International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
NATIONAL CONFERENCE on Developments, Advances & Trends in Engineering Sciences
(NCDATES- 09th & 10th January 2015)

RESEARCH ARTICLE OPEN ACCESS

Pipelined High Speed Double Precision Floating Point Multiplier Using Dadda Algorithm Based on FPGA

J.Rupesh Kumar, G.Ram Mohan, Sudershanraju.Ch
M. Tech Scholar, Dept. of ECE, BIT Institue of Technology
Asst. Prof. Dept. of ECE, BIT Institue of Technology
HOD, Dept. of ECE, BIT Institue of Technology

ABSTRACT
Floating Point (FP) multiplication is widely used in large set of scientific and signal processing computation. Multiplication is one of the common arithmetic operations in these computations. A high speed floating point double precision multiplier is implemented in HDL. This paper presents a high speed binary double precession floating point multiplier based on Dadda Algorithm. To improve speed multiplication of mantissa is done using Dadda multiplier replacing Carry Save Multiplier. In addition, the proposed design is compliant with IEEE-754 format and handles over flow, under flow, rounding and various exception conditions. The design achieved the operating frequency of 414.714 MHz with an area of 648 slices.
KEY WORDS: Dadda Algorithm, Double precision, Floating point, Multiplier, IEEE-754, Verilog HDL.

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 I shows the IEEE- 754 double precision binary format representation. Sign (S) is represented with one bit, exponent and fraction (Mor 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).

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

$$Value = (-1^{Sign\ bit}) * 2^{(Exponent - 1023)} * (1.Mantissa) \quad (2)$$

Floating point implementation has been the interest of many researchers. In an IEEE-754 single precision pipelined floating point multiplier is implemented with custom 16/18 bit three stage pipelined floating point multiplier, that doesn't support rounding modes [1]. L.Louca, T.A.Cook, W.H. Johnson [2] implemented a single precision floating point multiplier by using a digit-serial multiplier. The design achieved 2.3 MFlops and doesn't support rounding modes. The multiplier handles the overflow and underflow cases but rounding is not implemented. The design achieves 30 I MFLOPs with latency of three clock cycles. The multiplier was verified against Xilinx floating point multiplier core.

Figure1. IEEE 754 Double Precision Floating Point Format. The double precision floating point multiplier presented here is based on IEEE-754 binary floating standard. We have designed a high speed double precision floating point multiplier using Verilog language. It operates at a very high frequency of 414.714 MFlops and occupies 648 slices. It handles the overflow, underflow cases and rounding mode.

II. FLOATING POINT MULTIPLICATION ALGORITHM

1. Adding the exponent of the two numbers then subtracting the bias from their result.
2. Multiplying the significand of the two numbers
3. Calculating the sign by XORing the sign of the two numbers. In order to represent the multiplication result as a normalized number there should be I in the MSB of the result (leading one).

The following steps are necessary to multiply two floating point numbers.
1. Multiplying the significand i.e. (I.MI * I.M2)
2. Placing the decimal point in the result
3. Adding the exponents i.e. (E I + E2 - Bias)
4. Obtaining the sign i.e. sl xor s2
5. Normalizing the result i.e. obtaining I at the MSB of the results "significand"
6. Rounding the result to fit in the available bits
7. Checking for underflow/overflow occurrence

III. IMPLEMENTATION OF DOUBLE PRECISION FLOATING POINT MULTIPLTER

Double Precision:

In case of double precision 64 bits format, the MSB for normaliza- tion, Exponent is represented in 11 bits which is biased to 1023 and MSB of double precision is reserved for sign bit as shown in Figure 1 [9].

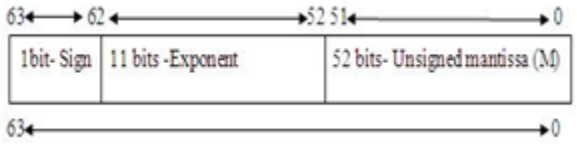

Figure 1. IEEE Double Precision Data Format

The value of the floating point number represented in double precision format is F= (-1)S1.M 2E-1023
where 1023 is the value of bias in double precision data format. Exponent E ranges between 1 to 2046, and E = 0 and E = 2047 are reserved for special values. The double precision format offers range from 2-1023 to 2+1023, which is equivalent 10-308 to10308[7].

Floating Point Multiplier

Multipliers are key components of many high Performance systems such as FIR filters, Microprocessors, Digital Signal Processors etc.

Systems performance is generally determined by the performance of the multiplier, because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a critical issue for an effec- tive system design.

