ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

FPGA Implementation of Single Precision Floating Point Adder

Rupali Dhobale, Soni Chaturvedi

M. Tech. Electronics (Communication), Dept. of EC, PIET, R.T.M. Nagpur University, Nagpur, Maharashtra, India

Assistant Professor, Dept. of EC, PIET, R.T.M. Nagpur University, Nagpur, Maharashtra, India

ABSTRACT: 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. Over the years, the VLSI community has developed many floating-point adder algorithms mainly aimed to reduce the overall latency. An efficient design of floating-point adder onto an FPGA offers major area and performance overheads. Our research was oriented towards studying and implementing standard, Leading One Predictor (LOP) and Leading One Detector (LOD) floating-point addition algorithms. Each of the sub-operation is researched for different implementations. The Objective of this paper to implement the 32 bit binary floating point adder with minimum time. Floating point numbers are used in various applications such as medical imaging, radar, telecommunications Etc. The language used for programming is VHDL, and is Synthesized using Xilinx ISE14.2 Suite. The target device selected is Spartan6 family, XC6SLX45T device. The results are obtained using ISim (VHDL/Verilog) Simulator.

KEYWORDS: *ASIC, FPGA, IEEE754, LOP*.

I. INTRODUCTION

Floating-point addition is the most frequent floating-point operation and accounts for almost half of the scientific operation. Therefore, it is a fundamental component of math coprocessor, DSP processors, embedded arithmetic processors, and data processing units. These components demand high numerical stability and accuracy and hence are floating- point based. Floating-point addition is a costly operation in terms of hardware and timing as it needs different types of building blocks with variable latency. In floating-point addition implementations, latency is the overall performance bottleneck. A lot of work has been done to improve the overall latency of floating-point adders. Various algorithms and design approaches have been developed by the Very Large Scale Integrated (VLSI) circuit community.

Field Programmable Gate Array (FPGA) is a silicon chip with unconnected logic blocks, these logic blocks can be defined and redefined by user at anytime. FPGAs are increasingly being used for applications which require high numerical stability and accuracy. With less time to market and low cost, FPGAs are becoming a more attractive solution compared to Application Specific Integrated Circuits (ASIC). FPGAs are mostly used in low volume applications that cannot afford silicon fabrication or designs which require frequent changes or upgrades. Devices with millions of gates and frequencies reaching up to 300 MHz are becoming more suitable for floating-point arithmetic reliant applications.

II. RELATED WORK

One of the first competitive floating-point addition implementation is done by L. Louca, T. Cook, and W. Johnson [8] in 1996. Single precision floating-point adder was implemented for Altera FPGA device. The primary challenge was to fit the design in the chip area while having reasonable timing convergence. The main objective of their implementation was to achieve IEEE standard accuracy with reasonable performance parameters. This is claimed to be the first IEEE single precision floating-point adder implementation on a FPGA, before this, implementation with only 18-bit word length was present [8].

Most of the algorithms implemented in FPGAs used to be fixed-point. Floating-point operations are useful for computations involving large dynamic range, but they require significantly more resources than integer operations.

DOI: 10.15680/ijircce.2015.0306081

With the current trends in system requirements and available FPGAs, floating-point implementations are becoming more common and designers are increasingly taking advantage of FPGAs as a platform for floating-point implementations. The rapid advance in Field-Programmable Gate Array (FPGA) technology makes such devices increasingly attractive for implementing floating-point arithmetic. Compared to Application Specific Integrated Circuits, FPGAs offer reduced development time and costs. Moreover, their flexibility enables field upgrade and adaptation of hardware to run-time conditions. A 32 bit floating point arithmetic unit with IEEE 754 Standard has been designed using VHDL code by Naresh Grover, M.K. Soni and all operations of addition, subtraction, multiplication and division are tested on Xilinx[9].

The IEEE 754 single precision format[4] is as shown below. It divides in three parts are as follows:-

S	8 bit Exponent-E	23bit fraction –F
0	1…………………………………8	9……………………………………………..31

IEEE 754 Single Precision Format

- Sign: 1 bit wide and used to denote the sign of the number i.e. 0 indicate positive number and 1 represent negative number.
- Exponent: 8 bit wide and signed exponent in excess-127 representation. The exponent field represents both positive and negative exponents.
- Mantissa: 23 bit wide and fractional component.

