BCD To Floating Point Converter With Floating Point Adder Unit Using VHDL

[1]Abhishek Kumar
Department of Electronics Engineering
Agnihotri College of Engineering,
Nagthana road, Wardha (MH) INDIA
E-mail: abhi111333@gmail.com

[2] Prof. Mayur S. Dhait
Department of Electronics Engineering
Agnihotri College of Engineering,
Nagthana road, Wardha(MH) INDIA

[3] Prof. Vijay R. Wadhankar
Department of Electronics Engineering
Nagthana road, Wardha(MH) INDIA
Agnihotri College of Engineering

Abstract— To perform numerical calculations on modern computers floating point arithmetic is a better way of approximating real number arithmetic. Its advantage 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. Among them the most complex one is the floating point addition. Adder is the most important 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 to represent 32 bit single precision floating point values. Then the converter will be placed at the input 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 13.1.

Keywords- *floating point arithmetic, IEEE 754 format, VHDL, Xilinx*

*****

I. INTRODUCTION

Nowadays the floating point arithmetic operations based applications has become the most demanding applications. It is hard to implement these operations on reconfigurable hardware i.e. on FPGAs because of the complexity of their algorithm. 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. VHDL code for floating point adder is written in Xilinx 13.1 and the Design process of Xilinx will outline various parameters.

Because of the increasing demand of floating point arithmetic operations, 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 given should be 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 based on IEEE single precision format and is of 32 bits wide. The floating point format, real arithmetic can be coded directly into hardware operations. So, this project emphasizes on utilization of 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 13.1.

II. IEEE FLOATING POINT REPRESENTATION

To represent real numbers in binary format floating point numbers are the best possible way. 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)

S	8 bit Exponent-E	23 bit Fraction (M or Mantissa)
0	1 8	31

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)

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

Where $M = m22\, 2^{-1} + m21\, 2^{-2} + m20\, 2^{-3} + \ldots\ldots.. + m1\, 2^{-22} + m0\, 2^{-23}$;

Bias = 127

Sign bit is used to determine the Sign of a number, which will be either 0 for a non-negative number or 1 for a negative number. For IEEE single precision format a bias of 127 is added to the actual exponent. 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.

III. BCD TO BINARY FLOATING POINT CONVERTER

A. Methodology

Methodological analysis used to convert the BCD number into Binary Floating point number is described below. To convert the BCD number into Binary floating point number IEEE-754 single precision format is used. Conversion of a BCD real number into an IEEE 754 binary 32 format uses the following outline:

1. Consider a BCD number.
2. To convert the number into floating point use the shift and subtract-3 algorithm.
3. Represent the output of shift and subtract-3 algorithm into normalized form.
4. To produce a proper final conversion adjust the result.

The example of conversion process is given below:

- Example- 255=$(0010\ 0101\ 0101)_{BCD}$
- To convert the BCD number into floating point use shift and subtract-3 Algorithm.
- The output of above number i.e $(0010\ 0101\ 0101)_{BCD}$ is given as $(11111111)_2$ by using shift and subtract-3 algorithm. Also in IEEE 754 binary32 format values need to be represented in normalized form given as $1.1111111x2^7$.
- From above, the exponent is 7, and in the biased form, it is add to the power with 127 to give the exponent as 127+7=134 represented as 10000110 in binary form.
- Therefore 127+7=134 = $(1000\ 0110)_2$.
- The fraction is 1111111 (looking to the right of the binary point)
- To produce a proper final conversion adjust the result.
- The resulting IEEE 754 (single precision) 32 bit format representation of 255 i.e. $(0010\ 0101\ 0101)_{BCD}$ is: 0-10000110-11111110000000000000000

We have done with the binary to floating point conversion in IEEE 754 format. This conversion has been done by using VHDL and later will be implemented in Xilinx FPGA.

B. Block diagram of converter

Block Diagram of BCD to floating point converter is shown below. In IEEE 754 BCD to Floating point converter the main modules are Shift and Subtract-3 module, and check and Forward block.
As an input we give BCD number, the internal blocks of the converter converts BCD number into floating point number.

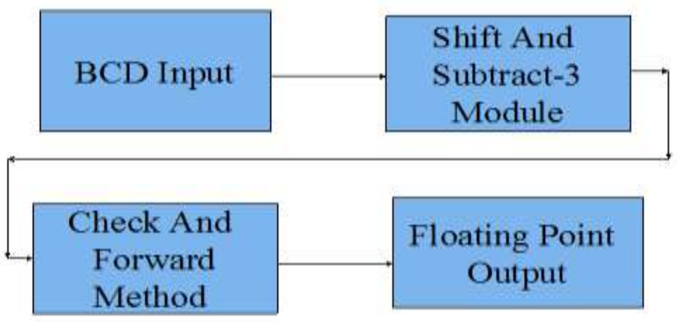

Figure 2: Block Diagram of BCD to Floating Point Converter

The principal purpose of this project is to provide a program that converts a BCD number with a decimal point and/or an exponent to a floating-point binary number. The floating-point binary number has a mantissa of 23 bits, an exponent byte consisting of a sign bit and seven magnitude bits, and a sign flag (one byte) for the mantissa.

