International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Review on Floating Point Adder and Converter Units Using VHDL

Abhishek Kumar[1], Mayur S. Dhait[2]

[1] Research Scholar, Agnihotri College of Engineering, Nagthana Road, Wardha (M.S), India

[2] Professor, Department of E&C, Agnihotri College of Engineering, Nagthana Road, Wardha (M.S), India

Abstract: ***Floating Point arithmetic is the most used way of approximating real number arithmetic for performing numerical calculations on modern computers. The advantage of floating-point representation is that it can support a much wider range of values rather than fixed point and integer representation. Addition/Subtraction, Multiplication and division are the common arithmetic operations in these computations. Among them floating point Addition is the most complex one. Adder is the core element of complex arithmetic circuits, in which input should be given in standard IEEE754 format. The main objective of the work is to design and implement a binary to IEEE 754 floating point converter for representing 32 bit single precision floating point values. Then the converter will be placed at the input side of the designed floating point adder module to improve the overall design. The modules are written using very high speed integrated circuit (VHSIC) Hardware Description Language (VHDL), and are then synthesized for Xilinx vertex E FPGA using Xilinx Integrated Software Environment(ISE) design suite 10.1.***

Keywords: Floating point arithmetic, single precision, IEEE 754 format, VHDL, Xilinx**.**

1. Introduction

The demand for floating point arithmetic operations in most of the commercial, financial and internet based applications is increasing day by day. Floating point operations are hard to implement on reconfigurable hardware i.e. on FPGAs because of their algorithm's complexity. While many scientific problems require floating point arithmetic with upper level of accuracy in their calculations. Therefore VHDL programming for IEEE single precision floating point adder have been explored. For implementation of floating point adder on FPGAs module various parameters like combinational delay, area, clock period, latency, total number of paths/destination ports, etc will be outline in the synthesis report. VHDL code for floating point adder is written in Xilinx 8.1i and the Design process of Xilinx will outline various parameters.

Since the demand for floating point arithmetic operations is increasing day by day. Hence it becomes essential to find out a technique to feed binary numbers directly as input for these applications. This helps in time saving and becomes much easier. In the current scenario, it is not possible, because, in the floating point adder, inputs should be given in IEEE 754 format i.e. the binary inputs cannot be given directly, because it needs to be converted to the sign, exponent and mantissa form. Hence in this project we have also designed a binary to floating point converter for single precision bits and will be directly given to the inputs of floating point adder which will solve this issue to an extent.

The converter is of is 32 bits wide and based on IEEE single precision format. The floating point format, real arithmetic can be coded directly into hardware operations. So, this project emphasizes on utilizing the capabilities of floating point format. The range of binary input given will be from 0-256 bits, which is the maximum input range that can be provided to satisfy the exponent range in the 32 bit IEEE 754 single precision format.

The modules are written using very high speed integrated circuit (VHSIC) Hardware Description Language (VHDL), and are then synthesized for Xilinx vertex E FPGA using Xilinx Integrated Software Environment(ISE) design suite 10.1.

2. IEEE Floating Point Representation

Floating point numbers are one possible way to represent real numbers in binary format. There are two basic formats described in IEEE 754 format, double-precision using 64-bits and single-precision using 32-bits.Table 1 shows the comparison between the important aspects of the two representations.

Table 1: Single and double precision format summary

Format	*Sign*	*Exponent*	*Mantissa*
Single Precision	1(31)	8 (23 TO 30)	23(0 TO 22)
Double Precision	1(64)	11(52 TO 63)	52(0 TO 51)

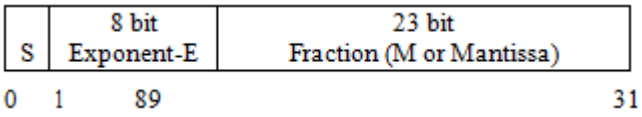

Figure 1: IEEE 754 single precision format

The IEEE 754 single precision binary format representation is shown in Fig. 1; [2] it consists of a one bit sign (S), an eight bit exponent (E), and a twenty three bit fraction (M or Mantissa). If the exponent is greater than 0 and smaller than 255, and there is 1 in the MSB of the significand then the number is said to be a normalized number; in this case the real number is represented by (1)

$$Z = (-1)S * 2(E - bias) * (1.M) \quad (1)$$

Where M = m22 2 -1 + m21 2 - 2 + m20 2 - 3 + ... + m1 2 - 22 + m0 2 - 23;

