IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 9, Issue 4 Ver. III (Jul – Aug. 2014), PP 01-05
www.iosrjournals.org

An Implementation of Double precision Floating point Adder & Subtractor Using Verilog

[1]Sonali M. Rane, Prof. [2]Mrs. Trupti Wagh, [3]Dr. Mrs. P. Malathi
[1,2,3] *Department of E&TC Engineering DYPCOE, Akurdi Pune, India*

Abstract: *The floating point operations are critical to implement on FPGAs due to their complexity of their algorithms. Thus, many scientific problems require floating point arithmetic with high levels of accuracy in their calculations. Therefore, in this paper the proposed work is explored FPGA implementation of Addition/Subtraction for IEEE double precision floating point numbers. This kind of unit can be tremendously useful in the FPGA implementation of complex systems that benefits from the parallelism of the FPGA device. The implementation of 64 bit double precision floating point Adder/Subtractor results are compared with the previous results. The design is in Verilog Hardware description language (HDL) and implemented on FPGA. The verilog code first simulated with isim and synthesized using Xilinx ISE14.1i. The proposed double precision adder/Subtractor Modules are compliant with IEEE754 format and handle the various rounding conditions. Floating point adder/Subtractor is the most frequent floating point operation. Floating point adders are critically important components in modern microprocessors and digital signal processors.*

Keywords**:** *FPGA, Floating Point, Verilog, Double precision, Adder/Subtractor, IEEE754.*

I. Introduction

Floating point numbers are widely adopted in many applications due their dynamic representation capability. Numbers representated in floating point format can represent a much larger range than possible using the same number of bits in fixed point format. This makes a floating point a natural choice for scientific and signal processing computations. The advantage of floating-point representation over integer and fixed-point representation is that, it can shore up a much wider series of values.

Now a days, new recent advances in FPGA technology made it good choice for giving high performance in many computationally scientific and Engineering applications. These recent FPGAs have large amounts of programmable logics, memory and often come with a large set of high speed IP cores to implement scientific functions. The FPGAs are becoming aggressive to general purpose processors in wide series of applications, like communications, scientific computing, cryptography and Image processing

FPUs Floating-point arithmetic is widely used in many areas, particularly in scientific computation, signal processing (like digital filters, image processing FFT, etc.) and; numerical processing .The IEEE defines the standard for single-precision and double-precision formats. Hardware completion of arithmetic operations for IEEE floating-point standard becomes a important part of almost all processors.

The application area is always looking for high-performance and area efficient implementation of floating-point arithmetic operation and, thus, an efficient implementation of floating point adder/subtraction module is of a major concern. As a result, this work is primarily aimed for improved implementation of double-precision floating-point addition/subtraction, on FPGA platform. The mantissa of double-precision floating-point numbers is 53 bits in length. In this work, an approach for the addition/ subtraction of double-precision floating-point numbers has been proposed which allows to using less amount of addition/ subtraction, achieving excellent performance at a relatively low cost of hardware resources. Comparison of results has been done with optimized implementation of Sandia) floating-point library. The real numbers represented in binary format are known as floating point numbers. Based on IEEE-754 standard, floating point formats are classified into binary and decimal interchange formats. Floating point operations are very important in DSP applications. This paper focus on double precision normalized binary transaction format. Figure 1 shows the IEEE- 754 double precision binary format representation. Sign (S) is represented with one bit, exponent (E) with 11 bit and Mantissa with fifty two bits. 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.

Double precision floating point number is calculated as

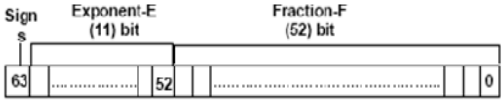

Fig. 1: Double precision floating point format.

Double precision format uses 1 bit for sign bit, 11 bit for bias exponent and 52 bits for representing the fraction as shown in figure 1.
Double precision floating point number is calculated as

Value= (-1ss)* (1FF)*2(EE-1023) (1)

The next segment provides a high level overview of the related work, the way our system is implemented followed by the results obtained and conclusion.