Floating Point Multiplication Algorithm

Multiplication of floating point numbers F1 and F2is a seven step process. The algorithm of IEEE compatible floating point multiplier is listed below. To multiply two floating point numbers the following is done:
1. Obtaining the sign; i.e. S1 xor S2.
2. Adding the exponents and subtracting the bias value; i.e. (E1+ E2 – Bias).
3. Multiplying the significand; i.e. (1.M1*1.M2).
4. Placing the decimal point in the significand result.
5. Normalizing the result; i.e. obtaining 1 at the MSB of the results significand.
6. Rounding the result to fit in the available bits.
7. Checking for underflow/overflow occurrence [5].

Pipelined Double Precision Floating Point Multiplier The aim in developing a floating point multiplier was to be pipeline that is each module in order to produce result at every clock cycle. In order to enhance the performance of the multiplier, three pipelining stages are used to divide the critical path thus increasing the operating frequency of the multiplier. As the number of pipeline stages is increased, the path delays of each stage are decreased and the overall performance of the circuit is improved. By pipelining the floating point multiplier, the speed increased, however, the area increased as well.

Different coding structures were tried in VHDL code in order to minimize size. Multiplying two numbers in floating point format is done by three main module i.e. multiplier module, rounding module and exceptions module. Figure 2 shows the Pipelined floating point multiplier with rounding and exceptions module. The multiplier, rounding and exceptions module that are implement independent and are done in parallel. All the modules in the FPM are realized using VHDL. In his paper, pipelined floating point multiplier structure organized in a three-stage.
Stage 1 performs the the mantissas, and the exclusive-or of the signs.
Stage 2 takes these results from stage 1 as inputs, performs the rounding operations.
Stage 3 checks the various exceptions conditions like overflow, underflow, invalid and inexact.

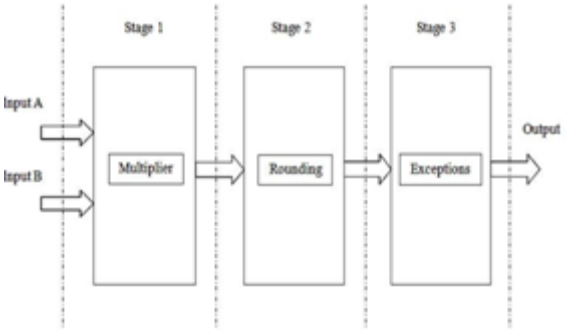

Figure 2. Pipelined Floating Point Multiplier with Rounding and Exceptions

Double precision floating point multiplier has two 64 bits inputs and one 64 bits output. 52 bits, 1 bit is added to the MSB for normalization, Exponent

is represented in 11 bits and the MSB of double precision is used for sign bit. In this paper, pipelined double precision floating point multiplier with rounding and exceptions is presented. Multiplying two numbers in double precision floating point format is done by three modules: multiplier module, rounding module and exceptions module. Top module of double precision floating point multiplier as shown in Figure 3.

Multiplier Module

The double precision multiplication operation is performed in the multiplier module. The Multiplier module receives two 64-bit floating point numbers operand A and operand B. First these numbers are unpacked by separating the numbers into sign, exponent, and mantissa bits. Pipelined double precision floating point multiplier as shown in Figure 4.

Multiplier module has three outputs i.e. sign, exponent and product result. Double precision floating point multiplication is carried out in following three units.

- **Sign calculation unit:**

In this unit, the sign of the product is obtain by performing a XOR operation on the sign bits of the two input operands. In double precision, MSB (63 bit) of the operand is used for sign bit.

Multiplying two numbers result in a negative sign number if one of the multiplied numbers is of a negative value.

signresult = signa xor signb

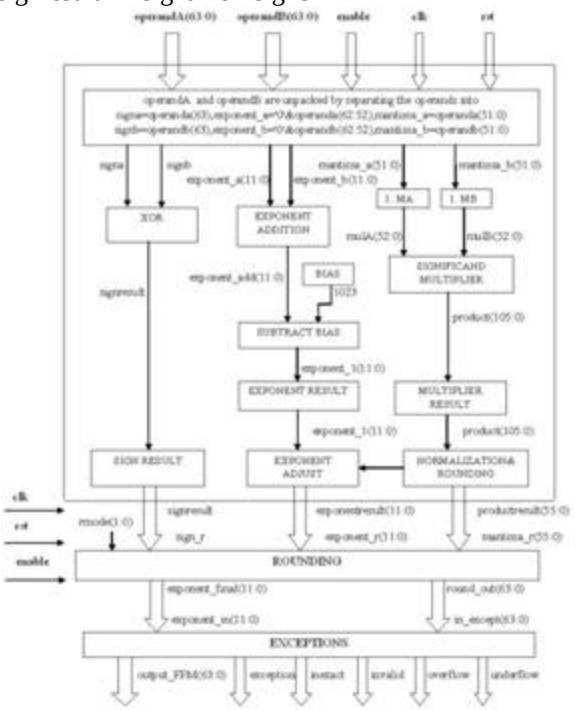

Figure 4. Pipelined double precision floating point multiplier

Exponent calculation unit:

