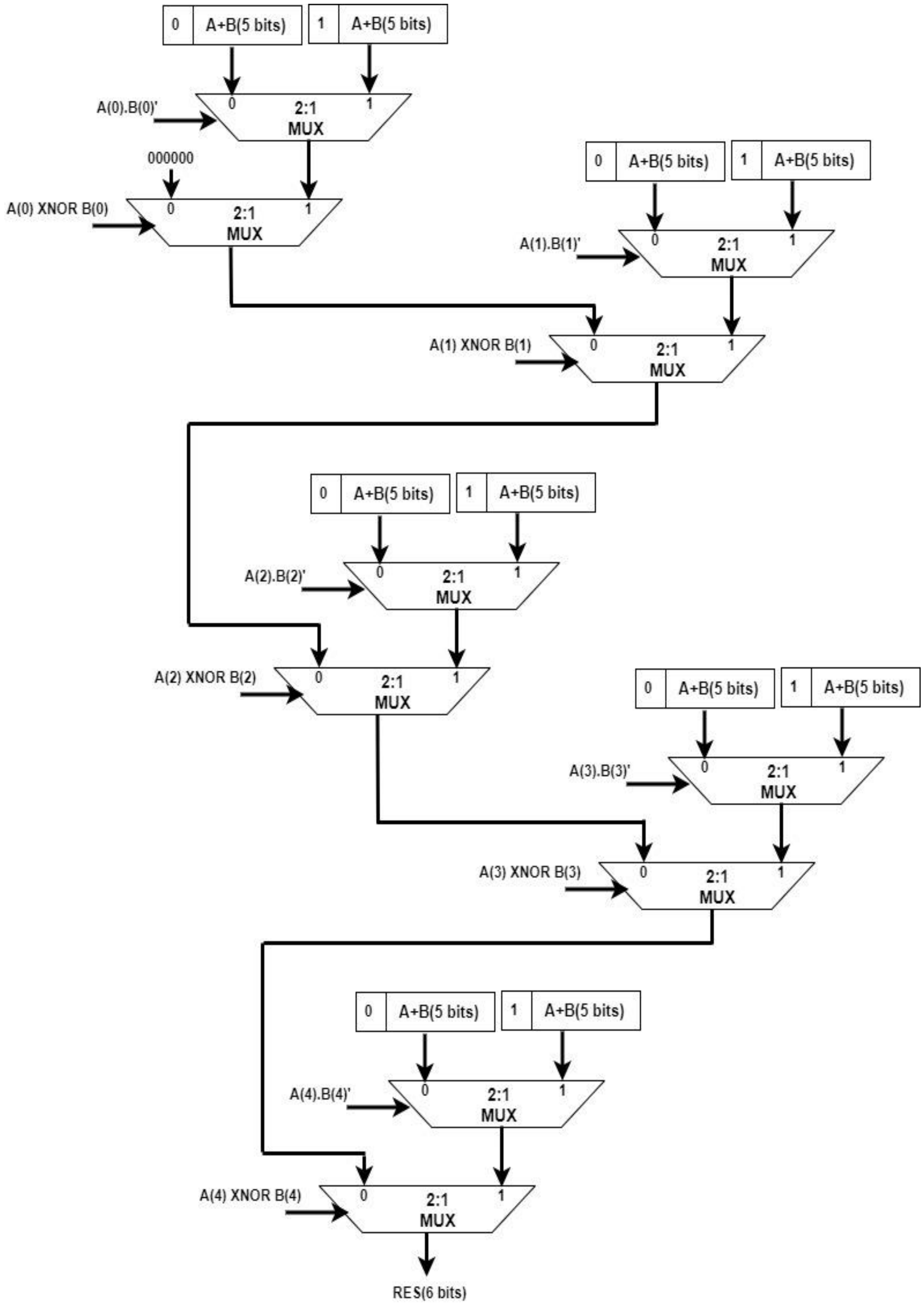


Fig 5.6: ECC with the same sign bit

The ECC is designed in cadence virtuoso and the design is shown in figure 5.7.

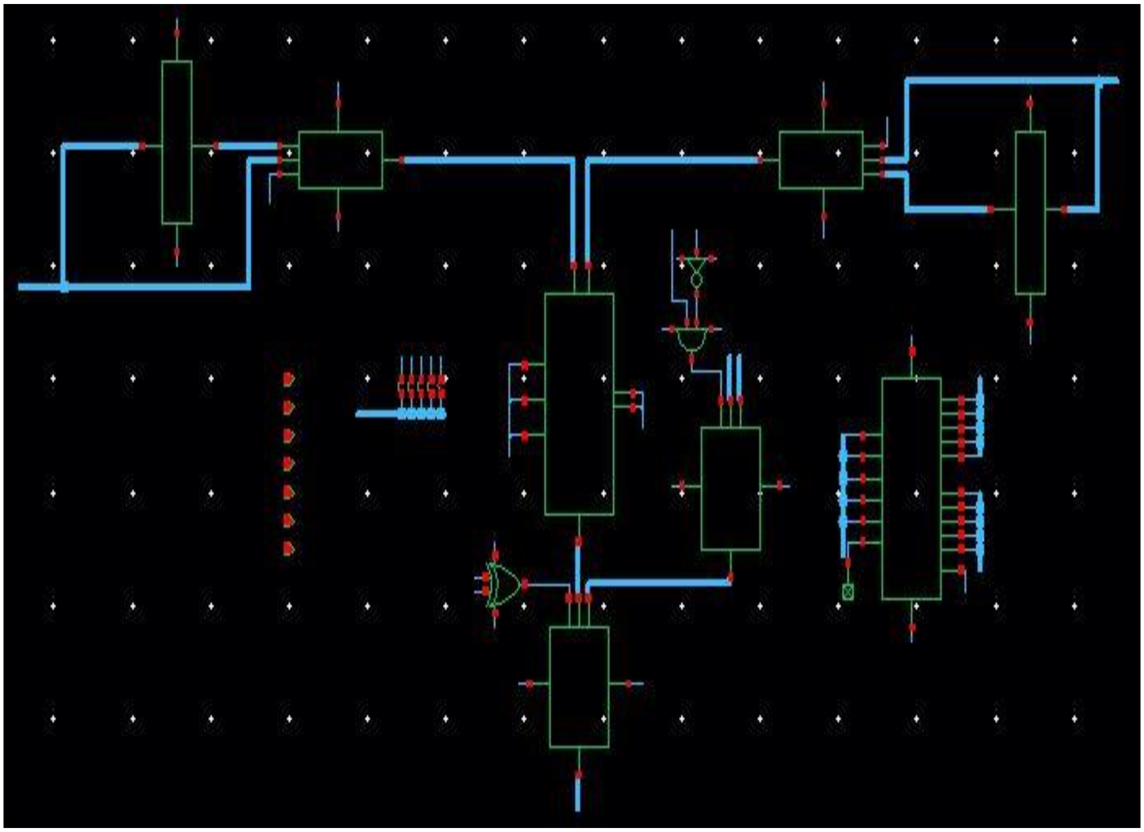


Fig 5.7: Exponent comparator circuit in Cadence Virtuoso

The one input to the exponents will be the output from the EA block and the other input will be from a previous output exponent. In the simulation, the EA block output is taken as the first input(Num_exp) and the Previous output exponent is taken as the second input(Prevop_exp).

Here Num_exp(21 in hexadecimal) which is equal to 100001 in binary and

Prevop_exp(02 in hexadecimal) which is equal to 000010 in binary.

Since the inputs are of different signs, the output will be selected from the adder as shown in the figure 5.8.

The addition of both the inputs will result in 100011 which is equal to 23 in hexadecimal