II. Related Works

A lot of work has been carried in the field Floating point implementation on FPGA. Since past two decades a lot of work has been done, related high performance implementation of floating point adder Subtractor unit.

Vogin G. Oklobdzija (1994) implemented 32 bit and 64 bit leading zero detector (LZD) circuit using CMOS and ECL Technology. In this the ECL implementation of the 64 bit LZD circuit simulated to perform in under 200ps for nominal speed. This circuit has been implemented in 0.6µ CMOS technology.

The algorithmic approach outperformed LS consistently with improvement in speed ranging from 12%-56% and improvements in layout area ranging from 14.5%-35%.The resulting LZD circuit has noteworthy presentation, which is important since it is often a part of the critical path in the floating point unit.[1].Nabeel Shirazi, Walters, and Peter Athanas (1995) implemented custom 16/18 bit three stage pipelined floating point multiplier that doesn't support rounding modes. By pipelining the each unit the sufficient number of times in order to maximize speed and to minimize area. They investigate the ways of implementing the floating point operators in the best combination of speed and area in order to minimize the area requirements small floating point representation formats are used. Pipeline techniques are applied to produce a result every clock cycle.[2].Loucas Louca, Cook, and Johnson (1996)explored FPGA implementation of addition and multiplication for single precision Floating point numbers that tradeoff the area and speed for the accuracy.The design achieved 2.3 MFlops and doesn't supports rounding modes. And7 MFlops for addition. [3].Barry Lee, and Neil Burgess (2002) implemented the floating point unit using primitives of Xillinx vertex-2 FPGA. The design achieved the operating frequency of 100MHz. with a latency of 4 clock cycles. They focused on using primitives of a specific FPGA to produce components with low delay and low area matrices.[4].Pavle Belanovic, and Miriam Leeser (2002) implemented the reconfigurable floating point unit using VHDL, which is mapped on to Xillinx XCV1000FPGA.They developed a library of fully parameterized hardware modules for format control, arithmetic operation and conversion to and from any fixed point format.[5].Ali Malik, and Seok-Bum Ko (2006) implemented the floating point adder using leading one predictor (LOP) algorithm instead of Leading one Detector (LOD) algorithm. The focal meaning of the LOP is to predict the leading number of zeros in the addition result, functioning in parallel with the 2's complement adder. The design implemented in Vertex2p FPGA. The improvement seen in LOP design is the level of logic reduced by 23%with an added expense of increasing the area by 38%.[6]. Dhiraj Sangwan, and Mahesh K. Yadav investigated adder/subtractor and multiplication unit for floating point arithmetic using VHDL. The floating point multiplication operation implemented using sequential architecture based on booths Radix-4 recording algorithm. For floating point addition, the sequential addition could have been complex so the combinational architecture has been implemented.[7]. K. Scott Hemmert, and Keith Underwood (2006), implemented open sourse library of highly optimized floating point units for Xilinx FPGAs. The units are fully IEEE complaint .The double precision add and multiply achieved the operating frequency of 230 MHz using a 10 stage adder pipeline and a 12 stage multiplier pipeline. The area requirement is 571 slices for adder. The floating point modules are hand-mapped and placed using JHDL as a design entry language. This presentation details the size and the performance of floating point operators in a library developed at Sandia National Labs.[8].

III. Design Description

In this section we will discuss about the hardware that we have used, block diagram of the proposed system.

The floating point module performs addition, subtraction. Also it implements all four rounding modes-round to nearest even, round to zero, round to positive infinity, round to negative infinity.

A. Block Diagram

The block diagram and the architectural schematic view of double precision floating point adder/Subtractor is shown in figures 2 and 3 respectively is shown in Fig. 2.
Each of the blocks along with its brief description is explained below:

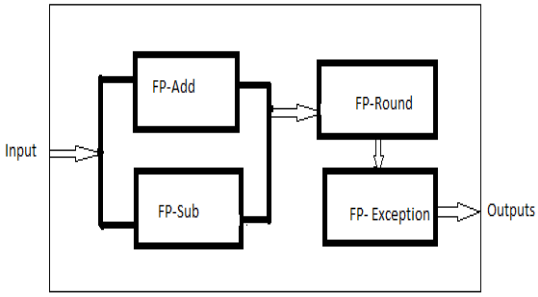

Fig.2: Block Diagram of Double Precision Floating point Adder/Subtractor.

