Vol 05, Article 07390 July 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Design and Implementation of Efficient Adder based Floating Point Multiplier

LOKESH BHARDWAJ[1], SAKSHI BAJAJ[2]
[1]Student, M.tech, VLSI, [2]Assistant Professor,Electronics and Communication Engineering Department, Thapar University, Patiala, Punjab, India
[1]lokeshbh786@gmail.com, [2]sakshi.bajaj@thapar.edu

ABSTRACT
In this paper a new idea is proposed to increase the speed of single precision floating point multiplier. In floating point multiplication adders are used at different places. The implementation uses efficient adders for compressing the partial products, adding the exponent and at final stage. First different adders are compared based on the delay and then multiplier is designed using the best adder at each stage. The multiplier and the adder modules have been written in Verilog HDL and then synthesized and simulated using Xilinx ISE 14.5 targeted on the Spartan 3E FPGA. The design is verified using chipscopeproanalyzer. Results show that the multiplier speed can be increased if carry save adder is used for adding the partial products and carry select adder for exponent and at final stage.
Keywords:*FPGA, floating point multiplication*

1. INTRODUCTION

Floating point numbers are used for representing very large and very small values. The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The IEEE 754 standard presents two different floating point formats. One is binary interchange format and other is decimal interchange format. In this paper binary interchange format is used. The IEEE 754 binary format for single precision divides 32 bit number in three fields. The lower 23 bits are used for mantissa, next 8 bits are used for exponent and the last bit is used for representing sign of the number. The general equation for representing a normalized floating point number in binary format can be written as

$$Z = (-1^{s}) * 2^{E-Bias} * (1.M) \quad (1)$$

Where M is mantissa part, E is exponent part and S is sign bit. Floating point multiplication is similar to integer multiplication, because floating point numbers are represented in signed format, the multiplier needs to only deal with unsigned integer numbers and normalization. Multiplication in binaryformat of floating point multiplier is done in three steps. In first step the sign bit of two numbers are xored to get the sign of product, in second step the exponents of two numbers are added and bias is subtracted to get the resultant exponent, in third step mantissa's of two numbers are multiplied after appending '1' to each number. So the equation for final product after multiplication of two numbers can be written as

$$Z = (-1^{S1\ XOR\ S2}) * (2^{E1+E2-Bias}) * (1.M1 * 1.M2) \quad (2)$$

The result should be normalized and rounded for obtaining better accuracy.Implementation of floating point multiplier has been the interest of many researchers. In [2],variable latency floating point multiplier architecture was purposed, which can save respectable power and energy at the expense of slight area acceptable delay overheads. In [4], implementation of multiplier was done using three pipelined stages and the design was verified against Xilinx floating point multiplier core. In [5], floating point multiplier was implemented with kogge-stone adder at each stage. In [8], vedic multiplication technique was used to implement floating point multiplier. In [9],a combined method to reduce area and increase speed was proposed.
The rest of this paper is organised as follows: Section 2 describes the various steps involved in single precision floating point multiplication. Section 3 explains about the idea used in this paper. Results and implementation are described in Section 4. The paper is concluded in Section 4.

2. STEPS IN FLOATING POINT MULTIPLICATION

Figure.1 shows the block diagram representing the structure of multiplier.
Following steps are involved in calculating product of two floating point numbers:
A. Calculating the sign: The sign bit of the product is calculated by performing exclusive-or operation on the sign bit of two operands. A zero sign bit represent positive number and a one sign bit represent negative number.
*B. Calculating exponent:*Exponent is calculated by adding exponents of two operands and subtracting the bias. Initially the operands are in biased notation, so the bias is added twice. Hence a bias is subtracted to get the result. Bias value is 127 for single precision multiplier. So we require two adders, one for adding the exponents and second for adding the intermediate result and 2's complement of the bias as shown below in Figure.2.
*C. Calculating significand:*In this part multiplication of mantissa bits is performed. First of all a '1' is appended after mantissa of each operand. Then multiplication is carried out in different steps. Direct multiplication yields

Vol 05, Article 07390 July 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

24 partial products. But it is very difficult to deal with large number of partial products, so different radix-based algorithm was proposed by booth, out of them radix-4 booth algorithm is fastest which reduces the number of partial products by a factor of two.