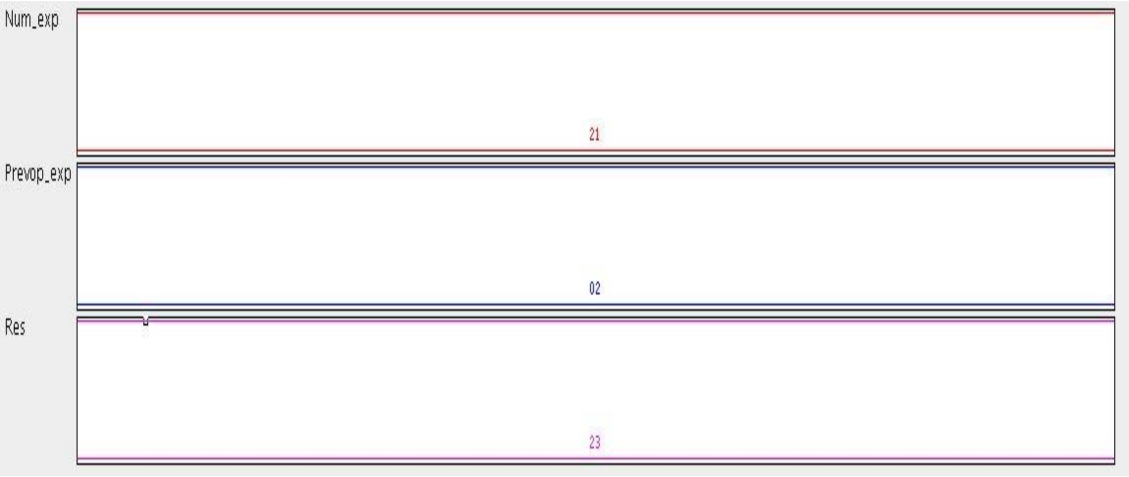


Fig 5.8: Simulated waveform of Exponential comparator circuit

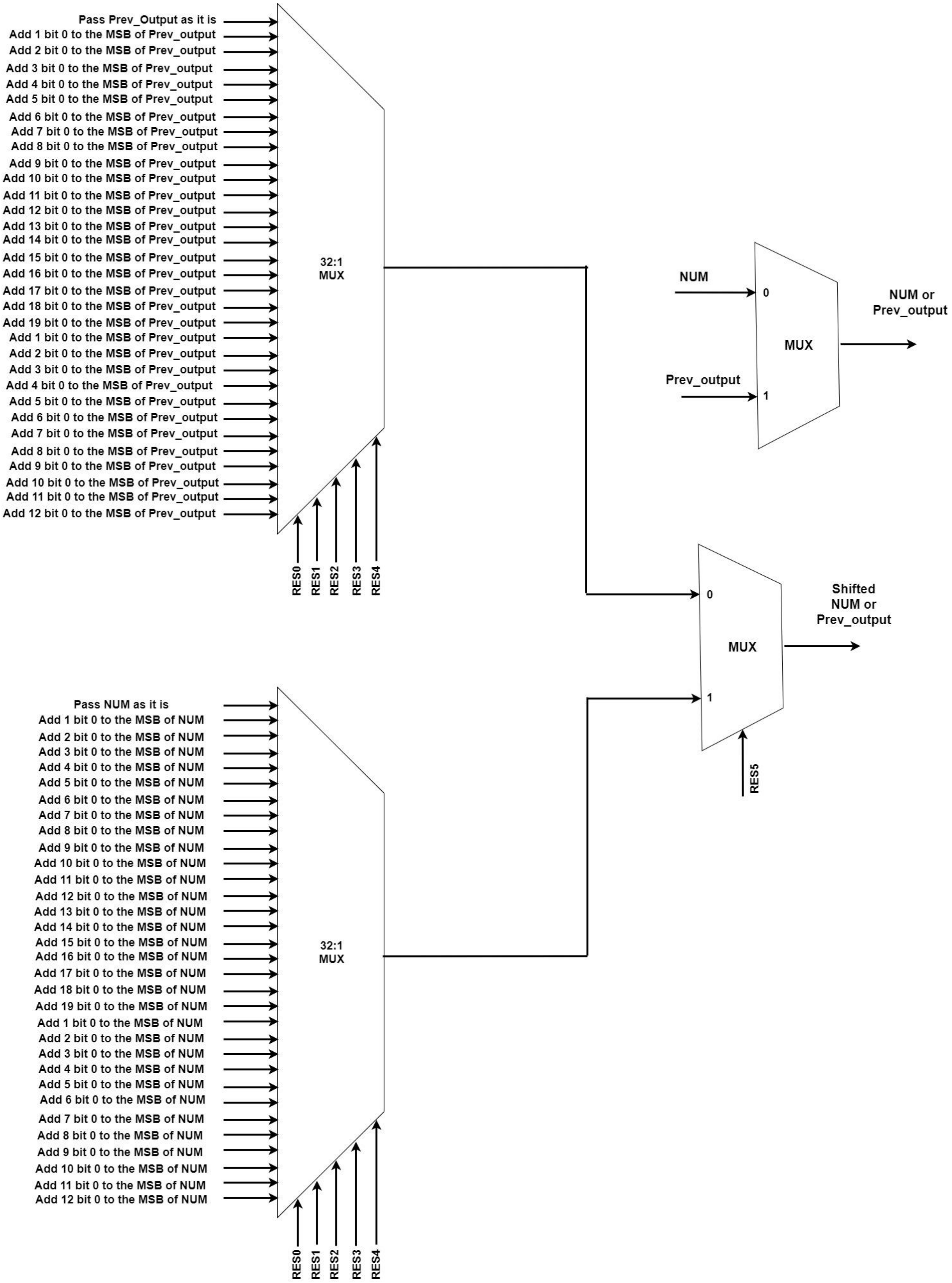
5.3 Exponential Shifter Circuit

The Exponential shifter is used for the shifting of the bits based on the Exponential comparator circuit output. The MSB bit is used to select the shifting operation between the Previous output and the present input which is coming from the 10-bit multiplier.

32-bit multiplexers are used for the shifting of the 20-bit previous output or the present input. The shifting is based on the decimal of equivalent of the remaining 5 bits of ECC output which is given as input to the selection lines of the multiplexer as shown in figure 5.9.

The inputs which need shifting is identified by the MSB bit of RES, if the MSB of the RES is 0 then the previous output is shifted to the right side by the equivalent of the remaining 5 bits, or else the present input (NUM) is shifted.

The inputs of the exponent shifter circuit which need no shifting are found by the MSB of the RES, if the MSB of RES is 0 then the present input (NUM) is passed as it is, or else the previous output is passed as it is.



Pass Prev_Output as it is
Add 1 bit 0 to the MSB of Prev_output
Add 2 bit 0 to the MSB of Prev_output
Add 3 bit 0 to the MSB of Prev_output
Add 4 bit 0 to the MSB of Prev_output
Add 5 bit 0 to the MSB of Prev_output
Add 6 bit 0 to the MSB of Prev_output
Add 7 bit 0 to the MSB of Prev_output
Add 8 bit 0 to the MSB of Prev_output
Add 9 bit 0 to the MSB of Prev_output
Add 10 bit 0 to the MSB of Prev_output
Add 11 bit 0 to the MSB of Prev_output
Add 12 bit 0 to the MSB of Prev_output
Add 13 bit 0 to the MSB of Prev_output
Add 14 bit 0 to the MSB of Prev_output
Add 15 bit 0 to the MSB of Prev_output
Add 16 bit 0 to the MSB of Prev_output
Add 17 bit 0 to the MSB of Prev_output
Add 18 bit 0 to the MSB of Prev_output
Add 19 bit 0 to the MSB of Prev_output
Add 1 bit 0 to the MSB of Prev_output
Add 2 bit 0 to the MSB of Prev_output
Add 3 bit 0 to the MSB of Prev_output
Add 4 bit 0 to the MSB of Prev_output
Add 5 bit 0 to the MSB of Prev_output
Add 6 bit 0 to the MSB of Prev_output
Add 7 bit 0 to the MSB of Prev_output
Add 8 bit 0 to the MSB of Prev_output
Add 9 bit 0 to the MSB of Prev_output
Add 10 bit 0 to the MSB of Prev_output
Add 11 bit 0 to the MSB of Prev_output
Add 12 bit 0 to the MSB of Prev_output
32:1
MUX
RES0
RES1
RES2
RES3
RES4
NUM
0
NUM or
Prev_output
MUX
Prev_output
1
0
Shifted
NUM or
Prev_output
MUX
1
RES5
Pass NUM as it is
Add 1 bit 0 to the MSB of NUM
Add 2 bit 0 to the MSB of NUM
Add 3 bit 0 to the MSB of NUM
Add 4 bit 0 to the MSB of NUM
Add 5 bit 0 to the MSB of NUM
Add 6 bit 0 to the MSB of NUM
Add 7 bit 0 to the MSB of NUM
Add 8 bit 0 to the MSB of NUM
Add 9 bit 0 to the MSB of NUM
Add 10 bit 0 to the MSB of NUM
Add 11 bit 0 to the MSB of NUM
Add 12 bit 0 to the MSB of NUM
Add 13 bit 0 to the MSB of NUM
Add 14 bit 0 to the MSB of NUM
Add 15 bit 0 to the MSB of NUM
Add 16 bit 0 to the MSB of NUM
Add 17 bit 0 to the MSB of NUM
Add 18 bit 0 to the MSB of NUM
Add 19 bit 0 to the MSB of NUM
Add 1 bit 0 to the MSB of NUM
Add 2 bit 0 to the MSB of NUM
Add 3 bit 0 to the MSB of NUM
Add 4 bit 0 to the MSB of NUM
Add 5 bit 0 to the MSB of NUM
Add 6 bit 0 to the MSB of NUM
Add 7 bit 0 to the MSB of NUM
Add 8 bit 0 to the MSB of NUM
Add 9 bit 0 to the MSB of NUM
Add 10 bit 0 to the MSB of NUM
Add 11 bit 0 to the MSB of NUM
Add 12 bit 0 to the MSB of NUM
32:1
MUX
RES0
RES1
RES2
RES3
RES4

Fig 5.9: Exponential Shifter Circuit

The exponential shifter circuit is designed in cadence virtuoso and the design is shown in figure 7.10. 32*1 multiplexer with 5 selection bits is used to select the shifting operation as shown in figure 5.11.