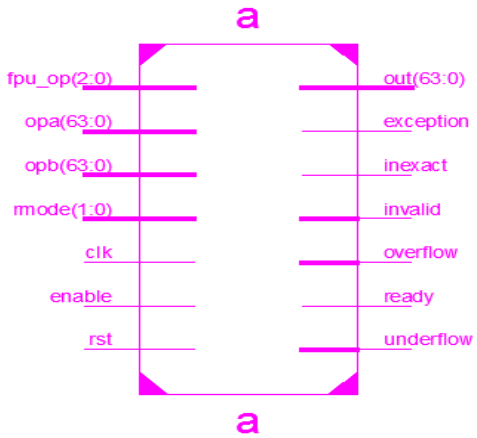

Fig.3: Black box view of Double Precision Floating point Adder/Subtractor

A. Addition

This module fp-add, will responsible to perform operation of addition. In the module fp_add., the input operands are separated into their mantissa and exponent components. Then the exponents are compared to check which operand is larger. Then the larger operand goes into "mantissa_large" and "exponent_large" with the smaller operand populating "mantissa_small" and "exponent_small". The sign and exponent of the output is the sign and exponent of the larger operand. To find the mantissa of the output, the difference between the exponents is found and the mantissa_small is right shifted equal to the "difference" amount. Finally, the addition is performed.

B. Subtraction

Subtraction is performed in the module fp_sub.v. Subtraction is similar to addition where the operands are separated, then compared and right shifted. Then subtraction is performed instead of addition.

C. Rounding

The rounding operation is performed in the module fpu-round. There are 4 possible rounding modes. Round to nearest, round to zero, round to positive infinity and round to negative infinity. . If after addition extra carry is generated, there is a need of increase in exponent by one, along with right shifting of the addition result by one bit. And after subtraction of mantissa, the MSB bit lost by the difference it required to check and the mantissa shifted by that value.

D. Exception

The exception performs in the FP-exception module. In this module, the special cases are checked for, and if they are found, the proper output is created and the individual output signals of overflow, underflow, exception, inexact, and invalid are asserted if the conditions are each case exist.

IV. Algorithm

Floating point addition is one of the difficult units in the floating point arithmetic operations. Addition and the related operation of subtraction is the most basic arithmetic operation. The hardware implementation of this arithmetic for floating point is a complicated operation due to the normalization requirements.

Implementation of double precision floating point adder/Subtractor has been presented here. The flowchart is used to both for addition and subtraction.

Steps for addition of two floating point numbers as below.

1. Comparing the exponents and mantissas of both input operands, Decide large exponent and mantissa and small exponent and mantissa.
2. Shift the mantissa to right allied with the smaller exponent, with the difference of exponent.
3. If signs are same, add both mantissas else subtract smaller mantissa from large mantissa.
4. After mantissa addition, do the rounding of the result.
5. If the subtraction results in loss of most significant bit (MSB), then the result must be normalized. To do this the most significant nonzero entry in the result mantissa must be shifted until it reaches the frontage. This is able by a "Leading one detector " followed by shift.
6. After that do the normalization and adjust large exponent accordingly.
7. Final output includes sign of larger number, normalized exponent and significant.

For subtraction, first reverse the sign of second operand and do the addition operation. The design is mainly uses the basic algorithm.

V. simulation results

The simulation results of double precision floating point adder, Subtractor is shown in fig. 5,fig 6, and fig.7 . Snapshot is nothing but every moment of the application while running. It will be useful for the new-fangled to understand for the future steps.

And also fig .9 shows the device utilization summary for rounding unit.