The unsigned adder is used for adding the exponent of the first input to the exponent of the second input and subtracting the Bias (1023) from the addition result.

exponentresult = exponent_a + exponent_b – Bias

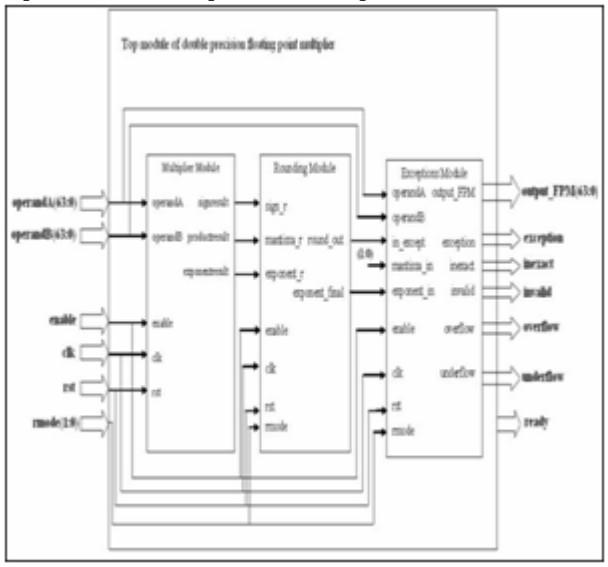

Figure 3. Top module of double precision floating point multiplier

IV. Simulation & Synthesis Results

The double precision floating point multiplier has been coded in VHDL. For simulation and synthesis purpose, Xilinx Integrated Software Environment ISE 13.2 software tool has been used. Pipelined double precision floating point multiplier was simulated in ISE simulator and synthesized using Xilinx ISE 13.2 tool. The double precision floating point multiplier is targeting on Xilinx Virtex-6 xc6vlx75t-3ff484 device. Double precision floating point multiplier has multiplier, rounding and exceptions module.

The RTL view and simulation results of complete double precision floating point multiplier are shown in following section.

A. RTL view of Double Precision Floating Point Multiplier

Top module of double precision Floating point multiplier consist of three modules multiplier, rounding and exceptions module. The IEEE standard specifies four rounding modes infinity, and round to negative infmity. Table 1 shows the rounding modes selected for various bit combinations of rmode. Based on the rounding changes to the mantissa

corresponding changes has to be made in the exponent part also.

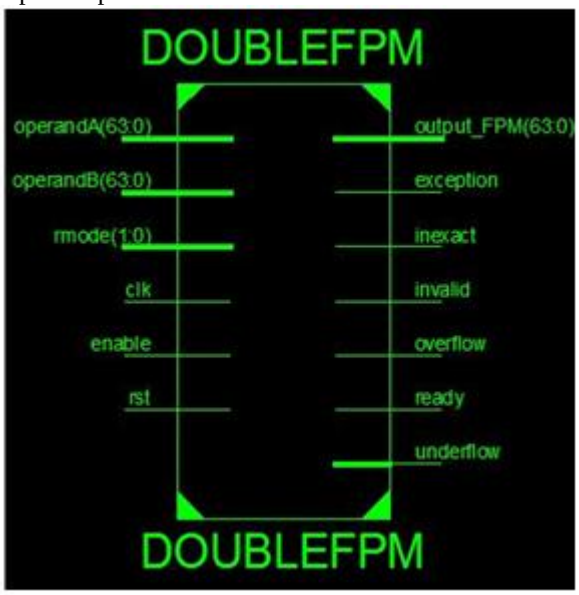

Figure 5. Top RTL view of floating point multiplier

V. Simulation Results of Double Precision Floating Point Multiplier

Double precision floating point multiplier has two 64 bits inputs and one 64 bits output output FPM. It as also two bits rmode input signal which is used for rounding mode. Output signals of underflow, overflow, inexact, exception, and invalid will be asserted if the conditions for each case exist.

