International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 11, Issue 07 (July 2015), PP.60-65

Comparison of Adders for optimized Exponent Addition circuit in IEEE754 Floating point multiplier using VHDL

Nishi Pandey[1], Virendra Singh[2]
Sagar Institute of Research & Technology Bhopal

Abstract:- Floating point arithmetic has a vast applications in DSP, digital computers, robots due to its ability to represent very small numbers and big numbers as well as signed numbers and unsigned numbers. In spite of complexity involved in floating point arithmetic, its implementation is increasing day by day. Here we compare three different types of adders while calculating the addition of exponent bits while calculating the single precision and double precision floating point multiplication. We also present the multiplication of mantissa/significand bits by decomposition of operands method for IEEE 754 standard multiplication. Here we break down the mantissa bits of each single precision floating point operand into 4 Parts, each of six bits. Likewise we breakdown the mantissa bits of double precision floating point operand into 4 Parts, 3 parts of 13 bits each and one part of 14 Bits. We get 16 partial product terms in each case. Careful addition of these partial product terms are required to get the product of mantissas.

Keywords:- IEEE754, Single Precision Floating Point (SP FP), Double Precision Floating Point (DP FP), Maximum Combinational Path Delay (MCPD).

I. INTRODUCTION

IEEE 754 and IEEE 854 are the two standards used to represent floating point numbers. IEEE 854 standard uses variable length of bits to represent the floating point numbers. IEEE 754 is the most widely used format for representing floating point numbers in computers, providing four level of precision. It is a fixed point representation [1]. Most of the algorithms implemented in FPGAs are to be fixed point. The floating point operations has many applications in the various fields because of its great dynamic range, easy operation rules and high precision. IEEE754 encodes floating point numbers in memory in a way such that it packs three fields the sign, exponent and significand as follows .The leading bit is the sign bit,0 for + and 1 for-.The next 8 bits (11 bits for double precision floating point number)hold a biased exponent .the last 23 bits (52 bits for double precision floating point number)hold the significand's magnitude. It is shown in figure 1. In IEEE754 floating point representation, the exponent is biased in the sense that it is offset value from the actual value by the exponent bias [2].
Exponent need to be signed values to represent both very small and huge values. Biasing makes the values of exponents within an unsigned range suitable for high speed comparison.

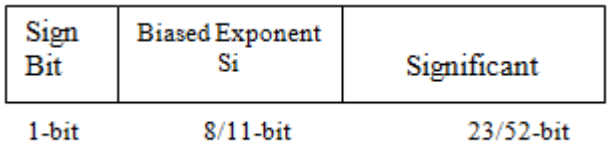

Figure 1: IEEE 754 Single Precision and Double Precision Floating Point Format

Greater data precision available with 64 bits representation (double precision floating point representation), but processing 64 bits of data uses twice as much RAM, cache , and bandwidth, thereby reducing the overall system performance as compare to 32 bits representation(single precision floating point representation) [3]. The word double here means double precision number uses twice as many bits as a regular floating point number (32 bits). The extra bits increase not only the precision but also the range of magnitudes that can be represented.

- Decimal to IEEE754 Standard Floating Point

The following algorithm needs to be follow to convert a decimal number in to IEEE754 standard floating point [4].

- Represent the decimal number into binary format.
- Normalize the binary number.
- Normalization is the process in which radix point is either shifted right or to the left such that the MSB is "1"(add or subtract the exponent accordingly).
- The biased exponent E=exponent value derived from the step 2(normalized exponent) +bias (127 for single precision floating point and 1023 for double precision floating point).
- Convert E into binary and finally we get exponent E.
- After normalization, bits after the radix point become significand bits. The first bit (1) is not including in the format that is why it is called the hidden bit.

II. IEEE 754 STANDARD FLOATING POINT MULTIPICATION ALGORITHM

A brief overview of floating point multiplication has been explained below [5-6].