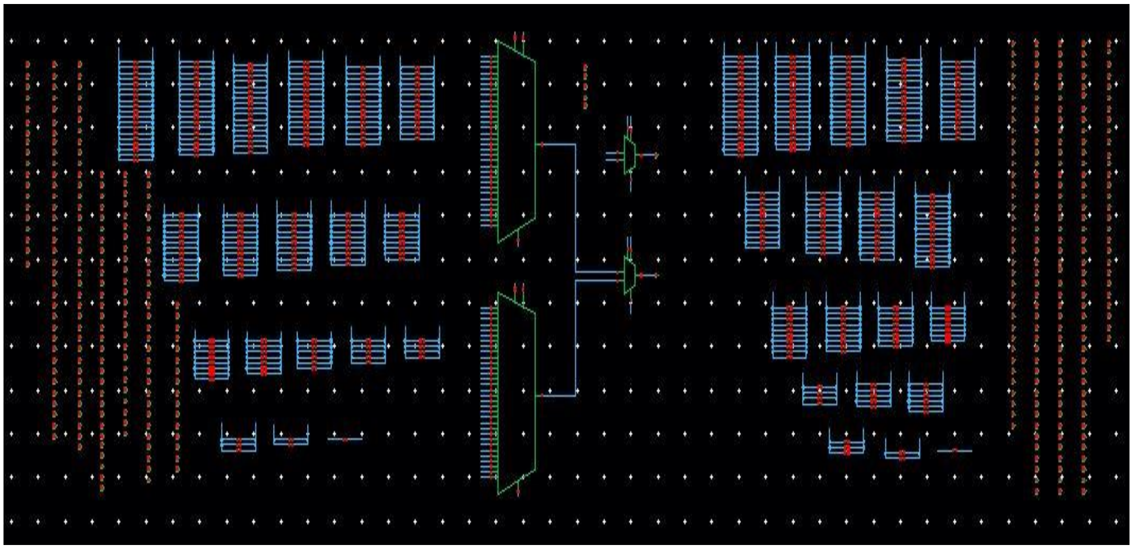


Fig 5.10: Exponential shifter circuit in Cadence Virtuoso

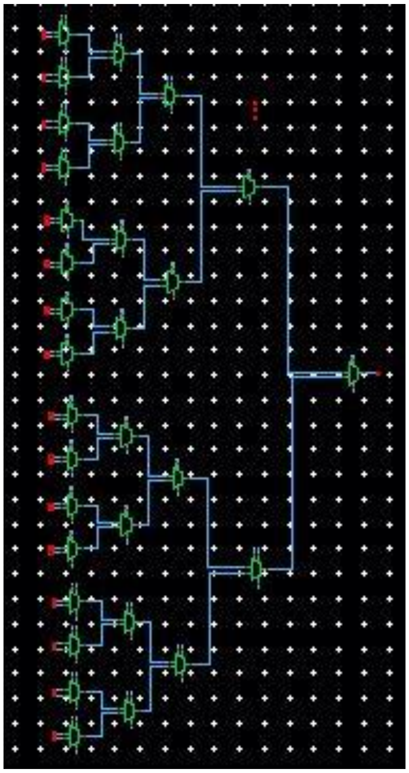


Fig 5.11: 32*1 multiplexer

The simulated waveform of Exponential shifter in cadence virtuoso is given in the figure 5.12.

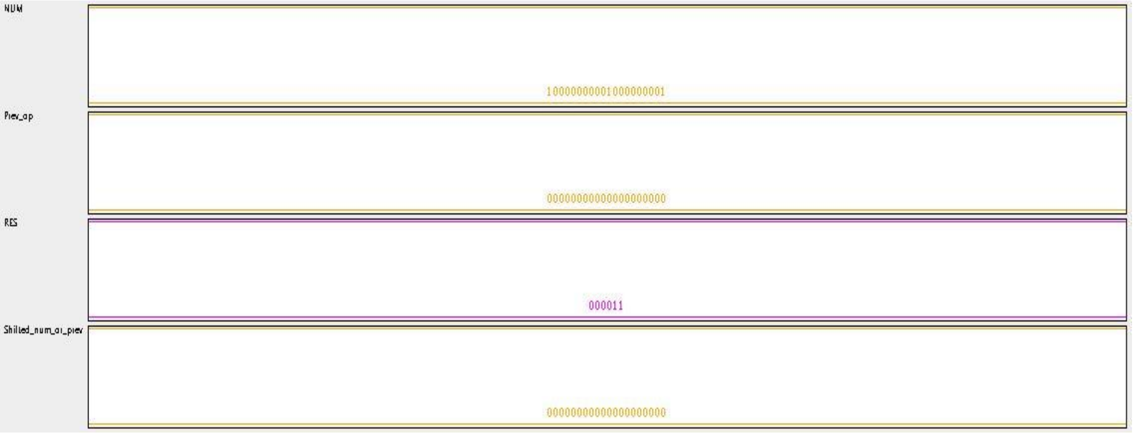


Fig 5.12: Simulated waveform of Exponential shifter circuit.

As shown in figure 5.12, there are two inputs given to the shifter circuit. One is a present number (NUM) from the multiplier and the other is the previous output (Prev_op) from the register. Based on the value of the RES, either of the inputs is shifted. Here, the MSB of RES is 0. So, Prev_op is shifted to the decimal equivalent of the remaining 5 bits. The decimal equivalent of RES[4:0] in the above figure is 3 which means that the Prev_op is shifted towards the right by 3 bits. Since all the inputs in the above case are 0's, the output is observed as all 0's.

5.4 Multiplier

Multiplication has various applications in the fields of digital signal processing, digital image processing, multimedia systems, etc., There are many types of multipliers like array multiplier, booth multiplier, and Wallace multiplier, etc., The Half-precision floating-point MAC requires a 10-bit mantissa which means the multiplier in the MAC architecture should be of 10* 10-bit size [33].

There are many stages involved in multiplication from partial product generation to the final sum generation. The partial products generation involves a large number of AND gates and the sum generation will require many adder circuits.

Taking the advantages of better optimization of partial products in the booth multiplier and partial production addition in Wallace tree multiplier [34][35][36][37][38][39][40], A new Universal compressor-based architecture has been developed in [41] Which yields better results. The regular Wallace tree adders are replaced with compressor circuits. Different sizes of compressors like 4:2, 9:4, etc., are used in the design. The 9*9 UCM architecture is shown in figure 5.13.

