Synthesis of area Optimized 64 Bit Double Precision Floating Point Multiplier Using VHDL

Kanwaljeet Kaur, Parminder Singh Jassal

***Abstract*— Floating point arithmetic is widely used in many areas. IEEE Standard 754 floating point is the most common representation today for real numbers on computers. The way of floating point operations are executed depends on the data format of the operands. IEEE standards specify a set of floating point formats for single precision and double precision. This paper presents synthesis of double precision floating point multiplier using VHDL. In this paper breakdown technique is used for Synthesis of double precision floating point multiplier. Double precision floating point multiplier with minimum delay 9.251 ns and with an area of 599 no of slices LUTS. Double precision FPM targeted on a Xilinx Virtex-6 xc6vlx75t-3ff484 device.The double precision floating point multiplier was simulated in MODEL SIM simulator and synthesized using Xilinx ISE 13.2 tool.) floating point multiplication is widely used in large set of scientific and signal processing computation. Floating Point Multiplication is one of the common arithmetic operations in these computations. An area optimized floating point double precision multiplier is implemented on a Virtex-6 FPGA. The design achieved with an area of 599 no of slices LUTs as compared to[1].**

***Index Terms*—LUTs, Virtex-6 FPGA, Xilinx ISE 13.2, FPM.**

I. INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. Multiplication based operations such as multiply and accumulate(MAC) and inner product are among some of the frequently used computation- intensive arithmetic functions currently implemented in many digital signal processing (DSP) applications such as convolution, fast fourier transform (FFT), filtering and in

microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of area optmized multiplier.[1]Floating point number system is a common choice for many scientific computations due to its wide dynamic range feature. For instance,floating point arithmetic is widely used in many areas, especially in scientific computation, numerical processing, image processing and signal processing. The term floating point is derived from the fact that there is no fixed number of digits before and after the decimal point, that is, the decimal point or binary point can float.[1] There are also representations in which the number of digits before and after the decimal or binary point is fixed; called fixed-point representations. The advantage of floating-point representation over fixed point representation is that it can support a much wider range of values. The floating point numbers is based on scientific notation. A scientific notation is just another way to represent very large or very small numbers in a compact form such that they can be easily used for computations. Binary floating point numbers multiplication is one of the basic functions used in digital signal processing (DSP) application. The IEEE 754 standard provides the format for representation of Binary Floating point numbers in computers. The Binary Floating point numbers are represented in Single and Double formats. The Single precision format consists of 32 bits and the Double precision format consists of 64 bits. The formats are composed of 3 fields; Sign, Exponent and Mantissa.

Floating point number consists of three fields:
1. SIGN (S): It used to denote the sign of the number i.e. 0 represent positive number and 1 represent negative number.
2. SIGNIFICAND OR MANTISSA (M): Mantissa is part of a floating point number which represents the magnitude of the number.
3. EXPONENT (E): Exponent is part of the floating point number that represents the number of places that the decimal point (binary point) is to be moved.Number system is[2] completely specified by specifying a suitable base β, significand (mantissa) M, and exponent E. A floating point number F has the value

F = $M\beta^E$

β is the base of exponent and it is common to all floating point numbers in a system.

Sign Bit	8 Bit Biased Exponent	23Bit Signifiacnd/ Mantissa/ Fraction

Table1.1.IEEE Single Precision Data Format

II. CHOICE OF FLOATING POINT REPRESENTATION

The way floating point operations are executed depends on the specific format used for representing the operands. The choice of a floating point format for the hardware implementation of floating point units is governed by factors like the dynamic range requirements, maximum affordable computational errors, power consumption etc.[6] The exponent bit width decides the dynamic range of floating point numbers while the significand bit width decides the resolution. The dynamic range offered by floating point units is much higher than that offered by fixed point units of equivalent bit width. Larger dynamic range is of significant interest in many computing applications like in multiply - accumulate operation of DSPs. But larger range is not needed in all the applications.

(a) IEEE single precision data format

Kanwaljeet Kaur, Mtech Student, Department Of Electronics, Yadavindra College Of Engineering, Talwandi Sabo(Pb)- INDIA.

Parminder Singh Jassal, Assistant Professor, Department Of Electronics, Yadavindra College Of Engineering, Talwandi Sabo(Pb)- INDIA.