The double precision floating point adder/Subtractor, supports the IEEE-754 binary interchange format. The proposed design is done in Verilog.

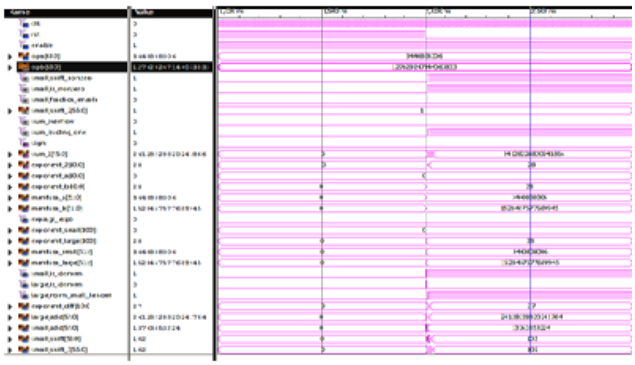

Fig.5: Simulation results of Addition operation

Fig.6: Simulation results of Subtraction operation

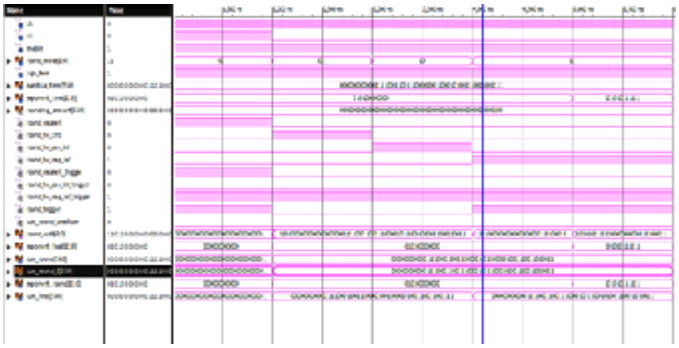

Fig.7 : Simulation results of Rounding operation

VI. Conclusion

We have presented a Double Precision Floating Point adder/Subtractor unit Using Verilog. The actual logic design of a floating point adder is not too difficult, and most of the design time was spent trying to minimize the logic resources used. The

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	176	768	22%
Number of Slice Flip Flops	248	1536	16%
Number of 4 input LUTs	195	1536	12%
Number of bonded IOBs	149	124	120%
Number of GCLKs	1	8	12%

Fig.8: Device utilization summary.

Results presented above show that these requirements have been satisfied to the great extent, this mean that further improvements are not possible. As a part of our plans, proper rounding will be added in the implementations. Our implementations give respectable performance, though not at the level of custom implementations. It can be designed for 64-quad double precision. It can be extended to have more mathematical operations like trigonometric, exponential and logarithmic functions.

References

[1] V. Oklobdzija, "An arithmetic and Novel Design of a leading zero detector circuit: comparison with logic synthesis", IEEE Transactions on very large scale Integration (VLSI) systems, Vol. 2, No. 1,(1994) March, pp.124-128.

[2] N. Shirazi, A. Walters and P. Athanas,'Quantitative Analysis of floating Point Arithmetic on FPGA Based Custom Computing Machines", Proceedings of IEEE symposium on FPGAs for custom Computing Machines (FCCM'95),(1995),pp.155-162.

[3] 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.

[4] B. Lee and N. Burgess, "Parameterisable Floating-point Operations on FPGA", Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems, and Computers, **(2002)**.

[5] P. Belanovic and M. Leeser, "A Library of Parameterized Floating-Point Modules and Their Use", in 12th International Conference on Field-Programmable Logic and Applications (FPL-02). London, UK: Springer-Verlag, **(2002)** September, pp. 657–666.

[6] Malik and S. -B. Ko, "A Study on the Floating-Point Adder in FPGAs", in Canadian Conference on Electrical and Computer Engineering (CCECE-06), **(2006)** May, pp. 86–89.

[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] K. Hemmert and K. Underwood, "Open Source High Performance Floating-Point Modules", in 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM-06), **(2006)** April, pp. 349–350.