Marode Shubham Hemant, 2022, 10:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

VLSI design and analysis of IEEE 754 Floating Point Multiplier

M.Tech. Scholar Marode Shubham Hemant, Asst. Prof. Dr Neetesh Raghuwanshi
Department of Electronics & Communication Engineering
Sarvepalli Radhakrishnan University, Bhopal

Abstract- There is an increasing demand for high-speed and low-power processors as new technology in the field of VLSI and embedded systems advances. The multiplier and adder performance of a processor have a significant impact on its speed. Despite the complexity of floating point arithmetic, its implementation is becoming more common. As a result, high-speed adder architecture is becoming increasingly critical. To improve the efficiency of the adder, several architecture designs have been proposed. In this paper, we provide an architecture that uses a carry choose adder to perform a high-speed IEEE 754 floating point multiplier (CSA). Two carry choose based designs are presented here. These designs are based on the Vertex device family from Xilinx.

Keywords- IEEE754, Single Precision Floating Point (SP FP), Double Precision Floating Point (DP FP, VLSI).

I. INTRODUCTION

Floating point numbers are real numbers that are expressed in binary representation. Floating point formats are divided into binary and decimal interchange formats according to the IEEE-754 standard. In dsp applications, floating point multipliers are crucial. The double precision normalized binary exchange format is the subject of this paper. The IEEE-754 double precision binary format is depicted in Figure 1.

Sign (s) is represented with one bit, exponent (e) and fraction (m or mantissa) are represented with eleven and fifty two bits respectively. For a number is said to be a normalized number, it must consist of 'one' in the MSB of the significand and exponent is greater than zero and smaller than 1023. The real number is represented by equations (i) & (2).

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

$$Value = (-1^{signbit}) \times 2^{(Exponent-1023)} \times (1.Mantissa) \quad \textbf{(2)}$$

Biasing makes the values of exponents within an unsigned range suitable for high speed comparison.

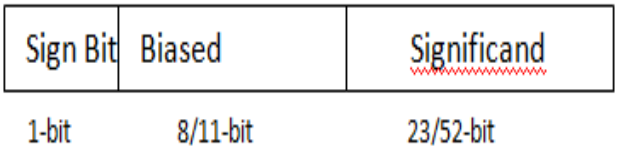

Fig 1. IEEE 754 Single Precision and Double Precision Floating Point Format.

1. IEEE 754 Standard Floating Point Multiplication Algorithm:

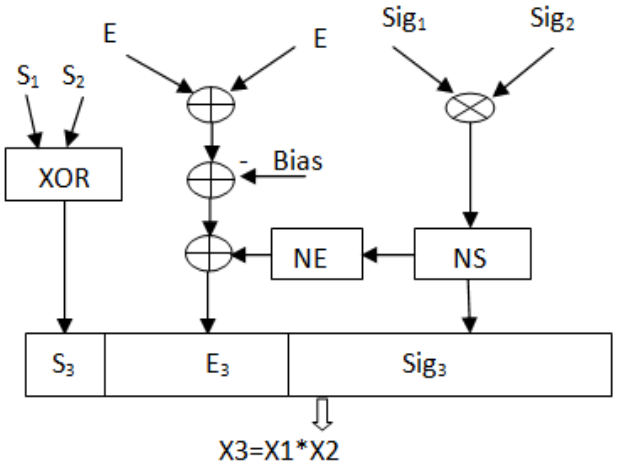

Fig 2. IEEE754 SP FP and DP FP Multiplier Structure, NE: Normalized exponent, NS: Normalized Significand.

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

- Both sign bits S1 and S2 must be Xored together, and the result will be the final product's sign bit.
- Both exponent bits E1 and E2 are put together, and then the bias value is subtracted. As a result, we obtain the final product's exponent field.
- Bits of significance both operands' Sig1 and Sig2 are multiplied, including their hidden bits.
- Change the exponent to match the normalized product calculated in step 3. The leading "1" will become the hidden bit after normalization.

Above algorithm of multiplication algorithm is shown in Figure 2.

II. DIFFERENT TYPES OF ADDER

1. Parallel Adder:

The addition speed of a parallel adder is boosted since it may add all bits in parallel, i.e. at the same time. Multiple full adders are employed in this adder to add the two corresponding bits of two binary numbers, as well as the previous adder's carry bit. For the following stage adder, it generates sum bits and carries bits.

