International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 06 | June-2017 www.irjet.net p-ISSN: 2395-0072

Design and Analysis of Multimode Single Precision Floating Point Arithmetic Unit Using Verilog

Sachin Saraswat[1], and Sunita Malik[2]

[1]*Student, Dept. of ECE (VLSI), DCRUST, Murthal, Haryana, India*
[2]Assistant Professor, Dept. of ECE (VLSI), DCRUST, Murthal, Haryana, India

---

ABSTRACT: *This paper presents design and implementation of multimode single precision floating point arithmetic unit using Verilog Hardware Description Language on FPGA. The multimode floating point arithmetic unit have addition, subtraction, multiplication and division operations. The target device is ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1) on which the proposed design will be physically verified. Our aim will be to design an efficient multimode floating point arithmetic unit for IEEE 754 floating point number system, which gives a better implementation in terms of area of hardware. We have four separate units for four different arithmetic operations, by combining addition and subtraction unit into one and multiplication and division unit into one and by efficient optimization of these combined units, the no of slices LUTs used in FPGA can be reduced. Thus the total area of hardware required will be reduced.*

KEYWORDS: Floating Point, Look Up Tables (LUTs), VERILOG, HDL, Adder, Multiplier.

I Introduction

Floating point numbers are basically a real numbers in binary format. The floating point arithmetic operations is generally used in business, financial and web based applications. The scientific applications are basically depends on the multimode computation. Multimode based computation are used to avoid underflow by removing multiplication by addition. Recent FPGA have a large number of look up tables (LUTs), registers, hardware multipliers and microprocessors [1]. There are lots of efforts that are made over the past few decades to improve performance of floating point computation. Floating point units are not only complex but also require more area and hence are more power consuming as compared to fixed point multiplier and the complexity of the floating point unit increases as accuracy becomes the major issue. Using these features the designs of multimode based arithmetic and floating point based circuits to be applicable to FPGAs. FPGA designers design a floating point arithmetic units on FPGA in 90's decade. Area is the main factor in all design. Even though scientific computations prefer floating point representation compared to fixed point representation, floating point arithmetic designs has increased complexity. Hence logarithmic number systems gains advantages over floating point systems [2].

So it is essential to seek out an option to feed binary numbers directly as input for these applications. By using this method the time is save and the method is easier, in current situation, this is unattainable, because within this adder/subtraction, input ought to lean in IEEE 754 format [3]. In floating point data format single precision consists of 32 bits and double precision consists of 64 bits. There are lots of efforts that are made over the past few decades to improve performance of floating point computation [4].

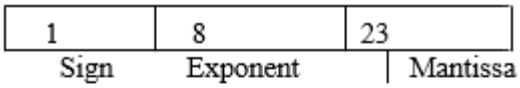

Fig 1. Single Precision

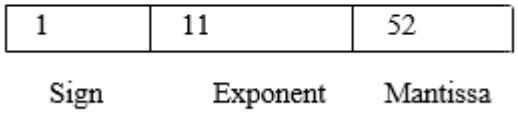

Fig 2. Double Precision

Floating point units are not only complex but also require more area and hence there are more power consumption as compared to fixed point multiplier and the complexity of the floating point unit increases as accuracy becomes the major issue. Even a small error in accuracy can cause large consequences. There are some scientific applications such as geometry computational, climate modelling require good computational requirements, for this it is required to have extreme precision in floating point calculations. But some applications do not require good precision. In that type of applications, such as even and approximate value will be sufficient for the correct

operation [5]. It would be a luxury for applications which require lower precision to use double precision of quadruple precision floating point units. But it waste area and also increases latency. These all numerous modules are written in Verilog Hardware Description Language [6] and is simulated in Xilinx. After that they are synthesized in Xilinx integrated software environment (ISE) design suite.

This paper is presented as follows: section II discus floating point Adder/Subtractions, section III discuss floating point numbers multiplier, section IV define the floating point number division section V discus Applications and Advantages of Floating point numbers and section VI discus conclusion.

