www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 6 June 2015, Page No. 12923-12926

Novel Technique for Parallel Pipeline Double Precision IEEE-754 Floating Point Adder

Shrikant Fulzele[1], Prof. Venkat Ghodke[2]

[1]G.S Moze College of Engineering, Balewadi, Pune-45, India
Shrikantfulzele5@gmail.com

[2]AISSMS Information of Technology, Shivaji nagar,
Pune-01, India
Venkatghodke@gmail.com

Abstract: ***The Floating Point Additions are critical to implement on FPGAs due to their complexity of their algorithms in hard real-time due to excessive computational burden associated with repeated calculations with high precision numbers. Thus, many scientific problems requires floating point arithmetic with high level of accuracy in their calculations. Moreover, at the hardware level, any basic addition or subtraction circuit has to incorporate the alignment of the significands. This Paper represents Novel technique for implementation of parallel pipeline Double precision IEEE-754 floating point adder that can complete a operation in two clock cycle. This kind of technique can be very useful for parallelism of FPGA device and this proposed technique can exhibits improvement in latency and also in operational chip area management. The proposed double precision floating point adder has been implemented with XC2V6000 and XC3SI500 Xilinx FPGA Device.***

Keywords: Floating Point Addition, IEEE-754 Standard, FPGA, Delay Optimization, VHDL.

1. 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 [1] over the span of last two decades.

The recent time in the area of Field Programmable Gate Array (FPGAs) has given many useful ways of doing things and tools for the development of dedicated and reconfigurable hardware employing complex digital circuits at the chip level. Therefore, FPGA technology can be productively used in order to develop digital circuits so that the problem of floating-point representation of numbers and the computational resources useful while performing the math and logical operations during execution of the set of computer instructions could be solved at the hardware level. This investigation presents a new technique

to represent a double precision IEEE floating-point adder that can complete the operation within two clock cycles.

A number of works have been reported in the literature with an aim to achieve a reduced latency realization of floating-point operations. [1-2] The algorithm in effectively finishes the floating-point addition within two clock cycles with the packet forwarding format for handling data hazards in deeply pipe lined floating-point pipelines. Our proposed technique has exhibited significant improvement in the latency reduction as well as also in the operational chip area management while implementing a dedicated double precision IEEE floating-point adder in FPGA based embedded system.

The proposed Double precision floating point adder has been implemented on FPGA device. All the parameters of FPGA device like use of slices, number of slice flip flop, number of 4 input LUTs and so on are observed. The significant improvement on previous algorithm and parallel pipeline improves its latency and helps to complete a operation in two clock cycle.

2. Related Work

Purna Ramesh Addanki, Venkata Nagartna Tilak Alapati andMallikarjuna Prasad Avana (2013) presented a high speed floating-point double precision adder/subtractor and multiplier, which are implemented on a virtex-6 FPGA using Verilog language. Their proposed designs were compliant with IEEE 754 standard format and handles overflow, underflow cases and rounding mode. The IEEE standard specifies four rounding modes and the rounding odes are selected for various bit combinations of mode. Based on the changes in the rounding to the mantissa corresponding changes has to be made in the exponent path also. They showed that, their presented design was achieved high operating frequency with better accuracy and considerably good performance. [9].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 key role of the LOP is to calculate the leading number of zeros in the addition result, working 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].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].

3. IEEE-754 Standard Floating-Point Numbers

An IEEE standard floating point numbers are of different types according to their precisions i.e. the number of their mantissa bit length. In accordance with IEEE 754-2008, there are half, single, double and quadruple precision binary numbers having a mantissa of bit length 16, 32, 64, 128 respectively. Out of these, the double precision number is most widely used in the area of binary applications. This type of representation of the numbers is advantageous due to fact that a large spectrum of numbers can be expressed with a limited number of bits. A double precision floating point number has a greater dynamic range and consists of 64 binary bits. Out of which the 1 sl bit is the sign bit, the next 11 bits are the exponent and the remaining 52 bits represent the mantissa. This has been explained in the Figure 1.

