JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-6, June 2019 E-ISSN: 2347-2693

Design and Analysis of Single and Double Precision Floating Point Matrix Multiplier using Partition Multiplier Method

Manjusha Kumari[1*], Vijay Yadav[2]

[1,2]Dept. of Electronics and Communication, LNCT, Bhopal

DOI: https://doi.org/10.26438/ijcse/v7i6.10411044 | Available online at: www.ijcseonline.org

Accepted: 09/Jun/2019, Published: 30/Jun/2019

***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

I. 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).

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

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

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

Sign Bit	Biased	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 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.

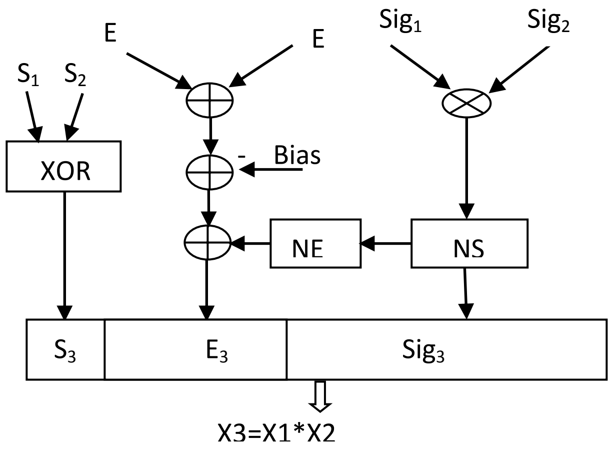

Figure 2: IEEE754 SP FP and DP FP Multiplier Structure, NE: Normalized exponent, NS: Normalized Significand

- Normalize the product found in step 3 and change the exponent accordingly. After normalization, the leading "1 "will become the hidden bit.

II. DIFFERENT TYPES OF ADDER

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.

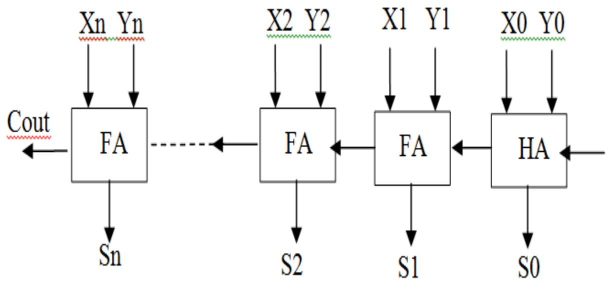

Figure 3: Parallel Adder (n=7 for SPFP and n=10 for DPFP)

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.

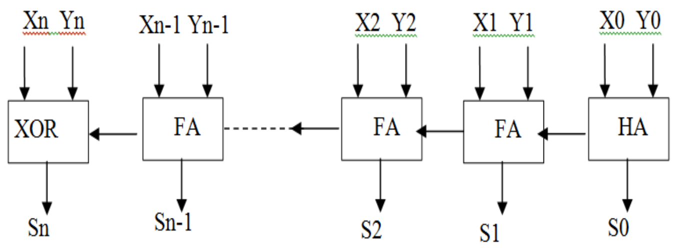

Figure 4: Modified Parallel Adder (n=7 for SPFP and n=10 for DPFP)

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.

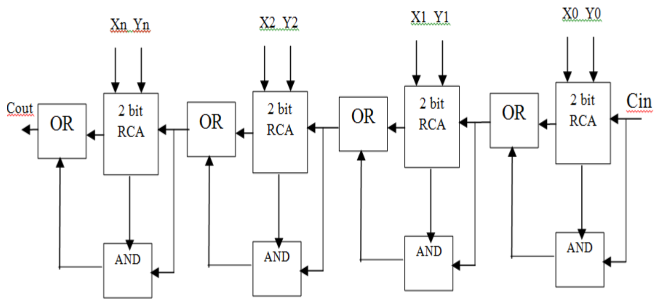

Figure 5: Carry Skip Adder

III. PROPOSED DESIGN

Proposed Parallel-Parallel Input and Multi Output(PPI-MO)
In this design, we opted for faster operating speed by increasing the number of multipliers and registers performing the matrix multiplication operation. From equation 2 we have derived for parallel computation of 3 × 3 matrix-matrix multiplication and the structure is shown in figure 6.
For an n×n matrix – matrix multiplication, the operation is performed using n^2 number of multipliers, n^2 number of registers and n^2-n number of adders. The registers are used to store the partial product results. Each of the n^2 number of multipliers has one input from matrix B and the other input is obtained from a particular element of matrix A.

