International Journal of Latest Trends in Engineering and Technology
Vol.(8)Issue(3), pp.263-269
DOI: http://dx.doi.org/10.21172/1.83.040
e-ISSN:2278-621X

DESIGN AND IMPLEMENTATION OF SINGLE PRECISION FLOATING POINT MULTIPLIER USING VHDL ON SPARTAN 3

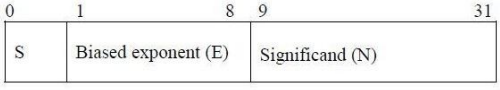
Pooja Hatwalne[1], Ameya Deshmukh[2], Tanmay Paliwal[3] and Krupal Lambat[4]

Abstract— **Floating-point arithmetic algorithms are highly useful for computations involving large dynamic range, high precision and ease of operation. Hence, they find lot of applications in the various fields for the requirements for high precision operation. This paper presents the design and implementation of single precision floating point multiplier using VHDL hardware description language. 8 bit Carry Look Ahead Adder is used for the purpose of exponent addition and 24 bit Vedic Multiplier based on Urdhva-Triyagbhyam Sutra in Vedic Mathematics is used for the purpose of Mantissa multiplication in the proposed floating point multiplier. The designs are synthesized and simulated in Xilinx ISE 14.7 targeted on xc3s50-5pq208 Spartan -3 device.**
Keywords— **single precision floating point multiplier, VHDL, carry look ahead adder, Vedic multiplier, Xilinx, Spartan-3.**

I. INTRODUCTION

Floating point multipliers have variety of applications in fields like finance, medical imaging, biometrics, high performance audio etc. and also Multipliers play a key role in microprocessors, DSP processors and various FIR filters. A performance of a system is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Since multiplication dominates the execution time of most DSP application so there is need of high speed multiplier. Hence increasing the speed and optimizing area of the multiplier is a major design issue.

This work deals with the design and implementation of optimized floating point multiplier. IEEE 754 single precision format is used to represent floating point numbers.



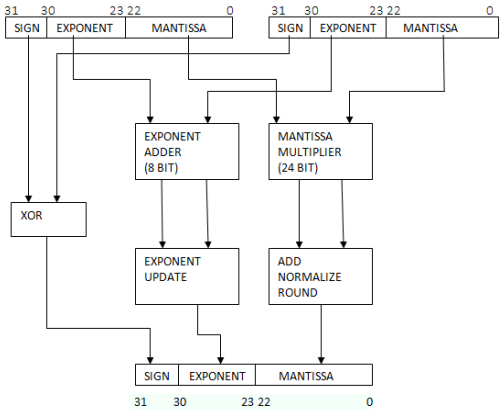
Single Precision = 32 bits

The general representation of floating point number is: **(-1) ^S h 1.F h 2^ (E-127),** where the range of E is -126 to +127.

II. IMPLEMENTATION OF FLOATING POINT MULTIPLIER

A. Block diagram:

[1] *Department of ECE, RCOEM Nagpur, MH*
[2] *Department of ECE, RCOEM Nagpur, MH*
[3] *Department of ECE, RCOEM Nagpur, MH*
[4] *Department of ECE, RCOEM Nagpur, MH*



B. Algorithm of floating point multiplication with example:

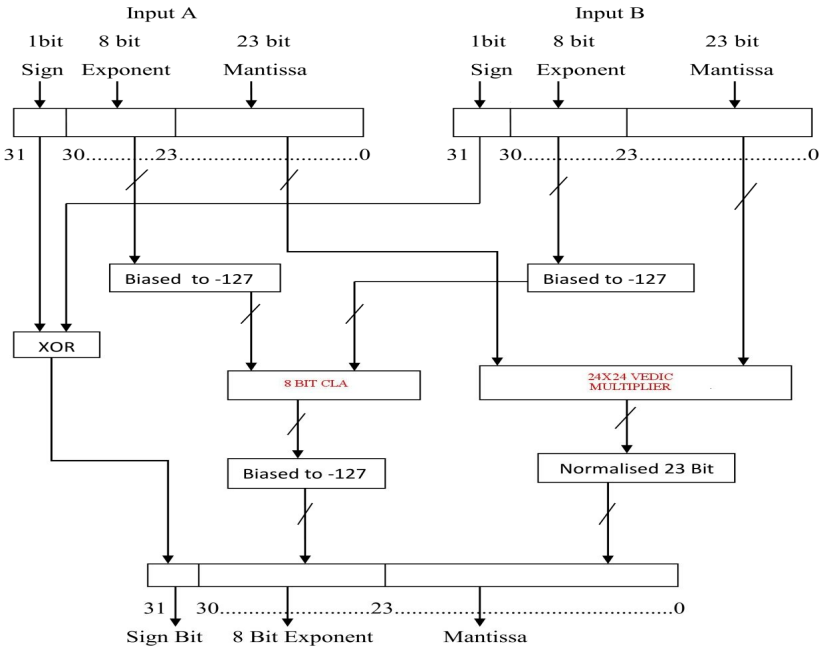
The example below explains the algorithm of floating point multiplier properly:
Multiply 189.1234 by -4.62 = -873.750108

