Design of Double Precision Floating Point Multiplier using VHDL

Er. Namrata R. Patel[1] Prof. Hetal G. Bhatt[2]
[1,2]Department of Electronics and Communication Engineering
[1,2]Sankalchand Patel College of Engineering, India

Abstract— Floating point arithmetic is widely used in many areas, especially in scientific computation and signal processing. Now a day the main applications of floating points are in the field of medical imaging, biometrics, motion capture and audio applications. Multipliers play an important role in digital signal processing and other applications. A system's performance is generally determined by the performance of the multiplier, because the multiplier is the slowest element in the system. The proposed design is compliant with IEEE-754 format and handles overflow, underflow, rounding and exception conditions. The design achieved the operating frequency of 35.89 MHz.

Key words: Double precision, Floating point, Multiplier, FPGA, IEEE-754 standard

I. INTRODUCTION

Floating point numbers are binary representation of real numbers. Based on IEEE-754 standard, floating point formats can be classified into binary interchange format and decimal interchange format. Floating point arithmetic is very important in DSP applications. This paper focuses on double precision normalized binary format. Figure 1 shows the IEEE- 754 single precision and double precision binary representation. Sign (S) is one bit long, exponent (E) and fraction (M or Mantissa) are eleven and fifty two bits long 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 for double precision. The real number is represented by equation (1).

Sign (1 bit)	Biased Exponent (8 bits)	Significand/fraction/mantissa its)

Fig. 1(a): IEEE single precision format

Sign (1 bit)	Biased Exponent (11 bits)	Significand/fraction/mantissa (52 bits)

Fig.1 (b): IEEE double precision format

Fig. 1: IEE-754 standard single and double precision floating point format

$$Z=(〖-1〗^{s})*2^{((E-bias))}*(1.M) \quad (1)$$

The double precision floating point multiplier presented here is based on IEEE-754 binary floating point standard. We have designed a high speed double precision floating point multiplier using VHDL language and ported on Xilinx Vertex-5 FPGA. It operates at a frequency of 35.89 MHz. It handles the overflow, underflow cases and rounding modes.

II. ALGORITHM FOR FLOATING POINT MULTIPLICATION

The following steps are needs to be performed 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 1 at the
6) MSB of the resultant "significand".
7) Rounding the result to fit in the available bits.
8) Checking for underflow/overflow occurrence.

III. DESIGN OF DOUBLE PRECISION FLOATING POINT MULTIPLIER

In this paper we designed a double precision floating point multiplier with exceptions cases and rounding modes. Figure 2 shows the multiplier structure that includes exponent addition, significand multiplication, and sign calculation. Figure 3 shows the multiplier, exceptions and rounding processes that are independent and are done in parallel.