- Both sign bits S_1, S_2 are need to be Xoring together, then the result will be sign bit of the final product.
- Both the exponent bits E_1, E_2 are added together, then subtract bias value from it. So, we get exponent field of the final product.
- Significand bits Sig_1 and Sig_2 of both the operands are multiply including their hidden bits.
- Normalize the product found in step 3 and change the exponent accordingly. After normalization, the leading "1 "will become the hidden bit.

Above algorithm of multiplication algorithm is shown in Figure 2.

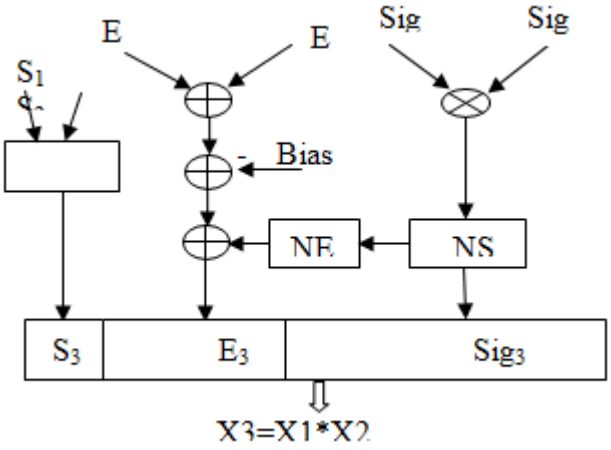

Figure 2: IEEE754 SP FP and DP FP Multiplier Structure, NE: Normalized exponent, NS: Normalized Significand

III. EXPONENT ADDER

In digital electronics most of the adders are used for adding numbers in binary and produces sum bits and carry bit. Adders can be constructed for any numerical representation such as binary coded decimal, Excess-3 etc. An adder plays an important role not only in arithmetic logic unit but also in other processors. Many types of adders are available using different logics to add the binary numbers. Half adder and full adder are the two most commonly used and basic building blocks for other adders. Half adder adds two bits and produces carry bit and sum bits whereas full adder adds 3 bits and produces sum bits and carry bit. These two adders are the simplest types of adders. More complex adders are also available which can adds more than three bits. Parallel adder, Carry skip adder and Cary select adder are some of them.

- Parallel adder

Parallel adder can add all bits in parallel manner i.e. simultaneously hence increased the addition speed. In this adder multiple full adders are used to add the two corresponding bits of two binary numbers and carry bit of the previous adder. It produces sum bits and carry bit for the next stage adder. In this adder multiple carry produced by multiple adders are rippled, i.e. carry bit produced from an adder works as one of the input for the adder in its succeeding stage. Hence sometimes it is also known as Ripple carry adder (RCA). Generalized diagram of parallel adder is shown in Figure 3.

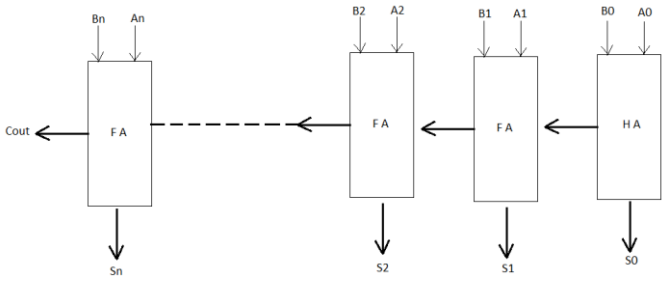

Figure 3: Block Diagram of Exponent Parallel Adder

- Carry Skip Adder

This adder gives the advantage of less delay over Ripple carry adder. It uses the logic of carry skip, i.e. any desired carry can skip any number of adder stages. Here carry skip logic circuitry uses two gates namely "and gate" and "or gate". Due to this fact that carry need not to ripple through each stage. It gives improved delay parameter. It is also known as Carry bypass adder. Generalized figure of Carry skip adder is shown in Figure 4.

