International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 5 146 – 148

Design and Simulation of Double Precision Floating Point Adder-A Literature Review

Sharon Bhatnagar
Student, Department of ECE
Lakshmi Narain College of Technology
Bhopal, India
sharon_bhatnagar1989@rediffmail.com

Soheb Munir
Assistant Professor, Department of ECE
Lakshmi Narain College of Technology
Bhopal, India
munir_soheb@yahoo.co.in

Abstract—Floating Point arithmetic is widely used in large set of scientific and signal processing computation. Floating point addition is a complex task in computers due to computational burden associated with repeated calculation with high precision numbers. IEEE-754 defines two format single precision and double precision for floating point numbers. This paper describes the analysis of work done in double precision floating point adder regarding its parameter latency, throughput, hardware, clock cycle also the future challenges and scope of work in this field.

Keywords—*Double precision,floating point,adder,IEEE-754,FPGA*

*****

I. INTRODUCTION

Integers provide an exact representation for numeric values but they suffer from two major drawbacks the inability to represent fractional values and a limited dynamic range. Two's complement representation deal with signed integer values only. Without modification, these formats are not useful in scientific or business applications that deal with real number values. Floating-point numbers remove this problem and this makes it a natural choice for scientific and signal processing computations. Most of today's computers are equipped with specialized hardware that performs floating-point arithmetic with no special programming required Floating-point numbers allow an arbitrary number of decimal places to the right of the decimal point 0.5 × 0.25 = 0.125. Computers use a form of scientific notation for floating-point representation Numbers written in scientific notation have three components:

Sign bit	Biased Exponent	Significand/Mantissa/Fraction

The one-bit sign field is the sign of the stored value. The size of the exponent field determines the range of values that can be represented. The size of the significand determines the precision of the representation.

The IEEE has established a standard for floating-point numbers.IEEE-754 single precision and IEEE 754 double precision. The IEEE-754 *single precision* floating point standard uses an 8-bit exponent (with a bias of 127) and a 23-bit significand. The IEEE-754 *double precision* standard uses an 11-bit exponent (with a bias of1023) and a 52-bit significand. In both the IEEE single-precision and double-precision floating-point standard, the significant has an implied 1 to the LEFT of the radix point. The format for a significand using the IEEE format is: 1.xxx…For example, 4.5 = .1001 x 2^3 in IEEE format is 4.5 = 1.001 x 2^2. The 1 is implied, which means is does not need to be listed in the significand (the significand would include only 001). For example, the floating point representation of the decimal number 3.12 will be 0l00000000I000 1111010111000010100011110101110000101000111101l0 when represented as a double precision floating point number. The sign bit '0' represents the positive sign, the exponent "10000000000",of which the 11th bit corresponds to the sign bit of the exponent, effectively making the range of the exponent [-1023,l024]. Thereafter, a bias of 1023 is used for determining the exponent. So the exponent of this number will be 0 and the mantissa has a hidden bit of value' l' before the msb. Therefore, the mantissa becomes (including the hidden bit 1.1000111101011100001010001111010111000010100011llO 110. The first bit is hidden because it is always 1.
However, for the preprocessing of the floating point numbers before the addition or subtraction we have to consider the hidden bit also.

Using the IEEE-754 single precision floating point standard: An exponent of 255 indicates a special value.If the significand is zero, the value is ± infinity.If the significand is nonzero, the value is NaN, "not a number," often used to flag an error condition.Using the double precision standard:The "special" exponent value for a double precision number is 2047, instead of the 255 used by the single precision standard.

Floating-point addition and subtraction are done using methods analogous to how we perform calculations using pencil and paper. The first thing that we do is express both operands in the same exponential power, then add the numbers, preserving the exponent in the sum. If the exponent requires adjustment, we do so at the end of the calculation. Floating-point overflow and underflow can cause programs to crash. Overflow occurs when there is no room to store the high-order bits resulting from a calculation. Underflow occurs when a value is too small to store, possibly resulting in division by zero.

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.

Recent advances in FPGA technology made it a good choice for speeding up many computationally intensive scientific and engineering applications. The variable-input Look-Up Table (LUT) architecture has been a fundamental component of the Xilinx Virtex architecture first introduced in 1998 and now in its fourth generation. This distinctive architecture enables flexible implementation of any function with eight variable inputs, as well as implementation of more complex functions Recent FPGAs have large amounts of programmable logic, memory and often come with a large set of high speed intellectual property (IP) cores to implement specific functions.The Virtex architecture enables users to implement these functions with minimal levels of logic. By collapsing levels of logic, users can achieve superior design performance. The design is still based on the fundamental Configurable Logic Block (CLB).

