VHDL Implementation of Addition and Subtraction unit for Floating Point Arithmetic Unit

Prerna Mandloi[1] Mr. Atush Jain[2]
[1]M.Tech [2]Assistant Professor
[1]Embedded Systems and VLSI Design [2]Electronics & Communication Engineering
[1,2]Acropolis Institute of Technology and Research, Indore

Abstract— *this paper describes implementation of Floating point adder and subs tractor unit using IEEE 754 format. A floating-point unit (FPU) is a part of a computer and it is specially designed to carry out operations on floating point numbers. Floating point representation can support a much wider range of values than fixed point representation. It describes a method of representing a rough calculation to real numbers in a way that it can support an ample range of values Floating Point arithmetic is by far the most used way of approximating real number arithmetic for performing numerical calculations on present computers. This paper presents implementation of a floating point, Addition and subtraction unit for double presicion floating point numbers. The implementation is done using conventional method. The whole design was implemented using VHDL and simulation was verified using Xilinx 13.1i simulation tool.*

Keywords— Floating point, Double precision, IEEE-754, VHDL

I. INTRODUCTION

The floating point operations have found intensive applications in the various fields for the requirements for high precious operation due to its great dynamic range, high precision and easy operation rules. High attention has been paid on the design and research of the floating point processing units. With the increasing requirements for the floating point operations for the high-speed data signal processing and the scientific operation, the requirements for the high-speed hardware floating point arithmetic units have become more and more exigent. The implementation of the floating point arithmetic has been very easy and convenient in the floating point high level languages, but the implementation of the arithmetic by hardware has been very difficult. With the development of the very large scale integration (VLSI) technology, a kind of devices like Field Programmable Gate Arrays (FPGAs) have become the best options for implementing floating hardware arithmetic units because of their high integration density, low price, high performance and flexible applications requirements for high precious operation. [1]

However, real numbers are not well-suited for general purpose computation, because their numeric representation as a string of digits expressed [12] in, say, base 10 can be very long or even infinitely long. Examples include π, e, and 1/3. In practice, computers store numbers with finite precision. Numbers and arithmetic used in scientific computation should meet a few general criteria

- Numbers should have minimum storage requirements
- Arithmetic operations should be efficient to carry out
- A level of standardization, or portability, is desirable–results obtained on one computer should closely match the results of the same computation on other computers. [13]

A. Floating point representation

Floating point describes a method of representing an approximation to real numbers in a way that it can support a wide range of values. Floating point numbers are one possible way of representing real numbers in binary format; the IEEE 754 [11] standard presents two different floating point formats, Binary interchange format and Decimal interchange format. Numbers are, in general, represented approximately to a fixed number of significant digits and scaled using an exponent. The base for the scaling is normally 2, 10 or 16 for binary, decimal and octal numbers respectively. Floating point numbers are represented by using equation 1 and figure 1.1.

$$\textbf{Significant digits} \times \textbf{base}^{\textbf{exponent}} \quad (1)$$

Sign	Exponent	Mantissa

Fig. 1: Floating point representation

B. IEEE-754 standards

The IEEE Standard for Floating-Point Arithmetic is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers. The arithmetic model specified as IEEE 754-1985 or "IEEE 754.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.

IEEE 754 specifies:

(1) Formats for binary and decimal floating point data for computation and data interchange.
(2) Different operations as addition, subtraction, multiplication and other operations.
(3) Two basic floating-point formats: single and double.
- The IEEE single format has a significant precision of 24 bits and occupies 32 bits overall.
- The IEEE double format has a significant precision of 53 bits and occupies 64 bits overall.

(4) Two classes of extended floating-point formats: single extended and double extended. The standard does not prescribe the exact precision and size of these formats, but it does specify the minimum precision and size. For example, an IEEE double extended format must have a significant precision of at least 64 bits and occupy at least 79 bits overall.
(5) Four rounding directions: toward the nearest represent able value, with "even" values preferred

whenever there are two nearest represent able values; toward negative infinity (down); toward positive infinity (up); and toward 0 (chop/truncate)

(6) Five types of IEEE floating-point exceptions, and the conditions for indicating to the user the occurrence of exceptions of these types. The five types of floating – point exceptions are invalid operation, division by zero, overflow, underflow, and inexact.