(b) IEEE double precision data format

Sign Bit	11Bit Biased Exponent	52Bit Significand/Fraction/Mantissa

Table1.2.IEEE Doublele Precision Data Format

III. FLOATING POINT MULTIPLICATION

Decimal number are also called floating point because a single number can be represented with one or more significant digits depending on position of the decimal point. Since the point floats between mass of digits that represent the number such numbers are termed as floating point number. floating-point multiplication, by complies with the IEEE 754 Standard, the two mantissas are to be multiplied, and the two exponents are to be added. The sign logic is a simple XOR. In order to perform floating-point multiplication, a simple algorithm is realized:[6]

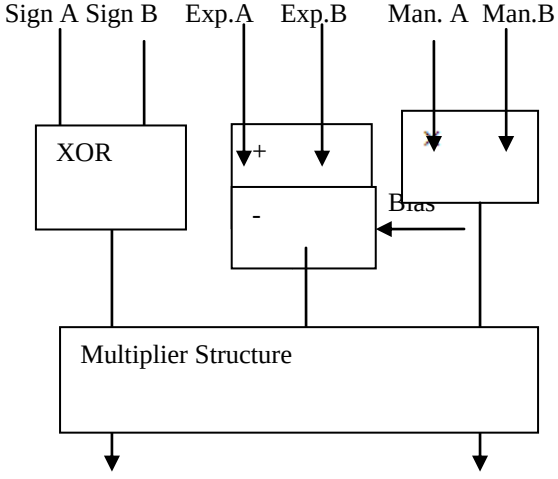
1. Adding the exponent of the two numbers then subtracting the bias from their result.
2. Multiplying the significant of the two numbers
3. Calculating the sign by XOR ing the sign of the two numbers.

2.5THE FOLLOWING STEPS ARE NECESSARY TO MULTIPLY TWO FLOATING POINT NUMBERS

1. Multiplying the significand i.e. (I.MI * I.M2) [1]
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 "significant"
6. Rounding the result to fit in the available bits
7. Checking for underflow/overflow occurrence

IV. IMPLEMENTATION OF DOUBLE PRECISION FLOATING POINT MULTIPLTER

In this paper we implemented a double precision floating point multiplier Figure 1.1 shows the multiplier structure that includes exponents addition, significant multiplication, and sign calculation [1]



1.5 The breakdown of the multiply in module (fJ:lU_mul) is broken up as follows :

product_a = mul_a[23:0] * mul_b[16:0]
product_b = mul_a[23:0] * mul_b[33:17]
product_c = mul_a[23:0] * mul_b[50:34]
product_d = mul_a[23:0] * mutb[52:51]
product_e = mul_a[40:24] * mul_b[16:0]
productj= mul_a[40:24] * mutb[33:17]
product_g = mul_a[40:24] * mul_b[52:34]
product_h = mul_a[52:41] * mul_b[16:0]
product_i =mul_a[52:41] * mul_b[33:17]
productj =mul_a[52:41] * mul_b[52:34]

V. VHDL CODE FOR MULTIPLICATION OF IEEE-754 DOUBLE PRECISION NUMBERS

VHDL code for multiplication of double precision (64-bit) numbers is being developed and then is simulated using Model Sim SE Plus 6.5. VHDL code is break down into small components to handle normalisation, rounding, Exponents are added and significands are further multiplied. Sign bit is computed with XOR operation. Various sets of inputs are fed to the block to get the results. The further part of the document deals with simulation and synthesis results.

6.4 Model Sim Simulation

Consider multiplication of two decimal numbers:
Inputs:
A and B Where A= -18.5 and B = 9.5
Binary representation of operands:
A= -10010.0
B = +1001.1
Normalized representation of operands:
A= - 1.001×2^4
B = + 1.0011×2^3
IEEE representation of the operands:
Operands A = 1 10000000011
0010000000000000000000000000000000000000000000000000 = 64'hc03000000000000
Operands B = 0 10000000010
0011000000000000000000000000000000000000000000000000 = 64'h4023000000000000
Outputs:
A× B = -18.0× 9.5 = -1.0101011×2^(1030-1023)
= -10101011.0 = =171
Output_FPM = 64'Hc065600000000000000000000000
underflow = 0, overflow = 0, inexact = 0, exception = 0, invalid = 0, ready = 1

VI. XILINX SYNTHESIS

This device has following attributes manifests in Table 1.2 shows the Device Utilisation Summary of the VHDL code, so written, it is been observed that number of device parameters used are very less. Hence, an optimum Device Utilisation is obtained. From the timing report obtained, it is found that the maximum combinational path delay is 9.251 ns. Maximum combinational path delay is only for paths that start at an input to the design and go to an output of the design without being clocked along the way.

