VLSI IMPLEMENTATION OF IEEE 754 FLOATING POINT MULTIPLIER USING PARTITION TECHNIQUE

[1]Markandeya Pandey, [2] Prof. Balram Yadav
[1]Research Scholar, [2]Assistant Professor
Scope college of Engineering

Abstract— Due to advancement of new technology in the field of VLSI and Embedded system, there is an increasing demand of high speed and low power consumption processor. Speed of processor greatly depends on its multiplier as well as adder performance. In spite of complexity involved in floating point arithmetic, its implementation is increasing day by day. Due to which high speed adder architecture become important. Several adder architecture designs have been developed to increase the efficiency of the adder. In this paper, we introduce an architecture that performs high speed IEEE 754 floating point multiplier using carry select adder (CSA). Here we are introduced two carry select based design. These designs are implementation Xilinx Vertex device family.

Keywords— IEEE754, Single Precision Floating Point (SP FP), Double Precision Floating Point (DP FP), Binary to Execess-1 Converter

1. INTRODUCTION

The real numbers represented in binary format are known as floating point numbers. Based on IEEE-754 standard, floating point formats are classified into binary and decimal interchange formats. Floating point multipliers are very important in dsp applications. This paper focuses on double precision normalized binary interchange format. Figure 1 shows the IEEE-754 double precision binary format representation. 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).

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

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

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

Sign Bit	Biased Exponent	Significand
1-bit	8/11-bit	23/52-bit

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

IEEE 754 Standard Floating Point Multiplication Algorithm
A brief overview of floating point multiplication has been explained below [5-6].

- Both sign bits S1, S2 are need to be Xoring together, then the result will be sign bit of the final product.
- Both the exponent bits E1, E2 are added together, then subtract bias value from it. So, we get exponent field of the final product.
- Significand bits Sig1 and Sig2 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.

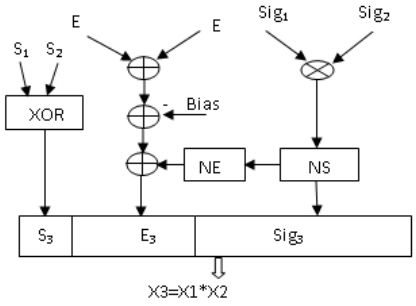


Figure 2: IEEE754 SP FP and DP FP Multiplier Structure, NE: Normalized exponent, NS: Normalized Significand

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.

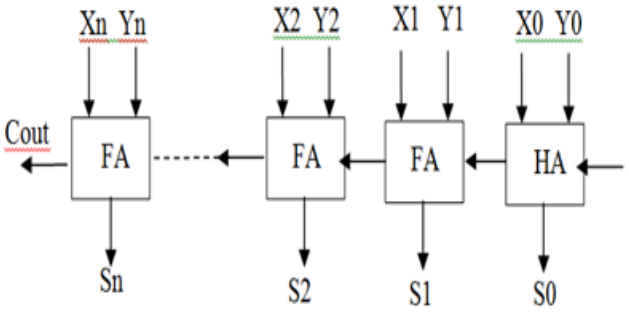


Figure 3: Parallel Adder (n=7 for SPFP and n=10 for DPFP)

An n-bit parallel adder has one half adder and n-1full adders if the last carry bit required. But in 754 multiplier's exponent adder, last carry out does not required so we can use XOR Gate instead of using the last full adder. It not only reduces the area occupied by the circuit but also reduces the delay involved in calculation. For SPFP and DPFP multiplier's exponent adder, here we Simulate 8 bit and 11 bit parallel adders respectively as show in figure 4.

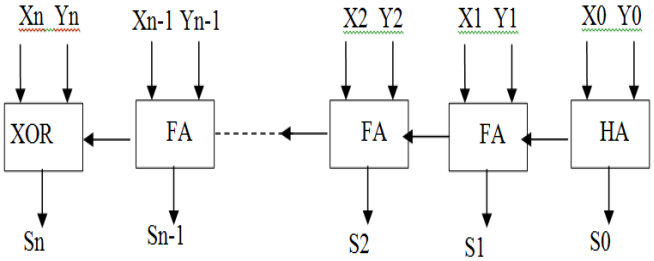


Figure 4: Modified Parallel Adder (n=7 for SPFP and n=10 for DPFP)

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 5.

