FPGA Implementation of Single Precision Floating Point Multiplier using High Speed Compressors

Sunil Kumar Mishra, Vishakha Nandanwar, Eskinder Anteneh Ayele, S.B. Dhok

Abstract- Floating point multiplier is one of the vital concerns in every digital system. In this paper, the concepts of High speed compressors are used for the implementation of a High speed single precision binary Floating point multiplier by using IEEE 754 standard. Since compressors are special kind of adder which is capable to add more number of bits at a time, the use of these compressors makes the multiplier faster as compared to the conventional multiplier. For Mantissa calculation, a 24×24 bit multiplier has been developed by using these compressors. Owing to these high speed compressors, the proposed multiplier obtains a maximum frequency of 1467.136MHz. It is implemented using Verilog HDL and it is targeted for Xilinx Virtex-5 FPGA.

Keywords- Compressors, Floating point multiplier, Mantissa, IEEE754 standard, Verilog HDL.

I. INTRODUCTION

Floating point Multipliers are important components for many high performance digital systems such as FIR/IIR filters, microprocessors and digital signal processors. In this paper single precision floating point multiplier is implemented using different high speed compressors [1]. Introduction of high speed compressors leads to increase the speed of multiplication. Inputs and output of multiplier are single precision floating point number based on IEEE 754 standard [6].Single precision floating point representation is given in fig1. The single precision floating point number is of 32 bit. Starting from MSB it has a one bit sign (S), an eight bit exponent (E), and a twenty three bit mantissa (M) [6].

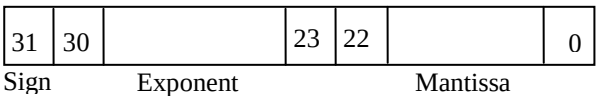


Fig1: Single Precision Floating Point Representation

Floating point is used to represent numbers and do arithmetic in computing machines, ranging from simple calculators to computers. In general, floating point representations are slower than fixed-point representations. Good quality about floating point is that they can handle a larger range of numbers [5].

Manuscript Received May 2014

Sunil Kumar Mishra, Department of Electronics Engineering, Visvesvaraya National Institute of Technology, Nagpur-440010, India.
Vishakha Nandanwar, Department of Electronics Engineering, Visvesvaraya National Institute of Technology, Nagpur-440010, India.
Eskinder Anteneh Ayele, Department of Electronics Engineering, Visvesvaraya National Institute of Technology, Nagpur-440010, India.
Dr. S. B. Dhok, Department of Electronics Engineering, Visvesvaraya National Institute of Technology, Nagpur-440010, India.

Expansion of the exponent component in floating point leads to achieve this greater range. Floating point multiplication is almost similar to integer multiplication. Because floating-point numbers are stored in sign-magnitude form, the multiplier needs to deal with unsigned integer numbers and normalization. Higher order multiplier requires a large number of adders for partial product addition. In this paper the numbers of adders are reduced by introducing High speed compressors logic. Compressors like 5-3, 6-3, 7-3, 8-4, 15-4 and so on are implemented for partial product addition [1] [2]. Binary counter property is merged along with compressor property for the implementation of higher order compressors [1]. Generally, compressors do the simple operation of addition that adds more number of bits at a time.

The paper is organized as follows. Section II presents the floating point multiplier algorithm. Section III contains the theory of high speed compressors. Section IV describes details of the proposed architecture and its implementation. Experimental results and Comparison are given in Section V. Finally, Section VI concludes the paper.

II. FLOATING POINT MULTIPLIER ALGORITHM

According to IEEE754 standard, the representation of a 32 bit binary floating point number is consists of sign, exponent and mantissa component. During calculation of floating point multiplication different operations are performed on each component. The detail algorithm is described as below [3], [4] and [5].

1. Calculation of the sign bit; i.e. SA XOR SB.
2. Exponent is calculated by adding the exponent of EA and EB. After that, bias the addition by 127 to get the final exponent. i.e. EA+ EB-127.
3. Add 1 before the mantissa bit to make the 23 bit into 24 bit after that multiplies together the 24 bit to get 48 bit result.
4. Normalizing the result, to get the required 23bit mantissa for the final result.
5. Combine the calculated sign, exponent and mantissa components to get the desired multiplication result.