This design has implemented, simulated on model sim and synthesized for VHDL. Simulation based verification is one of the method for functional verification for design. The test bench form top module that instantiates other module. The simulation verification ensure that design is functionally corrected when tested with a given set of inputs. Though it is not fully complete, by picking random set of inputs as well as corner cases simulation based verification can still yield reasonably good results. The result obtained shown in figure 1.3 and figure 1.4.The following snapshots are take from

Model Sim after timing simulation of the floating point multiplier.

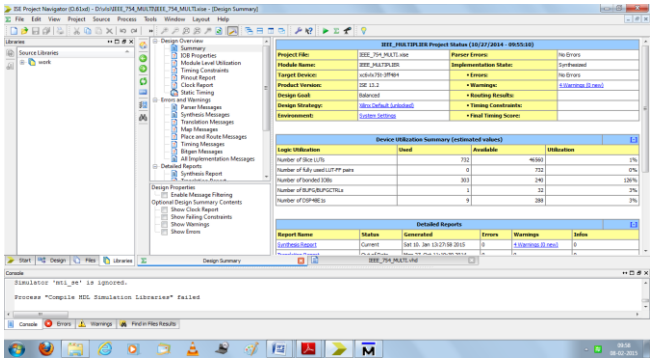


Figure 1.2 Design Summary Of Double Precision Floating Point Multiplier

A= -18.0
B= 9.5
M= 171

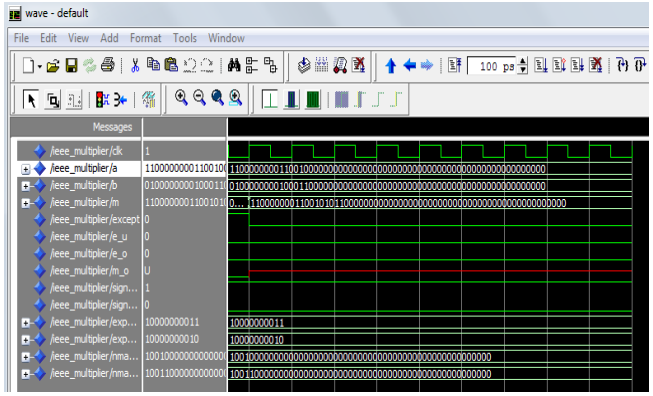


Figure 1.3 Simulation Results Of Double Precision Floating Point Multiplier

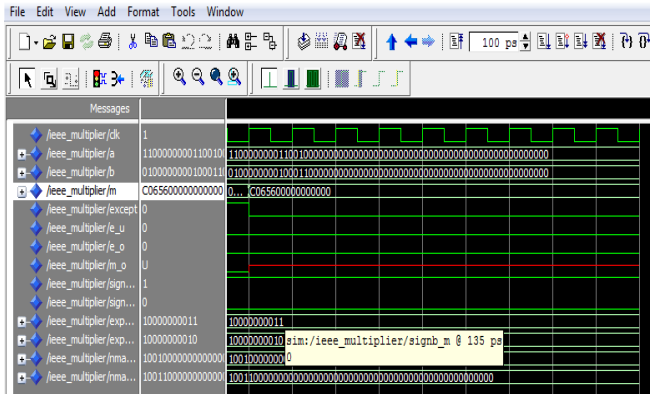


Figure 1.4 Simulation Results Of Double Precision Floating Point Multiplier(Hexadecimal)

Consider multiplication of two decimal numbers:
Inputs:
A and B Where A= 2.2 and B = 2.2
Binary representation of operands:
A= 10.001100110011
B = 10.001100110011
Normalized representation of operands:
A= 1.0001100110011〖 ×2〗 ^1
B = 1.0001100110011×2^1
IEEE representation of the operands:
Operands
A = 01000000000000011001100110000000000000000000000000000000000000000
B = 01000000000000011001100110000000000000000000000000000000000000000

Outputs:
A×B = 2.2 × 2.2 =
Output FPM = 40235BF0A4000000
X = 01000000001000110101101111110000101001000000000000000000000000000