First convert both of the floating point numbers to their equivalent IEEE754 formats:
• A = 0 10000110 01111010001111110010111 = 0x433d1f97
• B = 1 10000001 00100111101011100001010= 0xC093d70a

Step 1: Sign calculation:
The sign is calculated by xor-ing the sign bits s1 and s2 of individual floating point numbers
S =s1 XOR s2
= 0 XOR 1
S = 1.
Step 2: Exponent calculation:
for exponent calculation bias is added to individual exponents E1 and E2 and it is to be removed after adding exponents to obtain the final exponent
E_output = E1+E2-127
Here E1=10000110, E2 = 10000001
E_output = E1+E2 -127 =10000110 + 10000001 – 10000001 = 10001000
Step3: Mantissa calculation:
Mantissa is of 23 bits but an implicit one is added for normalization so it becomes of 24 bits
For mantissa calculation the, individual mantissa M1 and M2 are added to get final mantissa of 48 bits.
Here we get: M_output = 1.M1*1.M2 = 100101111001011000000100000000000110011100110

Step 4: Normalization:
In the output of the mantissa product the leading one comes in 46th position .As mentioned in the flow chart if the leading 1 comes in 46th position then consider the exponential part calculated in step 2 and take the bit position [45:23] which is calculated in step 3.
To get final output now we combine sign, exponent and normalized mantissa calculation:
1 10001000 10110100111000000000010 = -873.750108

C. Proposed Floating Point Multiplier:



III CARRY LOOK AHEAD ADDER

For the Purpose of carry Propagation, Carry look Ahead Adder constructs Partial Full Adder, Propagation and generation Carry block. It avoids Carry propagation through each adder. In order to implement Carry Look Ahead Adder, first implement Partial Full Adder and then carry logic using propagation and generation block.