A. Multiplication Operation

The above algorithm can be better explained by considering the multiplication of two floating point numbers A and B as an example. Let's consider A= -17.85, B=12.57. The IEEE754 standard format representation of the two operands A and B are given in Table-1 in view of single precision format. Here MSB of the operand shows the sign bit, the next 8 bit represent exponent, and the mantissa is

represented by rest 23bit. The exponent is expressed in excess 127 bit.

Table-1: IEEE754 Standard Representation of the Two Operands

Operand	A	B
Decimal Value	-17.85	12.57
Sign	1	0
Exponent	10000011	10000010
Mantissa	00011101100110011001101	10010010001111010111000

By computing the XOR operation on sign bit of operands A and B, we obtain the sign bit of output. Here sign bit of output is S_A XOR S_B. In this case sign bit get hold of is "1". For resultant exponent, the addition operation is used to add exponent of both the operands. In this implementation 8 bit ripple carry adder is used for addition of E_A and E_B. After addition the result is again biased to decimal 127. For this purpose 127 is subtracted from the addition result of E_A and E_B. So the final resultant is $E_R=E_A+E_B-127$. Where, E_A and E_B are the exponents of operand A and B respectively. E_R is the final resultant exponent. For this typical example, E_R =10000110. In this paper the subtraction is implemented using two's complement method [9].

Mantissa multiplication is done by using 24x24 bit binary multiplier. Further the binary multiplier is implemented by using high speed compressors. Before binary multiplication the 23 bit mantissa is normalized to 24 bit by adding 1 at MSB. In this illustration the normalized mantissas are

M_A=100011101100110011001101
M_B=110010010001111010111000

The above two 24 bit normalized mantissa are multiplied by using high speed compressors based binary multiplier and the acquired result will be of 48 bit as

011100000010111111101111100110000011100101011000

Now normalizing the resultant 48 bit into 23 bit mantissa is done by simply ignoring the MSB and hence the final mantissa, R_M=11000000101111111011111.

After combining the three results of sign, exponent and mantissa components, the final obtained result is shown in Table 2.

Table-2: Results of Sign, Exponent and Mantissa Component

Decimal Value	Sign	Exponent	Mantissa
224.3748	1	10000110	11000000101111111011111

III. HIGH SPEED COMPRESSORS

In binary multiplication a large number of partial product additions are carried out. In conventional multiplier full adders are used for partial product addition [3]. With full adder maximum of 3 inputs can be added at a time. Accordingly to add all the partial products large numbers of full adders are required. Hence, to reduce the number of adders in this implementation High speed compressors are introduced. Different compressors are developed based upon the concept of binary counter property [1]. In 5-3 compressor, maximum of five inputs can be added at a time wherein the output is a three bit number [1]. Table-3 shows the counter property of 5-3 compressor.

Table-3: Counter Property of 5-3 Compressor

Input Condition	Out3	Out2	Out1	Decimal Value
All inputs are zero	0	0	0	0
Any one input is one	0	0	1	1
Any two input are one	0	1	0	2
Any three input are one	0	1	1	3
Any four input are one	1	0	0	4
Any five input are one	1	0	1	5

A. Architecture of Different Compressors

Some basic different compressors architecture are designed and discussed by using the same logic in [1]. Fig. 2 shows the block diagram of 4-3 compressor which is designed by using one full adder and two half adder. 5-3 compressor is implemented by using two full adders and one half adders as shown in Fig.3.