The single- precision floating-point number is calculated as $(-1)^{S} \times 1.F \times 2^{(E-127}$

III. PROPOSED ALGORITHM

Description of the Proposed Algorithm:

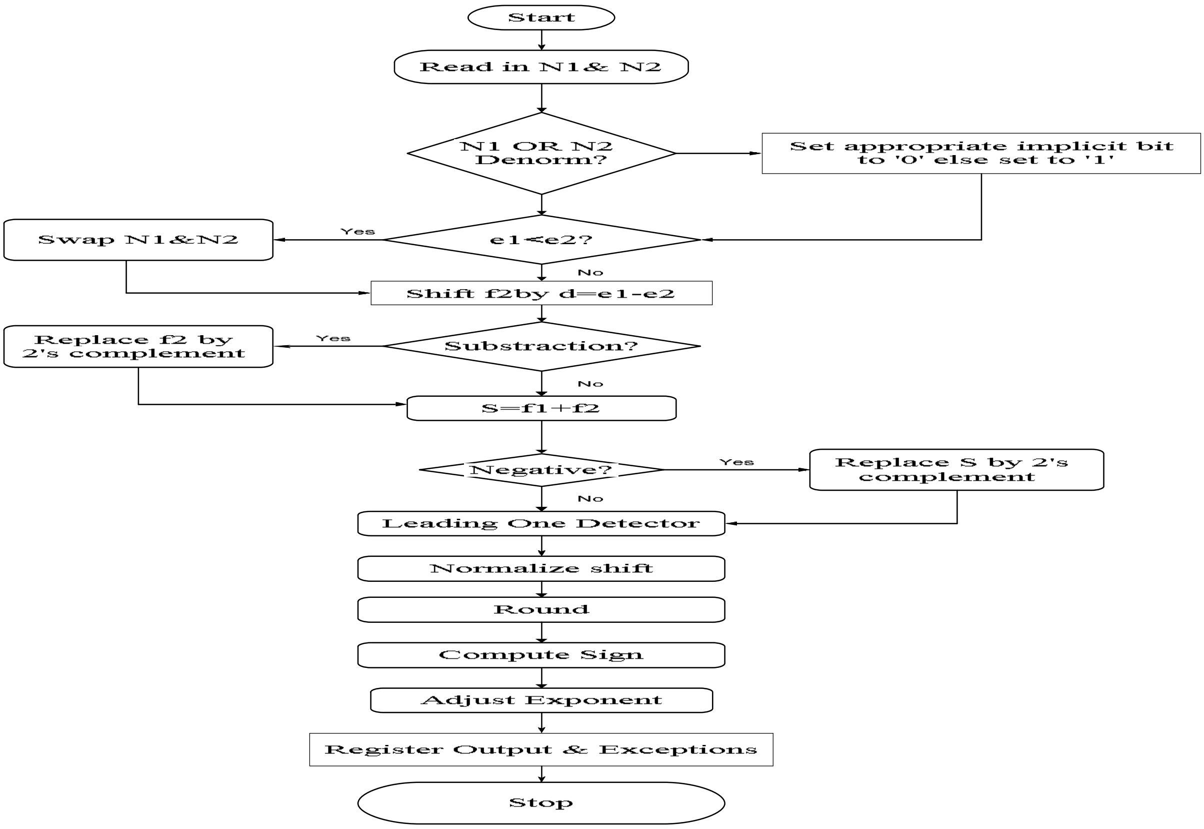


Fig. 1 : Flowchart of standard floating point adder Algorithm

DOI: 10.15680/ijircce.2015.0306081

Fig 1 shows the flowchart of standard floating point adder algorithm. Let s1; e1; f1 and s2; e2; f2 be the signs, exponents, and significands of two input floating –point operands, N1 and N2, respectively. A description of the standard floating point adder algorithm is as follows.

1. The two operands, N1 and N2 are read in and compared for demoralization and infinity. If numbers are demoralized, set the implicit bit to 0 otherwise it is set to 1. At this point, the fraction part is extended to 24 bits.
2. The two exponents, e1 and e2 are compared using 8-bit subtraction. If e1 is less than e2, N1 and N2 are swapped i.e. previous f2 will now be referred to as f1 and vice versa.
3. The smaller fraction, f2 is shifted right by the absolute difference result of the two exponents' subtraction. Now both the numbers have the same exponent.
4. The two signs are used to see whether the operation is a subtraction or an addition.
5. If the operation is a subtraction, the bits of the f2 are inverted.
6. Now the two fractions are added using a 2's complement adder.
7. If the result sum is a negative number, it has to be inverted and a 1 has to be added to the result.
8. The result is then passed through a leading one detector or leading zero counter. This is the first step in the normalization step.
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. If the carry out from the rounding adder is 1, the result is left shifted by one.
12. Using the results from the leading one detector, the exponent is adjusted. The sign is computed and after overflow and underflow check, the result is registered.