The main objective of implementation of floating point adder on the reconfigurable hardware i.e. on FPGAs is to utilize less chip area, less number of destination paths/ports, less clock period, less combinational delay and faster speed. Less chip area means less number of slices is used in reconfigurable hardware i.e. on FPGAs. Less combinational delay means less latency i.e. less time is required to appear an output after the input response is applied and if there is less latency then there will be the faster speed. If there is less number of components are used on FPGAs then less number of paths are used to connect them.

II. IEEE COMPATIBLE FLOATING POINT ADDERS ALGORITHM

For addition of two numbers in floating-point format standard algorithm is already defined. The steps in this algorithm are-

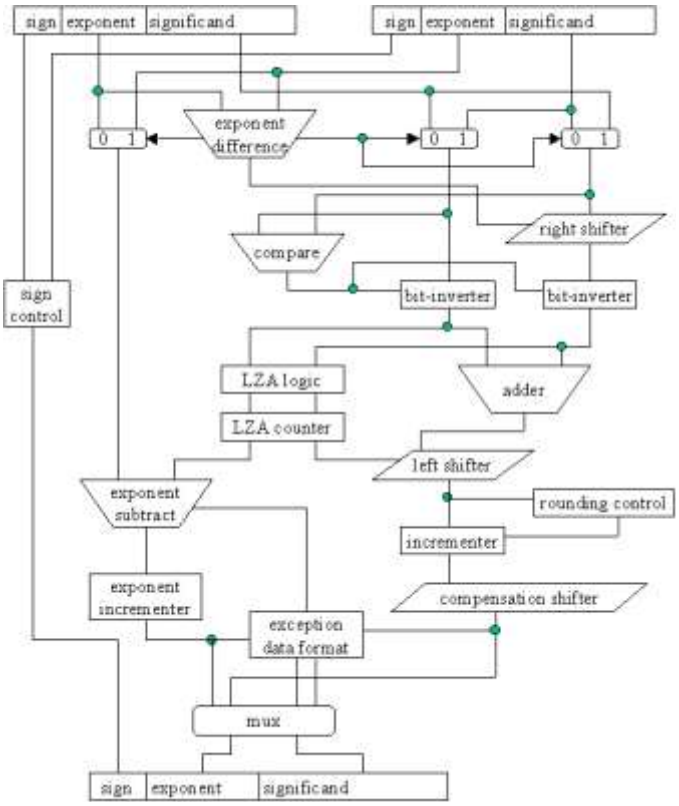
Step 1- Compare the exponents of two numbers for (or) and calculate the absolute value of difference between the two exponents (). Take the larger exponent as the tentative exponent of the result

Step 2 -Shift the significand of the number with the smaller exponent, right through a number of bit positions that is equal to the exponent difference. Two of the shifted out bits of the aligned significand are retained as guard (G) and Round (R) bits. So for p bit significands, the effective width of aligned significand must be p + 2 bits. Append a third bit, namely the sticky bit (S), at the right end of the aligned significand. The sticky bit is the logical OR of all shifted out bits.

Step 3- Add/subtract the two signed-magnitude significands using a p + 3 bit adder. Let the result of this is SUM.

Step 4 -Check SUM for carry out (Cout) from the MSB position during addition. Shift SUM right by one bit position if a carry out is detected and increment the tentative exponent by 1. During subtraction, check SUM for leading zeros. Shift SUM left until the MSB of the shifted result is a 1. Subtract the leading zero count from tentative exponent.
Evaluate exception conditions, if any.

Step 5 -Round the result if the logical condition R"(M0 + S'') is true, where M0 and R'' represent the pth and (p + 1)st bits from the left end of the normalized significand. New sticky bit (S'') is the logical OR of all bits towards the right of the R'' bit. If the rounding condition is true, a 1 is added at the pth bit (from the left side) of the normalized significand. If p MSBs of the normalized significand are 1's, rounding can generate a carry-out in that case normalization (step 4) has to be done again.



III. LITERATURE SURVEY

Lots of research work has been done in this field. One of the first competitive floating-point addition implementation is done by L. Louca, T. Cook, and W. Johnson in 1996. Single precision floating-point adder was implemented for Altera FPGA device.