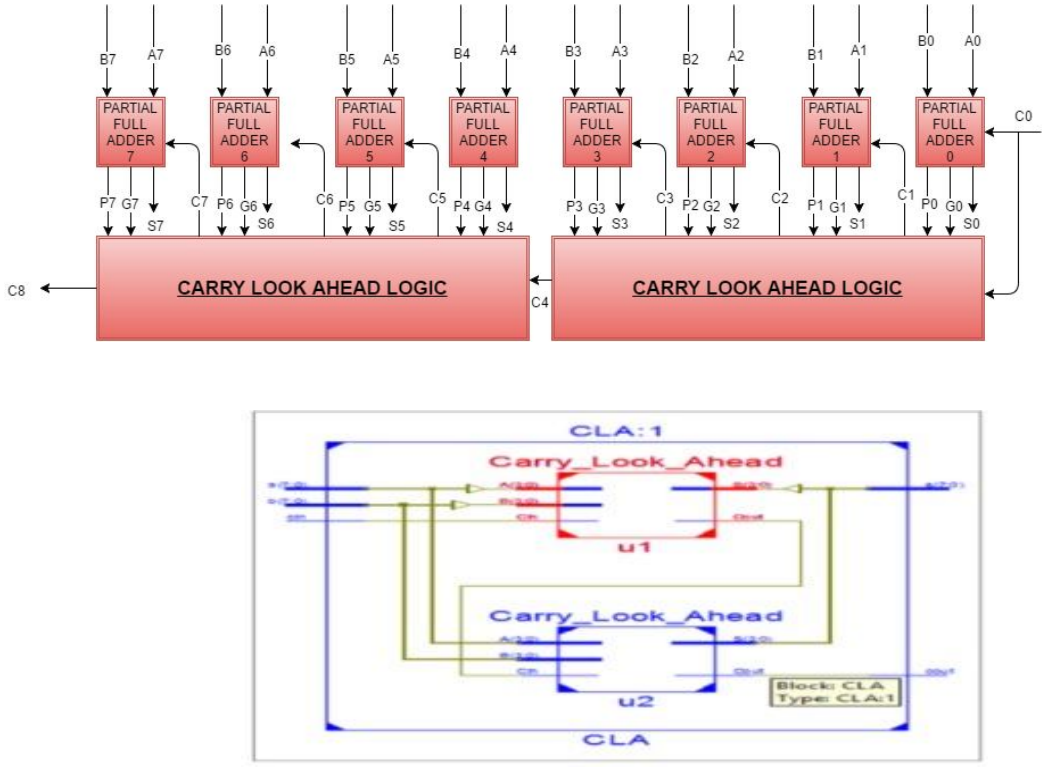


Fig 1. RTL VIew of Carry Look Ahead Adder

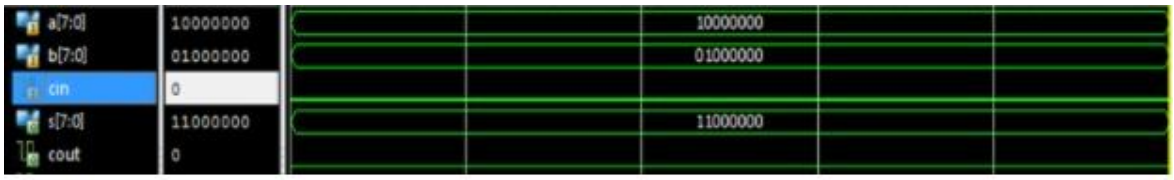
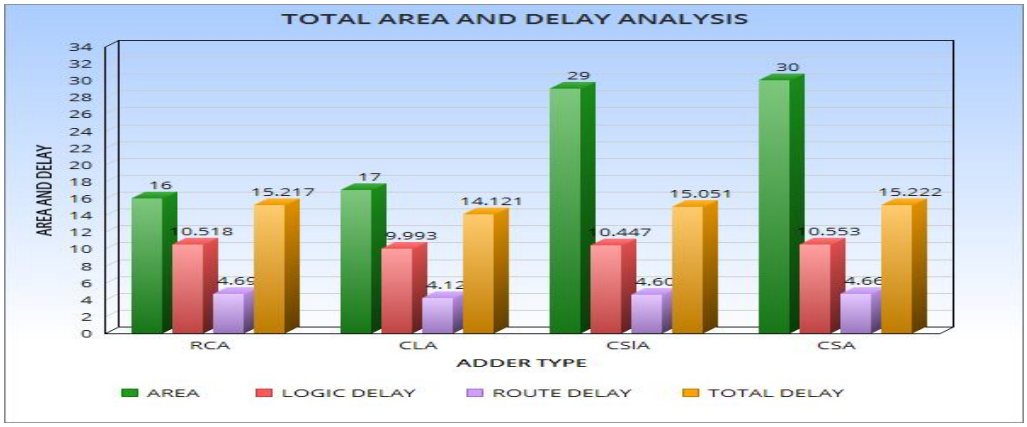


Fig 2. Simulation of Carry Look Ahead Adder

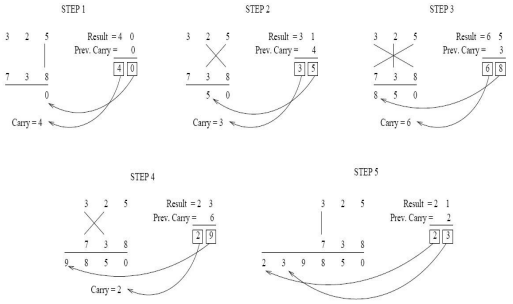
A. *Performance Analysis of Adders:*

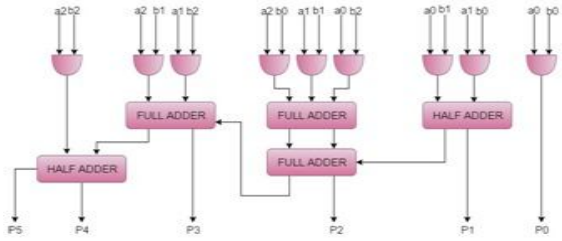
Comparison of various adders like ripple carry adder, carry look ahead adder,carry save adder and carry select adder was done on the basis of area i.e no of input LUTs and time delay. It was found that carry look ahead adder is the most efficient. Hence it is used for exponent calculation in the proposed floating point multiplier.