IV. BLOCK DIAGRAM OF SINGLE PRECISION FLOATING POINT ADDER

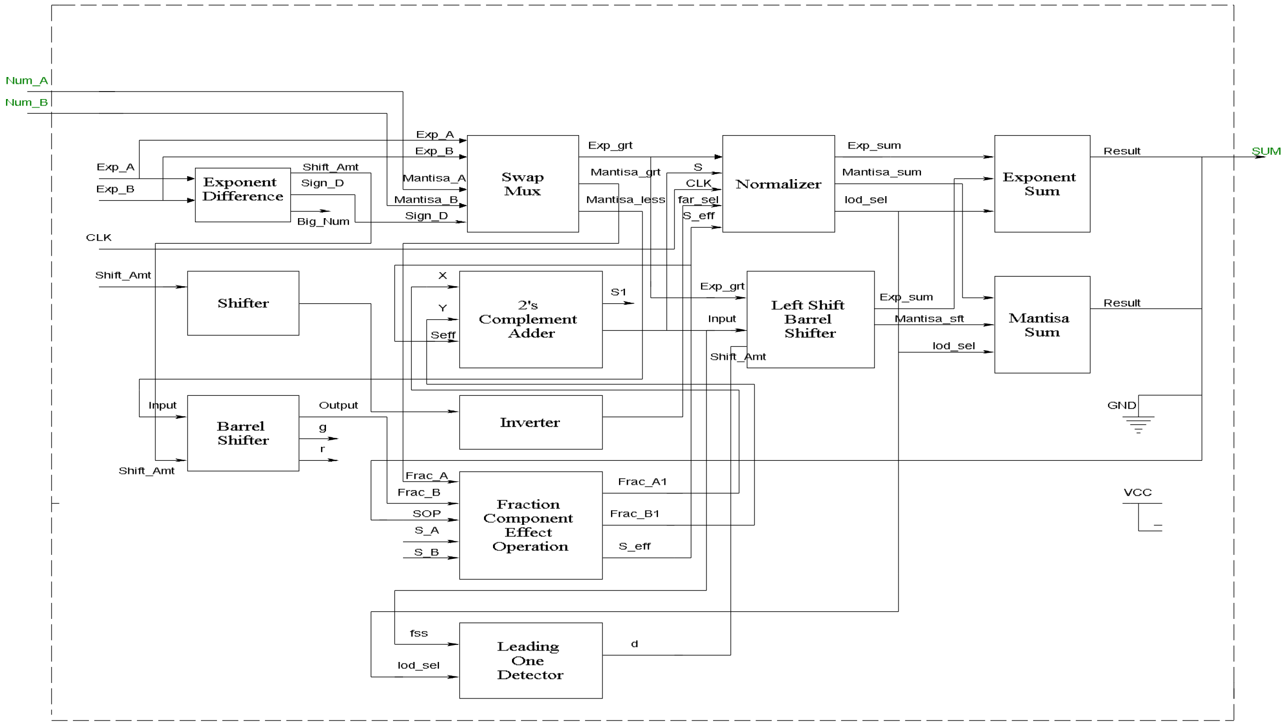


Fig 2: Block Diagram of Single Precision floating-point Adder

Fig 2 shows the block diagram of single precision floating point adder. Num_A , Num_B ,CLK ,GND and VCC are inputs of the block diagram of single precision floating point adder. Output SUM is calculated by performing addition of Num_A and Num_B using floating point addition algorithm shown in Fig 1. It shows the main hardware modules necessary for floating point addition. The different modules are Exponent Difference Module, Swap Multiplexer,Shifter, Barrel Shifter , Fraction Component Effect Operation , 2's Complement Adder , Inverter , Normalizer , Leading One Detector , Left Shift Barrel Shifter, Exponent Sum and Mantissa Sum.