Multiple carry bits produced by multiple adders are rippled in this adder, i.e. a carry bit produced by one adder becomes one of the inputs for the adder in the next stage. As a result, it's also known as the Ripple Carry Adder (RCA). Figure 3 depicts a generalized parallel adder diagram.

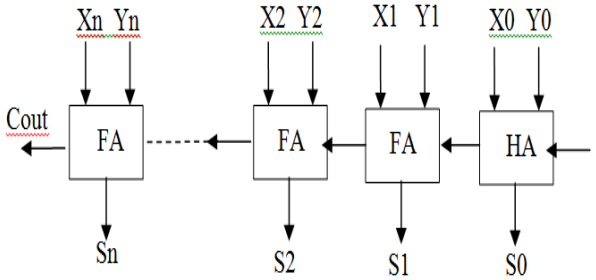

Fig 3. Parallel Adder (n=7 for SPFP and n=10 for DPFP).

If the last carry bit is required, an n-bit parallel adder has one half adder and n-1 complete adders. However, because final carry out is not required in the 754 multiplier's exponent adder, we can utilize the XOR Gate instead of the last full adder. It not only minimizes the amount of space taken up by the circuit, but it also reduces the amount of time it takes to calculate. We simulate 8 bit and 11 bit parallel adders for SPFP and DPFP multiplier exponent adders, respectively, as shown in figure 4.

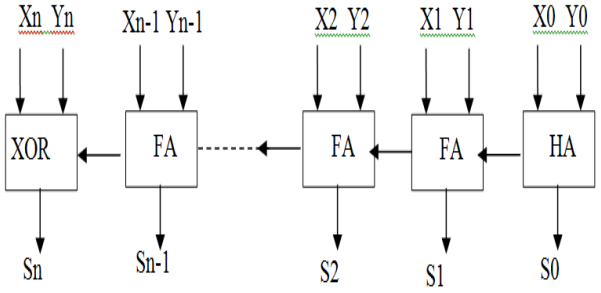

Fig 4. Modified Parallel Adder (n=7 for SPFP and n=10 for DPFP).

2. Carry Skip Adder:

In comparison to the Ripple carry adder, this adder has a shorter delay. It employs carry skip logic, which means that any desired carry can skip any number of adder stages. The "and gate" and "or gate" gates are used in the carry skip logic circuits. As a result, carry does not need to ripple across each stage. It improves the delay parameter. Carry bypass adder is another name for it. Figure 5 is a generalized Carry Skip Adder figure.

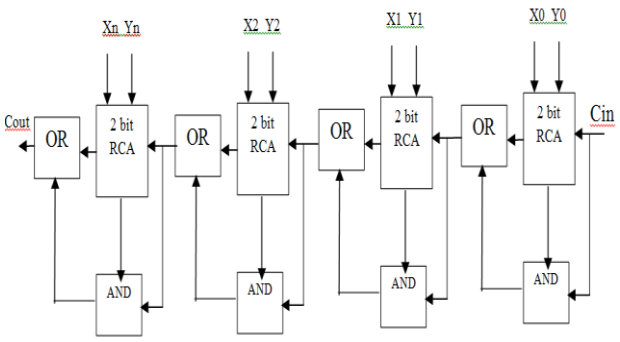

Fig 5. Carry Skip Adder.

3. 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 3.9. Adders are the basic building blocks of most of the ALUs (Arithmetic logic units) used in Digital signal processing and various other applications**.** Many types of adders are available in today's scenario and many more are developing day by day.

Half adder and Full adder are the two basic types of adders. Almost all other adders are made with the different arrangements of these two basic adders only. Half adder is used to add two bits and produce sum and carry bits whereas full adder can add three bits simultaneously and produces sum and carry bits.