C. Subtract-3 Module

Subtract-3 Module states that if binary value in any of BCD column is greater than 7 then subtract - 3 from that number.
Truth Table For Subtract -3 module is shown below.

Table 2: Truth table for subtract-3 module

A3	A2	A1	A0	S3	S2	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

Figure below shows subtract-3 block. Each subtract-3 block contain 4 bit of input and 4 bit output.

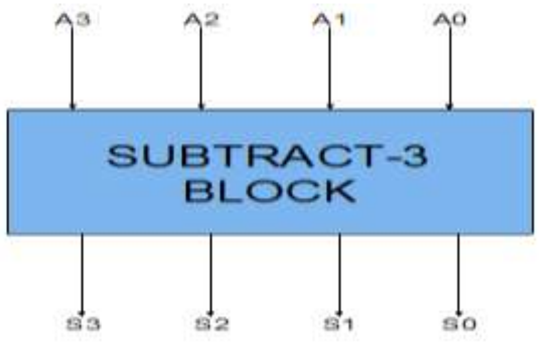

Figure 3: Block Diagram of subtract-3 module

D. Mathematical analysis of shift and subtract-3 algorithm

The working of Shift & Subtract-3 algorithm is described below
1. Shift BCD number right one bit and examine each decade. Subtract 3 from each 4-bit decade containing a binary value greater than 7.
2. Shift right, examine, and correct after each shift until the least significant decade contains a number smaller than eight and all other converted decades contain zeros.
Mathematical Analysis of Shift & Subtract-3 algorithm is shown below.

Table 3: Mathematical analysis of shift and subtract-3 algorithm

0010	**0101**	**0101**			**Start**
001	0010	1010	1		Shift 1
001	0010	0111	1		Sub-3
00	1001	0011	11		Shift 2
00	0110	0011	11		Sub-3
0	0011	0001	111		Shift 3
	0001	1000	1111		Shift 4
	0001	0101	1111		Sub-3
	000	1010	1111	1	Shift 5
	000	0111	1111	1	Sub-3
		0011	1111	11	Shift 6
		001	1111	111	Shift 7
		00	1111	1111	Shift 8

E. Output simulation and RTL view of BCD to Floating point converter

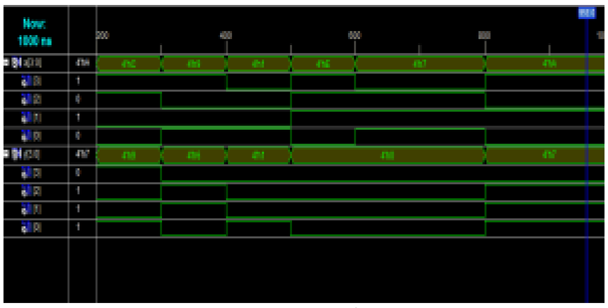

Figure 4: output simulation of Subtract-3 module.

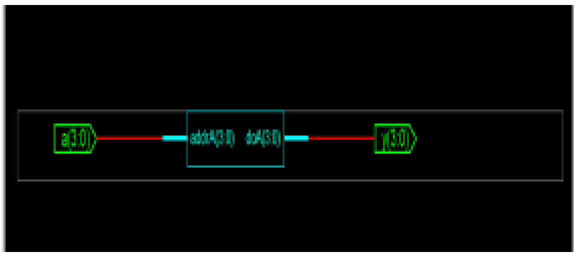

Figure 5: RTL schematic for subtract-3 module.

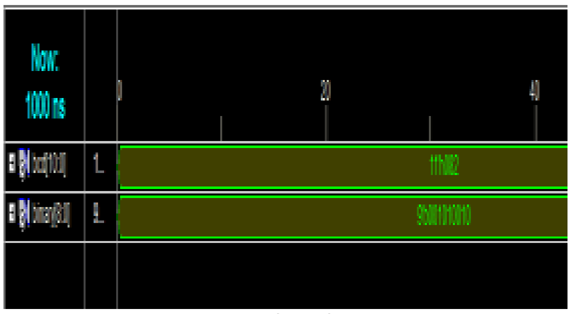

Figure 6: output simulation of Shift and Subtract-3algorithm.