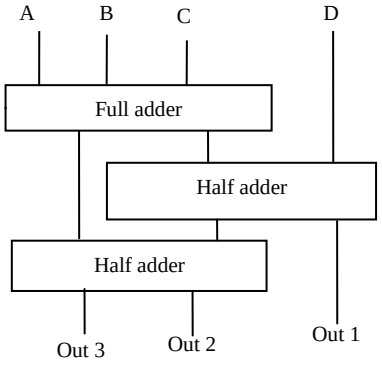


Fig-2: Structural Design of 4-3 Compressor

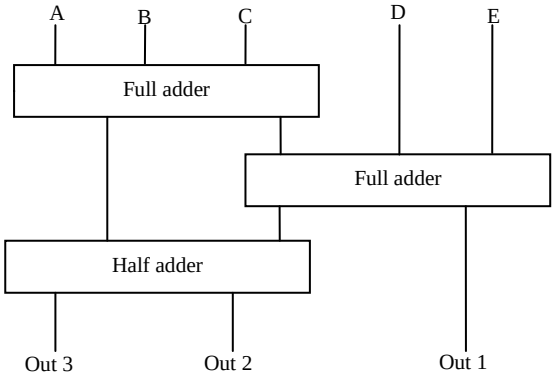


Fig-3: Structural Design of 5-3 Compressor

For the design of 6-3 compressor, two full adders are used. Fig.4 shows the block diagram of 6-3 compressor [1]. For the parallel addition purpose ripple carry adder as seen in Fig.5 is used. Similarly for the design of 7-3 compressor in Fig.6, one full adder and one 4-3 compressor are used. Its parallel addition unit is given in fig.7.

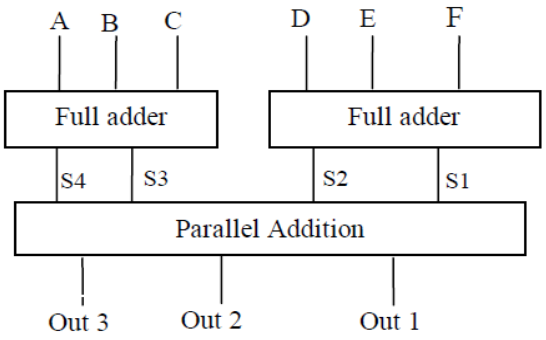


Fig-4: Structural Design of 6-3 Compressor

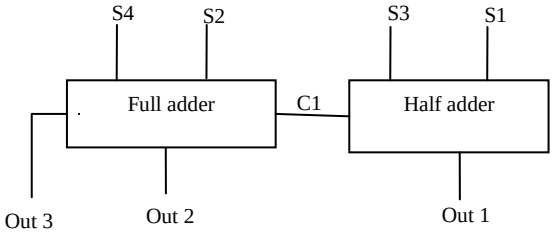


Fig-5: Parallel Addition Unit for 6-3 Compressor

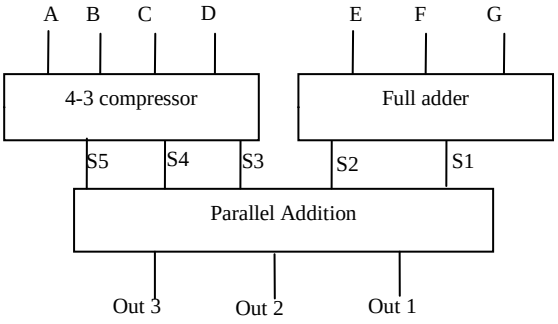


Fig-6: Structural Design of 7-3 Compressor

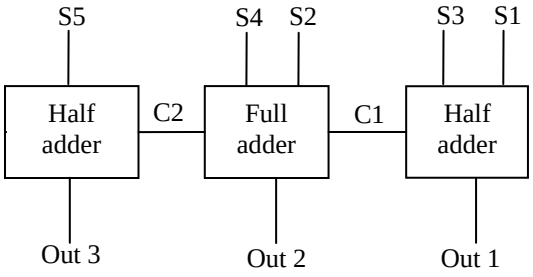


Fig-7: Parallel Addition Unit for 7-3 Compressor

For the implementation of 15-4 compressor five full adders, two compressors, and parallel adder unit are used [2]. Fig.8 shows the block diagram of 15-4 compressor and its optimized parallel adder is shown in fig.9