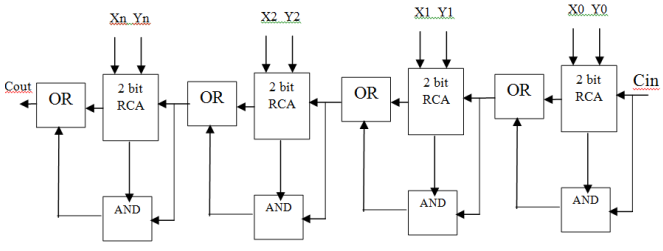


Figure 5: Carry Skip Adder

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 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.

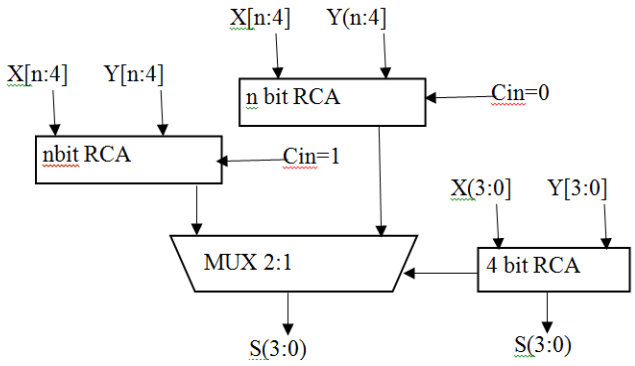
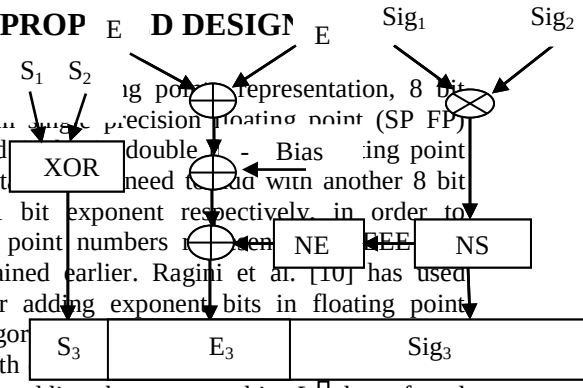


Figure 6: Carry Select Adder

2. PROP E D DESIGN E



In IEEE754 stan ng po representation, 8 bit Exponent field i precision floating point (SP FP) representation and double - Bias ing point (DP FP) represent need d ad with another 8 bit exponent and 11 bit exponent respectively in order to multiply floating point numbers en EEE standard as explained earlier. Ragini et al. [10] has used parallel adder for adding exponent bits in floating point multiplication algor modified CSA with RCA and BEC for adding the exponent bits. We have found the improved area of 8-bit modified X3=X1*X2 th RCA and BEC over the 8-bit modified CSA with dual RCA.

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

o Sign bit
To calculate the
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.

o 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.

o 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 significant 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.