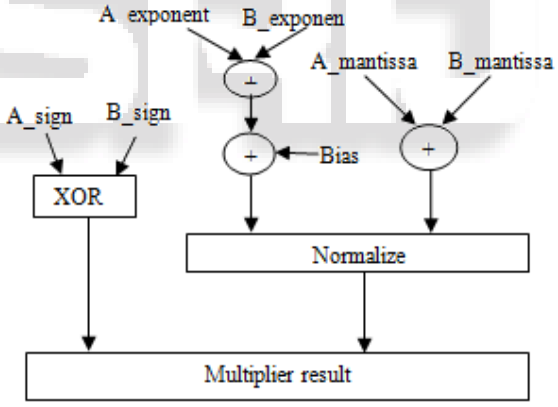

Fig. 2: multiplier structure

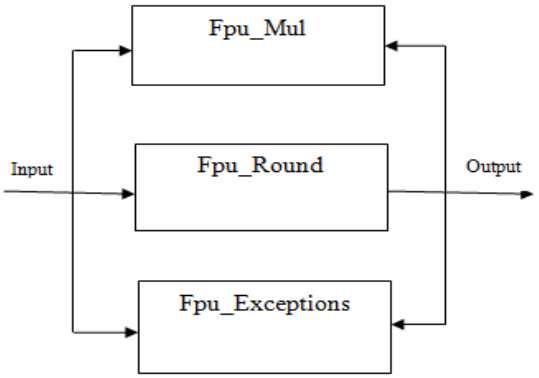

Fig. 3: multiplier structure with rounding and exceptions

A. Multiplier

The black box view is shown in figure 4 for the double precision floating point multiplier. The inputs of Multiplier are two 64-bit floating point numbers. First these numbers are divided in three parts by separating the numbers into sign, exponent, and mantissa fields. For resultant sign we just have to simple XORing two sign bits of operands. The exponents of the two numbers are added and then subtracted with a bias. For double precision floating point number bias is 1023. Mantissa multiplier performs multiplication operation. After this the output of mantissa multiplication is normalized, i.e., if the MSB of the result obtained is not 1, then it is left shifted to make the MSB 1. If there are changes made by shifting then corresponding changes also has to be made in exponent. The multiplication operation is performed in the module (fJ:lU_mul). The mantissa of operand A and the leading '1' (for normalized numbers) are stored in the 53-bit register (mul_ a). The mantissa of operand B and the leading ' 1' (for normalized numbers) are stored in the 53-bit register (mul_b). Multiplying all 53 bits of mul_ a by 53 bits of mul_ b would result in a 106-bit product. 53 bit by 53 bit multipliers are basically not available in the most popular Xilinx and Altera FPGAs, so the whole multiplication would be broken down into smaller multiplies and the results would be added together to give the final 106-bit product. The module (fJ:lU_mul) breaks up the multiply into smaller 24-bit by 17-bit multiplies. The Xilinx Virtex-5 device contains DSP48E slices with 25 by 18 twos complement multipliers, which can perform a 24-bit by 17-bit unsigned multiply.

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]

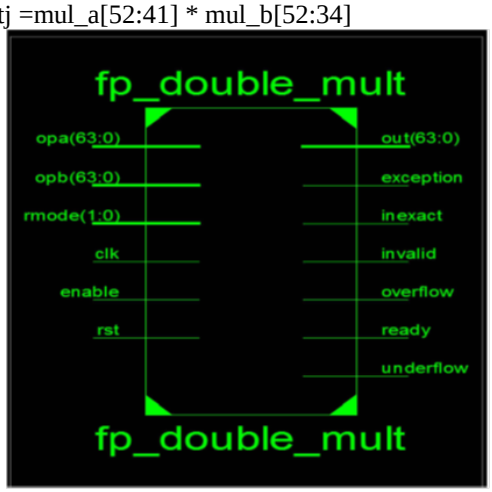

Fig. 4: black box view of double precision floating point multiplier.

The products (a-j) are added together, with the appropriate offsets based on which part of the mul_ a and mul_b arrays they are multiplying. In this work the adders in the Virtex-5 DSP48E slices have been used that follow each 24 by 17 multiply block. The final 106-bit product is stored in the registers. The output will be left-shifted if there is not a '1' at the MSB of product result. The number of leading zeros in register is counted by signal (product_shift). The output exponent will also be reduced by (product_shift). The exponent fields of both operands A and B are added together and then the bias (1023) is subtracted from the sum of A and B. If the exponent result is less than 0, then the (product) register needs to be right shifted by the amount. This value is stored in register (exponent_under). The final exponent of the output operand will be 0 in this case, and the result will be a denormalized number. If exponent_under is greater than 52, then the mantissa will be shifted out of the product register, and the output will be 0, and the "underflow" signal will be asserted. The mantissa output from the (fJ:lU_mul) module is in 56-bit register (product_7). The MSB is a leading '0' to allow for a potential overflow in the rounding module. The first bit '0' is followed by the leading '1' for normalized numbers, or '0' for denormalized numbers. Then the 52 bits of the mantissa follow. Two extra bits follow the mantissa, and are used for rounding. The first extra bit is taken from the next bit after the mantissa in the 106- bit product result of the multiply. The second extra bit is an OR of the 52 LSB's of the 106-bit product.

B. Rounding modes and exceptions

IEEE standard gives four rounding modes which are round to nearest value, round to zero, round to positive infinity, and round to negative infinity. The rounding modes selection for various bit combinations of rmode shown in figure 1. Based on the rounding changes to the mantissa exponent part also be changed.

Bit's combination	Mode of rounding
00	Round nearest even
10	Round to positive infinity
01	Round to zero
11	Round to negative infinity

Table 1: rounding mode selection

In the exceptions module, all of the special cases are Check for, and if any exception case found, the appropriate output is created, and the individual output signals of underflow, overflow, inexact, exception, and invalid will be asserted if the conditions for each case exist.