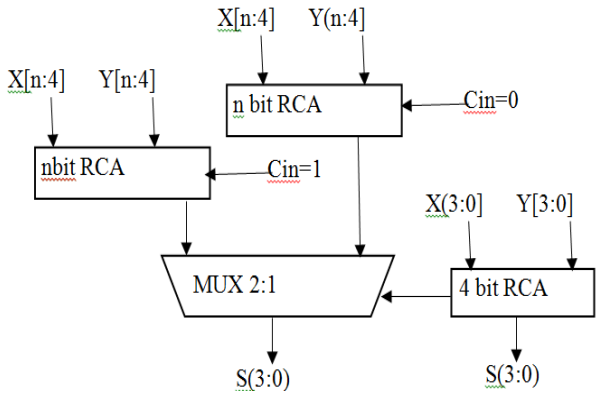

Fig 6. Carry Select Adder.

III. PROPOSED DESIGN

In order to multiply floating point numbers represented in IEEE 754 standard, the 8 bit Exponent field in single precision floating point (SP FP) representation and the 11 bit Exponent field in double precision floating point (DP FP) representation must be added with another 8 bit exponent and 11 bit exponent, respectively.

In their floating point multiplication technique, Ragini et al. [10] used a parallel adder to add exponent bits. For adding the exponent bits, we proposed using 8-bit modified CSA with dual RCA and 8-bit modified CSA with RCA and BEC. We discovered that the 8-bit modified Carry select adder with RCA and BEC has a better area than the 8-bit modified CSA with dual RCA.

1. 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 XOR 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.

2. 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.

3. 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.

4. Partition Multiplier:

Algorithm for partition method

- t1 : in STD_LOGIC_VECTOR (7 downto 0);
- t2 : in STD_LOGIC_VECTOR (7 downto 0);
- t3 : out STD_LOGIC_VECTOR (15 downto 0));
- h1<=t1(3 downto 0);
- h2<=t1(7 downto 4);
- h3<=t2(3 downto 0);
- h4<=t2(7 downto 4);
- su1<=h1*h3;
- su2<=h1*h4;
- su3<=h2*h3;
- su4<=h2*h4;
- ad1<=("00000000" & su1);
- ad2<=("0000" & su2 & "0000");

- ad3<=("0000" & su3 & "0000");
- ad4<=(su4 & "00000000");
- t3<=ad1 + ad2 + ad3 + ad4;

IV. SIMULATION RESULT

VHDL's engineering makes it extremely versatile, allowing designers, electronic plan mechanization groups, and the semiconductor industry to experiment with new dialect concepts to ensure low plan costs and information interoperability.

After planning the various DSP configurations, we are now moving on to the product union of these designs using VHDL. We've set up the optimum channel yields in the following sections, using different VHDL scripts for each design. In the Appendix, the plan codes have been displayed. On XILINX ISE plan suite 6.2i, the construction was analyzed to see if it could be completed or blended efficiently.

1. Number 4-input LUTs:
LUT stands for look up table that reduces the complex mathematics calculations and provide the reduced processing time. Look up table uses so many complex applications such as signal processing, image processing, device modeling and other digital processing etc.

2. Number of Slices:
If the devices are connected in parallel form then it is called array of the devices. Generally look up table are comprised with number of slices. If the numbers of slices are increased then area will be increased. Numbers of slices are used less as possible as for better result and speed.

3. Propagation Delay
Generally, the ideal condition of the result is the output of the digital circuit should from level 0 to level 1 or level 1 to level 0 in zero time. But in practice, it takes finite time to switch output levels. The time required to change output levels is called output switching time. It defines separately for switching from level 0 to level 1 and level 1 to level0

The spread postponement of the gadget is essentially the time interim between the utilization of an information beat and the event of the subsequent yield beat. The proliferation deferral is an essential normal for rationale circuits since it restricts the velocity at which they can work. The shorter the spread defer, the higher the rate of the circuit and the other way around.

4. Number of IOBs:
Input output buffers are related to the fan in and fan out of the circuit. Number of gates is dependent on numbers of IOBs. So, for low propagation delay IOBs must be less.

SPFP multiplier is a combinational logic circuit with E1, E2, M1, M2, S1, S2 inputs and E3, M3, S3 outputs depends on the requirement. Figure 7 shows the view technology schematic of SPFP multiplier.