C. Double precision

This paper focuses on double precision normalized binary interchange format. Figure 1.2 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 significant and exponent is greater than zero and smaller than 1023. The real number is represented by equations (2).

r = (-1)s 2e-1023 1.f (2)

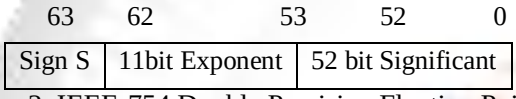

Fig. 2: IEEE-754 Double Precision Floating Point Formats

Double format bit pattern	***Value***
0<e<2047	r= $(-1)^s$ 2^{e-1023} 1.f (normal no.)
e= 0, f ≠ 0 (at least 1bit of f is not zero)	r= $(-1)^s$ 2^{e-1023} 0.f (subnormal no.)
e=0, f=0	r= $(-1)^s$ 0.0 (signed zero)
s=0, e=2047, f=0	+INF(positive infinity)
s=1, e=2047, f=0	-INF(negative infinity)
S= u, e=2047, f≠0	Nan

Table 1: Double Format Bit Pattern

II. Literature Survey

A few research works have been conducted to explain the concept of Floating Point Numbers. D. Goldberg [2] explained the concept of Floating Point Numbers used to describe very small to very large numbers with a varying level of precision. They are comprised of three fields, a sign, a fraction and an exponent field. B. Parhami [3] proposed IEEE-754 standard defining several floating point number formats and the size of the fields that comprise them. [4,5]The standard specifies two formats for floating-point numbers, *basic* (single precision) and *extended* (double precision), it also specifies the basic operations for both formats which are addition and subtraction of operations. Then different conversions are needed, as integer to floating-point, basic to extend and vice versa. Finally, it describes the different floating-point exceptions and their handling, including non-numbers (NaNs) This Standard defines several rounding schemes, which include round to zero, round to infinity, round to negative infinity, and round to nearest. Michael L. Overton [6] defines the subnormal numbers, normal numbers and special cases. D. Sangwan [7] gives the concept of addition subtraction and multiplication of floating point numbers all these operation requires either comparator or shifter or both of that. Shirazi [8] perform all the arithmetic operation on FPGA but this does not support rounding modes. Likewise L. Louca [9] designs an IEEE Single Precision Floating Point Addition and Multiplication unit, unit support 32 bit format given by IEEE std. 754-1985.

III. Floating Point Addition & Subtraction

A floating-point operation takes two inputs with p bits of precision and returns a p- bit result. The ideal algorithm would compute this by first performing the operation exactly, and then rounding the result to p bit.

Problems Associated With Floating Point Addition & Subtraction: For the input the exponent of the number may be dissimilar. And dissimilar exponent can't be added directly. So the first problem is equalizing the exponent. To equalize the exponent the smaller number must be increased until it equals to that of the larger number. Then significant are added. Because of fixed size of mantissa and exponent of the floating-point number cause many problems to arise during addition and subtraction. The second problem associated with overflow of mantissa. It can be solved by using the rounding of the result. The third problem is associated with overflow and underflow of the exponent. The former occurs when mantissa overflow and an adjustment in the exponent is attempted the underflow can occur while normalizing a small result. Unlike the case in the fixed-point addition, an overflow in the mantissa is not disabling; simply shifting the mantissa and increasing the exponent can compensate for such an overflow. Another problem is associated with normalization of addition and subtraction. The sum or difference of two significant may be a number, which is not in normalized form. So it should be normalized before returning results.

IV. Implementation Of Adder Subtractor Unit

The IEEE (Institute of Electrical and Electronics Engineers) has produced a Standard to define floating-point representation and arithmetic the standard brought out by the IEEE come to be known as IEEE 754. The IEEE 754 Standard for Floating-Point Arithmetic is the most widely-used standard for floating-point computation, and is followed by many hardware (CPU and FPU) and software implementations. In this paper, VHDL implementation of a adder and Subtractor unit is carried out using IEEE 754 formats and operations.

The figure shown above is the generalized block diagram of adder/Subtractor unit. This unit comprise of a comparator, shift register, normalizing, adder and Subtractor unit. Addition is a complex operation because depending on the signs of the operands, it may actually be a subtraction. If it is an addition, there can be carryout on the left. If it is subtraction there can be cancellation also.