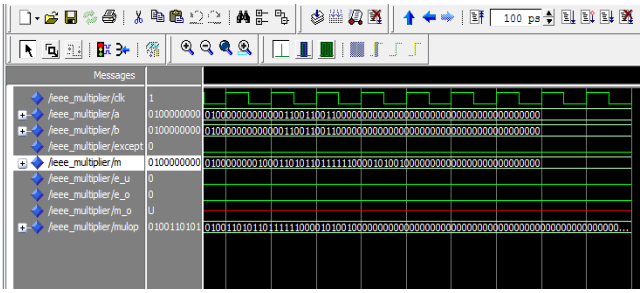


Figure 1.5 Simulation Results Of Double Precision Floating Point Multiplier

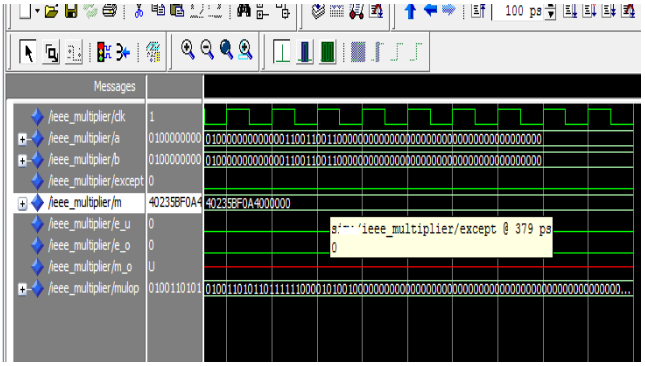


Figure 1.6 Simulation Results Of Double Precision Floating Point Multiplier(Hexadecimal)

Table Of Comparison

Device parameter	Our Work Double Precision	Addanki, Tilak,Prasad[1] Double Precision
No Of LUTs	599	648
Delay	9.251ns	

VII. CONCLUSION AND FUTURE SCOPE

The double precision floating point multiplier supports the IEEE-754 binary interchange format, targeted on a Xilinx Virtex-6 xc6vlx75t-3ff484 FPGA. The design achieved with minimum dealy 9.251 and an area of 599 slices. The implemented design is verified with double precision floating point multiplier [6]. this double precision floating point multiplier uses a breakdown technique that reduced the area compared to single precession. This design handles the overflow, underflow, and normalization rounding mode

VIII. FUTURE SCOPE

• Speed can improved using advanced algorithm like BREAKDOWN TECHNIQUE, BOOTH, ARRAY, DADDA, Algorithm etc.
• ALU Can be designed for DSP Applications.

REFERENCE

[1.] Addanki Puma Rameshl, A. V. N. Tilak2, A.M.Prasad " An FPGA Based High Speed IEEE-754 Double Precision Floating Point Multiplier using Verilog." Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT), IEEE International Conference on Print ISBN: 978-1-4673-5300-7 January 2013.

[2.]R Dhanabal, Ushasree G, Dr Sarat kumar sahoo " VLSI Implementation of a High Speed Single Precision Floating Point Unit Using Verilog" Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013) ISSN 978-1-4673-5758 2013.

[3.] R. Sai Siva Teja1, A. Madhusudhan " FPGA Implementation of Low-Area Floating Point Multiplier Using Vedic Mathematics" International Journal of Emerging Technology and Advance Engineering ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 12, December 2013.

[4.] Shaifali, Sakshi, "FPGA Design of Pipelined 32-bit Floating Point Multiplier" IJCEM International Journal of Computational Engineering & Management, Vol. 16 Issue 5, ISSN (Online): 2230-7893 September 2013.

[5.] Anurag Sharma "BIST Architecture and Implementation of 64-Bit Double Precision Floating Point Multiplier Using VHDL" International Journal of Scientific Engineering and Technology ISSN : 2277-1581 Volume No.2, Issue No.8, pp : 776-779 1 Aug. 2013.

[6.] Manish Kumar Jaiswal Ray C.C. Cheung, "Area-Efficient Architectures for Large Integer and Quadruple Precision Floating Point Multipliers" 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines. ISSN 978-0-7695-4699 2012.

[7.] Addanki Purna Ramesh, Rajesh Pattimi, "High Speed Double Precision Floating Point Multiplier" International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 9 ISSN (Print) : 2319-5940 ISSN (Online) : 2278-102 , November 2011.

[8.] Geetanjali Wasson, "IEEE-754 compliant Algorithms for Fast Multiplication of Double Precision Floating Point Numbers" International Journal of Research in Computer Science Volume 1 Issue 1pp. 1-7 ISSN 2249-8257 2011.

[9.] Mohamed Al-Ashrafy, Ashraf Salem, Wagdy Anis, "An Efficient Implementation of Floating Point Multiplier" IEEE ISSN 978-1-4577-0069 2011