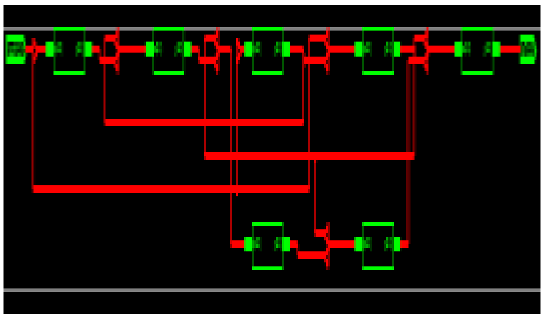
Figure 7: RTL schematic for Shift and Subtract-3algorithm.

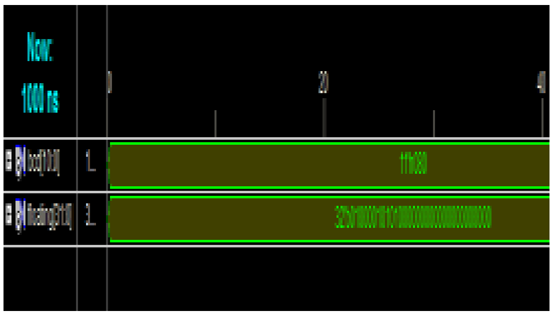

Figure 8: output simulation of BCD to floating point converter.

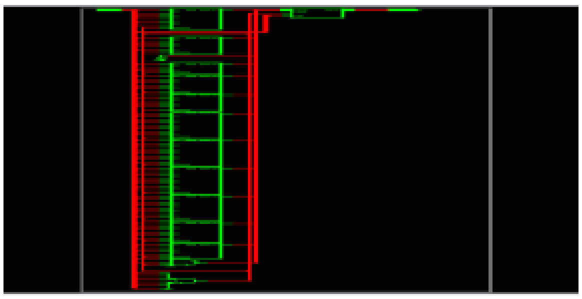
Figure 8: RTL schematic for BCD to floating point converter.

IV. FLOATING POINT ADDER USING BCD TO FLOATING POINT CONVERTER AS INPUT

Design method for adder using BCD to floating point converter as input is described below. To improve the overall performance of the design we use the output of the BCD to floating point converter in IEEE-754 single precision floating point format because of its greater accuracy and precision. Adder is used to add two numbers, here we give two BCD numbers as the input, and the converter we designed will convert that BCD number into binary floating point numbers.

Then by using floating point adder these floating point numbers will be added.

Standard algorithm for floating point adder is given below, at its input it takes two floating point numbers, and then add them by following the steps given in the algorithm and give the result in floating point format.

A. Floating Point Addition Algorithm

1. Read and compare the two operands, N1 and N2 for denormalization and infinity. Set the implicit bit to 0 if numbers are denormalized, otherwise it is set to 1. At this point, the fraction part is extended to 24 bits.
2. Using 8-bit subtraction the two exponents, e1 and e2 are compared. Swap N1 and N2 if e1 is less than e2 i.e. previous f2 will now be referred to as f1 and vice versa.
3. Shift the smaller fraction, f2 to right by the absolute difference result of the two exponents' subtraction. Now both the numbers have the same exponent.
4. To check whether the operation is a subtraction or an addition the two signs are used.
5. The bits of the f2 are inverted for the operation of Subtraction.
6. Now using a 2's complement adder the two fractions are added.
7. If the result of adder is a negative number, then it has to be inverted and a 1 has to be added to the result.
8. Now in the first step of the normalization the result is passed through a leading one detector or leading zero counter.
9. Using the results from the leading one detector, the result is then shifted left to be normalized. In some cases, 1-bit right shift is needed.
10. The result is then rounded towards nearest even, the default rounding mode.
11. The result is left shifted by one if the carry out from the rounding adder is 1.
12. The exponent is adjusted by using the results from the leading one detector. After overflow and underflow check the sign is computed, and then the result is registered.

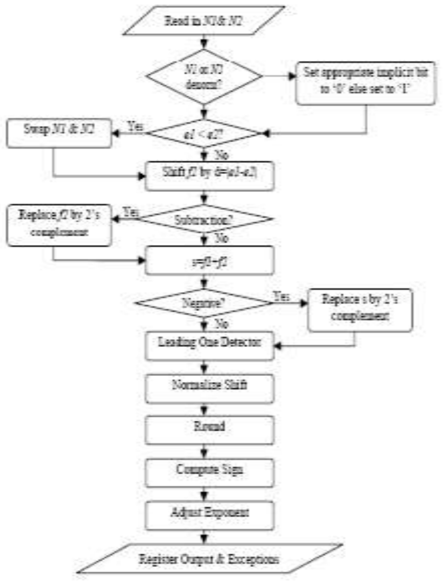

Figure 9: Flow chart for standard floating point adder

B. Pipeline standard floating point adder

Pipelining is used to decrease clock period, to run the operations at a higher clock rate, and to increase speedup by increasing the throughput. By distributing the main module into smaller operations pipelining can be achieved. In pipelining the whole operation may take more clock cycles to complete but new inputs can be added with every clock cycle increasing the throughput. Pipeline architecture for floating point adder is given below. In the first stage the two operands are compared to identify denormalization and infinity. To obtain the exponent difference the two exponents are subtracted. To pre normalize the smaller mantissa the right shifter is used. In the third stage along with the leading one detection the addition is done. In the fourth stage to post normalize the result left shifter is used. In the last stage the exponent out is calculated and rounding is done. To set overflow or underflow flags results are compared.