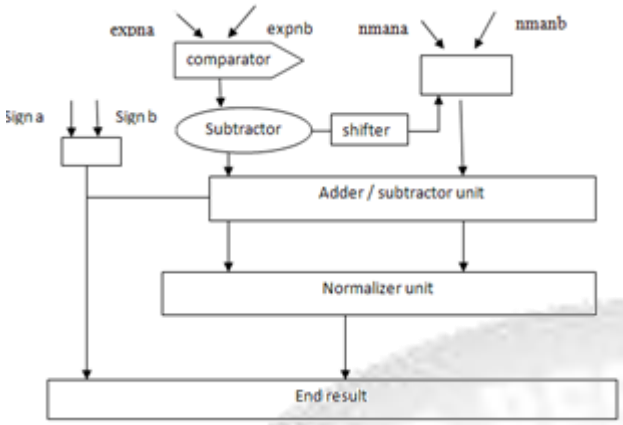

Fig. 3: Block Diagram of Adder/Subtractor Unit

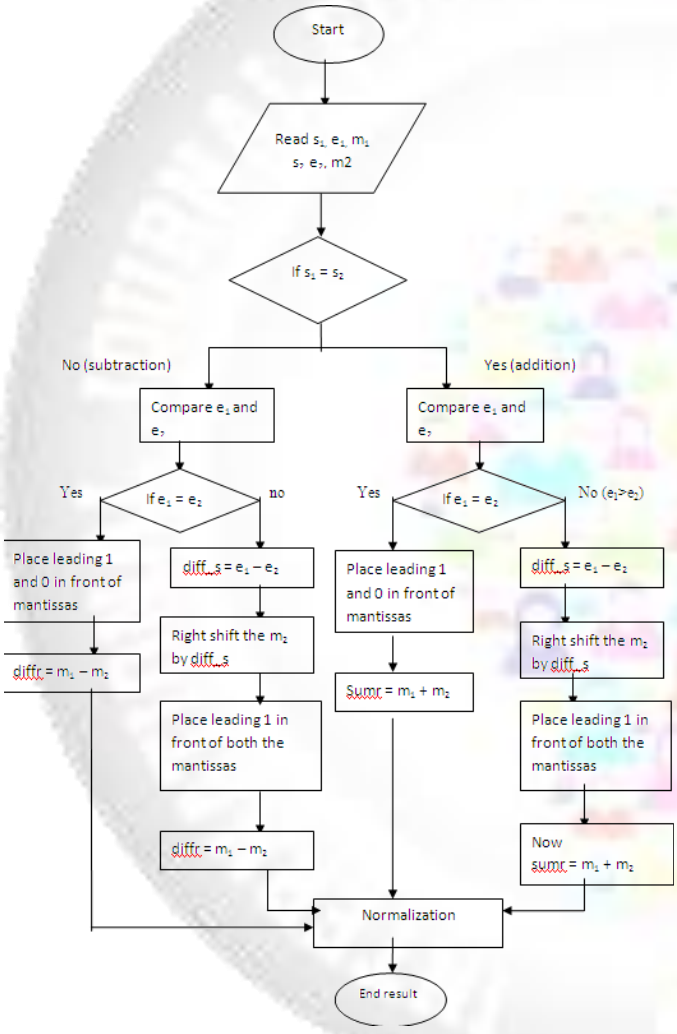

Fig. 4: Flow Diagram of Adder-Subtractor Unit

Addition and Subtraction block is implemented using the algorithm as shown below. Figure 1.4 shows the flow of design unit. The algorithm is as below:

(1) Compare exponent and mantissa of both numbers. Find the large exponent and mantissa and small exponent and mantissa.
(2) If both exponents are equal then put leading 1 and 0 in front of each mantissa for overflow, *i.e.*, 52+2 bits.
(3) If overflow occurs the exponent must be increased by one. 54th bit will be shifted out of mantissa and this will be saved for rounding purpose.
(4) The leftmost 1 in the result becomes leading 1 of mantissa and next 52 bits are actual mantissa.
(5) If there is overflow 1 in result, the exponent of larger operand is increased by one. Then add both mantissas.
(6) If one exponent is larger than other then subtract smaller exponent from larger exponent, and difference is saved.
(7) Now shift smaller mantissa by that subtracted exponent difference.
(8) If exponents are equal, *i.e.*, larger exponent is equal to smaller exponent after shifting then put leading 1 in both mantissas and perform addition for mantissas.
(9) For subtraction if both exponents are equal then put implied 1 in front of mantissa and perform subtraction.
(10) Store that result in "diff" register. Count the number of zeros in "diff" before the left most 1.
(11) Reduce the large operand exponent by number of zeros in front of least 1.
(12) The left most 1 will be leading 1 in front of mantissa.
(13) Finally normalization is done.

V. Simulation And Results

The double precision floating point adder and Subtractor unit is implemented in VHDL language and it is simulated using Xilinx ISE 13.2i on XC3s1600efg484-4.