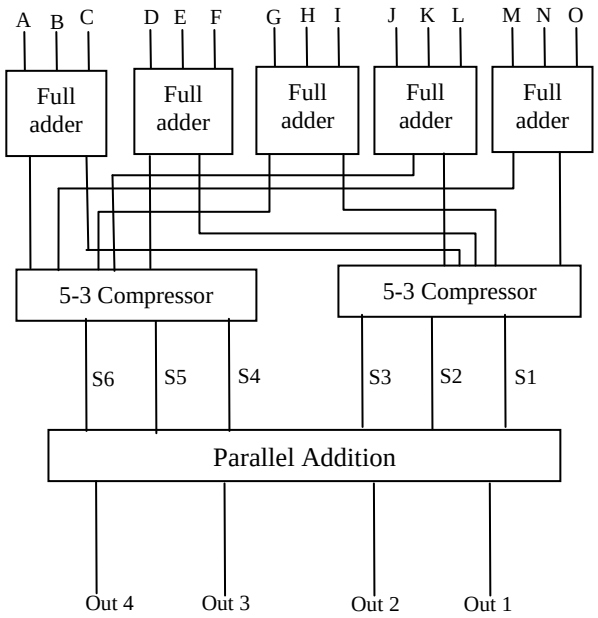


Fig-8: Architecture of 15-4 Compressor

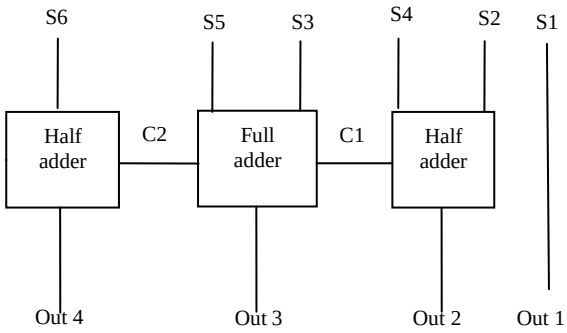


Fig-9: Parallel Adder Unit for 15-4 Compressor

By using the concept of adders all the compressors are implemented. These compressors are used in the calculation of partial product addition in different stages of binary multiplication.

IV. PROPOSED ARCHITECTURE OF FLOATING POINT MULTIPLIER

The proposed architecture for Single Precision Floating Point Multiplier using High Speed Compressors is given in Fig.10.

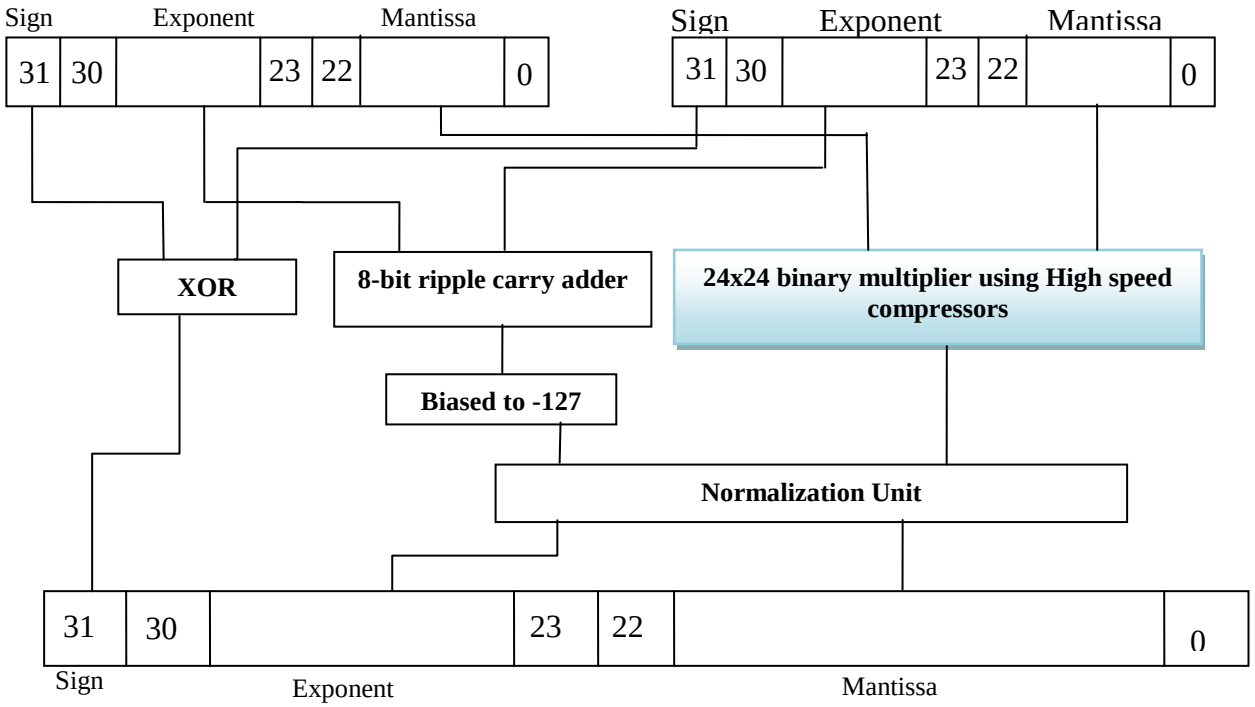
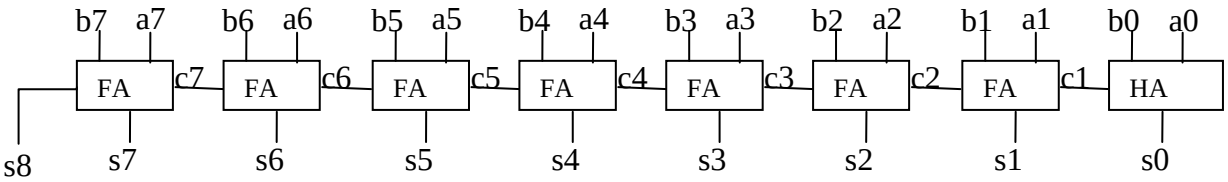


