IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 5, Issue 1, Ver. I (Jan - Feb. 2015), PP 50-53
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Implementation of 32 Bit Binary Floating Point Adder Using IEEE 754 Single Precision Format

Rupali Dhobale[1], Soni Chaturvedi[2]
[1]*(III SemMtech(Communication) , ECE,PIET,Nagpur/ R.T.M.N.U., India)*
[2]*(Asst.Professor, ECE,PIET,Nagpur/ R.T.M.N.U., 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. 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. Here pipelined architecture is used in order to increase the performance and the design is achieved to increase the operating frequency. The logic is designed using VHDL. This paper discusses in detail the best possible FPGA implementation will act as an important design resource. The performance criterion is latency in all the cases. The algorithms are compared for overall latency, area, and levels of logic and analyzed specifically for one of the latest FPGA architectures provided by Xilinx.*
Keywords: *ASIC,FPGA,IEEE 754,pipelining,VHDL*.

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. IEEE 754 Precision Binary Format

The IEEE 754 Single Precision Binary Format is as shown below:-

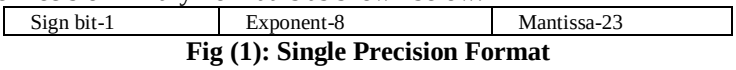

Fig (1): Single Precision Format

Following are the steps for Converting a decimal number to floating point number :-

1. Convert a Decimal number to Binary number
   $(975.75)_{10} = (1111001111.11)_2$
2. Normalize the number
   $1.11100111111 * 2^9$
3. From this normalized number we can fill all 32-bits of floating point number
   Sign bit = 0 (number is positive)
4. Exponent = Bias + 9 = 127 + 9 = $(136)_{10} = (1000\ 1000)_2$
5. Fraction part will contain all the bits after decimal point.
6. $(975.75)_{10}$ is expressed as shown below in single precision floating point format.

0	1000 1000	11100111111000000000000

Fig (2): Single Precision Format of $(975.75)_{10}$

III. Flowchart For Proposed Implementation

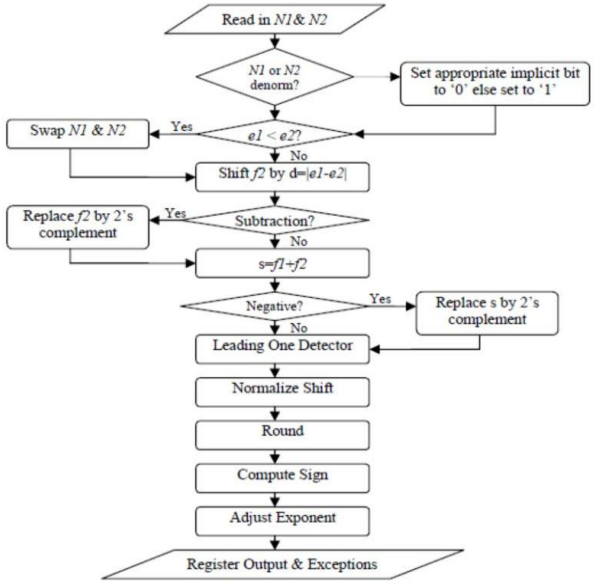

Fig (3) : Flowchart of proposed Implementation

A description of the proposed implementation algorithm is as follows:-

1. The two operands, N1 and N2 are read in and compared for denormalization and infinity. If numbers are denormalized , 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 riht 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 Standard Floating Point Adders

The block diagram of standard floating point adder is shown in Fig(4) . It shows the main hardware modules necessary for floating-point addition. 1)The exponent difference module, 2) Right shift shifter, 3) 2's complement adder, 4) Leading one detector, 5) Left shift shifter, and 6) The rounding module. The bit-width as shown in Fig(4) .

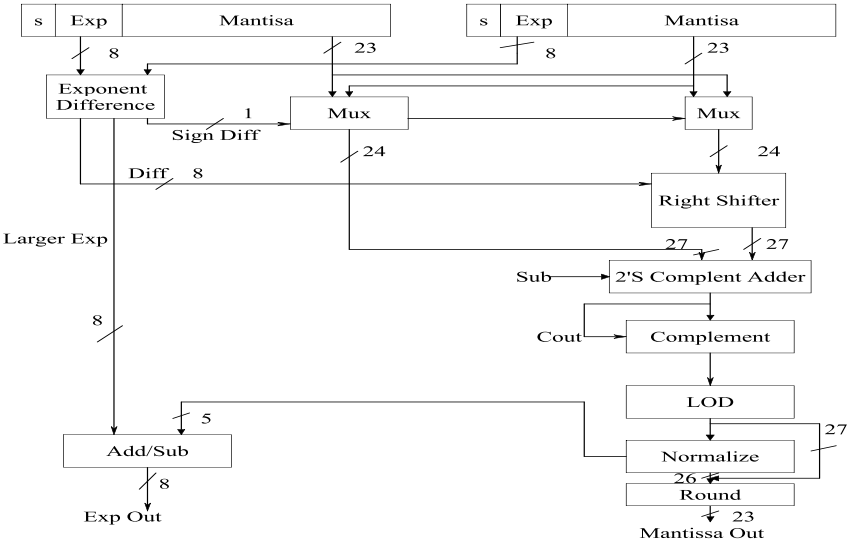

Fig (4): Block diagram of standard floating point adder

The main hardware modules for a single-precision floating-point adder are:

1) The exponent difference module: It has the following two functions:

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

2) **Right shift shifter:** The right shifter is used to shift right the significand of the smaller operand by the absolute exponent difference. This is done so that the two numbers have the same exponent and normal integer addition can be implemented. Right shifter is one of the most important modules when designing for latency.

3) **2's complement adder:** 2's complement adder is a simple integer addition process which adds or subtracts the pre-normalized significands.

4) **Leading one detector:** 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.

5) **Left shift shifter:** Using the results from the LOD, the result from the adder is shifted left to normalize the result. That means now the first bit is 1. This shifter can be implemented using "shl" operator in VHDL or by describing it behaviorally using 'case' statements.

6) **The rounding module**: Rounding is done using the guard, round and sticky bit of the result. REN mode is accomplished by rounding up if the guard bit is set, and then pulling down the lowest bit of the output if the r and s bits are 0. A 1 is added to the result if r and s bit are 1 or r and either of the last two bits of the normalized result is 1. This step is really important to assure precision and omit loss of accuracy.

V. Advantages And Applications

Advantages:

1. This floating point adder algorithm mainly reduces the overall latency and improves the performance.
2. It reduced power consumption of floating point unit without sacrificing correctness.
3. Design challenges of the redundant format, namely the leading digit detection and the rounding.
4. This floating point adder unit performs the addition and subtraction using substantially the same hardware used for floating point operation. This advantage causes saving the core area by minimizing the number of element.
5. Real floating point hardware uses more sophisticated means to round the summed result but in this paper take the simplification of the truncating bits if there are more bits that can be represented.

Applications:

- Floating point adder is generally used in the arithmetic calculations.
- This adder can be used in the instruments which involve mathematical processes.
- The floating point adder is used in such as :

1. CPU
2. Calculators

3. Watches
4. Pocket adder
5. 32 bit ALU

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

[9]. John P Hay, 'Computer Architecture and organization' Tata Mc-graw hill 3 rd edition.

[10]. J.Bhaskar, 'VHDL Primer', PHI publication 3rd Edition .