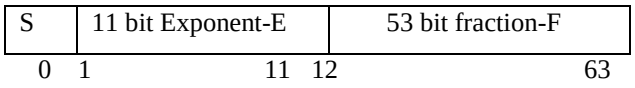

Figure 1. IEEE-754 double precision format

For example, the floating point representation of the decimal number 3.12 will be 0l00000000001000-1111010111000010100011110101110000101000111101l0 when represented as a double precision floating point number. The sigh 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 [-1023,1024]. 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) 1.1000111101011100001010001111010111000010100011110l0 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. Computation of the IEEE representations of the rounded sum:

$$rnd(sum)=rnd((-1)^{sa}.2^{ea.}.fa+(-1)^{(SOP+sb)}.2^{eb}.fb) \qquad (1)$$

Let the effective sign of operation be

$$S.EFF = sa \oplus sb \oplus SOP$$

So, for S.EFF = 0 the circuit will perform an essential addition and if S.EFF = 1 then the arithmetic operation will essentially be a subtraction. From these two numbers, and the exponent difference 0, the small operand is defined as (ss, es, fs) and the large operand is denoted as, (sl, el, fl). The resulting sum can be computed as [1]:

$$Sum=(-1)^{sl}.2^{el}.(fl+(-1)^{S.EFF}(fs.2^{-|\delta|})) \qquad (2)$$

4. Proposed Algorithm

We have followed a similar approach as [1] for designing the basic algorithm for this implementation. The floating point arithmetic in [1] is two stage pipe lined which are divided into two paths, namely "R-Path" and "N-Path". The two paths are selected on the basis of the exponent difference. The new algorithm has been arrived at by following a few implemental changes in the algorithm of [1].

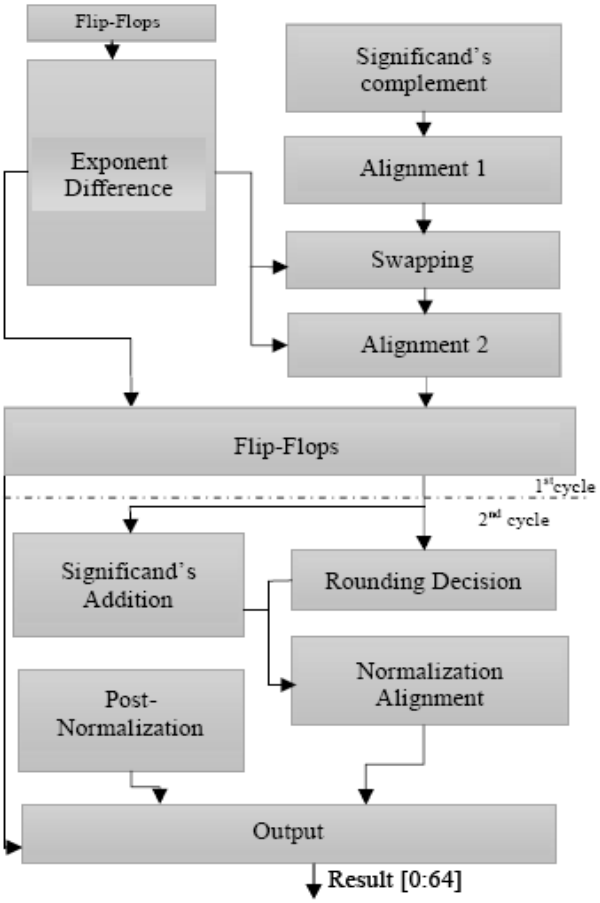

Figure 2. IEEE-754 double precision format

This algorithm is broken into two pipeline stages, which are executed in two different clock cycles. The advantage of the pipelining mechanism is that, despite having a higher input-output sequential length, they offer an unmatched throughput by virtue of their assembly line structure. An overview of the proposed algorithm is explained by Figure 2.

A. First Clock Cycle Operation

This is the first stage in the pipeline mechanism. The components of the Floating Point number, in terms of bit vector, are,

(S, E [0:10], F [0:52])

The basic algorithm operates only with normalized FP numbers, i.e. f ε [1, 2]. The basic operation is performed within two clock stages, and is determined by the parameter.