Fig-10: Proposed Architecture of Single Precision Floating

Point Multiplier



HA- Half adder

FA- Half adder

Fig-11: Eight-Bit Ripple Carry Adder

From the calculation perspective whole floating point multiplication is divided into four sections.

A. Sign section
B. Exponent section
C. Mantissa section
D. Normalization section

A. Sign Section

In sign section, the sign bit for the final result is calculated. The sign bit is calculated by performing XOR operation on the sign bits of the two operands. The truth table for the XOR operation is given in Table-4.

Table-4: Sign Bit Operation

E_A	E_B	Sign
0	0	0
0	1	1
1	0	1
1	1	0

B. Exponent Section

In this section the two operands are added directly by using 8-bit ripple carry adder. Fig.11 shows the 8-bit ripple carry adder. After addition, biasing is required to get the final exponent. For biasing decimal value 127 is subtracted .Here subtraction is done by using 2's complement method [9].

C. Mantissa Section

Mantissa calculation is the most important part of the floating point multiplier. The whole performance of the floating point multiplier is depending upon the mantissa calculation section. In mantissa calculation 24x24 bit binary multiplier is required to do the multiplication of mantissas of two operands. In this paper binary multiplier is implemented by using high speed compressors, which reduces the delay. In single precision floating point number, mantissa is consists of 23 bit. So initially normalization is done in the mantissa by adding 1 at the MSB.

After normalization the mantissa of each operand became 24 bit. After binary multiplication of the two normalized 24 bit operands a 48 bit number will generate as a multiplication output. Again this 48 bit number is normalized to 23 bit to get the final mantissa.

In this paper 24 bit multiplier is proposed and implemented by using the concept of high speed compressors. After multiplication of 24x24 binary multiplier large numbers of partial products are obtained. Further those partial products are added by using different compressors at different stage. At first stage of partial product addition, no need of any addition because only one partial product is there. Which directly goes as it is as LSB. At the second stage one half adder is required because only two partial products are here. The sum bit taken as output for that stage and carry will go to the next stage. At third stage, partial products are increased to three and one carry from previous stage and hence a total four partial products are present. Addition of these four partial products is done by using 4-3 compressor. In the next stages the number of partial products goes on increasing. So, higher order compressors are required for addition. Similarly in the4th stage 7-3 compressor, 5thstage 8-4 compressor, 6th stage 9-4 compressor, and 7th stage 10-4 compressor are used. As the stage increases higher order compressors are used. At 24th stage, partial products are increased 28 including the carry from previous stages. For the partial addition of 24th stage 28-5 compressor is used. After 24th stage number partial products are decreasing .So for the addition of those partial products previously used compressors are used. This process of addition of partial product is better explained by considering a 4x4 binary multiplier as shown in Fig-12.