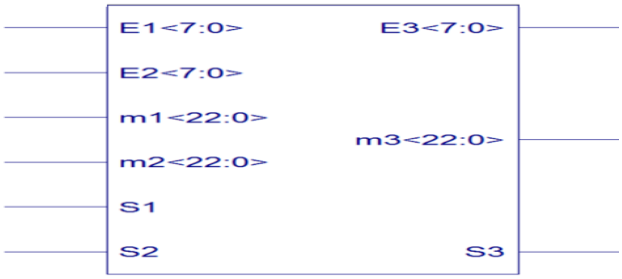

Fig 7. View Technology Schematic of SPFP Multiplier.

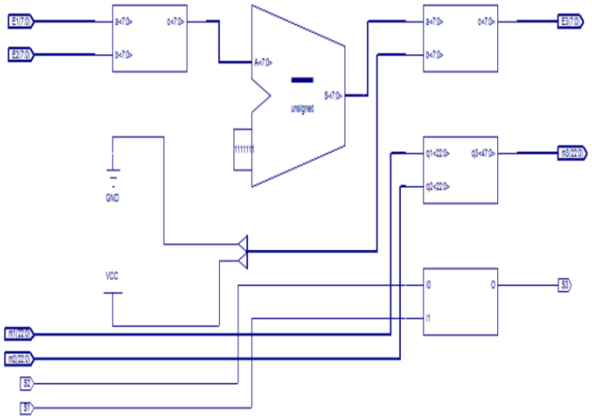
Fig 8. RTL View of SPFP Multiplier.

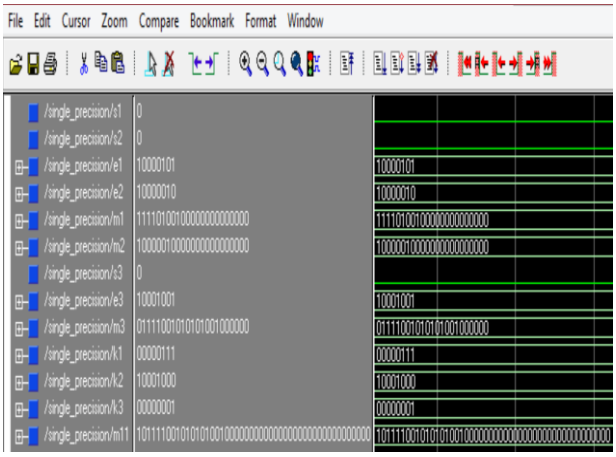

Fig 9. Output Waveform of SPFP Multiplier.

```
Device utilization summary:
---------------------------

Selected Device : 2vp2fg256-7

 Number of Slices:                121  out of   1408     8%
 Number of 4 input LUTs:          226  out of   2816     8%
 Number of bonded IOBs:            96  out of    140    68%
 Number of MULT18X18s:             16  out of     12   133% (*)
Timing Summary:
---------------
Speed Grade: -7

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 33.970ns
```

Fig 10. Device Utilization summary of SPFP Multiplier.

DPFP multiplier is a combinational logic circuit with E1, E2, M1, M2, S1, S2 inputs and E3, M3, S3 outputs depends on the requirement. Figure 11 shows the view technology schematic of DPFP multiplier.

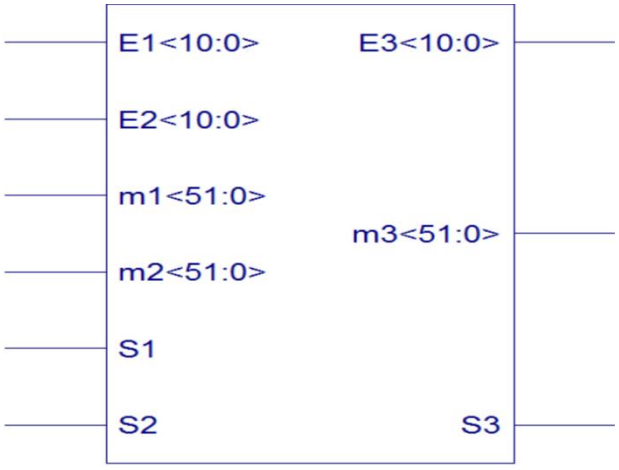

Fig 11. View Technology Schematic of DPFP Multiplier.