Bias = 127

Sign bit determines the sign of a number, which is either 0 for a non-negative number or 1 for a negative number. A bias of 127 is added to the actual exponent for IEEE single precision format. The mantissa or significand is composed of an implicit leading bit (to the left of the binary point)with value 1,unless the exponent and 23 fraction bits to the right of the binary point is all filled with zeros. The numbers are always normalized and thus there is no need to explicitly show the implicit '1' bit, thereby precision is increased. The IEEE 754 standard specifies some special values. The standard also specifies some rounding modes like: Round toward positive infinity; round toward negative infinity and Round toward zero; Round to nearest, ties to even; Round to nearest, ties away from zero. These special cases will be considered in this project.

3. Binary to Floating Point Conversion

Decimal number (base 10 real number) can be converted into an IEEE 754 binary32 format by using the following steps: [1]

- Consider a real number with an integer and a fraction part such as 13.375.
- Convert and normalize the integer part into binary.
- Convert the fraction part using the following method shown below.
- Add two results and adjust them to get a final conversion

3.1 Conversion of the Fractional Part

Consider 0.375, the fractional part of 13.375. To convert it into a binary fraction, multiply the fraction part by 2, take the integer part and re-multiply new fraction part by 2 until a fraction of zero is found or until the precision limit is reached which is 23 fraction digits for IEEE 754 binary32 format.

0.375 x 2 = 0.750 = 0 + 0.750 => b_{-1} = 0, the integer part represents the binary fraction digit. Next step is to re-multiply 0.750 by 2 to proceed.
0.750 x 2 = 1.500 = 1 + 0.500 => b_{-2} = 1
0.500 x 2 = 1.000 = 1 + 0.000 => b_{-3} = 1,
fraction = 0.000, terminated.

We found that $(0.375)_{10}$ can be represented in binary as $(0.011)_2$. Not all decimal fractions can be represented in a finite digit binary fraction. For example decimal 0.1 cannot be represented in binary exactly. So it can be only approximated.

Therefore $(13.375)_{10} = (13)_{10} + (0.375)_{10} = (1101)_2 + (0.011)_2 = (1101.011)_2$

Also in IEEE 754 binary32 format real values need to be represented in normalized form. Hence it becomes 1.101011 x 2^3 i.e. the exponent is 3 (and in the biased form it is therefore 127+3=130 = $(1000\ 0010)_2$). Now the fraction is 101011 (right of the binary point). The 32 bit IEEE 754 binary32 format representation of 13.375 as:
0-10000010-10101100000000000000000 = 41460000_H.

We have done the binary to floating point conversion in IEEE 754 format in this project. This conversion had been done by using VHDL and later implemented in Xilinx FPGA.

3.2 Block diagram of converter

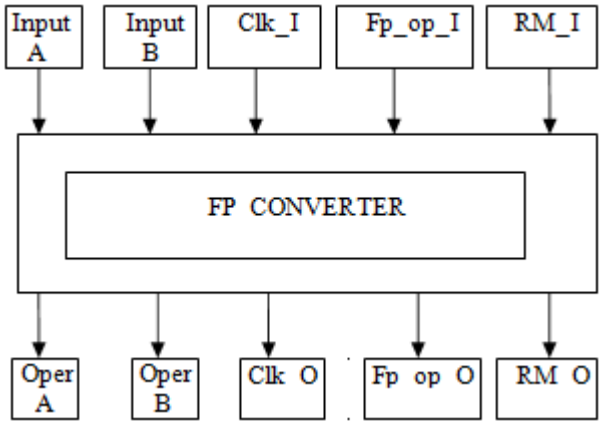

Figure 2: Block Diagram of Binary to Floating Point Converter

This is the block diagram of the binary to floating point converter which will be implemented. The Input A and Input B are the 255 bits wide inputs to the floating point converter. And Clk_I is a clock bit. Other inputs are Floating point operation Input (Fp_op_I), which can take either value-0 or 1 which specifies addition or subtraction operation, And RM_I is the input, which constitutes the rounding mode bits. Mostly four rounding modes are considered in the adder/subtractor block and they are [00,01,10,11].