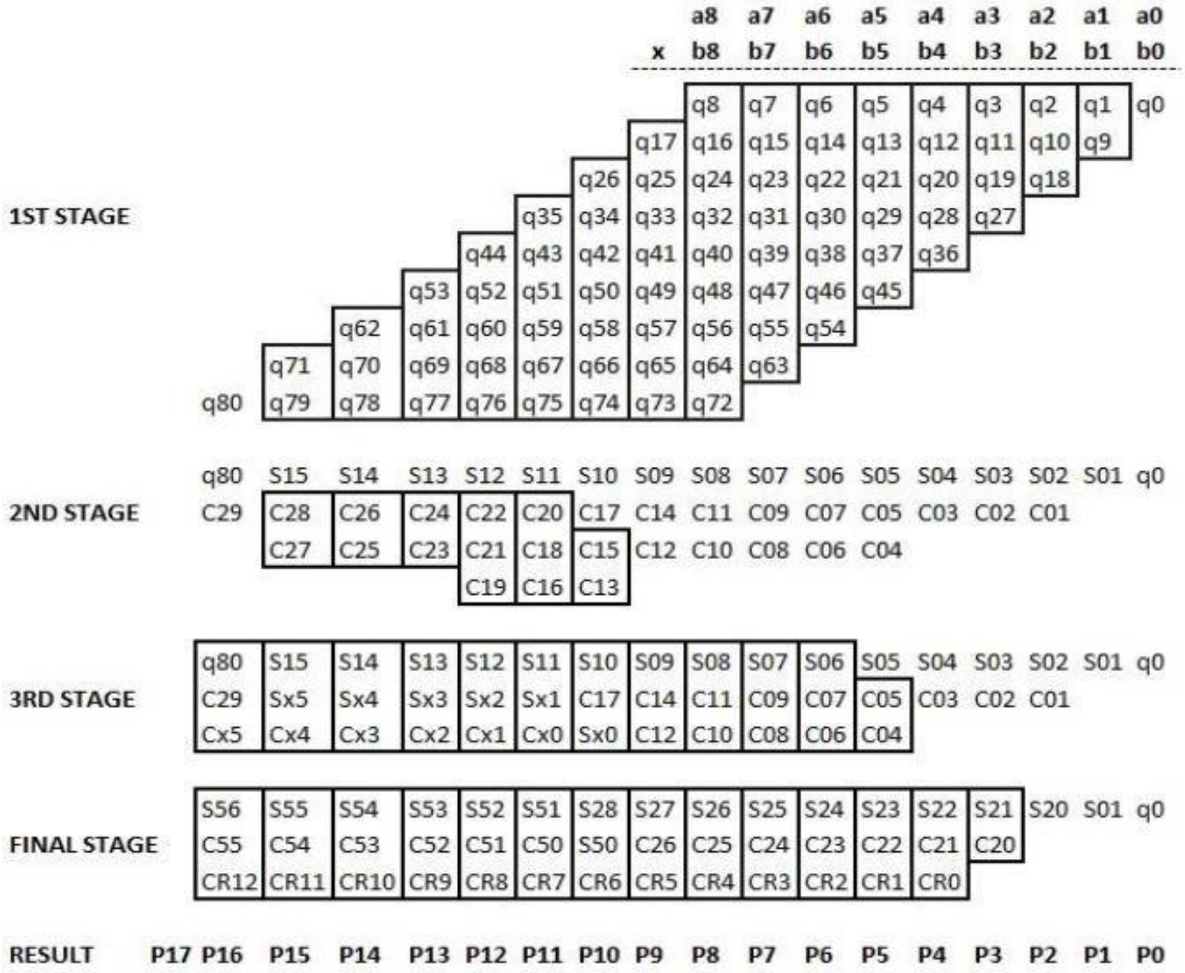

a8 a7 a6 a5 a4 a3 a2 a1 a0
x b8 b7 b6 b5 b4 b3 b2 b1 b0
q8 q7 q6 q5 q4 q3 q2 q1 q0
q17 q16 q15 q14 q13 q12 q11 q10 q9
q26 q25 q24 q23 q22 q21 q20 q19 q18
1ST STAGE
q35 q34 q33 q32 q31 q30 q29 q28 q27
q44 q43 q42 q41 q40 q39 q38 q37 q36
q53 q52 q51 q50 q49 q48 q47 q46 q45
q62 q61 q60 q59 q58 q57 q56 q55 q54
q71 q70 q69 q68 q67 q66 q65 q64 q63
q80 q79 q78 q77 q76 q75 q74 q73 q72
2ND STAGE
q80 S15 S14 S13 S12 S11 S10 S09 S08 S07 S06 S05 S04 S03 S02 S01 q0
C29 C28 C26 C24 C22 C20 C17 C14 C11 C09 C07 C05 C03 C02 C01
C27 C25 C23 C21 C18 C15 C12 C10 C08 C06 C04
C19 C16 C13
3RD STAGE
q80 S15 S14 S13 S12 S11 S10 S09 S08 S07 S06 S05 S04 S03 S02 S01 q0
C29 Sx5 Sx4 Sx3 Sx2 Sx1 C17 C14 C11 C09 C07 C05 C03 C02 C01
Cx5 Cx4 Cx3 Cx2 Cx1 Cx0 Sx0 C12 C10 C08 C06 C04
FINAL STAGE
S56 S55 S54 S53 S52 S51 S28 S27 S26 S25 S24 S23 S22 S21 S20 S01 q0
C55 C54 C53 C52 C51 C50 S50 C26 C25 C24 C23 C22 C21 C20
CR12 CR11 CR10 CR9 CR8 CR7 CR6 CR5 CR4 CR3 CR2 CR1 CR0
RESULT
P17 P16 P15 P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 P0

Fig 5.13: 9*9 UCM architecture

The architecture is designed in Cadence virtuoso and the partial products block is shown in figure 5.14. For 10*10 bit multiplier, the number of partial products produced would be 100.

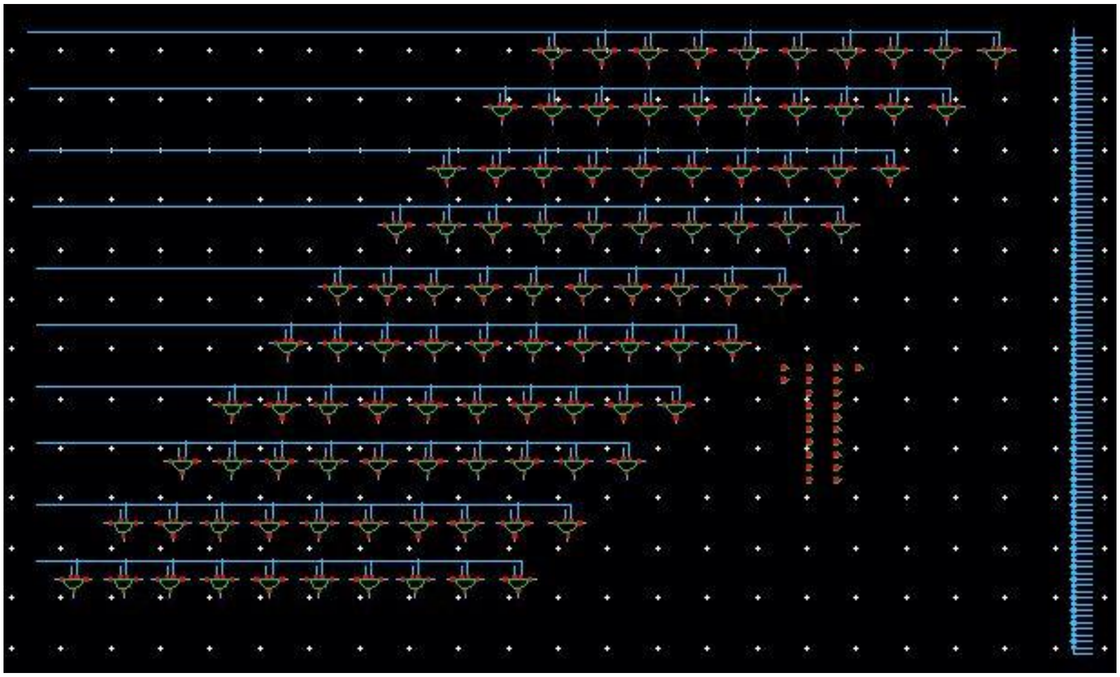


Fig 5.14: partial product generation of the multiplier in Cadence virtuoso

The partial products are added by the AND-XOR gates in the second stage. The number of stages required will be given in the formula (1).

$$2^n\text{-}1 \geq I \qquad \text{- (1)}$$

Where I am the number of partial products to be added and n will be the number of levels required.

For example, for adding 9 partial products it will take 4 AND-XOR stages.

$$= 2^n\text{-}1 \geq 9$$

$$= 2^n \geq 10$$

To satisfy the above condition, n should be 4, so a total of 4 stages will be required. The AND-XOR stages are given in figure 5.15.

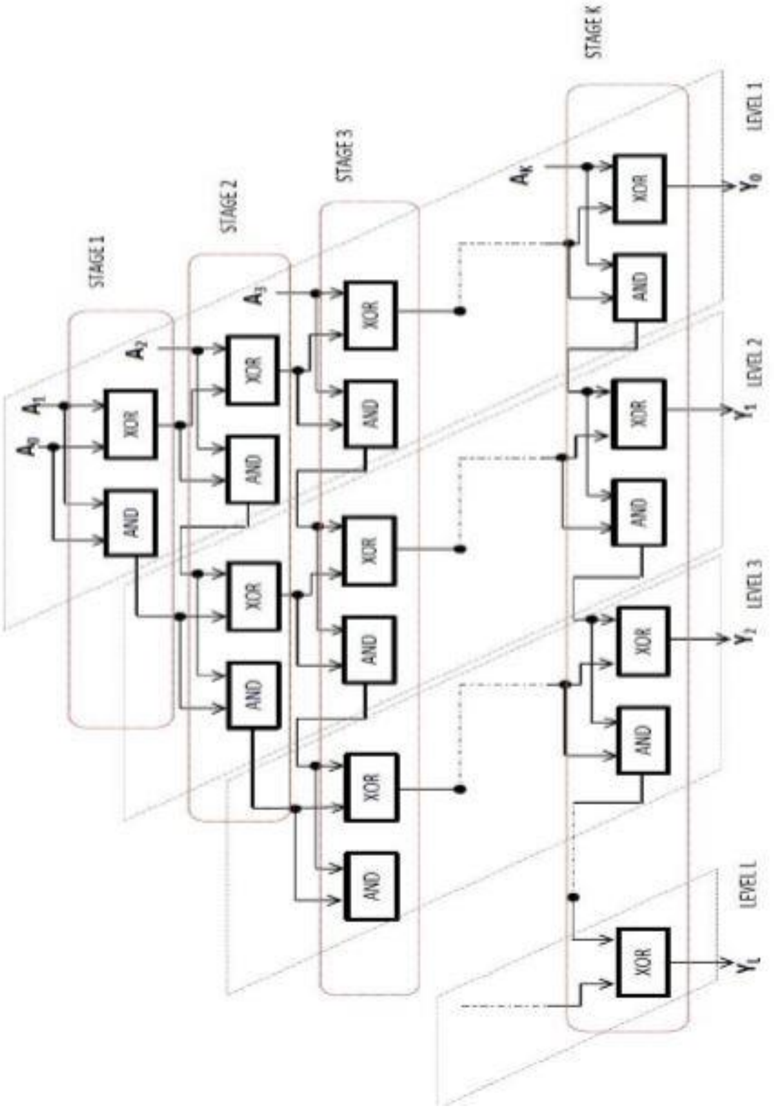


Fig 5.15: AND – XOR stages for partial product addition

The output produced from the AND-XOR gates is added with the help of Full adders and Half adders. The last stage in the multiplier is the ripple carry adder which produces the final output. The multiplier designed in the cadence is given in figure 5.16.

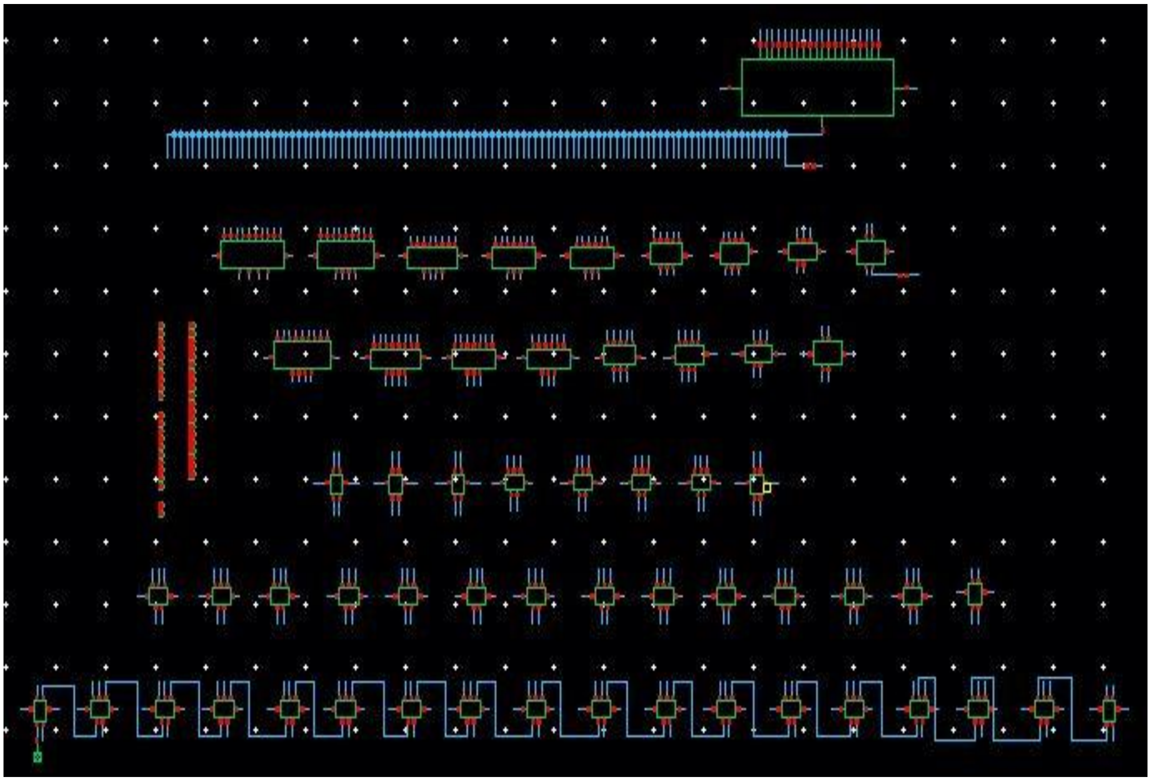


Fig 5.16: 10*10 bit UCM architecture in Cadence Virtuoso.