IV. RESULTS

The design of double precision floating point multiplier was simulated in Modelsim and synthesized using Xilinx ISE 12.2i which was mapped on to Virtex-5 FPGA. The simulation results of 64-bit floating point double precision multiplier are shown in figure5. The 'opa' and 'opb' are the inputs and 'out' is the output. Table2 shows the device utilization for implementing the circuit on Virtex-5 FPGA. Table 3 shows the timing summary of double precision floating point multiplier. The implemented design provides high operating frequency with more accuracy.

Parameters	Values
Frequency (MHz)	35.89 MHz
Minimum period(ns)	27.85 ns

Table 3: timing summary

Vertex 5 5vlx20tff323-1		
Logic utilization	**Used**	**Available**
No. of slice registers (flip-flops)	1824	12480
No. of slice LUTS	3803	12480
No. of bonded IOBs	206	172

Table 2: device utilization summary

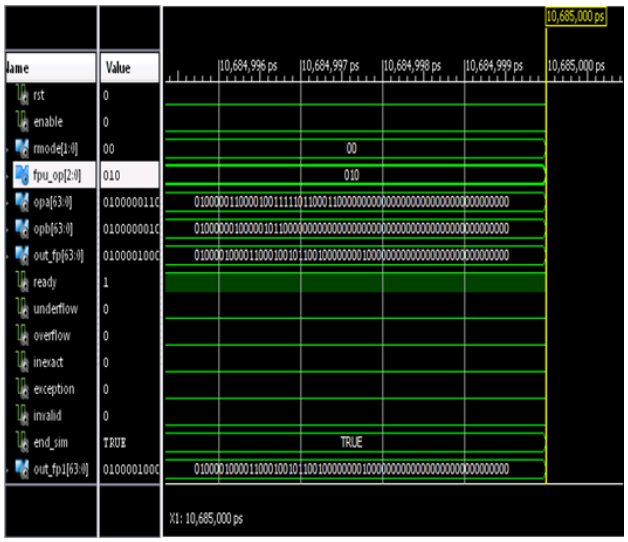

Fig. 5: simulation results of double precision floating point multiplier

V. CONCLUSION

The designed double precision floating point multiplier supports the IEEE-754 binary interchange format, targeted on a Xilinx Virtex-5 FPGA. The design achieved the operating frequency of 35.89 MHz. This design handles the overflow, underflow, exception cases and four rounding modes.

REFERENCES

[1] P. Gayatri, P. Krishna Kumari, V. Vamsi Krishna, T. S. Trivedi, V.Nancharaiah, "Design of Floating Point Multiplier Using Vhdl", International Journal of Engineering Research and Development, Volume 10, Issue 3 (March 2014), PP.73-78

[2] Gargi S. Rewatkar, "Implementation of Double Precision Floating Point Multiplier in VHDL", International Journal of Innovative Research in Electronics and Communications (IJIREC) Volume. 1, Issue 1, April 2014, pp.45-50

[3] Pallavi Ramteke, Dr. N. N. Mhala, Prof. P. R. Lakhe, " An Efficient Implementation of Double Precision Floating Point Multiplier Using Booth Algorithm" International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Issue 7, July 2014

[4] Sukhvir Kaur, Parminder Singh Jassal, "Synthesis of Double Precision Floating Point Multiplier Using vhdl" Journal of Research in Electrical and Electronics Engineering (ISTP-JREEE), Volume 3, Issue 2, March 2014.

[5] Addanki Puma Ramesh, A. V. N. Tilak, A.M.Prasad, "An FPGA Based High Speed IEEE-754 Double Precision Floating Point Multiplier Using Verilog" 2013 IEEE

[6] Sukhvir Kaur and Parminder Singh Jassal, "An Efficient Field Programmable Gate Array Implementation of Double Precision Floating Point Multiplier using VHDL" International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 7, July 2013

[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, November 2012

[8] Alok Baluni, Farhad Merchant, S.K. Nandy, "A Fully Pipelined Modular Multiple Precision Floating point Multiplier with Vector Support", IEEE, 2011.

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

[10] Vibha Mishra, Vinod Kapse "Hardware Implementation of 64 Bit Floating Point Arithmetic Using VHDL" International Journal of Advanced Engineering and Nano Technology (IJAENT) ISSN: 2347-6389, Volume-1, Issue-3, February 2014

[11] Chaitali V. Matey, Dr. S. D. Chede, Prof. S. M. Sakhare, "Design and Implementation of Floating Point Multiplier Using Wallace and Dadda Algorithm" International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 3, Issue 6, June 2014.