SOP ε {0, l}

It is supplied as an input to the algorithm. The mathematical operation to be performed is determined by calculating the effective sign of operation,

S.EFF = sa ⊕ sb ⊕ SOP

After this, some initial preprocessing operations are done before adding or subtracting the two numbers. The exponent difference is obtained and is represented as $^{\delta=ea-eb}$ then the number with the smaller magnitude is sorted out through various operations based on conditions derived from the effective sign and the resultant of the exponent difference. In case the exponent difference is in the range [-63, 64] the smaller significand is shifted by MAG_MED positions to the right. The amount of alignment shift in medium range is determined by the modular value of the exponent difference 8,

i.e. MAG_MED. The alignment shift can be formulated as:

$$(-1)^{SIGN\_MED}.\langle MAG\_MED\rangle = \begin{cases} \delta - 1 & if\ 64 \geq \delta \geq 1 \\ \delta & if\ 0 \geq \delta \geq -63 \end{cases} \quad (3)$$

B. Second Clock Cycle Operation

This is the second stage of the pipelining mechanism. The two "preprocessed" significands are added and the result is rounded in accordance with the IEEE standard rounding algorithm. Here the rounding algorithm from [4] has been implemented. At the end, it is normalized. The output result is a 64 bit binary floating point number.

$$\text{rnd (sum)} = \text{rnd}\left((-1)^{sa}.2^{ea}.fa+(-1)^{(SOP+sb)}.2^{eb}.fb\right) \quad (4)$$

C. Algorithm for Addition

Let s1; e1; f1 and s2; e2; f2 be the signs, exponents, and significands of two input floating-point operands, N1 and N2, respectively. Given these two numbers, Figure 4 shows the flowchart of the standard floating-point adder algorithm. A description of the 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 53 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

5. Implementation Details

The implementation of this algorithm is has been accomplished using the Xilinx XC3S1500 device of Spartan 3 family. The Xilinx ISE 14.1 is used to generate a code and also a designing tool. Also there are scopes for further development. A report has been generated for estimation of usage of resources in below table. I

Table no. I Device Utilization Summary

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
No. of slice LUTs	333	407	47%
No. of fully used LUT-FF pairs	604	1408	42%
No. of bounded IOBs	190	108	178%
No. of BUFS/BUFGCTRL	1	24	4%

6. Simulation Results

The Floating point adder has been simulated using Xilinx 14.1. While the device utilization summary has been showed above. Figure 3(a) and 3(b) shows the simulation result of addition and subtraction respectively. And Figure 3(c) shows a schematic diagram of a design.