The simulated results for the multiplier are given in figure 5.17.

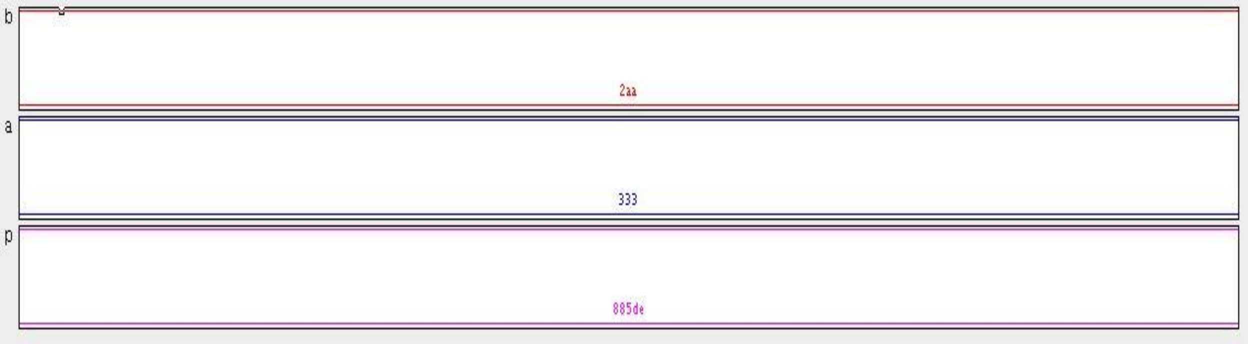


Fig 5.17: Simulated waveform of 10*10 multiplier

The inputs to multiplier are

A = 2aa which is equal to 682 in decimal and

B = 333 which is equal to 819 in decimal.

The produced multiplication result is 885de which is equal to 558558

5.5 20-bit Adder

In the MAC architecture, the adder is used in the accumulation part. The output of the exponential shifter circuit is passed through the 2's complement multiplexers to the 20-bit adder. The inputs passed to the adder are the shifted 20-bit previous output/ present input or the non-shifted number. The output of the adder is passed through 2's complement multiplexers to get the output in non-complemented form. Then the output is stored in a 20-bit register to be used for the next cycle of inputs and the same is taken as the final output for the cycle. The 20-bit adder designed in the Cadence virtuoso is shown in figure 5.18 where 20 Full adder circuits are used to form a ripple carry adder. The Full adder used in the design is shown in figure 5.19.

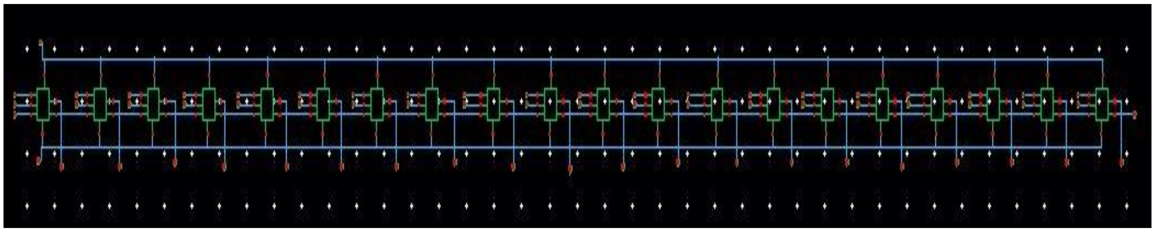


Fig 5.18: 20-bit Adder

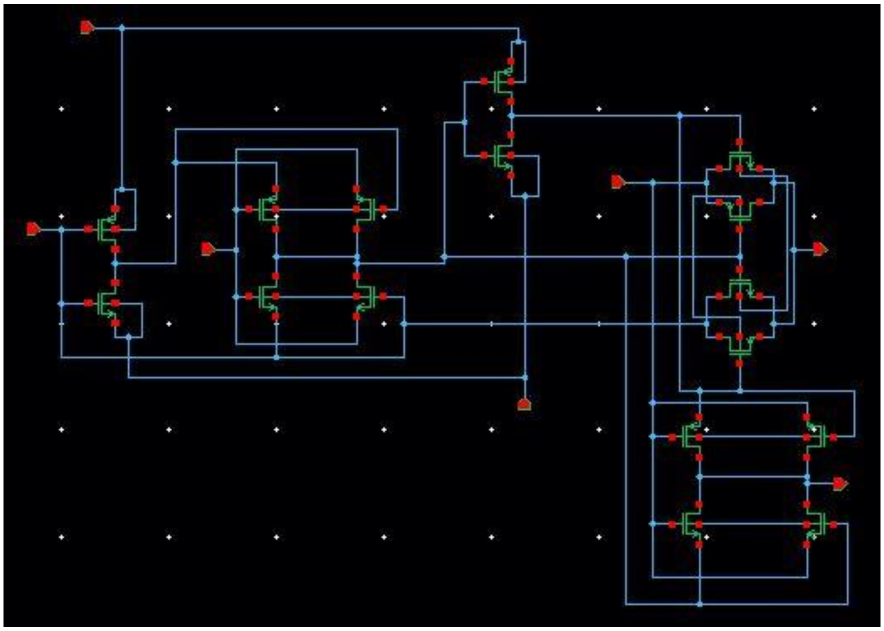


Fig 5.19: Full adder in cadence virtuoso

The simulated waveform for the 20-bit adder is given in the figure 7.20 where 2 sets of are given.

For the first cycle, the inputs are a = b = ffffe which is equal to 1048574 in decimal. The generated output from the adder is out = 1ffffc which is equal to 2097148.

In the second cycle, the inputs are a = b = 00001 which is equal to 1 in decimal. The generated output is out = 000002 which is equal to 2 in decimal.

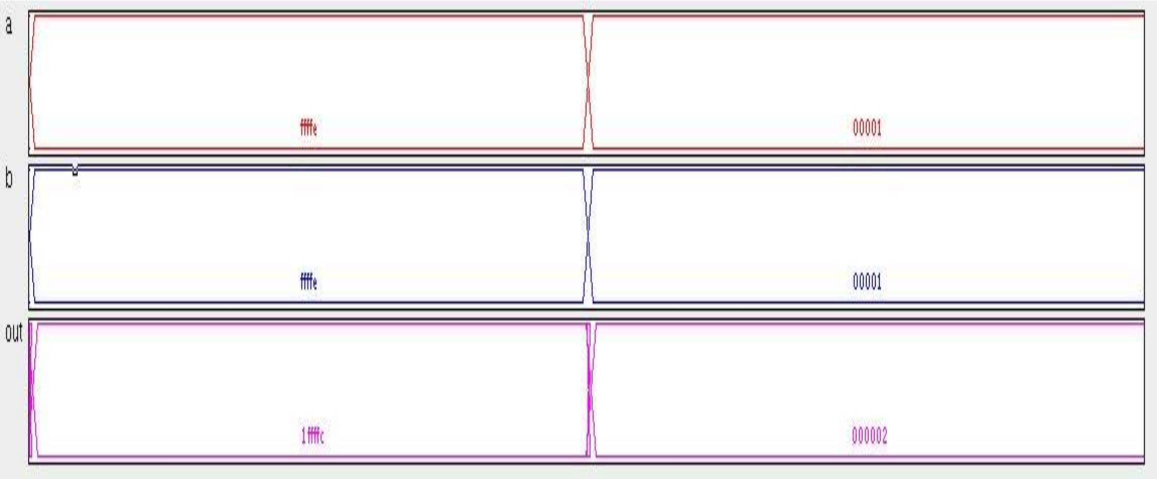


Fig 5.20: Simulated waveform for 20-bit adder

5.6 Register

A 20-bit Register is used to store the output from the adder. The stored data in the register is sent to the Exponential shifter circuit for the next cycle. A 6-bit register is used to store the updated exponents from the Exponential shifter circuit. The exponents stored in this register will be used for the next cycle. A 1-bit register given in figure 5.21 is used to design the 6-bit and 20-bit registers in the MAC architecture.