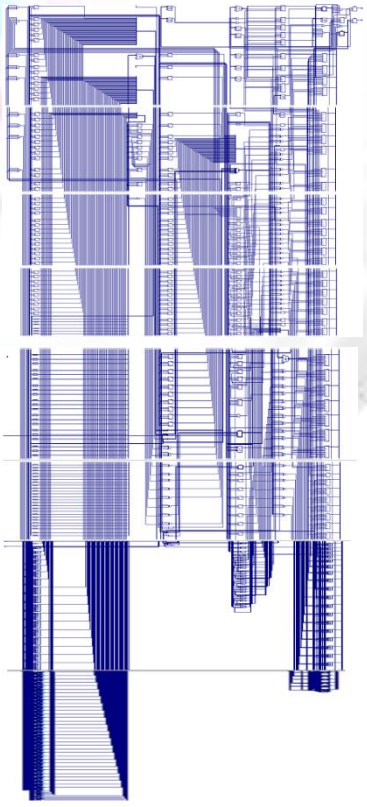

Fig. 5: RTL View of Adder/Subtractor Unit

The Add/Sub Unit is synthesized using XC3s Spartan IC. The synthesis report is shown in table below

	Utilized	**Available**	**Utilization %**
Number of Slices	808	14752	5%
Number of 4 input LUTs	1555	29504	5%
Number of bonded IOBs	5	376	1%

Table 2: Device Utilization Table

ROMs	1
Adder/Subtractor	3
Registers	202
Comparators	11
Multiplexers	1

Table 3: Advanced Synthesis Report

The simulation results are shown in the figure below. In the simulation results 'ieee754/a' and 'ieee754/b' are the inputs and 'ieee754/s and ieee754/d' are the outputs of adder and subtractors.

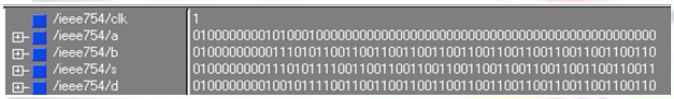

Fig. 6: Simulation result of Adder/Subtractor Unit

A	12.25	01000000001010001000000000000000 00000000000000000000000000000000
B	7.35	01000000000111010110011001100110 01100110011001100110011001100110
Sum	19.6	01000000001100111001100110011001 10011001100110011001100110011010
Diff	4.9	01000000000100111001100110011001 10011001100110011001100110010011

Table 4: Result of Adder-Subtractor unit

VI. Conclusion

The VHDL code is written for double precision floating point adder/Subtractor unit and it has been implemented and tested on Xilinx ISE 13.1. IEEE 754 standard based floating point representation has been used. The code has been synthesized and simulated successfully. The standard provides a method for computation with floating-point numbers that will give up the similar results when processing it either on hardware or software by providing the same input data. It can be extended to have more mathematical operations like multiplication, division, square root etc.

References

[1] Naresh Grover, M.K.Soni," *Design of FPGA based 32-bit Floating Point Arithmetic Unit and verification of its VHDL code using MATLAB*", I.J. Information Engineering and Electronic Business, 2014, 1, 1-14 Published Online February 2014 in MECS

[2] D. Goldberg, "*What every computer scientist should know about floating-point arithmetic*" ,ACM Computing Surveys vol. 23-1 , pp. 5-48 ,1991

[3] B. Parhami, "*Computer Arithmetic: Algorithms and Hardware Designs*", Oxford University Press, 2000.

[4] Institute of Electrical and Electronics. "*Standard for Binary floating point arithmetic*", IEEE Std 754-1985

[5] Institute of Electrical and Electronic Engineers, "*Draft Standard for Floating-Point Arithmetic*," October, 2006

[6] Michael L. Overton, "*Numerical Computing with IEEE Floating Point Arithmetic*," Published by Society for Industrial and Applied Mathematics, 2001.

[7] 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

[8] 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), (1995), pp. 155–162.

[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] Basit Riaz Sheikh and Rajit Manoharn " *An Operand-Optimized Asynchronous IEEE 754 Double-Precision Floating –Point Adder*" Computer Systm Laboratory

[11] IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.

[12] The Unicode Standard, Version 5.0, The Unicode Consortium, Addison-Wesley Professional, 27October 2006, ISBN 0-321-48091-0.54

[13] Ms. Anjana Sasidharan, Mr. M.K. Arun, "*VHDL Implementaion of IEEE 754 Floating Point Unit*", IJAICT Volume 1, Issue 2, June 2014