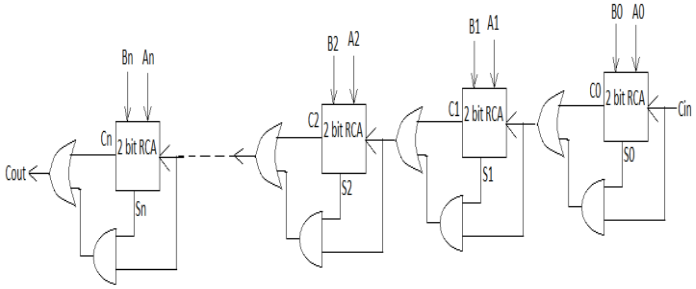

Figure 4: Block Diagram of Exponent Carry Skip Adder

- Carry select adder

Carry select adder uses multiplexer along with RCAs in which the carry is used as a select input to choose the correct output sum bits as well as carry bit. Due to this, it is called Carry select adder. In this adder two RCAs are used to calculate the sum bits simultaneously for the same bits assuming two different carry inputs i.e. '1' and '0'. It is the responsibility of multiplexer to choose correct output bits out of the two, once the correct carry input is known to it. Multiplexer delay is included in this adder. Generalized figure of Carry select adder is shown in Figure 5.

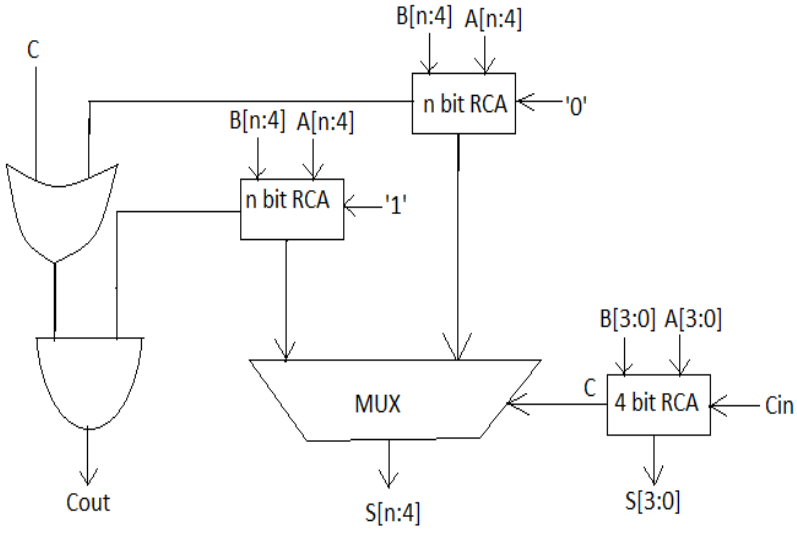

Figure 5: Block Diagram of Exponent Carry Select Adder

IV. PROPOSED DESIGN

In IEEE754 standard floating point representation, 8 bit Exponent field in single precision floating point (SP FP) representation and 11 bit in double precision floating point (DP FP) representation are need to add with another 8 bit exponent and 11 bit exponent respectively, in order to multiply floating point numbers represented in IEEE 754 standard as explained earlier. Ramesh et al [1] has used parallel adder for adding exponent bits in floating point multiplication algorithm. We proposed the use of carry select adder and Carry skip adder for adding the exponent bits. We have found the improved path delay of Carry select adder and Carry skip adder over the parallel adder. We have designed parallel adders, Carry select adders and Carry skip adders for exponent addition of single precision floating point multiplier (8 bit) and double precision floating point multiplier (11 bit).

- Sign bit calculation

To calculate the sign bit of the resultant product for SP FP and DP FP multiplier, the same strategy will work. We just need to xoring together the sign bits of both the operands. If the resultant bit is '1', then the final product will be a negative number. If the resultant bit is '0', then the final product will be a positive number.

- Exponent bit calculation

Add the exponent bits of both the operands together, and then the bias value (127 for SPFP and 1023 for DPFP) is subtracted from the result of addition. This result may not be the exponent bits of the final product. After the significand multiplication, normalization has to be done for it. According to the normalized value, exponents need to be adjusted. The adjusted exponent will be the exponent bits of the final product.