Similarly the output ports of the converter consists of Operand A, Oper A and Operand B, Oper B, both of them should be std_logic_vectors of size 32. It consists of sign bit, exponent bits and mantissa bits in the output corresponding to the inputs. Other output ports are the clock output (Clk_O), floating point operation output (Fp_op_O), it may be 0 or 1, and the Rounding Mode Output (RM_O), which can be any of these values [00,01,10,11]. To obtain the full functionality of the converter unit, we have integrated it into an adder module.

3.3 RTL view and output simulation of converter

Figure 3: RTL schematic for Binary to Floating point Converter.

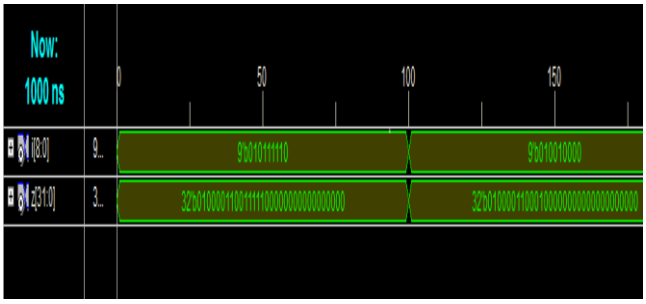

Figure 4: Output simulation for Binary to Floating point Converter.

4. Algorithm For Single Precision Floating Point Adder

1) Separating signs, exponents and mantissas of both A and B numbers.
2) Considering the special cases:
   Operation with A or B equal to zero
   Operation with ∞
   Operation with NaN
3) Specifying which type of numbers are given:
   Normal
   Subnormal
   Mixed
4) Shifting the mantissa of lower exponent number to the right [*Exp*1- *Exp*2] bits. Considering the output exponent as the highest exponent.
5) Working with the operation symbol and both signs to calculate the output sign and determine the operation to do.
6) Addition/Subtraction of the numbers and detection of mantissa overflow (carry bit).
7) Standardizing mantissa by shifting it to the left up, the first one will be at the first position and according to the carry bit updating the value of the exponent and shifting over the mantissa.
8) Detecting overflow or underflow of the exponent (result NaN or ∞)

5. Floating Point Adder Design

5.1 Existing Method

There are two existing cases for the floating point addition algorithm. [7]
Case I: When both the numbers are of same sign i.e. when both the numbers are either +ve or –ve. It means that the MSB of both the numbers are either 1 or 0.
Case II: when both the numbers are of different sign i.e. when one number is +ve and other one is –ve, it means that the MSB of one number is 1 and other is 0.

5.2 Proposed Method

The black box view and block diagram of the single precision floating point Adder is shown in figures 5 and 6 respectively. The input has been separated into their sign, mantissa and exponent components.

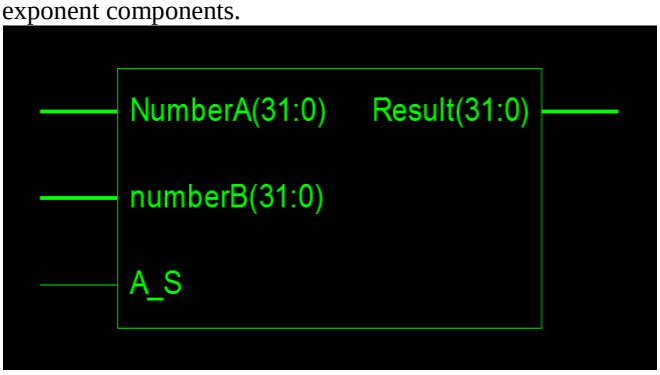

Figure 5:.Black box view of single precision Adder

The main hardware modules for a single-precision floating-point adder are the exponent difference module, right shift shifter, 2's complement adder, leading one detector, left shift shifter, and the rounding module.