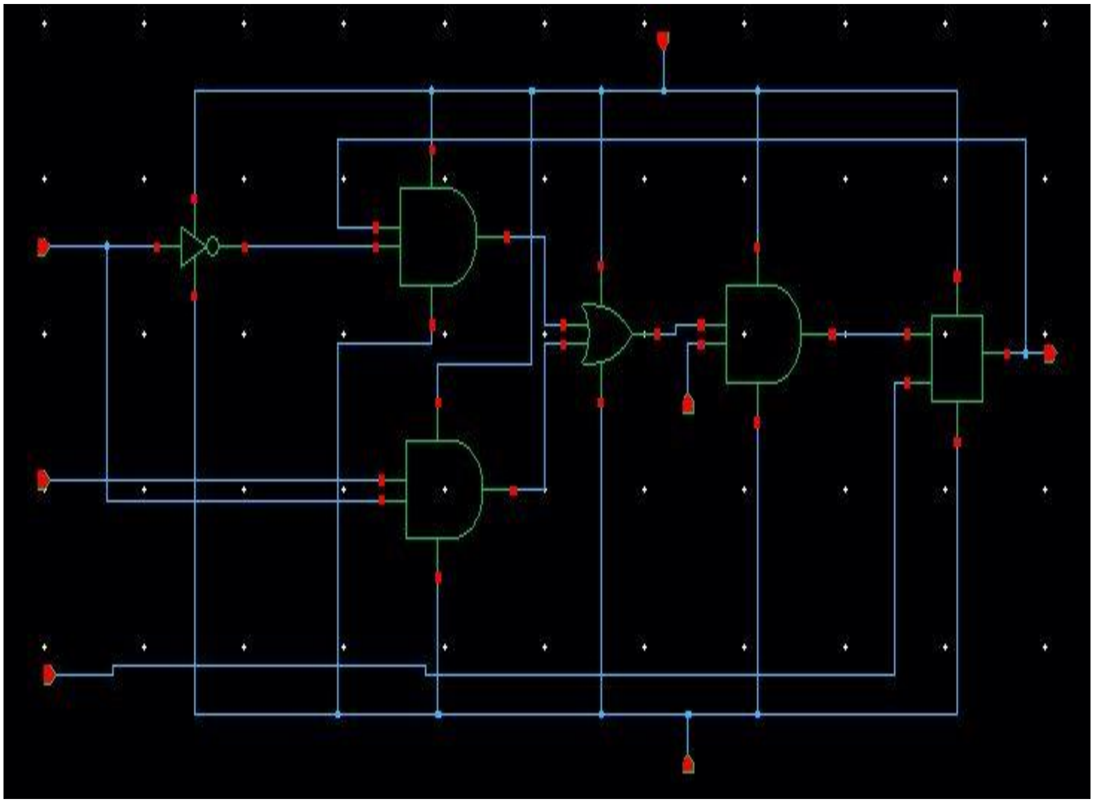


Fig 5.21: 1-bit register in cadence virtuoso

5.7 2’s complement circuit

The 2’s complement circuit is a small but very important component in the MAC architecture. It plays a major role in the addition of negative numbers. This circuit will convert the negative numbers into its 2’s complement form. The process starts by converting all the bits to their complement form by using the inverters and then a 1 is added at the LSB with the help of a Half adder. The output from the series of half adders gives the 2’s complemented form of the given negative number as shown in figure 5.22. In our architecture, we have used different sizes of 2’s complement circuits like 4-bit, 5-bit, and 6-bit, etc.,

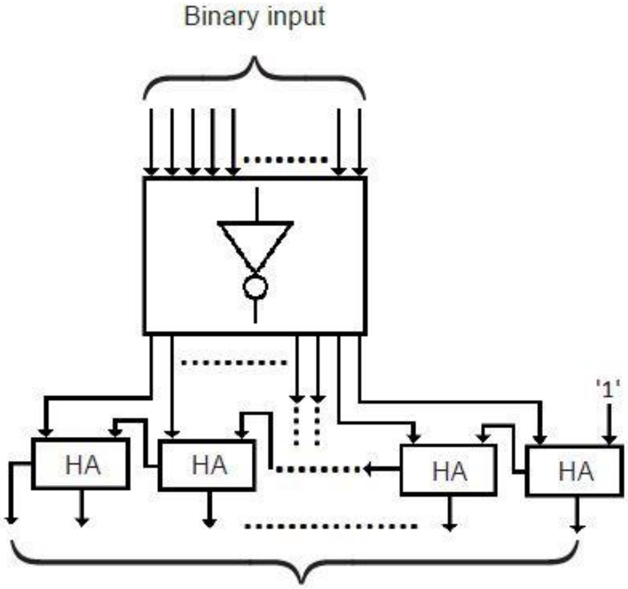


Fig 5.22. 2’s complement block

5.8 Multiplexer

A multiplexer is a component that gives a single output through the circuit from the circuit of multiple inputs. The inputs are selected based on the selection lines. It selects one output from 2^n inputs where n is the number of selection lines. In this MAC architecture, we have used different sizes of multiplexers like 2*1, 4*1, and 32*1. The basic 2*1 multiplexer is shown in figure 5.23.

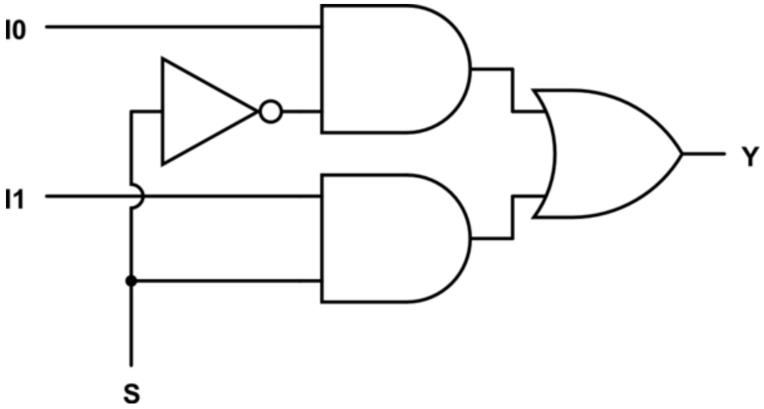


Fig 5.23: 2*1 multiplexer

5.9 Half adder, full adder

Half and Full adders are used in many parts of the MAC architecture like the exponential adder, multiplier, and exponential comparator circuit, etc., The half adder will produce the sum and carry by taking 2 inputs(a,b) whereas a full adder will produce sum carry by taking 3 inputs(a,b, ci). The basic half adder and full adder circuits are given in Figures 5.24. and 5.25

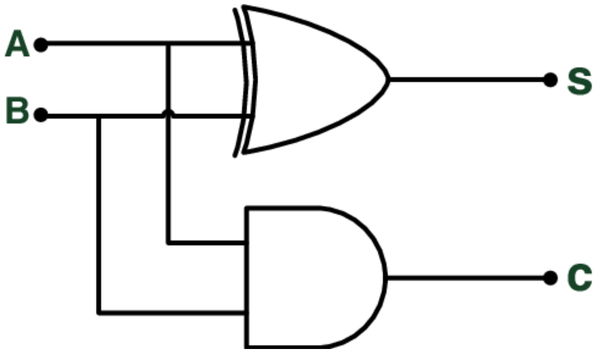


Fig 5.24. Half Adder

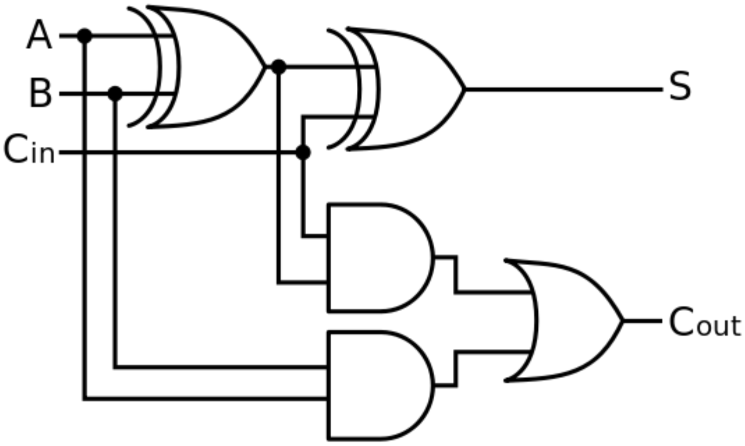


Fig 5.25. Full Adder

5.10 PED latch block

The number of blocks used in the SFMAC are high and the individual blocks delays are also different which can result in the wrong output. To avoid this situation, PED latch block is used. All the functional blocks are connected to positive edge detector (PED) block. A common clock is given as input to all the PED blocks which will ensure that the block will give output only for the positive edge of the clock. The block diagram for the PED latch block is given in fig 5.26.