- Significand bit calculation

Significand bits including the one hidden bit are need to be multiply, but the problem is the length of the operands. Number of bits of the operand will become 24 bits in case of SP FP representation and it will be 53 bits in case of DP FP representation, which will result the 48 bits and 106 bits product value respectively. In this paper we use the technique of break up the operands into different groups then multiply them. We get many product terms, add them together carefully by shifting them according to which part of one operand is multiplied by which part of the other operand. We have decomposed the significand bits of both the operands ain four groups. Multiply each group of one operand by each group of second operand. We get 16 product terms. Then we add all of them together very carefully by shifting the term to the left according to which groups of the operands are involved in the product term.

V. SIMULATION RESULT

We functionally verified each unit presented in this paper including all three adders and two IEEE754 multipliers. We have been found from the results shown in Table .1, that number of slices used is same in case of Carry select adder and Parallel adder which is less than slices used in Carry skip adder, but also Carry select adder gives least amount of path delay. So we designed SP FP and DP FP multiplier using Carry select adder whose device utilization summary is given inTable.2.

Table 1: Comparisons result for Exponent Adder used in SPFP and DPFP Multiplier

Design	Exponent Adder	No. of slices	No. of 4 input LUTs	MCPD (ns)
DP FP	Carry Skip adder	16	27	17.977
	Parallel adder	13	22	20.253
	Carry select adder	13	23	15.573
SP FP	Carry Skip adder	11	20	12.865
	Parallel adder	9	16	16.299
	Carry select adder	9	16	12.801

Table 2: Device utilization summary (VertexE-XCV50e-8cs144) of SP FP and DP FP multiplier using Carry select adder

	SP FP Multiplier	**DP FP Multiplier**
No. of Slices	353	1480
No. of 4 input LUTs	610	2620
No. of bounded IOBs	121	246
Maximum combinational path delay(in ns)	22.021	31.624

We have implemented the Carry Select Adder for SP FP and DP FP, also the multipliers on vertexE whose RTL (Resistor Transistor Logic) view are shown as below. a and b are the exponent inputs bit, s, c is the exponent output bit as shown in figure 6(a) and 6(b) respectively.

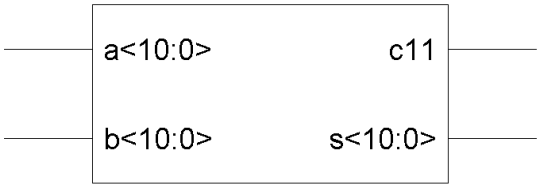

Figure 6 (a): RTL view of Carry select adder (11 bit)

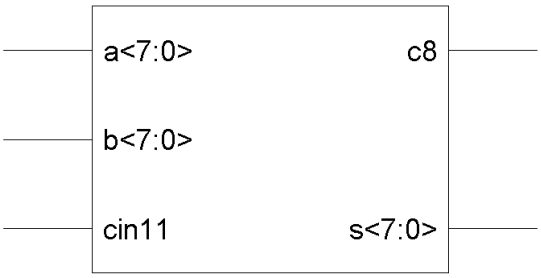

Figure 6 (b): RTL view of Carry select adder (8 bit)

The RTL view of the SPFP and DPFP multiplier using exponent carry select adder are shown in Figure 7(a) and 7(b) respectively. In this figure e1 and e2 are the exponent bit, s1 and s2 are the sign bit whereas m1 and m2 are the mantissa (significand) bit of the two operands in SP FP and DP FP IEEE754 format. S3, e3 and m3 are the sign bit, exponent bit and mantissa bit respectively of the product.