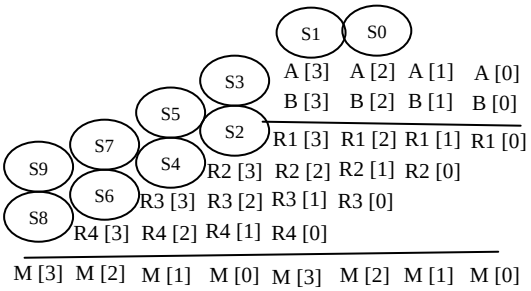


Fig-12: 4x4 Binary Multiplier

In the first stage of partial product addition only one partial product is at hand, R1 [0]. So it is directly taken as M [0]. For the second stage of partial products addition one half adder is used. Sum bit of half adder's result is taken as M [1] and carry is move to the next stage. In the 3rd stage, four partial products are present including the carry and one 4-3 compressors used in this stage. Similarly in 5th stage and 6thstage 5-3 and 4-3 compressors are used respectively for partial products addition. In 7th and 8th stage less number partial products are present so only one full adder and one half adder is required.

D. Normalization Section

In the Normalization section, normalization of exponent and mantissa are performed. According to the 47th bit (result of the 24x24 bit binary multiplier) normalization is done.

i. When 47th bit of 24X24 bit binary multiplier is binary one ,mantissa is normalized to 23 bit by taking 46th to 24th bit position number and exponent is increased by decimal value one.

ii. When 47th bit of 24X24 bit binary multiplier is binary zero, mantissa is normalized to 23 bit by taking 45th to 23th bit position number and there is no increment in the exponent value.

V. RESULT AND COMPARISON

A test bench program is written for the implemented single precision floating point multiplier and the results are verified. The code is written and executed by Xilinx 14.2 .The implemented design had been synthesized using synthesis XST tool and it is targeted on Vertex-5 FPGA, Device-XC5VLX20T, Package-FF323, speed-2.

The comparison of implemented single precision floating point multiplier with the other floating point multiplier has been shown in table-5.

Table-5: Area and Frequency Comparison between the Implemented Floating Point Multiplier with Others

Parameters	Implemented Floating Point Multiplier	Floating Point Multiplier by using Dadda Algorithm [3]	Conventional Floating Point Multiplier
LUT & FF Pair Used	1122	1146	1110
CLB Slices	866	1149	1112
Maximum Frequency (MHz)	1467.136	526.857	401.711

Table-6 and Table-7 shows the device utilization summary and power supply summary of implemented floating point multiplier respectively. In fig .13 RTL schematic of multiplier is shown.