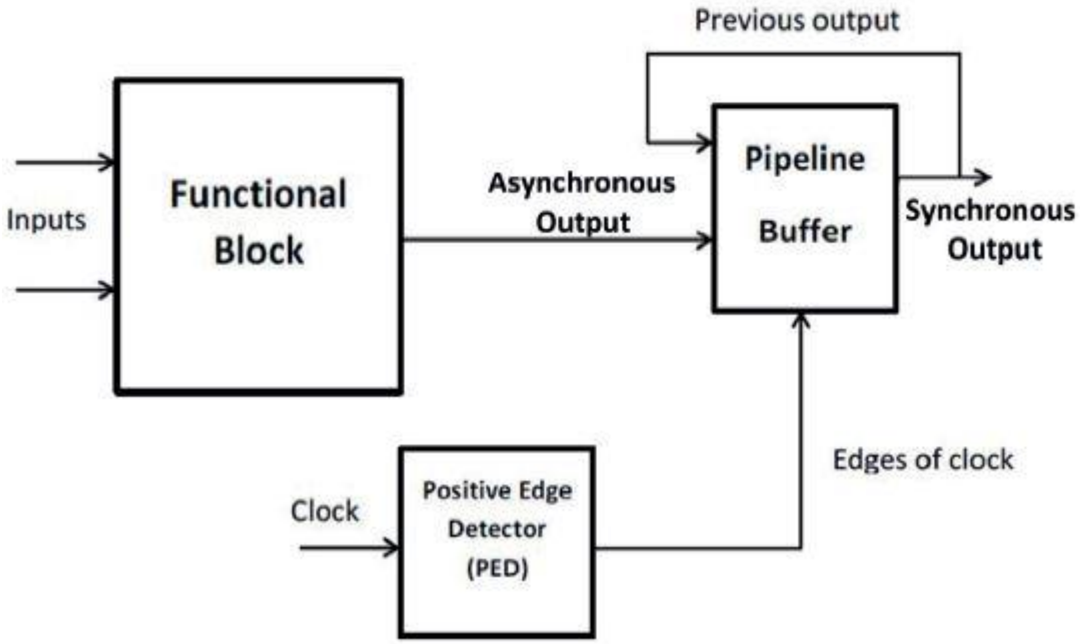


Figure 5.26. PED latch block.

The SFMAC architecture that is designed by using the Exponential Adder, ECC, Exponential shifter, registers, multiplexers is given in the below figure 5.27.

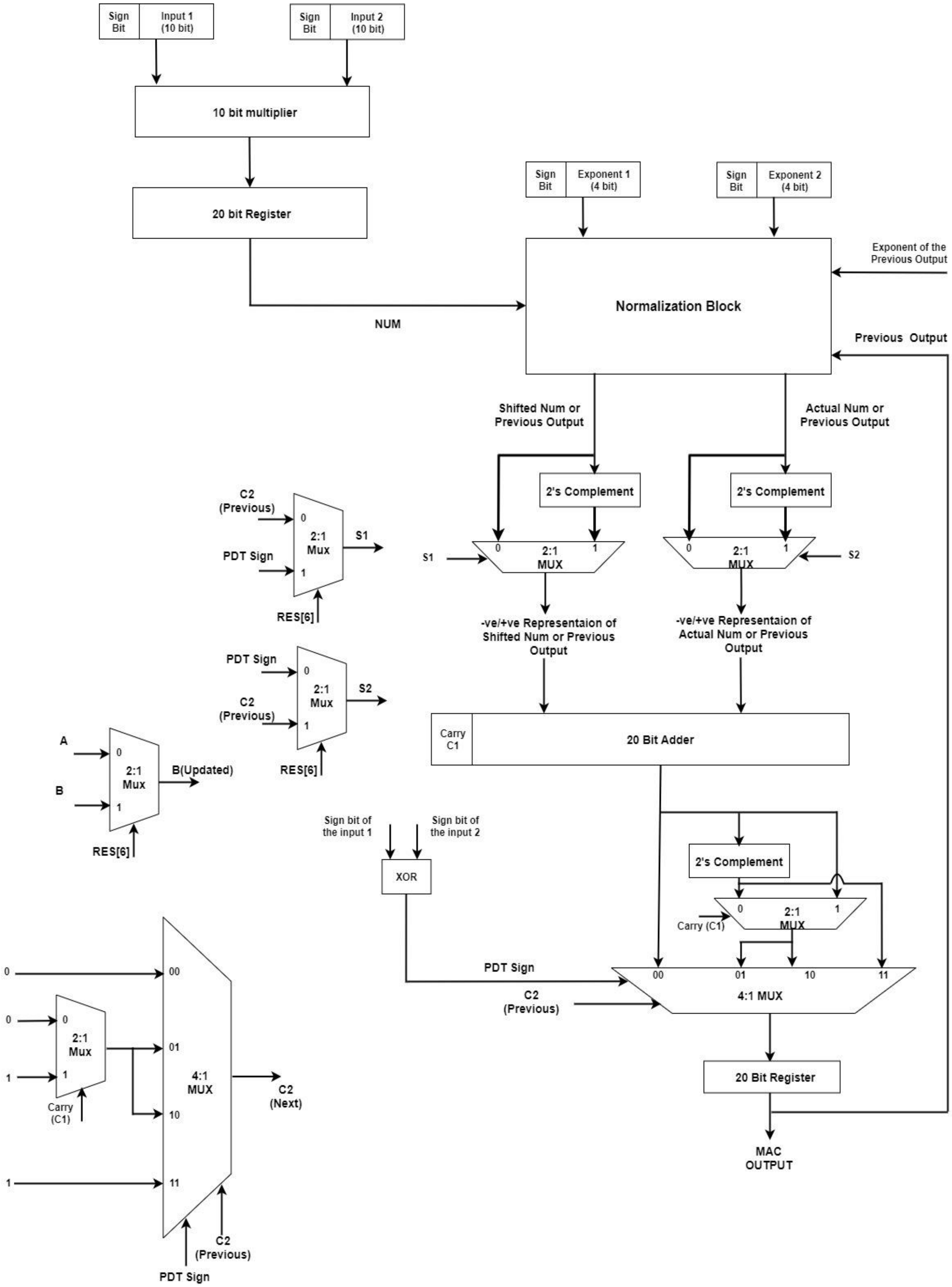


Fig 5.27. SFMAC Architecture.

As shown in the SFMAC architecture, the signed inputs are given to the 10-bit multiplier and the multiplied 20-bit output is stored in a 20-bit Register. The exponent part of the input is given to the Normalization block where there are three blocks in it. The exponents are first given to the Exponential part which will add the exponents and its output is given to the Exponential comparator circuit which will compare the present input from the 10-bit multiplier and the previous input and the output is given to the Exponential shifter. The Exponential shifter will shift the present input or the previous output based on the output from the ECC. The output from the Normalization block is given to the 20-bit adder through the help of 2’s complement and 2*1 multiplexer circuits as shown in the figure 5.27. The 20-bit adder output is stored in a Register and is used for the next cycle of inputs.

The designed SFMAC architecture in Cadence virtuoso is shown in the figure 5.28.

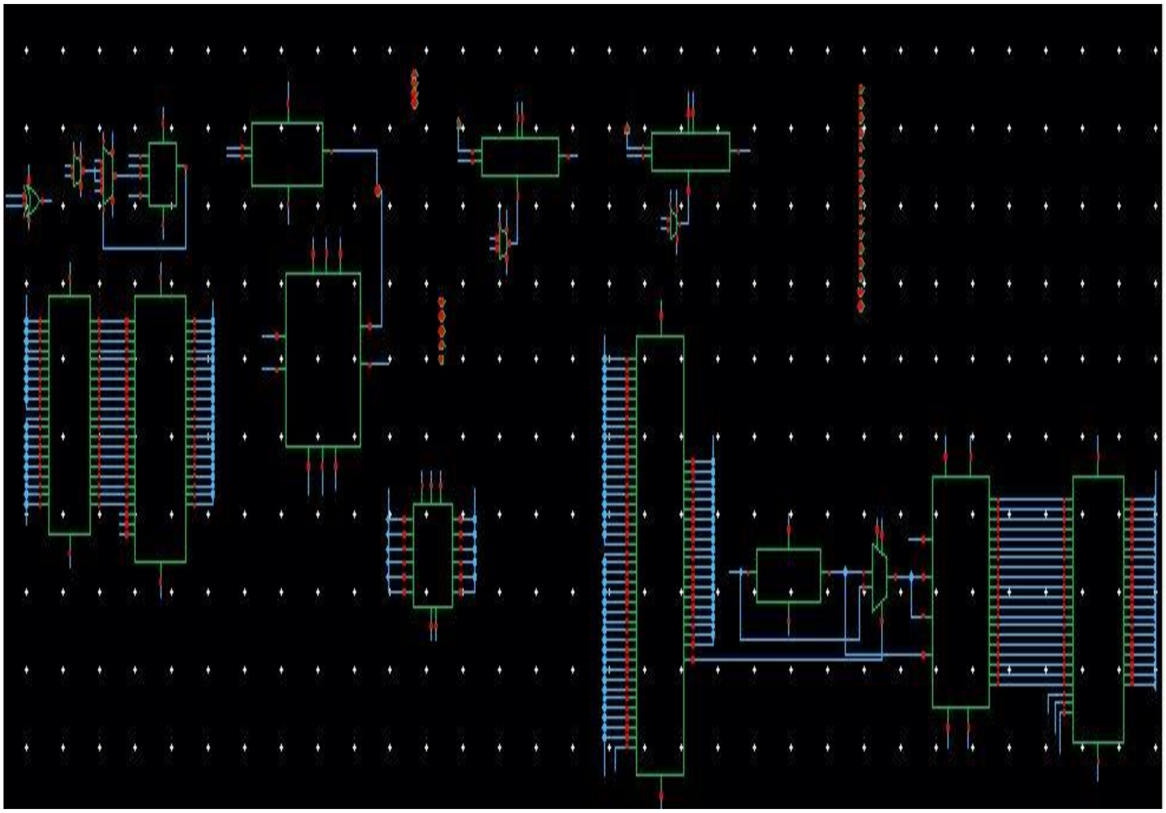


Fig 5.28. SFMAC Architecture in Cadence virtuoso.