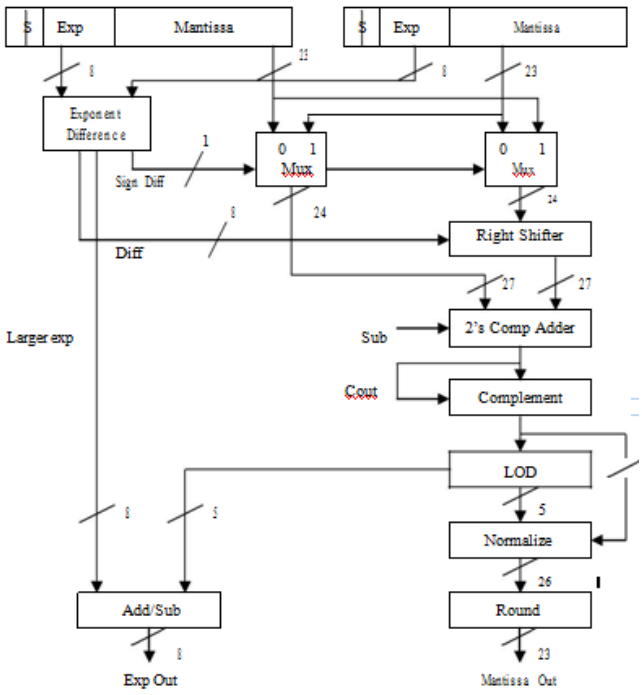

Figure 6:.Micro-architecture of standard floating-point Adder

5.3 Exponent Difference Module

The exponent difference module has the following two functions:

- To compute absolute difference of two 8-bit numbers.
- To identify if e1 is smaller than e2.

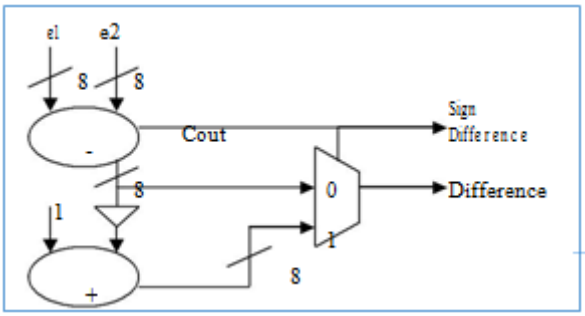

Figure 7:.Block diagram of exponent difference module.

The block diagram of the exponent difference module, RTL schematic and Output simulation are shown in Fig.7, 8 and 9 respectively.

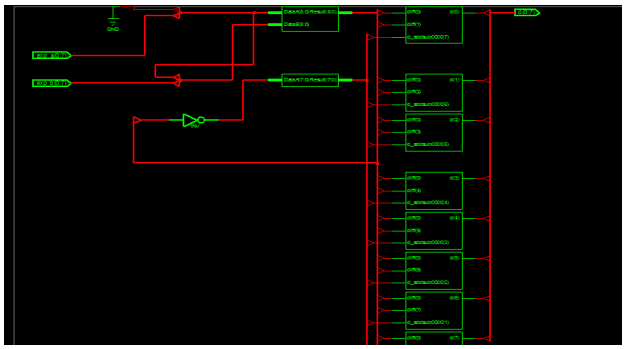

Figure 8: RTL schematic of exponent difference module

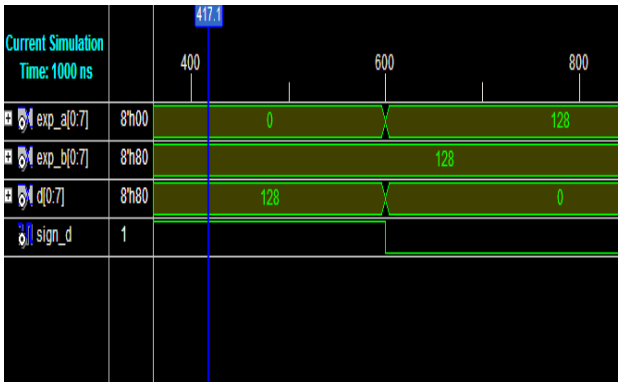

Figure 9: Output simulation for exponent difference module

This exponent difference module has been designed by using Hardware Description Language (VHDL), and are then synthesized for Xilinx vertex E FPGA using Xilinx Integrated Software Environment(ISE) design suite 10.1. In the similar manner we are also designing the other modules like right shift shifter, 2's complement adder, leading one detector, left shift shifter, and the rounding module.

6. Literature Review

Field Programmable Gate Arrays (FPGA) are increasingly being used to design high end computationally intense microprocessors capable of handling both fixed and floating point mathematical operations. Addition is the most complex operation in a floating-point unit and offers major delay while taking significant area.