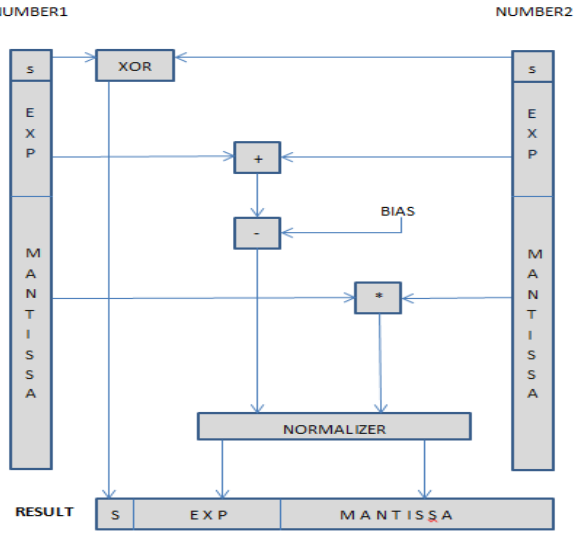

Figure.1 Floating point multiplier block diagram

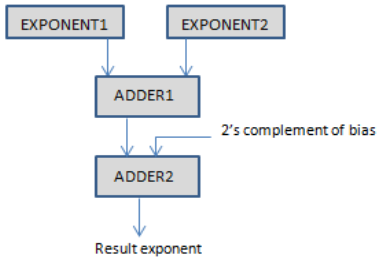

Figure.2 Exponent addition

Since the amount of hardware and the delay depends on the number of partial products to be added, this reduces the hardware cost and improves performance. So, in single precision multiplication 24 partial products are reduced to 12 partial products. These partial products are taken as inputs for the compressors.

Compressors [3] are used to make accumulation of partial products easy. These are used to convert large input bit strings into lesser output bit strings.A 3:2 compressor converts three input bit strings into two. Using chain of compressors twelve partial products are converted into two bit strings. At final stage there is a need of ripple carry adder or other fast adder for adding these two bits strings. This adder plays an important role in determining delay of the multiplier. The output of final stage adder is given to normalizer as the result is still not normalized. The complete procedure for calculation of mantissa is shown in Figure.3.

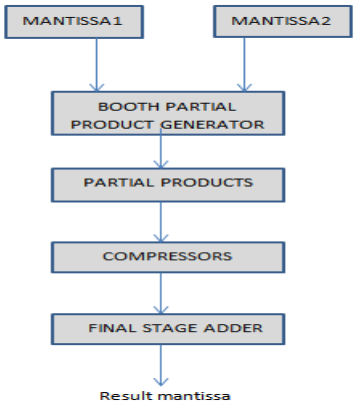

Figure.3 Mantissa calculation

D. Normalizing the result: Normalizing means getting a '1' at MSB. So a '1' is detected and exponent is adjusted accordingly. A one may be either at 46^{th} bit or at 47^{th} bit of final 48 bit final result. If leading '1' is at 46^{th} bit the number is already a normalized number and no shift is needed. While if the leading '1' is at 47^{th} bit then result is shifted to the right and exponent is incremented by one.

Vol 05, Article 07390 July 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

E: Rounding the result: Precision is lost when bits are shifted in normalizing the number. So rounding is implemented to get the best result from the given bits.Various rounding modes are used to get accurate result. Generally round to nearest even mode is used for obtaining correct results.
*E: Checking the exceptions:*Exceptions occur when the resultant number after multiplication is out of range for that particular representation. These exceptions can be checked by raising the flag register. The exceptions in IEEE standard are invalid operation, divide by zero, overflow, underflow and inexact.

3. PROPOSED IDEA

There are different steps involved in calculation of product of two floating point numbers. The major steps involved are calculation of sign, calculation of significand and calculation of exponent. For calculating the significand there is a need of compressors and a final stage adder. Adders are also used in calculation of exponent. The maximum time of multiplication is elapsed in accumulating the partial product and at the final stage adder. So the multiplication time can be reduced if the partial products are accumulated with the help of fast adders. In this brief, the approach is to use fast adders at each part of the floating point multiplier. In the proposed approach firstly the different adders are compared in terms of delay and then on the basis of the synthesis results obtained, thefloating point multiplier is designed using the fastest adder at appropriate place.