CHAPTER 6

Summary and Conclusions

The basic circuits in the Half- Precision MAC architecture like Exponential adder, Exponential comparator, Exponential shifter, multiplier, adder, register are designed and analyzed using the cadence virtuoso. The design is done gpdk 90nm technology node. The Half-precision designed with these components will give a highly efficient architecture. The use floating-point in the architecture will give high precision at the cost of more power and delay. With the help of pipelining, the power can be reduced further. The advanced Universal compressor multiplier more efficient in providing the multiplications for the designed MAC architecture. The standard used for the Floating-point numbers ‘IEEE-754’ when included in the architecture, the MAC will be much more efficient and can be used universally. Comparisons can be done with other available technology files like gpdk 45nm, tsmc 130nm and finfet technologies. The Final obtained MAC architecture with Integrated IEEE 754 can be used in all the MAC applications like Digital signal processing, digital image processing, multimedia systems, and so on.

CHAPTER 7

References

References

[1] Shanthala S, Kulkarni. S.Y.,”VLSI Design and Implementation of Low Power MAC Unit with Block Enabling Technique,” *European Journal of Scientific Research* ISSN 1450-216X.

[2] Sen, Avisek, Partha Mitra, and Debarshi Datta. "Low power mac unit for DSP processor." *International Journal of Recent Technology and Engineering (IJRTE)* 1.6 (2013): 93-95.

[3] Israel Koren, Computer Arithmetic Algorithms, A K Peters, second edition, 2002.

[4] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw-Hill, 1984.

[5] S. J. Jou, C. Y. Chen, E. C. Yang and C.C.Su, “A pipeline Multiplier-Accumulator using a high speed, low power static and dynamic full adder design”, *IEEE custom Integrated circuit conference*, 1995, pp. 593-596

[6] Saravanan, R., P. Balaji, and R. Prabu. "Design of 16-bit floating point multiply and accumulate unit." *IJMTES Int. J. Mod. Trends Eng. Sci* 3.01 (2015).

[7] Mehta, Sonali, Balwinder Singh, and Dilip Kumar. "Performance Analysis of Floating Point MAC Unit." *International Journal of Computer Applications* 78.1 (2013).

[8] Jyoti Singh Chouhan, Nitin Jain. “Fused floating point mac (multiply and add) unit with configurable architecture” Journal of Scientific Research in Allied Sciences. ISSN No. 2455-5800 (2016).

[9] L. A. Tawalbeh, “Radix-4 ASIC Design of a Scalable Montgomery Modular Multiplier using Encoding Techniques,” M.S. Thesis, Oregon State University, USA, October 2002.

[10] John L. Hennessy and David A. Patterson. Computer Architecture A Quantitative Approach, Second Edition. Morgan Kaufmann, 1996.

[11] Deepika Setia, Charu Madhu, “Novel Architecture of High Speed Parallel MAC using Carry Select Adder”, *International Journal of Computer Applications* (0975 – 8887) Volume 74– No.1, July 2013.

[12] N. J. Babu and R. Sarma, "A novel low power and high speed Multiply-accumulate (MAC) unit design for floating-point numbers," *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, Chennai, 2015, pp. 411-417, doi: 10.1109/ICSTM.2015.7225452.

[13] M. J. Flynn, S. F. Oberman, AdvancedComputer Arithmatic Design. John Wiley & Sons, Inc, 2001.

[14] West and Harris, CMOS VLSI Design: a circuits and systems perspective, Addison-Wesley Publishing Company,3rd ed

[15] R. P. Rao, N. D. Rao, K. Naveen and P. Ramya, "implementation of the standard floating-point mac using ieee 754 floating point adder", *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, 2018, pp. 717-722.

[16] Saokar, Sandesh S., R. M. Banakar, and Saroja Siddamal. "High speed signed multiplier for digital signal processing applications." *2012 IEEE International Conference on Signal Processing, Computing and Control*. IEEE, 2012.

[17] A. Goldovsky and et al., “Design and Implementation of a 16 by 16 Low-Power Two’s Complement Multiplier”. *In IEEE International Symposium on Circuits and Systems*, 5, pp 345–348, 2000.

[18] Al-Ashrafy, Mohamed, Ashraf Salem, and Wagdy Anis. "An efficient implementation of floating point multiplier." *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*. IEEE, 2011.

[19] Kahan, William. "IEEE standard 754 for binary floating-point arithmetic." *Lecture Notes on the Status of IEEE* 754.94720-1776 (1996): 11.

[20] Yadagiri Karri, Prof. Rajesh Misra. “Implementation of 32 Bit Floating Point MAC Unit to Feed Weighted Inputs to Neural Networks”. *International Journal of Research and Scientific Innovation.* Volume II, Issue IV, ISSN 2321 – 2705 (2015).

[21] Grabinski, Wladyslaw, Bart Nauwelaers, and Dominique Schreurs, eds. *Transistor level modeling for analog/RF IC design*. The Netherlands: Springer, 2006

[22] Basiri M, Mohamed Asan, and Noor Mahammad Sk. "An efficient hardware-based higher radix floating point MAC design." *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 20.1 (2014): 1-25.

[23] Dhananjaya, A., and Dr Deepali Koppad. "Design of high speed floating point MAC using Vedic multiplier and parallel prefix adder": *International Journal of Engineering Research & Technology (IJERT)* Vol. 2 Issue 6, june-2012." (2013): 2278-0181.

[24] Zhang, Hao, Hyuk Jae Lee, and Seok-Bum Ko. "Efficient fixed/floating-point merged mixed-precision multiply-accumulate unit for deep learning processors." *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018.

[25] Dhanabal, R., et al. "Implementation of floating point MAC using residue number system." *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. IEEE, 2014.

[26] Karthikkumar, M., D. Manoranjitham, and K. Praveenkumar. "Implementation of Efficient 16-Bit MAC Using Modified Booth Algorithm and Different Adders." *International Journal of Scientific and Research Publications* 4.3 (2014).

[27] S. S., C. Raj and S. Y. Kulkarni, "Design and VLSI Implementation of Pipelined Multiply Accumulate Unit," *2009 Second International Conference on Emerging Trends in Engineering & Technology*, Nagpur, 2009, pp. 381-386, doi: 10.1109/ICETET.2009.72.

[28] P. Zicari, S. Perri, P. Corsonello and G. Cocorullo, "An optimized adder accumulator for high speed MACs," *2005 6th International Conference on ASIC*, Shanghai, 2005, pp. 757-760, doi: 10.1109/ICASIC.2005.1611425.

[29] Paidimarri, Arun, et al. "FPGA implementation of a single-precision floating-point multiply-accumulator with single-cycle accumulation." *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*. IEEE, 2009.

[30] O. T. -. Chen, Nan-Ying Shen and Chih-Chien Shen, "A low-power multiplication accumulation calculation unit for multimedia applications," *2003 IEEE International*

Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)., Hong Kong, 2003, pp. II-645, doi: 10.1109/ICASSP.2003.1202449.

[31] Das, Shishir Kumar, Aniruddha Kanhe, and R. H. Talwekar. "Design and Implementation of High-performance MAC unit." *International Journal of Scientific & Engineering Research* 4.6 (2013).

[32] Sarma, R., C. Bhargava, and S. Jain. "A MUX based signed-floating-point MAC architecture using UCM algorithm." *Bulletin of the Polish Academy of Sciences. Technical Sciences* 68.4 (2020).

[33] M. Liao, C. Su, C. Chang, and A. C. Wu, "A Carry-Select-Adder Optimization Technique for High-Performance Booth-Encoded Wallace-Tree Multipliers," *IEEE International Symposium on Circuits and Systems,* ISCAS 2002, 2002.

[34] R. D. Kshirsagar, E. V. Aishwarya, A. S. Vishwanath, and P. Jayakrishnan, "Implementation of Pipelined Booth Encoded Wallace Tree Multiplier Architecture," in *Proceedings of the International Conference on Communication and Green Computing Conservation of Energy (ICGCE),* Chennai, 2013.

[35] S. Khan, S. Kakde, and Y. Suryawanshi, "VLSI Implementation of Reduced Complexity Wallace Multiplier Using Energy Efficient CMOS Full Adder," in *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research,* 2013.

[36] T. Y. Kuo and J. S. Wang, "A Low-Voltage Latch-Adder Based Tree Multiplier," in *Proceedings of the IEEE International Symposium on Circuits and Systems,* Seattle, WA, 2008.

[37] M. Liao, C. Su, C. Chang, and A. C. Wu, "A Carry-Select-Adder Optimization Technique for High-Performance Booth-Encoded Wallace-Tree Multipliers," *IEEE International Symposium on Circuits and Systems,* ISCAS 2002, 2002.

[38] X. V. Luu, T. T. Hoang, T. T. Bui, and A. V. Dinh-Duc, "A High-speed Unsigned 32-bit Multiplier Based on Booth encoder and Wallace-tree Modifications," in

Proceedings of the International Conference on Advanced Technologies for Communications (ATC'14), 2014.

[39] C. Paradhasaradhi, M. Prashanthi, and N. Vivek, "Modified Wallace Tree Multiplier using Efficient Square-Root Carry Select Adder," in *Proceedings of the International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE),* Coimbatore, 2014.

[40] M. J. Rao and S. Dubey, “A High Speed and Area Efficient Booth Recoded Wallace Tree Multiplier for fast Arithmetic Circuits,” in *Proceedings of the Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PRIMEASIA),* BITS Pilani, Hyderabad, 2012.

[41] R. Sarma, C. Bhargava, and S. Jain, “UCM: A Novel Approach for Delay Optimization”, *Int J Performability Eng* 15(4),1190–1198 (2019).