II FLOATING POINT NUMBERS ADDER/SUBTRACTOR

A floating-point adder/subtractor consists of a fixed-point adder for the aligned significant, plus support circuitry to deal with the signs, exponents, alignment pre shift, normalization post shift, and special values (0,+- ∞, etc). Floating point adder is an important part of large and complex units of complex data processing such as DSP processors, graphic processors, and implementation of custom science algorithms and so on [7]. Therefore the adder should have low latency and high throughput. It also needs to be relatively simple as floating point addition is inherently more complex as compared to integer addition. A lot of research has been done on floating point addition algorithms and many designs have been tested and developed. So the main purpose of this work is to further optimize the existing designs, thus reducing latency with preservation of simplicity Separation of sign, exponent and significant for each operand and reinstating the hidden Conversion of operands to the internal format of different (e.g. single extended or double-extended) [8]. Then testing for special operands and exceptions. The difference between the two exponents is utilized to determine the amount of alignment right shift and the operand to which it should be applied. To economize on hardware, pre shifting capability is often provided for only one of the two operands, with the operand swapped if other one needs to be shifted, Since, the computed sum or difference may have to be shifted to the left in the post normalization step, several bits of the right-shifted operand, which normally would be discarded as they move off the right end, may be kept for the addition.

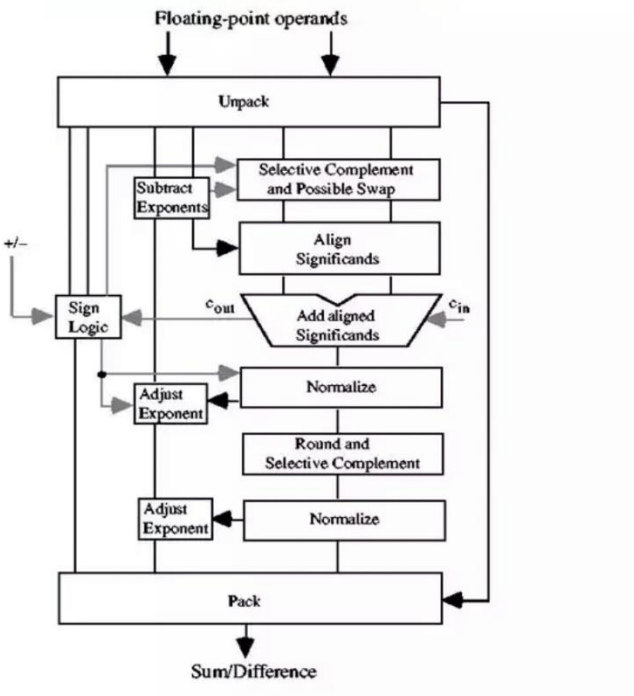

Fig 3. Block diagram of Adder/Subtractor

Thus, the significant adder is typically wider than the significant of the two output numbers. Note that in the block diagram, unlike the unpacking step, conversion between the internal and external formats is not included in the packing process. This is because converting a wider significant to a narrower one requires rounding and a best accomplished in the rounding stage which produces the result with the desired output precision [9].

III FLOATING POINT NUMBERS MULTIPLIER

Floating point multiplication is one of the crucial operation in many applications such as image processing, signal processing, etc. But every application requires different working features. Some need high precision, some need low power consumption, some need low latency, etc. DSP space application have found extensive use of multiplication of floating point numbers. Multiplication of mantissas which uses 24x24 bit integer multiplier is the most critical part in floating point multiplication [10]. Optimal run time reconfigurable hardware implementation may need the use of custom flow point formats. Floating point multiplier can have six modes of operation depending upon the accuracy or application requirement.

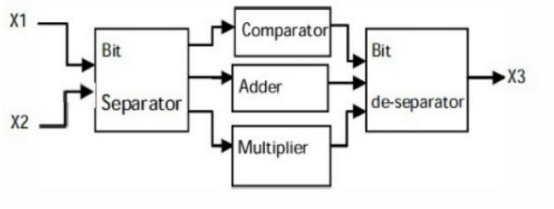

Fig 4. Block diagram of Multiplier

With the use of optimal design with custom intellectual properties a better implementation can be done by truncating the inputs before multiplication which further increases the efficiency of the multiplier. Improvement in the speed of the multiplication improves the speed of the system. Floating point multiplier can handle overflow, underflow and rounding.

IV FLOATING POINT NUMBER DIVISION

A floating point divider has the same overall structure as a floating point multiplier.

Fig 5 is generic block diagram for a floating point divider.