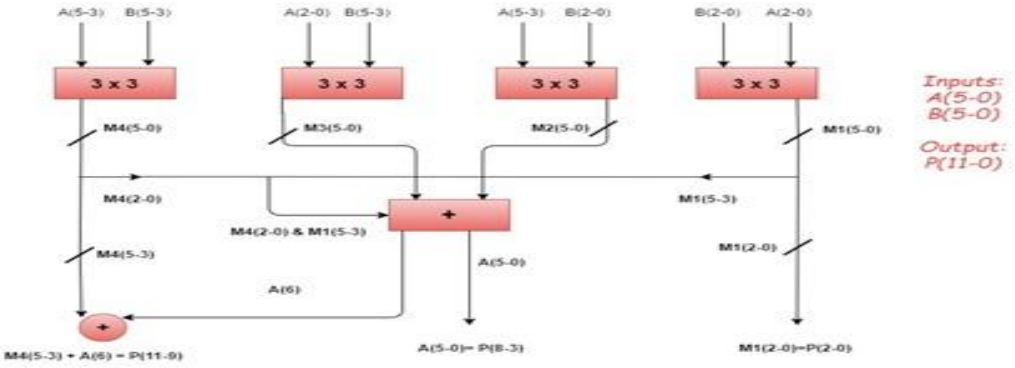
IV. VEDIC MULTIPLIER

In Vedic Mathematics, Urdhva-Tiryakbhyam Sutra is a multiplication algorithm which is applicable to all cases of multiplication. Thus the multiplier will require the same amount of time to calculate the product and hence is independent of the clock frequency. In this method the partial products are generated simultaneously which itself reduces delay and makes this method fast.





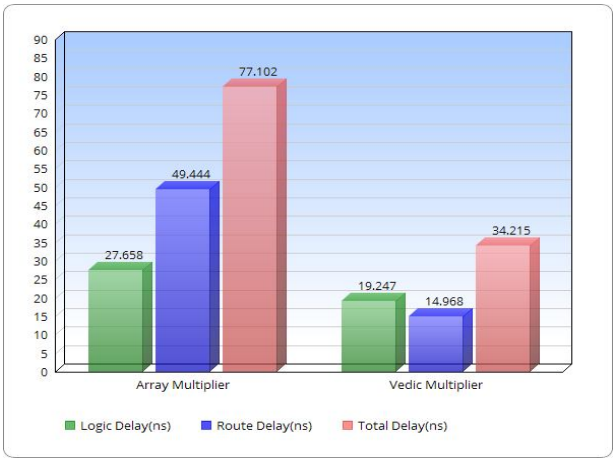
Hardware Implementation of 3x3 Block



Hardware Implementation of 6x6 from 3x3 Block

For preparing a 24x24 multiplier, initially 3x3 block is made(as shown in fig1 above).Using it, 6x6 block is generated(as shown in fig2 above)which is used to design a 12x12 block and then a 24x24 block is finally obtained using 12x12.

A. Delay Analysis of Multipliers:



Comparison of two multipliers namely array multiplier and vedic multiplier was done on the basis of area i.e no of input LUTs and time delay. It was found that vedic multiplier is the most efficient. Hence it is used for mantissa calculation in the proposed floating point multiplier.