Somsubhra ghosh implement a double precision IEEE floating-point adder that can complete the operation within two cycles with improved latency and chip area [1].He implemented the adder in XC2V6000 and XC3S1500 Xilinx FPGA devices. So many other research work has been done in this field. Michael Nachtigal implement a reversible floating-point adder architecture[2].Manish Kumar Jaiswal implement an adder architecture which is both area as well as performance optimal. His proposed design has optimized the individual complex components of adder module to achieve the better overall implementation[3].Chi.Huang implemented a design of high speed double precision floating-point adder using macro modules with chip area 1.44mm2 and clock frequency is 518Mhz[4].Kikkeri.n Seidel implemented a fully verified double precision IEEE floating-point adder. Proposed design incorporates many optimizations like a non-standard sepration into two paths, a simple rounding algorithm, unification of rounding cases for addition and substraction resulting implementation has a total latency13.6ns on an altera stratix 2 device. He partitioned the design into a 2stage pipeline running at a frequency of 14.7 Mhz[5].Liang-Kai Wang presents a novel design for a decimal floating-point adder and a decimal injection-based rounding in his research. Meenu Talwar Implemented a floating-point adder on FPGA With 349 slices and 69.987n sec consuming 187244 kbytes of

Memory with 1 global clock [6].

IV. FUTURE SCOPE

One can do work in area of latency, optimizing hardware Increasing speed using VHDL or Verilog through FPGA.

REFERENCES

[1] Somsubra Ghosh,Prarthana Bhattacharyya,Arka Dutta "FPGA Based Implementation of a double Precision IEEE Floating-Point Adder"7th International conference on intelligent systems and control, pp. 271-275 4-5 Jan 2013.

[2] Michael Nachtigal, Himanshu Thapliyal "Design of a reversible floating- Point adder architecture", 11th IEEE international conference on Nanotechnology,pp.451-456,15-18 August 2011.

[3] Manish Kumar Jaiswal And Ray C.C Cheung, Sharmelee Thangjam, "High performance FPGA Implementation of double precision floating Adder/subtractor", International Journal of Hybrid information Technology,vol 4,no 4,pp 71-80,Oct 2011.
Computer Applications, vol. 46, no. 9, pp. 1-5, May 2012.

[4] Chi.Huang,Xingu Wu, " Design of High Speed Double Precision
Floating point adder using macro modules",Design Automation
Confrence,2005. Proceedings of the ASP-DAC 2005.pp.11-12
18-21,Jan 2005.

[5] N. Kikkeri, P.M. Seidel, "An FPGA Implementation of a Fully Verified Double Precision IEEE Floating-Point Adder",
Proc. of IEEE International Conference on Application-specific
Systems, Architectures and Processors, pp. 83-88, 9-11 July 2007.

[6] Meenu Talwar,Karan Gumber, Sharmelee Thangjam,"Performance
Analysis of Floating Point Adder using sequential processing on
Reconfigurable Hardware", International Journal of Engineering
Research and Applications, vol. 2, no. 9, pp. 1226-1229, May-jun 2012.

[7] A. Beaumont-Smith, N. Burgess, S. Lefrere, C. Lim,"Reduced Latency IEEE Floating-Point Standard Adder Architectures," Proc. of 14th IEEE Symposium on Computer Arithmetic, pp. 35-43, 1999.

[8] N. Quach, N. Takagi, and M. Flynn, "On fast IEEE Rounding", Technical Report CSL-TR-91-459, Stanford Univ., Jan. 1991.

[9] P.-M. Seidel, "On the Design of IEEE Compliant FloatingPoint
Units and their Quantitative Analysis", PhD thesis, Univ. of Saarland, Germany, Dec. 1999.

[10] P.-M. Seidel, G. Even, "How Many Logic Levels Does Floating-Point Addition Require?", Proc. of International Conference on Computer Design (ICCD '98): VLSI, in Computers & Processors, pp. 142-149, Oct. 1998.

[11] We. Park, T.D. Han, S.D. Kim, S.B. Yang, "Floating Point Adder/Subtractor Performing IEEE Rounding and Addition/
Subtraction in Parallel", IE ICE Trans. on Information and Systems, vol. 4, pp. 297-305, 1996.

[12] S. Oberman, H. A1-Twaijry, and M. Flynn, "The SNAP Project:
Design of Floating Point Arithmetic Units", Proc. of 13th IEEE
Symposium on Computer Arithmetic, pp. 156-165, 1997.

[13] S. Oberman, "Floating-Point Arithmetic Unit Including an Efficient Close Data Path," AMD, US patent 6094668, 2000.

[14] V. Gorshtein, A. Grushin, and S. Shevtsov, "Floating Point Addition Methods and Apparatus." Sun Microsystems, US patent 5808926, 1998.

[15] G. Even, P.M. Seidel, "A comparison of three rounding algorithms for IEEE floating-point multiplication", Proc. of 14th IEEE Symposium on Computer Arithmetic, pp. 225-232, 1999.

[16] IEEE Computer Society, "IEEE Standard for Floating-Point

[17] Arithmetic", IEEE Std. 754T>1-2008 (Revision of IEEE Std