4. RESULTS AND IMPLEMENTATION

From above discussion it is clear that for calculation of exponent and mantissa part there is a requirement of adders. So different adders are designed in Verilog HDL,synthesized and simulated using the Xilinx ISE 14.5. Table I shows the total delay and area utilization summary in terms of number of slices for 8,16 and 32 bitripple carry adder, carry look-ahead adder, carry skip adder, carry save adder and carry select adder.Based upon synthesis reports of different adders available, carryselect adder gives the best performance in terms of delay. So carry select adder can be used for exponent addition and at final stage of the floating point multiplier. Carry save adder used as a compressor for compressing the partial products. In carry save adder carry is not rippled through different bits rather calculated for each bit. Thus its total consumption time is less than ripple carry adder. So implementation of single precision floating point multiplier has been done by using carry save adders for adding partial products and carry select adder for adding exponent and at final stage. This multiplier has been designed in Verilog HDL, synthesized and simulated using the Xilinx ISE 14.5 targeted on the Spartan 3E and Virtex 4 FPGA.Table II and III show the synthesis reports containing the area utilization and total delay of the single precision floating point multiplier for Spartan and virtex device.The multiplier is simulated using ISIM simulator. Functional simulation result for the single precision floating point multiplier is shown in Figure.4.
The implemented multiplier is verified using chipscope pro analyzer. It is the in-built verification tool provided in the ISE pack of xillinx. As the number of inputs are 64 and outputs are 32. So, direct implementation using on-board inputs outputs is very difficult on FPGA. So chipscopeproanalyzer is used for implementing on FPGA. Two IP cores are generated, one for virtual input output operation and other for controlling the generated VIO(virtual input/output) core. The top module instantiating the main module, VIO core and ICON(Integrated controller) core is shown in Figure.6. From the block diagram it is clear that the inputs of main module are connected to the outputs of VIO core and the outputs are connected to the inputs of the VIO core. So we get the output from sync_in port of the VIO core after applying input from async_out port. The system is synchronized with clock. The output results are shown in Figure.5 which is similar to simulation results obtained.

Table I
Delay and area utilization of different adders

S.No.	Adders (8,16 and 32 bit)	No. of slices(out of 4656) for 8/16/32 bit adder	Total Delay(in ns) for 8/16/ 32 bit adder
1,	Ripple carry adder	9/18/37	14.8/24.6/44.3
2.	Carry-look ahead adder	9/39/79	12.8/19.6/26.7
3.	Carry skip adder	13/28/56	13.9/18.9/29.0
4.	Carry-select adder	14/31/66	11.5/16.7/27.3
5.	Carry-save adder	13/29/61	13.7/23.2/42.2

Table II
Synthesis report of floating point multiplier for Spartan 3E

	Spartan 3E (XC3S500E)
No. of Slices	972/4656(20%)
No. of 4 input LUTs	1897/9312(20%)
Total Delay	27.6ns(18.260 logic 9.340 route delay)

Vol 05, Article 07390 July 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Table III
Synthesis report of floating of point multiplier for Virtex 4

	Virtex4 (XC4VLX15)
No. of Slices	968/6144(15%)
No. of 4 input LUTs	1889/12288(15%)
Total Delay	12.142ns(4.396 logic 7.746 route delay)