Table-6: Device Utilization Summary

Slice Registers	46
Slice LUTs	1110
LUT Flip Flop pairs	1122
IOB	98

Table-7: Power Supply Summary

Total Power (W)	Dynamic Power (W)	Quiescent Power (W)
0.258	0.011	0.247

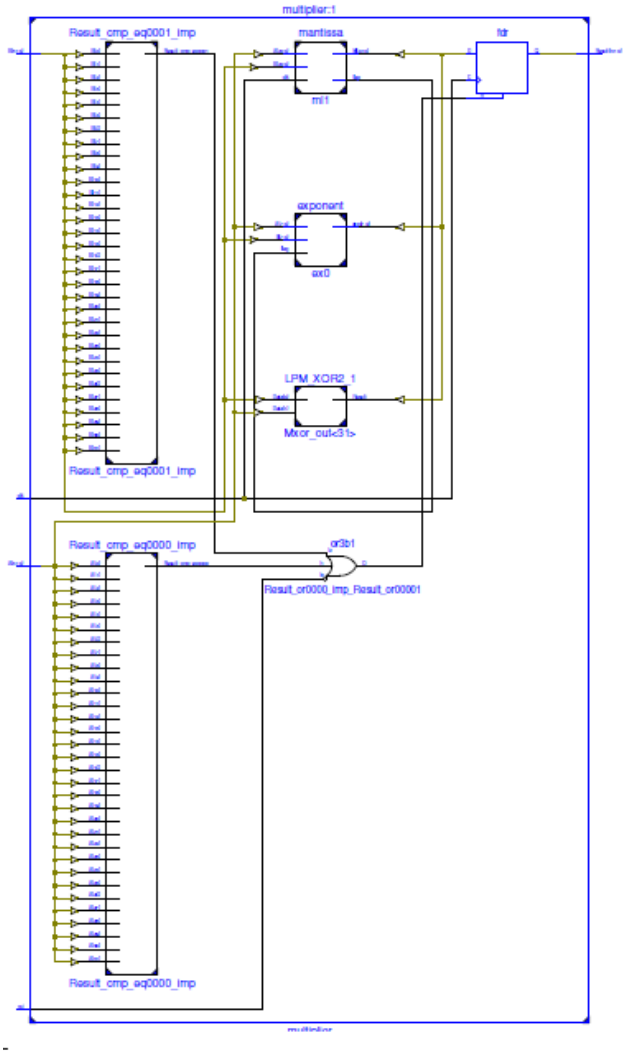


Fig-13: RTL Schematic of Multiplier

VI. CONCLUSION

In this paper, single precision floating point multiplier based on the IEEE-754 format is successfully implemented on FPGA. The modules are written in Verilog HDL to optimize implementation on FPGA. In this implementation the obtained maximum frequency is 2.78 times more than that of the multiplier implemented by using Dadda algorithm. Different compressors are used to speed up the multiplication process. Since the main idea behind this implementation is to increase the speed of the multiplier by reducing delay at every stage using the optimal compressors design, it gives the advantage of less delay in comparison to other method. The results obtained using the proposed algorithm and implementation is better not only in terms of speed but also in terms of hardware used.

REFERENCES

1. A. Dandapat, S. Ghosal, P. Sarkar, D. Mukhopadhyay, "A 1.2-ns16×16-Bit Binary Multiplier Using High Speed Compressors", International Journal of Electrical and Electronics Engineering, 2010.
2. Shubhajit Roy Chowdhury, Aritra Banerjee, Aniruddha Roy, Hiranmay Saha,"Design, Simulation and Testing of a High Speed Low Power 15-4 Compressor for High Speed Multiplication Applications", First International Conference on Emerging Trends in Engineering and Technology, 2008.
3. B. Jeevan, S. Narender, Dr C.V. Krishna Reddy, Dr K. Sivani,"A High Speed Binary Floating Point Multiplier Using Dadda Algorithm",IEEE,2013.
4. Loucas Louca, Todd A. Cook, William H. Johnson, "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs", IEEE,1996.
5. Shaifali, Sakshi, " FPGA Design of Pipelined 32-bit Floating Point Multiplier", International Journal of Computational Engineering & Management, Vol. 16, 5th September 2013.
6. IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.
7. Mohamed Al-Ashrafy, Ashraf Salem, Wagdy Anis, "An Efficient Implementation of Floating Point Multiplier", IEEE, 2008.
8. Guy Even, Silvia M. Mueller, Peter-Michael Seidel," A dual precision IEEE floating-point multiplier", INTEGRATION the VLSI journal, pp167-180, 2000.
9. M. Morris Mano, "Digital Design",3rd edition, Prentice Hall,2002

AUTHORS PROFILE

Mr. Sunil Kumar Mishra is 4th semester student of M.Tech in VLSI Design of Electronics Engineering department at Visvesvaraya National Institute of Technology, Nagpur, Maharashtra. He obtained his B.Tech degree from Biju Patnaik Univesity of Technology, Rourkela, Odisha. His research interests area are Digital Design & Signal Processing.

Miss. Vishakha Nandanwar is 4th semester student of M.Tech in VLSI Design of Electronics Engineering department at Visvesvaraya National Institute of Technology, Nagpur, Maharashtra. She obtained her B.Tech degree from Samrat Ashok Technological of Institute, Vidisha, Bhopal (M.P). Her research interest domain is Digital Image Processing.

Eskinder Anteneh Ayele received his B-Tech degree in electrical engineering from DEC, Ethiopia and M-Tech degree in Control and Instrumentation from IIT, Madras, India in 2001 and 2005 respectively. Currently, he is a PhD scholar in the Department of electronics engineering, VNIT, Nagpur, India. His area of interests is Signal and Image Processing, Electronic Instrumentation, and Control Systems.

Dr. S. B. Dhok is Associate Professor in Electronics Engineering Department at Visvesvaraya National Institute of Technology, Nagpur (INDIA). He has done his PhD from VNIT Nagpur. He is a member of IEEE society. He has published many research papers in national and international journals and conferences. His area of interest includes Signal Processing, Image Processing, Data Compression, Wireless Sensor Networks and VLSI Design.