Exp_a and Exp_b are the exponents of inputs Num_A and Num_B resp. Exp_a and Exp_b can be positive or negative numbers. The output shift_amt is given to the shifter and barrel shifter block for further calculation. Sign_d is given two the Swap Mux. Swap_Mux assigns greater Mantisa to Mantisa_grt and Lesser value of Mantisa to Mantisa_less and greater exponent is calculated and assigned to Exp_grt. Exp_grt output is given to the normalizer block, Mantisa_grt is given to the Fraction Component Effect Operation Block and Mantisa_less is given to the Barrel shifter Block respectively.The shifter is used to shift the significant of the smaller operand by the absolute exponent difference. Shifter is used to shift the data bits. The Output Shift_amt of Exponent Difference Block is given as input to the Shifter block which gives output shifted_amt is further given to the inverter block. The inverter is used to invert the data bits of shifted amount. The normalizer block gives us normalized result. After the addition, the next step is to normalize the result. The first step is to identify the leading or first one in the result. This result is used to shift left the adder result by the number of zeros in front of the leading one. In order to perform this operation, special hardware, called Leading One Detector (LOD) or Leading Zero Counter (LZC), has to be implemented. Exponent sum and mantissa sum blocks are used to calculate the exponent and mantissa of output SUM and sign bit SOP is initially considered as 1.

V. SIMULATION RESULTS

The single precision floating point adder is designed through VHDL Coding. The program uses two 32 bits numbers i.e. Num_A& Num_B for addition gives output SUM which is again of 32 bit.

For example, Num_A & Num_B , two numbers used for addition are as follows-

0	1000 0010	11111011000000000000000

Format for Num_A

0	1000 0010	11111000000000000000000

Format for Num_B

The result after addition of above two numbers Num_A and Num_B by considering value of SOP=1 is as follows:-

0	01111011	01111111111111110000000

Format for Num_B

Simulation result for this example is as shown below in below Fig 5. Fig 3 shows RTL schematic of single precision floating point adder and Fig 4. shows internal structure of RTL Schematic of single precision floating point adder which are obtained on the synthesis / Implementation using Xilinx 14.2 ISE Software. The RTL Schematic shows inputs clk , rst , A & B two single precision floating point numbers which gives the output sum i.e. floating point addition of A & B. Fig 6 shows the device utilization summary of single precision floating point adder.