According to Reshma Cherian And Nisha Thomas they have implemented binary to floating point converter, which was based on IEEE 754 single precision format, and has delay of 17.381 n sec and power utilization was 0.295 W In " Implementation of Binary to Floating Point Converter using HDL".

According to Sunita S. malaj, S.B. Patil, Bhagappa R. Umarane, In "VHDL Implementation of Interval Arithmetic Algorithms for Single Precision Floating Point Numbers", The author proposes a new approach where the design and implementation of single precision (32bit) Interval Arithmetic Adder/subtractor unit is carried using VHDL for computing interval arithmetic operations & functions suited for hardware implementation.

According to Jairaj Bhattacharya, Aman Gupta, and Anshul Singh "A High Performance Binary TO BCD Converter for Decimal Multiplication ", this paper presented a novel architecture for Binary to BCD conversion used in decimal multiplication. The proposed converters flexible and can be plugged into any homogeneous multiplication architectures to achieve better performance irrespective of the method used to generate binary partial products. The proposed architecture shows, on an average, an improvement of 28% in terms of power-delay product.

7. Conclusion

This paper shows the efficient use of floating point Converter and floating point Adder module. This paper presents an Implementation of an efficient 32 bit floating point Adder with floating point Converter module at its input port to support IEEE 754 standard with optimal chip area and high performance using VHDL. Based on the above discussion, it is clear that a floating point adder element in any processor design and a processor spends considerable amount of time in performing floating point conversion and addition. Hence optimizing the speed and area of the module is a major design issue. An improvement in conversion and addition speed by using new techniques can highly improve system performance. So the aim of our project is to analyze the problem and study the different ways to overcome the problems in a order to enhance the system performance.

References

[1] Reshma Cherian#, Nisha Thomas*, Y.Shyju# "Implementation of Binary to Floating Point Converter using HDL"pp. 461-64,©2013 IEEE

[2] Sunita.S.Malaj, S.B.Patil, Bhagappa.R.Umarane, "VHDL Implementation of Interval Arithmetic Algorithms for Single Precision Floating Point Numbers" International Journal of Scientific & Engineering Research Volume 4, Issue3, March-2013.

[3] Jairaj Bhattacharya, Aman Gupta, Anshul Singh.,"A High Performance Binary TO BCD Converter for Decimal Multiplication" International Symposium on VLSI Design, Automation and Test, 2010.
[4] Guillermo Marcus, Patricia Hinojosa, Alfonso Avila and Juan Nolazco-Flores " A Fully Synthesizable Single-Precision,Floating Point Adder/Substractor and Multiplier in VHDL for General and Educational Use," Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, Dominican Republic, Nov.3-5, 2004.
[5] "Design and Implementation of IEEE-754 Addition and Subtraction for Floating Point Arithmetic Logic Unit",V.vinay chamkur,
[6] W. Kahan "IEEE Standard 754 for Binary Floating-Point Arithmetic,"1996
[7] Preeti Sudha Gollamudi, M. Kamaraju, " Design of High performance IEEE-754 single precision (32 bit) floating point adder using VHDL. IJERT, Vol.2 Issue 7, pp. 2264-75, July-2013.
[8] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754, 1985.
[9] Metin Mete, Mustafa Gok, "A multiprecision floating point adder" 2011 IEEE.
[10] Ali malik, Soek bum ko , "Effective implementation of floating point adder using pipelined LOP in FPGAss," ©2010 IEEE.
[11] Karan Gumber,Sharmelee Thangjam "Performance Analysis of Floating Point Adder using VHDL on Reconfigurable Hardware" in International Journal of Computer Applications (0975 – 8887) Volume 46– No.9, May 2012

Author Profile

Abhishek Kumar received the B.E. in Electronics & Telecommunication Engineering from Shri Shankarprasad Agnihotri College of Engineering, Wardha, India in 2012.Currently he is research scholar and pursuing M.Tech in Electronics from Agnihotri College of Engineering, Nagthana, Wardha India.

Prof. Mayur S. Dhait received his B.E in Electronics & Communication from KITs Ramtek, India and M.Tech in Embedded system design from IIIT, Pune, India. He is working as Prof. in Agnihotri College of Engineering, Nagthana, Wardha, India.