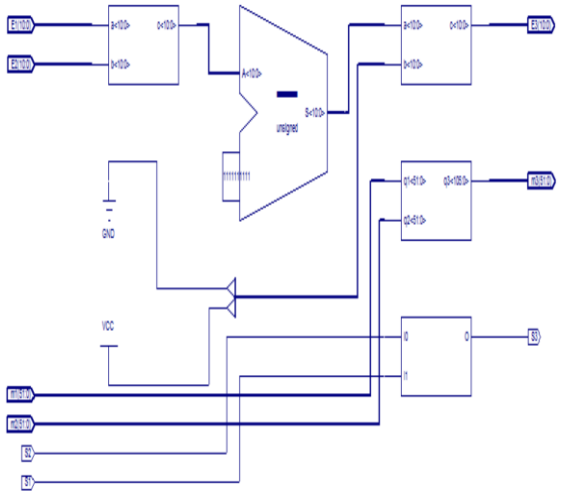

Fig 12. RTL View of DPFP Multiplier.

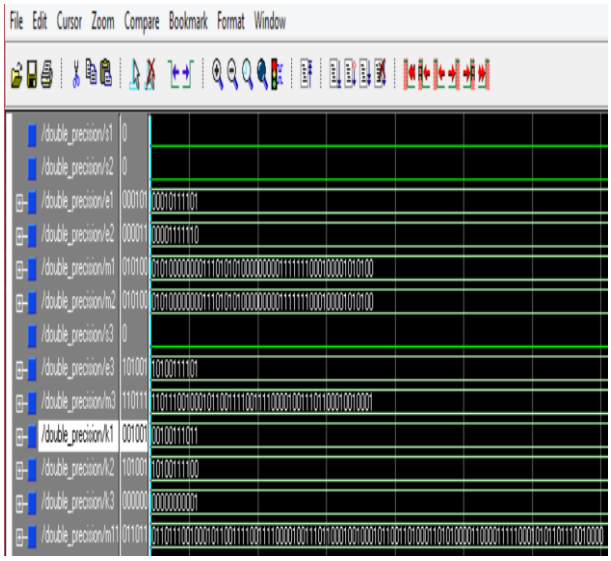

Fig 13. Output Waveform of DPFP Multiplier.

```
Device utilization summary:
---------------------------

Selected Device : 2vp2fg256-7

 Number of Slices:                377  out of   1408    26%
 Number of 4 input LUTs:          682  out of   2816    24%
 Number of bonded IOBs:           192  out of    140   137% (*)
 Number of MULT18X18s:             16  out of     12   133% (*)
Timing Summary:
---------------
Speed Grade: -7

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 70.700ns
```

Fig 14. Device Utilization summary of Double Precision Floating Point Multiplier.

Table 1. Comparison Result.

Parameter	Previous SPFP Algorithm	Implemented SPFP Multiplier using Partition Method	Previous DPFP Algorithm	Implemented DPFP Multiplier using Partition Method
Number of Slice LUTs	705	226	5153	682
Number of Input Output Bonded	96	96	192	192
Maximum Combinational Path Delay	44.823 ns	33.97 ns	83.169 ns	70.70 ns

V. CONCLUSION

Single precision floating point and double precision floating point are the two primary formats for encoding floating point values that IEEE754 standardizes. Floating point arithmetic has several applications in fields such as robotics and digital signal processing. The two most important elements to consider are the amount of delay offered and the amount of space required by hardware.

Using two separate adders, modified CSA with dual RCA and modified CSA with RCA and BEC, we provide a single precision floating point multiplier. Modified CSA with RCA and BEC has the shortest Maximum Combinational Path Delay of the two adders (MCDP). It also requires the fewest number of slices, occupying the smallest amount of space of the two adders.

REFRENCES

[1] Soumya Havaldar, K S Gurumurthy, "Design of Vedic IEEE 754 Floating Point Multiplier", IEEE International Conference on Recent Trends in Electronics Information Communication Technology, May 20-21, 2016, India.

[2] Ragini Parte and Jitendra Jain, "Analysis of Effects of using Exponent Adders in IEEE- 754 Multiplier by VHDL", 2015 International Conference on Circuit, Power and Computing Technologies [ICCPCT] 978-1-4799-7074-2/15/$31.00 ©2015 IEEE.

[3] Ross Thompson and James E. Stine, "An IEEE 754 Double-Precision Floating-Point Multiplier for Denormalized and Normalized Floating-Point Numbers", International conference on IEEE 2015.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.