The main 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. This adder is specially designed for single-precision addition.

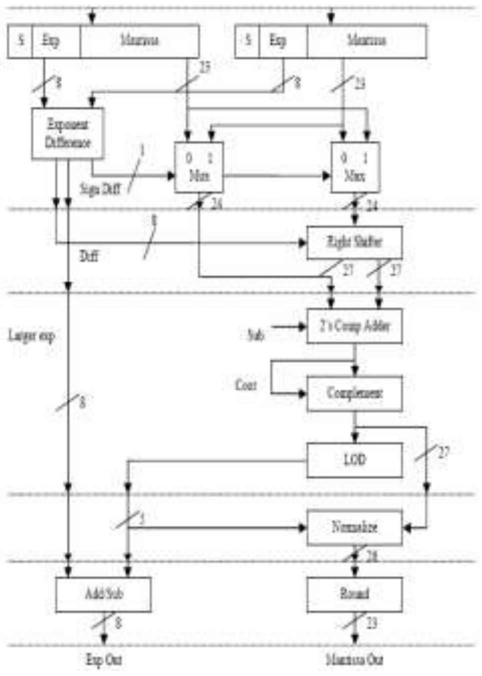

Figure 10: Pipeline standard floating point adder

C. Output simulation And RTL view of Floating Point Adder

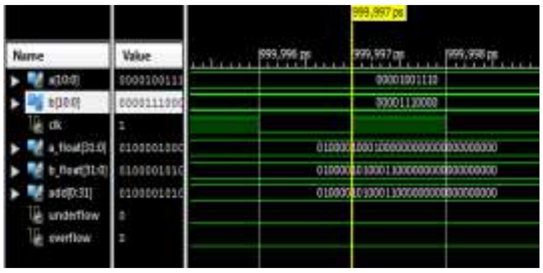

Figure 11: Simulation Result for floating point adder

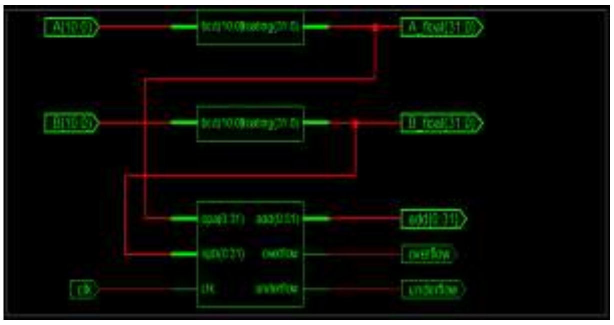

Figure 12: RTL Schematic for floating point adder

V. 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.

VI. CONCLUSION

This paper shows the efficient use of floating point Converter and floating point Adder module together. 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 at the output of converter we have of 32 bit binary floating point number with high accuracy and precision with sign, exponent and mantissa form. And then by using it as input to a floating point adder we get an improved result with greater accuracy and precision. An improvement in conversion and addition speed can highly improve system performance by using new techniques. So the aim of our project is to analyze the problem and study the different ways to overcome the problems in an order to enhance the system performance.

REFERENCES

[1] Abhishek Kumar, Mayur S. Dhait" Review on Floating point adder and converter unit using VHDL" in International Journal of Science & Research, Volume 4 Issue-3, March2015.

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

[3] 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.

[4] 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.

[5] 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.

[6] "Design and Implementation of IEEE-754 Addition and Subtraction for Floating Point Arithmetic Logic Unit",V.vinay chamkur,

[7] W. Kahan "IEEE Standard 754 for Binary Floating-Point Arithmetic,"1996

[8] 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.

[9] Shubhangi Bende, Prof. Sanjay Tembhurne, "Design of BCD to floating point Converter based on single precision format", International journal os Scientific Research.

[10] Metin Mete, Mustafa Gok, "A multiprecision
floating point adder" 2011 IEEE.

[11] Ali malik, Soek bum ko , "Effective implementation of floating point adder using
pipelined LOP in FPGAss," ©2010 IEEE.

[12] 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.

[13] Rupali Dhobale, Soni Chaturvedi, "Implementation of 32 Bit Binary Floating Point Adder Using IEEE 754 Single Precision Format" IOSR Journel of VLSI and Signal Processing (IOSR_JVSP) Volume 5, Issue 1, Ver. I (Jan_Feb.2015).