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;
```

3. SIMULATION RESULT

VHDL is exceptionally versatile, inferable from its engineering, permitting fashioners, electronic plan mechanization organizations and the semiconductor business to try different things with new dialect ideas to guarantee great plan costs and information interoperability.

Having planned the different DSP arrangements, we currently continue to the product union of this plans utilizing VHDL. In the accompanying segments, we have set up the ideal channel yields utilizing separate VHDL codes for each plan. The codes of the plans have been displayed in the Appendix. The construction so was examined effectively executed or blended on XILINX ISE plan suite 6.2i.

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.

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.

Propagation Delay

Generally, the ideal condition of the result is the output of the digital circuit should from level to level or level to level 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 to level and level to level .

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.

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.

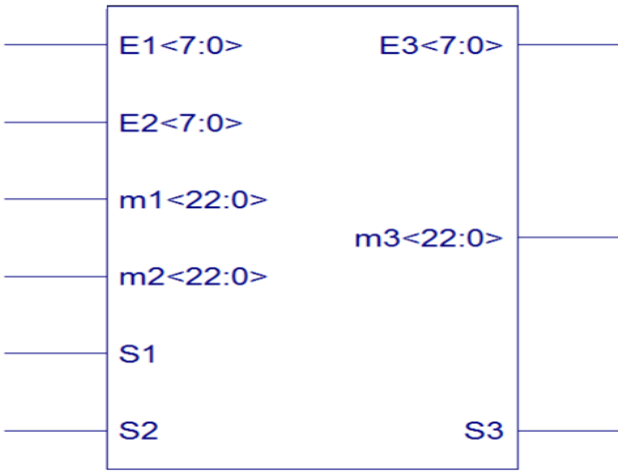


Figure 7: View Technology Schematic of SPFP Multiplier

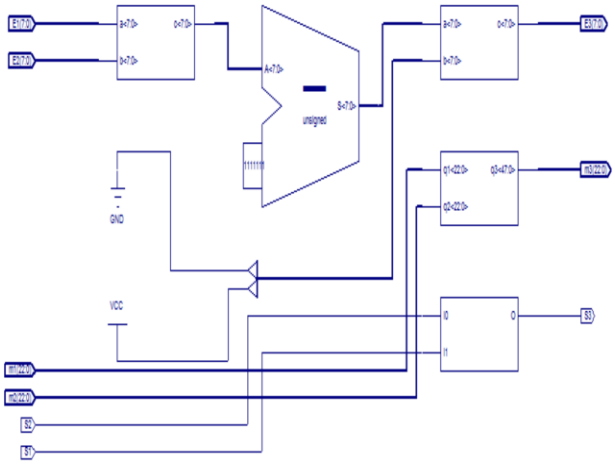


Figure 8: RTL View of SPFP Multiplier

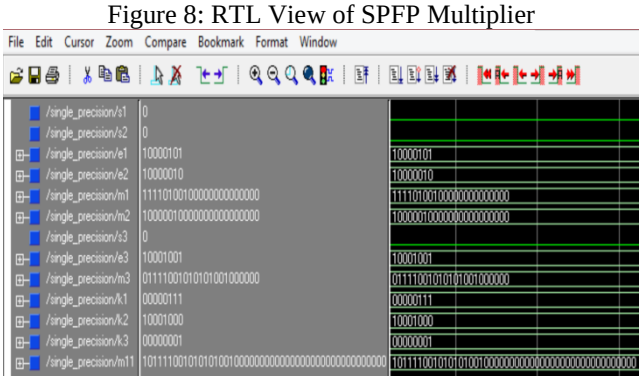


Figure 9: Output Waveform of SPFP Multiplier

```
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
```

Figure 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.

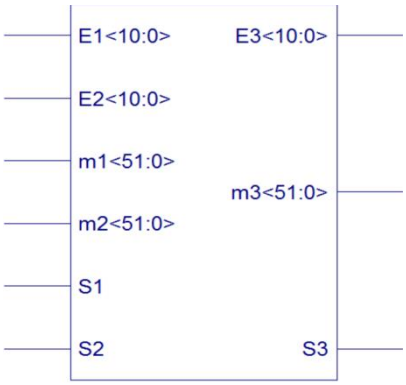


Figure 11: View Technology Schematic of DPFP Multiplier

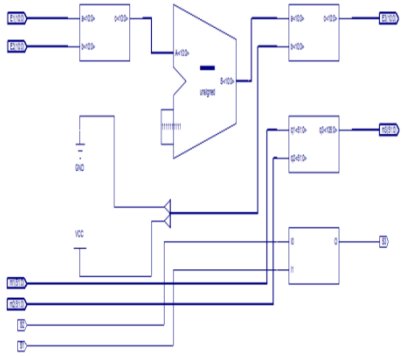


Figure 12: RTL View of DPFP Multiplier

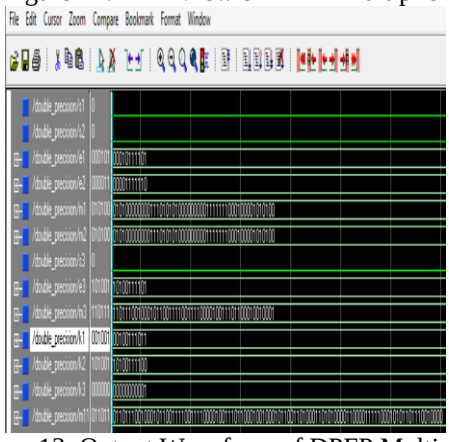


Figure 13: Output Waveform of DPFP Multiplier

```
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
```

Figure 14: Device Utilization summary of Double Precision

Floating Point Multiplier

Table I: 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

4. CONCLUSION

IEEE754 standardize two basic formats for representing floating point numbers namely, single precision floating point and double precision floating point. Floating point arithmetic has 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 by using two different adders namely modified CSA with dual RCA and modified CSA with RCA and BEC.

Among all two adders, modified CSA with RCA and BEC is the least amount of Maximum combinational path delay (MCDP). Also, it takes least number of slices i.e. occupy least area among all 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, “Parameterisable 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.