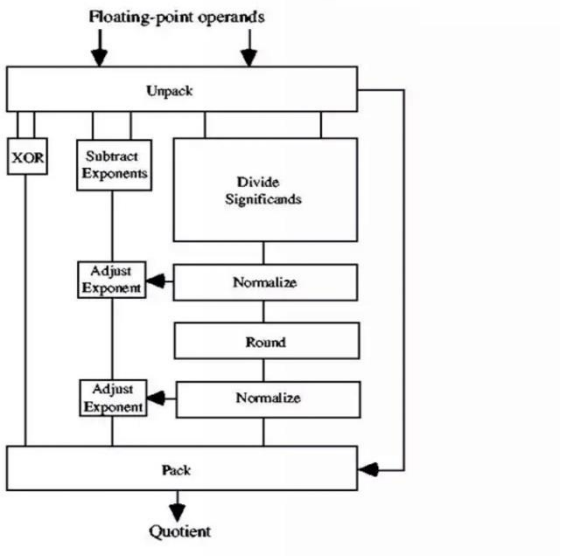

Fig 5. Block diagram of Division

Unpacking and packing have the same roles here as those discussed for floating point adders. The sign of the quotient is obtained by XORing the operand signs. The significant divider is the two operands of the floating point division are unpacked, the resulting components pass through several computation steps, and the final result is packed into the appropriate format for output. Slowest and the most complex part of the unit shown in Fig 5. The ratio of the two significant in [1, 2) is in the range (1/2, 2). Thus, the result may have to be normalized by shifting it one position to the left and decrementing the tentative exponent. Rounding the result may necessitate another normalizing shift and exponent adjustment

V APPLICATION AND ADVANTAGES

The application of Floating point numbers are widely used in digital applications such as digital filters, image transformation, and signal processing applications, the fundamental difference between fixed-and floating point DSPs is their respective numeric representation of data. While fixed-point hardware performs strictly integer arithmetic, floating-point DSP supporting industry-standard single-precision operations, the mantissa is 24 bits and the exponent is 8-bits. With its wider word width and exponential, this device provides a 16M range of precision-a vastly greater dynamic range than is available with the fixed-point format. Devices that implement industry-standard double precision (64 bits, including a 53-bit mantissa and an 11-bit exponent) can achieve even greater accuracy.

VI CONCLUSION

This review paper presented a design and analysis of Multimode Single Precision Floating Point Arithmetic Unit. In this we give brief introduction of the floating-point number, along with that floating-point numbers adder/subtractor, multiplier and divider is also described in detail

REFERENCES

[1] Smaranika Rout, Dr. S.K.Mandal, "Implementation of Low Power and High Speed Single Precession Floating" in International Journal of VLSI System Design and Communication System, vol.04, Issue.09, pp.0688-0674, sep-2016.
[2] N.Ramya Rani, V.Subbiah and L.Sivakumar, "Design of Logarithm Based Floating Point Multiplication and Division on FPGA" in ARPN journal of engineering and applied sciences, vol. 11, no. 2, January 2016.
[3] Mohamed Al-Ashrfy, Ashraf Salem and Wagdy Anis "An Efficient implementation of Floating Point Multiplier" IEEE Transaction on VLSI 978-1-4577-0069-9/11, 2011.
[4] G.Sruthi, M.Rajendra Prasad, "An Efficient Implementation of Floating Point Multiplier using Verilog", in international journal of innovation

technologies, vol.03, issue.11, pp.2107-2112, dec-2015.

[5] Elby C Varghese, Merlin Thomas, "Implementation of Single Precision Floating Point Processor Using Residue Number System" in internal journal of advanced research in electrical Electronics and Instrumentation Engineering, vol.04, Issue 11, pp.227-231, nov-2015.

[6] Whytney J. Townsend, Earl E. Swartz, "A Comparison of Dadda and Wallace multiplier delays". Computer Engineering Research Center, the University of Texas.

[7] BehroozParhani, "Computer Arithmetic Algorithm and Hardware Designs", New York Oxford University, 2000.

[8] Ishan A Patil, Vishwas V. Balpande, "Implementation of Floating-Point Fused Basic Arithmetic Module Using Verilog", IEEE ICCSP conference, 2015.

[9] DjordjePesic, Ivan Ratkovic, "An Efficient FPGA Implementation Of Floating-Point Addition", 23 Telecomunications forum TELFOR, 2015.

[10] Arish S, R.K. Sharma, "Run time Reconfigurable Multi-Precision Floating-Point Multiplier Design For High Speed, Low Power Application", 2nd International Conference on signal Processing and Integrated Networks, 2015.

[11] Arish S, R.K.Sharma, "An Efficient Floating-Point Multiplier Design For High Speed Applications Using Karatsuba algorithm and Urdhva-TiryagbhyamAlgorithm",IEEE, 2015.

[12] RupaliBhoyar, PrasannaPalsodkar, "Design and Implementation of Goldschmidts Algorithm for Floating Point Division and square root",IEEE ICCSP conference, 2015.

[13] Paldurai.K, Dr. K. Hariharn, "FPGA Implementation of Delay Optimized Single Precision Floating.