V. SYNTHESIS, SIMULATION & TIMING PERFORMANCE OF FLOATING POINT MULTIPLIER ON SPARTAN 3

The proposed Floating Point Multiplier is synthesized in Xilinx ISE 14.7 and simulated in Xilinx Isim Simulator and the RTL Schematic and Simulation results are shown below:

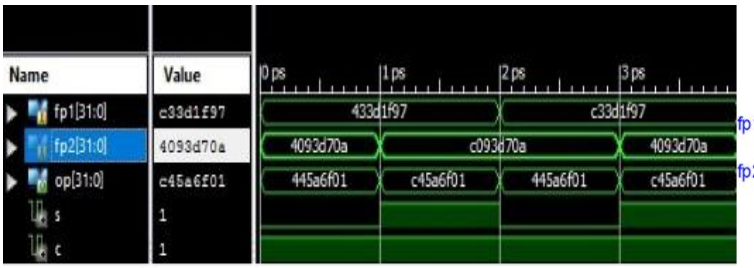


Fig 2. Simulation Result of Floating Point Multiplier

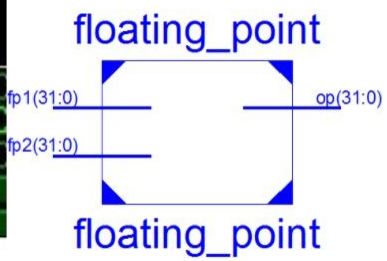


Fig 1. RTL Schematic of Floating Point Multiplier

```
    OBUF:I->O                 4.909          op_28_OBUF (op<28>)
    ----------------------------------------
    Total                    36.190ns (20.040ns logic, 16.150ns route)
                                      (55.4% logic, 44.6% route)
=========================================================================


Total REAL time to Xst completion: 23.00 secs
Total CPU time to Xst completion: 22.68 secs

-->

Total memory usage is 339640 kilobytes
```

Total Delay

Fig 3. Delay of Floating Point Multiplier

The proposed floating point multiplier was run on FPGA Spartan 3 and sample outputs are as shown in the figure below:

The above figure on the left hand side shows Floating point multiplier output for the value of 189.1234x4.62 On fpga spartan 3 where seven segment displays show the mantissa output in hex format,first LED in row one indicates sign bit and second row LEDs indicate exponent value.

The above figure on the right hand side shows Floating point multiplier output for values 189.1234x-4.62 On fpga spartan 3 where seven segment displays show the mantissa output in hex format,first LED in row one indicates sign bit and second row LEDs indicate exponent value

VI. CONCLUSION

This paper shows the implementation of IEEE754 single precision floating point multiplier on FPGA Spartan 3 using carry look ahead adder for exponent calculation and a Vedic multiplier for mantissa calculation. The proposed floating point multiplier shows optimized and better timing performance with a good accuracy. The results calculated are faster with total delay of 36.19ns. Due to an increasing demand of multipliers in scientific applications there is a need for increased precision floating point calculations using 64 bits and always there is a scope for improving the speed with the fast multiplication techniques and minimal time delay. So further the proposed work can be extended for the reconfigurable architecture and Double Precision Floating Point Multiplication.

VII. REFERENCES

[1] Paldurai.K, Dr.K.Hariharan, "FPGA Implementation of Delay Optimized Single Precision Floating point Multiplier" 2015 International Conference on Advanced Computing and Communication Systems (ICACCS -2015), Jan. 05 – 07, 2015, Coimbatore, INDIA.

[2]Mr. S.S.Mohanasundaram, A.Nirmal kumar, T.Arul prakash, "Design of Floating Point Multiplier Using Vedic Mathematics", IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 1, January 2015.

[3]Bhavesh Sharma , Amit Bakshi, "Comparison of 24X24 Bit Multipliers for Various Performance Parameters" International Journal of Research in Advent Technology (E-ISSN: 2321-9637) Special Issue 1st International Conference on Advent Trends in Engineering, Science and Technology "ICATEST 2015", 08 March 2015.

[4]Sneha Khobragade, Mayur Dhait, "Review on Floating Point Multiplier Using Vedic Mathematics" International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064.

[5] C. Renard, "FPGA implementation of an IEEE Standard floating-point unit," Tech. Rep. Thayer School of Engineering, Dartmouth College, NH USA.

[6] I.V.Vaibhav, K.V.Saicharan, B.Sravanthi and D.Srinivasulu, "VHDL Implementation of Floating Point Multiplier using Vedic Mathematics", International Conference on Electrical, Electronics and Communications (ICEEC) , ISBN-978-93-81693-66-03 , June 2014 pp.110-115.