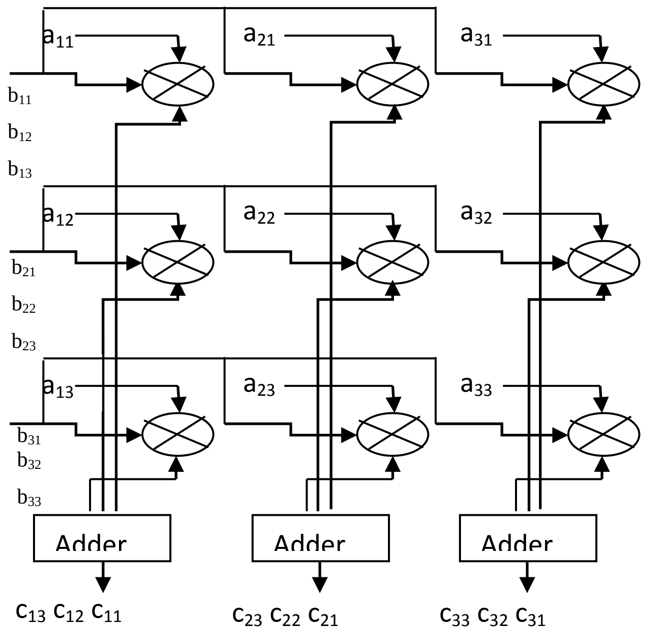

Figure 1: Proposed PPI – MO Design for n = 3

The dataflow for matrix B is in row major order and is fed simultaneously to the particular row of multipliers such that the i^{th} row of matrix B is simultaneously input to the i^{th} row of multipliers, where $1 < i < n$. The elements of matrix are input to the multipliers such that, $(j,i)^{th}$ element of matrix A is input.

The $(i, j)^{th}$ multiplier, where$1 < i,j < n$. The resultant products from each column of multipliers are then added to give the elements of output matrix C. In one cycle, n elements of matrix C are calculated, so the entire matrix the elements of matrix C are obtained in column major order with n elements multiplication operation requires n cycles to complete.

Let us consider the example of a 3×3 matrix – matrix multiplication operation, for a better analysis of the design (as shown in figure 1). The hardware complexities involved for this design are 9 multipliers, 9 registers and 6 adders. Elements from the first row of matrix B (b_{11} b_{12} b_{13}) are input simultaneously to the first row of multipliers (M_{11} M_{12} M_{13}) in 3 cycles. Similarly, elements from other two rows of matrix B are input to the rest two rows of multipliers. A single element from matrix A is input to each of the multipliers such that, $(j,i)^{th}$ element of matrix A is input to the multiplier M_{ij}, where $1 < i,j < 3$. The resultant partial products from each column of multipliers (M_{1k} M_{2k} M_{3k} where $1 < k$ 3) are added up in the adder to output the elements of matrix C. In each cycle, one column of elements from matrix C is obtained (C_{1k} C_{2k} C_{3k} where$1 < k < 3$) and so the entire matrix multiplication operation is completed in 3 cycles.

IV. FLOATING POINT

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. Ragini et al. [10] has used parallel adder for adding exponent bits in floating point multiplication algorithm. We proposed the use of 8-bit modified CSA with dual RCA and 8-bit modified CSA with RCA and BEC for adding the exponent bits. We have found the improved area of 8-bit modified Carry select adder with RCA and BEC over the 8-bit modified CSA with dual RCA.

- 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.

- 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.

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;
```

V. SIMULATION RESULT

All the designing and experiment regarding algorithm that we have mentioned in this paper is being developed on Xilinx 6.2i updated version. Xilinx 6.2i has couple of the striking features such as low memory requirement, fast debugging, and low cost. The latest release of ISE^{TM} (Integrated Software Environment) design tool provides the low memory requirement approximate 27 percentage low. ISE 6.2i that provides advanced tools like smart compile technology with better usage of their computing hardware provides faster timing closure and higher quality of results for a better time to designing solution.

These designs were compared with IEEE-754 floating point multiplier architecture proposed by Ragini et al. [2] to show for the improvements obtained.

So Ragini et al. [2] architecture is best in all these architectures. Implementing the Ragini et al. [2], proposed architecture IEEE-754 floating point design has been captured by VHDL and the functionality is verified by RTL and gate level simulation. To estimate the number of slice, number of 4-i/p LUTs and maximum combinational path delay (MCPD).

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

VI. 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.

REFERENCE

[1] Lakshmi kiran Mukkara and K.Venkata Ramanaiah, “A Simple Novel Floating Point Matrix Multiplier VLSI Architecture for Digital Image Compression Applications”, 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018) IEEE.

[2] 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.

[3] 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.

[4] 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.

[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] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.