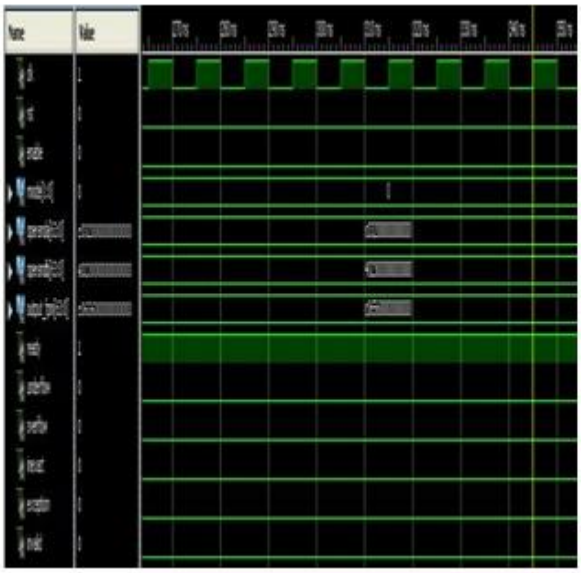

Figure 6. Simulation results of double precision floating point multiplier

Multiplication of two floating point numbers:
Inputs:
A and B, where A= -18.0
and B = 9.5 Binary representation of the
operands: A = -10010.0
B = +1001.1
Normalized representation of the
operands: A = $-1.001*2^4$
B = $+1.0011*2^3$
IEEE representation of the
operands: operand A = 1
10000000011
0010000000000000000000000000000000000000000000000000
00000000
00 = 64‟hC032000000000000
operand B = 0 10000000010
0011000000000000000000000000000000000000000000000000
00000000
00

Outputs:
= 64‟ h402300
A * B = -18.0 * 9.5 = -1.0101011 * 21030-1023
= - 10101011.0 = -171.
output_FPM=
64‟hC065600000000000000000000000
underflow =0, overflow=0, inexact= 0, exception=0,
invalid= 0, ready=1

VI. Conclusion

In this paper, double precision floating point multiplier has been coded with VHDL. Pipelining technique is used for synthesis of the double precision floating point multiplier. Double precision floating point multiplier using three stage pipelining technique achieved the maximum frequency of 489.045 MHz with minimum delay 2.045 ns and area of 888 slices. The pipelined double precision FPM supports the IEEE-754 binary interchange format, targeted on a Xilinx Virtex-6 xc6vlx75t-3ff484 device. Pipelined double precision floating point multiplier is compared with other floating point multiplier it provide high speed with minimum delay. Pipelined double precision floating point multiplier also support rounding mode and handle various exceptions like under flow, overflow and invalid. The double precision floating point multipliers was simulated in ISE simulator and synthesized using Xilinx (integrated software environment) ISE 13.2 tool.

References

[1] Addanki Puma Ramesh, A.V.N. Tilak and A.M. Prasad, “An FPGA Based High Speed IEEE-754 Double Precision Floating Point Multiplier using Verilog”, IEEE International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System

(ICEVENT), Tiruvannamalai, pp. 1– 5, January 2013.

[2] Riya Saini and R.D.Daruwala, "Efficient Implementation of Pipelined Double Precision Floating Point Multiplier", International Journal of Engineering Research and Applications, Vol. 3, Issue 1, pp.1676-1679,January -February 2013.

[3] Manish Kumar Jaiswal and Ray C.C.Cheung, "Area-Efficient Architectures for Double Precision Multiplier on PGA, with Run-Time-Reconfigurable Dual Single Precision Support", Elsevier Microelectronics Journal,pp. 421–430, March 2013.

[4] Anna Jain, Baisakhy Dash and Ajit Kumar Panda,"FPGA Design of a Fast 32-bit Floating Point Multiplier Unit", IEEE Conference Publications, pp. 545–547, 2012.

[5] Addanki Purna Ramesh and Rajesh Pattimi, "High Speed Double Precision Floating Point Multiplier", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 1, Issue 9, pp. 647–650, November 2012.

[6] Xia Hong and Jia Jinging, "Research and Optimization on Rounding Algorithms for Floating-Point Multiplier", IEEE Conference Publications, International Conference on Computer Science and Electronics Engineering, pp. 137 – 142, 2012.

[7] Puneet Paruthi,Tanvi Kumar and Himanshu Singh, "Simulation of IEEE 754 Standard Double Precision Multiplier using Booth Techniques", International Journal of Engineering Research and Applications,Vol. 2, Issue 5, pp. 1761-1766, September-October 2012.

[8] B.Sreenivasa Ganesh, J.E.N.Abhilash and G. Rajesh Kumar, "Design and Implementation of Floating Point Multiplier for Better Timing Performance", International Journal of Advanced Research in Computer Engineering & Technology, Vol. 1, Issue 7, September 2012.

[9] Aniruddha Kanhe, Shishir Kumar Das and Ankit Kumar Singh, "Design and Implementation of Floating Point Multiplier based on Vedic Multiplication Communication, Information & Computing Technology (ICCICT),Oct. 19-20, Mumbai, India, pp. 1–4, 2012.

[10] Mohamed Al-Ashrafy, Ashraf Salem and Wagdy Anis, "An Efficient Implementation of Floating Point Multiplier", IEEE Electronics, Communications and Photonics Conference (SIECPC), Saudi International, pp.1 - 5, 2011.