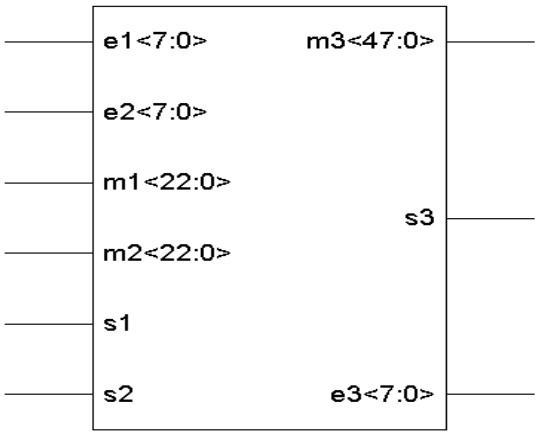

Figure 7(a): RTL view of SP FP Multiplier using Carry select adder

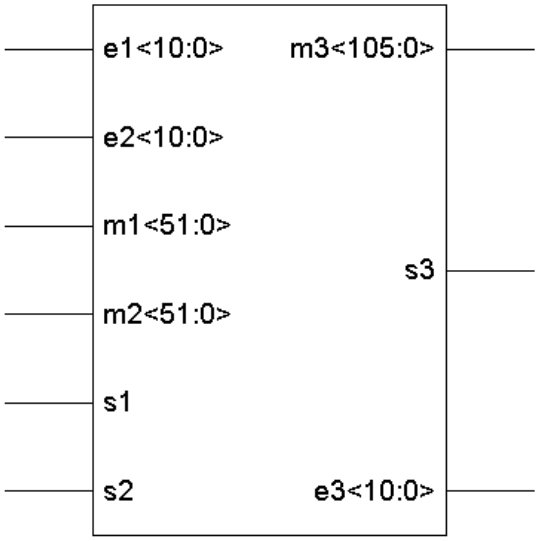

Figure 7(b): RTL view of DP FP Multiplier using Carry select adder

VI. CONCLUSION

IEEE754 standarize two basic formats for representing floating point numbers namely, single precision floating point and double precision floating point. Floating point arithmetics has a vast applications in many areas like robotics and DSP. Delay provided and area required by hardware are the two key factors which are need to be consider Here we present single precision floating point multiplier and double precision multiplier by using three different adders namely parallel adder,Carry skip adder and Carry Select aader.

Among all three adders, carry select adder provides the least amount of Maximum combinational path delay (MCDP). Also, it takes least number of slices i.e. occupy least area among all three adders. So, we implement the SP FP multiplier and DP FP multiplier by using Carry select adder on virtex E XCV50e-8cs144.

REFERENCES

[1]. B. Fagin and C. Renard, "Field Programmable Gate Arrays and Floating Point Arithmetic," IEEE Transactions on VLS1, vol. 2, no. 3, pp. 365-367, 1994.

[2]. N. Shirazi, A. Walters, and P. Athanas, "Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines," Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM"95), pp.155-162, 1995.

[3]. Malik and S. -B. Ko, "A Study on the Floating-Point Adder in FPGAs", in Canadian Conference on Electrical and Computer Engineering (CCECE-06), (2006) May, pp. 86–89.

[4]. D. Sangwan and M. K. Yadav, "Design and Implementation of Adder/Subtractor and Multiplication Units for Floating-Point Arithmetic", in International Journal of Electronics Engineering, (2010), pp. 197-203.

[5]. M. K. Jaiswal and R. C. C. Cheung, "High Performance FPGA Implementation of Double Precision Floating Point Adder/Subtractor", in International Journal of Hybrid Information Technology, vol. 4, no. 4, (2011) October.

[6]. L. Louca, T. A. Cook and W. H. Johnson, "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs", Proceedings of 83rd IEEE Symposium on FPGAs for Custom Computing Machines (FCCM□96), (1996), pp. 107–116.

[7]. Jaenicke and W. Luk, "Parameterized Floating-Point Arithmetic on FPGAs", Proc. of IEEE ICASSP, vol. 2, (2001), pp. 897-900.

[8]. Lee and N. Burgess, "Parameterisable Floating-point Operations on FPGA", Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems, and Computers, (2002).

[9]. M. Al-Ashrafy, A. Salem, W. Anis, "An Efficient Implementation of Floating Point Multiplier", Saudi International Electronics, Communications and Photonics Conference (SIECPC), (2011) April 24-26, pp. 1-5.