DOI: 10.15680/ijircce.2015.0306081

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

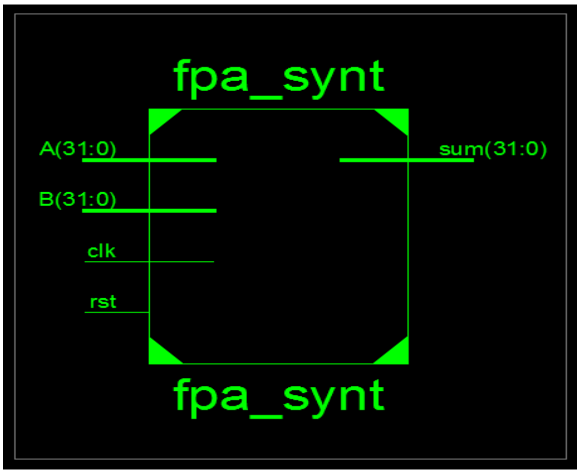


Fig. 3 RTL schematic of single precision FPA

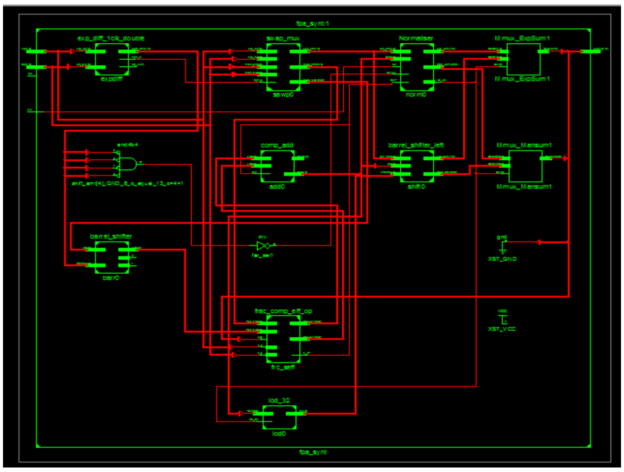


Fig 4. Internal Structure of RTL Schematic single precision FPA

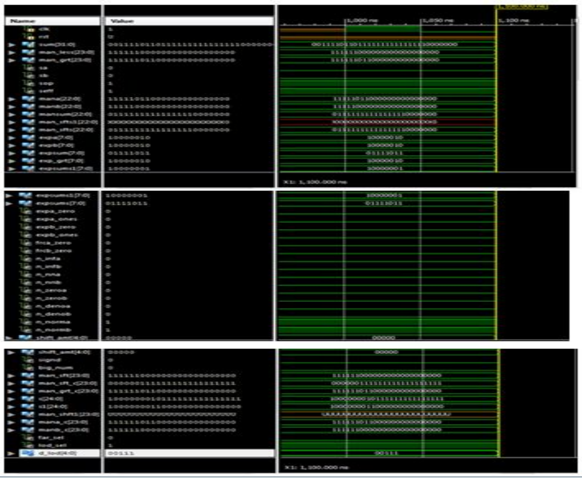
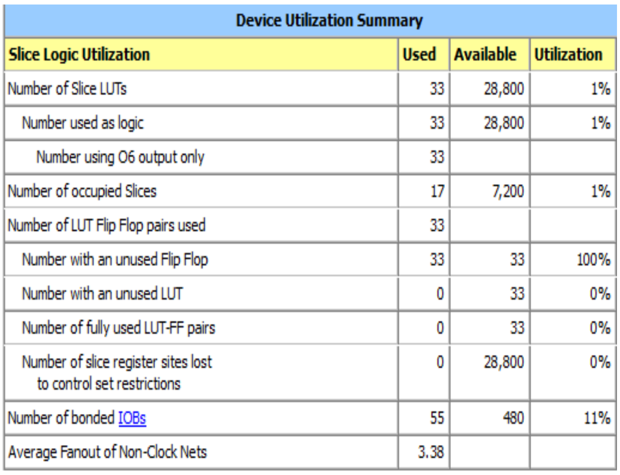


Fig. 5. Simulation result of FPA



Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	33	28,800	1%
Number used as logic	33	28,800	1%
Number using O6 output only	33		
Number of occupied Slices	17	7,200	1%
Number of LUT Flip Flop pairs used	33		
Number with an unused Flip Flop	33	33	100%
Number with an unused LUT	0	33	0%
Number of fully used LUT-FF pairs	0	33	0%
Number of slice register sites lost to control set restrictions	0	28,800	0%
Number of bonded IOBs	55	480	11%
Average Fanout of Non-Clock Nets	3.38		

Fig 6. Device Utilization summary

VI. Conclusion and Future Work

A single precision floating-point adder is implemented in this paper. The main contribution of our work is to implement and analyze floating-point addition algorithms and hardware modules were implemented using VHDL and and is Synthesized using Xilinx ISE14.2 Suite. The results are obtained using ISim (VHDL/Verilog) Simulator. In order to expand our paper further some of the works can be proposed in order to accommodate any exponent and mantissa length. This will gives more versatility while choosing the design criteria. The design can also be pipelined further for different number of pipeline stages to give even more adaptability and flexibility.

References

1. Ronald Vincent , Ms.Anju.S.L "Decimal Floating Point Format Based on Commonly Used Precision For Embedded System Applications." International Conference on Microelectronics, Communication and Renewable Energy (ICMiCR-2013).
2. Somsubhra Ghosh, Prarthana Bhattacharyya and Arka Dutta "FPGA Based Implementation of a Double Precision IEEE Floating-Point Adder", Proceedings of7'h International Conference on Intelligent Systems and Control (ISCO 2013).
3. Maarten Boersma, Michael Kr¨oner, Christophe Layer, Petra Leber, Silvia M. M¨uller, Kerstin Schelm "The POWER7 Binary Floating-Point Unit", 2011 20th IEEE Symposium on Computer Arithmetic.
4. Reshma Cherian, Nisha Thomas, Y. Shyju "Implementation of Binary to Floating Point Converter using HDL"P-461-P464.
5. Anand Mehta, C. B. Bidhul, Sajeevan Joseph, Jayakrishnan. P " Implementation of Single Precision Floating Point Multiplier using Karatsuba Algorithm", 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE).

6. Libo Huang, Li Shen, Kui Dai, Zhiying Wang "A New Architecture For Multiple-Precision Floating-Point Multiply-Add Fused Unit Ge Zhang,'Low Power Techniques on a High Speed Floating-point Adder Design' .

7. Ge Zhang,'Low Power Techniques on a High Speed Floating-point Adder Design' Proceedings of the 2007 IEEE International Conference on Integration Technology.

8. Loucas Louca, Todd A. Cook,William H. Johnson "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs"1996IEEE.

9. 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.

10. Douglas L. Perry, 'Programming By Example', Tata McGrew-Hill, Fourth edition.

BIOGRAPHY

Rupali Dhobale is a student in the Electronics & Communication Department, Priyadarshini Institute of Engineering & Technology, R.T.M.Nagpur University. She received B.E (Electronics) degree in 2013 from K.D.K.C.E, Nagpur, MS, India.

Mrs. Soni Chaturvedi is Head of Department, Priyadarshini Institute of Engineering & Technology, R.T.M.Nagpur University, MS, India