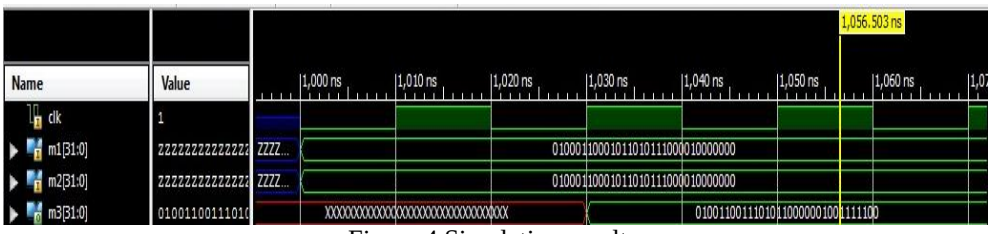

Figure.4 Simulation results

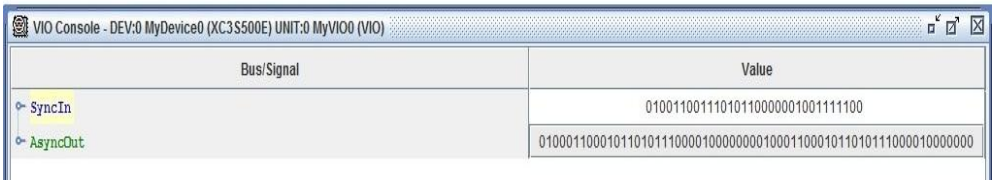

Figure.5 Output results on chipscopeproanalyzer

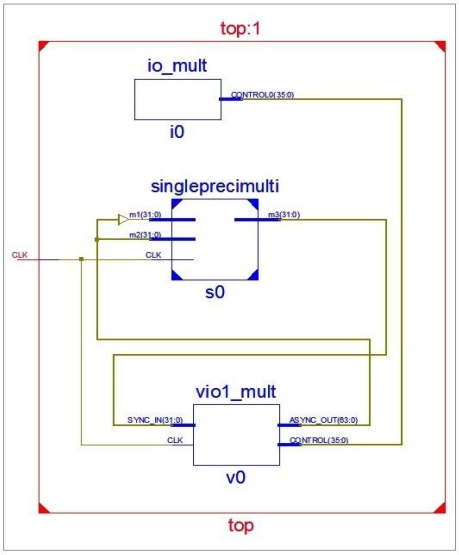

Figure.6 Block diagram for FPGA implementation

5. CONCLUSION

In this paper, first different adders are compared based on area utilization and total delay. According to synthesis results carry-select adder individually gives the best performance in terms of delay. So based upon these results the floating point multiplier is implemented using carry select adder for exponent addition and at final stage while carry save adder for adding partial products. As compared to [5] the proposed multiplier provided 24% improvement in delay and 30% improvement in area for Spartan device and 54% improvement in delay and 31% in area for the virtex device.

REFERENCES

[1] IEEE 754-2008, IEEE standard for Floating-Point Arithmetic,2008
[2] S. Kuang, J. Wang and H. Hong, "Variable-latency Floating Point Multipliers for Low-Power Applications," in *IEEE Transactions On Very Large Scale Integration Systems*, vol.18, no.10, pp.1493-1497, Oct. 2010

[3] J. Kaur, N.K. Gaahlan and P.shukla, "Delay-Power performance comparison of array multiplier in VLSI design,"in*International Journal of Advanced Research in Computer Science and Electronics Engineering*, vol.1, no.3, May.2012
[4] M. Al-Ashrafy, M. Salem and W. Anis, "An efficient implementation of floating point multiplier," in Proc. *Electronics,Communications and Photonics Conference(SIECPC)*, pp.1-5, 24-26 Apr.2011
[5] A. Jain, B. Dash and A. Panda, "FPGA Design of a Fast 32-bit Floating Point Multiplier Unit," in Proc. *International Conference on Devices, Circuits and systems(ICDCS)*, pp.545-547,15-16 Mar.2012
[6] R.P.P. Singh, P. Kumar and B. Singh, "Performance Analysis Of Fast Adders Using VHDL," in Proc. *International conference on Advances in Recent Technologies in Communication and Computing*, pp.189-193,27-28 Oct.2009
[7] P. Gurjar, R. Solanki, P. Kansliwal and M. Vucha," VLSI implementation of adders for high speed ALU," in Proc. *India conference(INDICON)*, pp.1-6,16-18 Dec.2011
[8] A. Kanhe, S.K. Das and A.K. Singh, "Design and Implementation of Floating Point Multiplier based on Vedic Multiplication Technique," in Proc. *International Conference on* Communication, Information & Technology(ICCICT), pp. 1-4, 19-20 Oct.2012
[9] S.V. Siddamal, R.M. Banakar and B.C Jinaga, " Design of High-speed floating point multiplier," in *4th IEEE International Symposium on Electronic Design, Test and Applications*, pp. 285-289, 23-25 Jan.2008