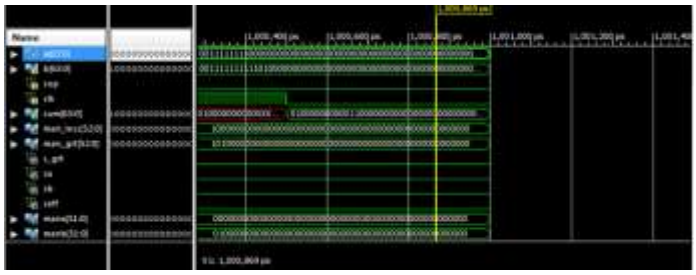

Fig no. 3(a) Simulation result of Addition

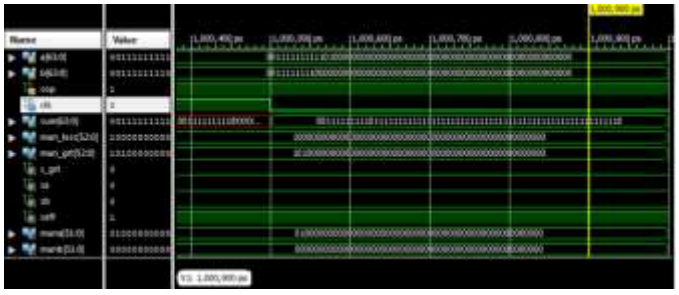

Fig No. 3(b) Simulation result of Subtraction

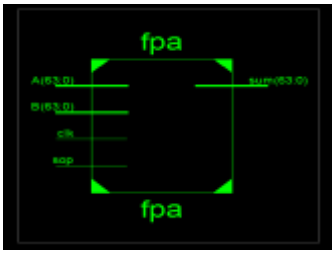

Fig No. 3(c) Schematic View of Design

Conclusion

This paper has successfully demonstrated an implementation of a high speed, IEEE 754, double precision floating point adder with a significant decrease in latency. This manifest in the fact that FPGA based embedded systems has a higher advantage of lower computational aspects. Also, an implementation work of this algorithm, on the Xilinx Spartan-3 FPGA would give results with further improvement.

Acknowledgment

The authors would like to take this opportunity to acknowledge Prof. Venkat Ghodke AISSMS IOT Shivaji Nagar, Pune for being a great source of encouragement, ADRIN for presenting us with the scope of learning, and the anonymous reviewers for their insightful comments.

References

[1] Somsubhra Ghosh, Prarthana Bhattacharyya, and Arka Dutta, “FPGA Based Implementation of a Double Precision IEEE Floating-Point Adder”, Proc. of 7th International Conference on Intelligent Systems and Control (ISCO 2013), pp 271-275, 2013

[2] Adarsha KM, Ashwini SS, Dr. MZ Kurian, "Double Precision IEEE-754 Floating-Point Adder Design Based on FPGA", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Issue 4, April 2014

[3] Purna Ramesh Addanki, Venkata Nagaratna Tilak Alapati and Mallikarjuna Prasad Avana, “An FPGA Based High Speed IEEE – 754 Double Precision Floating Point Adder\Subtractor and Multiplier Using Verilog”, International journal of advanced science and technology. Vol. 52, March, 2013.

[4] Paschalakis, S., Lee, P., “Double Precision Floating-Point Arithmetic on FPGAs”, In Proc. 2003 2nd IEEE International Conference on Field Programmable Technology (FPT ‘03), Tokyo, Japan, 2003.

[5] Peter-Michael Seidel, Guy Even, "Delay-Optimized Implementation of IEEE Floating-Point Addition", IEEE Trans. on Computers, vol. 53, no.2, pp. 97-113, Feb. 2004.

[6] L. Louca, T. Cook, and W. Johnson, “Single precision floating-point adder for Altera FPGA device”, IEEE Trans. on Information and Systems, vol. 4, pp. 297-305, 1996.

[7] W. Ligon, S. McMillan, G. Monn, F. Stivers, and K. Underwood, “Reconfigurable hardware to perform high precision operations on FPGAs”, Proc. of IEEE International Conference on Application-specific Systems, Architectures and Processors, pp. 83-88, 1999

[8] E. Roesler, B. Nelson, “Novel optimizations for arithmetic hardware”, Proc. 2002 2nd IEEE International Conference on Field Programmable Technology (FPT ‘02), 2002.

[9] J. Liang, R. Tessier, and O. Mencer, Floating Point Unit Generation and Evaluation for FPGAs, in the Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, California, April 2003.

[10] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna, “Analysis of HighPerformance Floating-Point Arithmetic on FPGAs”, proc. IEEE Trans. on Computers, vol. 49, no. 1, pp. 33-47, 2003.

[11] Peter-Michael Seidel, Guy Even, "Delay-Optimized Implementation of IEEE Floating-Point Addition", IEEE Trans. on Computers, vol. 53, no. 2, pp. 97-113, Feb. 2004.

Author Profile

Shrikant Fulzele received B.E Degree in Electronics Engineering from Nagpur University (2008-2012) and M.E Degree in VLSI and Embedded System from Savitribai Phule Pune University (2013-2015)

Prof. Venkat Ghodke received the B.E. degree in Electronics Engineering from Dr.B.A.M.University, Aurangabad, Maharashtra, India, in 1997, and the M.E degree in Electronics Engineering specialization with Digital System from Pune University, Maharashtra, India, in 2010.
Currently He is an Assistant Professor in AISSMS’S Institute of information technology of Savitribai Phule University of Pune, India. His research interests include digital image processing and embedded system area.