Design and Simulation of Double Precision Floating-Point Adder

Sharon Bhatnagar
Student, Department of ECE
Lakshmi Narain College of Technology
Bhopal, India

Soheb Munir
Assistant Professor, Department of ECE
Lakshmi Narain College of Technology
Bhopal, India

***Abstract*—Floating point numbers are very important part of computer processing. Addition imposes a great challenge due to its processing time. This paper presents a technique for addition of IEEE 754 double precision floating-point numbers within two clock cycles. This paper results also show improvements in power utilization, operational chip area management and optimization of hardware. This proposed adder is implemented with the help of 6slx45tfgg484-3 Spartan family as well as 5vfx70tff1136-1 of Virtex Xilinx FPGA devices.**

Keywords—IEEE 754,Floating Point Addition,Adder,FPGA

I. INTRODUCTION

By using floating point numbers in computation we can represent number in a way that it can tradeoff between range as well as precision. As we know computer memory is limited so we cannot store any number with infinite precision so we use floating point numbers. From early times so many formats have been used but since 1990 IEEE make a standard for floating point numbers (IEEE 754).In this there are two kinds of format IEEE 754 single precision and IEEE 754 double precision format. Floating point numbers are represented in the form.

Sign Bit	Biased exponent	Significand/mantissa/fraction

Sign bit represents the sign of the number if it is zero then it represents a positive number and if it is zero then represents negative number, exponent field determine range of numbers which can be represented, significand represents precision of number.

PARAMETER	IEEE-754 SINGLE PRECISION	IEEE-754 DOUBLE PRECISION
Total no. of bits	32	64
Significand bits	23+1	52+1
Exponent bits	8	11
Sign bit	1	1
Special e value	255	2047
Bias	127	1023

Here we concentrate on double precision floating point numbers. Simply if we want to convert a number into IEEE 754 double precision format. First we need to convert that number into binary then it should be normalized and exponent is biased and then written in standard format. Suppose we have number (-3.75). Then binary conversion of 3 is (11) and for 0.75 it is (1011).So (3.75) is equal to (1.11011*2^1), as exponent is one here it must be biased means (1023+1)=1024 it is again converted into binary. The final conversion of (-3.75) into IEEE 754 double precision format is 1 10000000000111000000000000000000000000000000000 000000000000000.

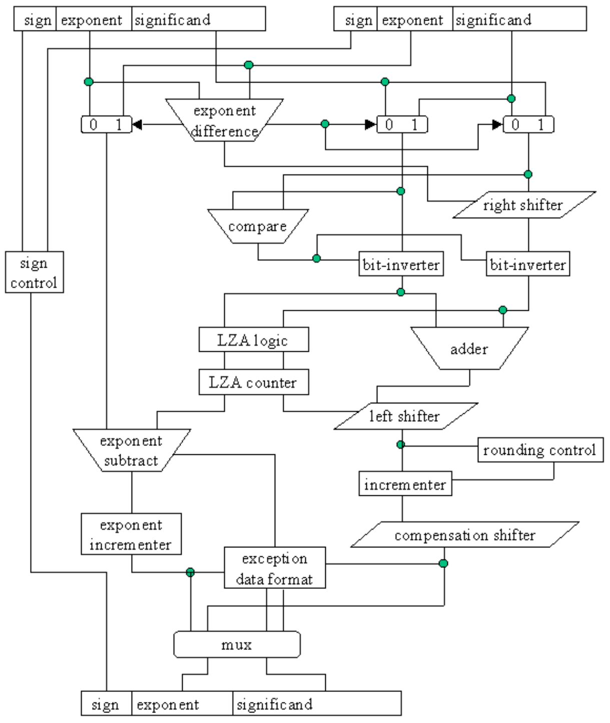
Lots of work have been done in this field. First implementation is done by l.louca and T.Cook and W.Johnson in 1996.The algorithm in[2] implement adder, subtractor and multiplier designs and achieved the operating frequencies of 363.76Mhz and 414.714hz with an area of 660 and 648 slices. Error detection technique is presented in [3].Optimization of each individual component of adder has been done in [4].Overall speed is the main concern of the operation due to large number of bits.

The proposed algorithm is implemented with the help of Xilinx FPGA device 6slx45tfgg484-3 and 5vfx70tff1136-1.Various parameters like number of slice flip-flops, Number of 4 input LUTs, number of global clks have been observed. Proposed technique show improvements in hardware optimization, latency and chip area compare to algorithm used in [1].

Rest of the paper is organized as section II presents addition of two floating point numbers. Section III describes the proposed algorithm. Section IV shows the device usage section V shows the simulation results. Section VI contain conclusion and future scope is provided in section VII.

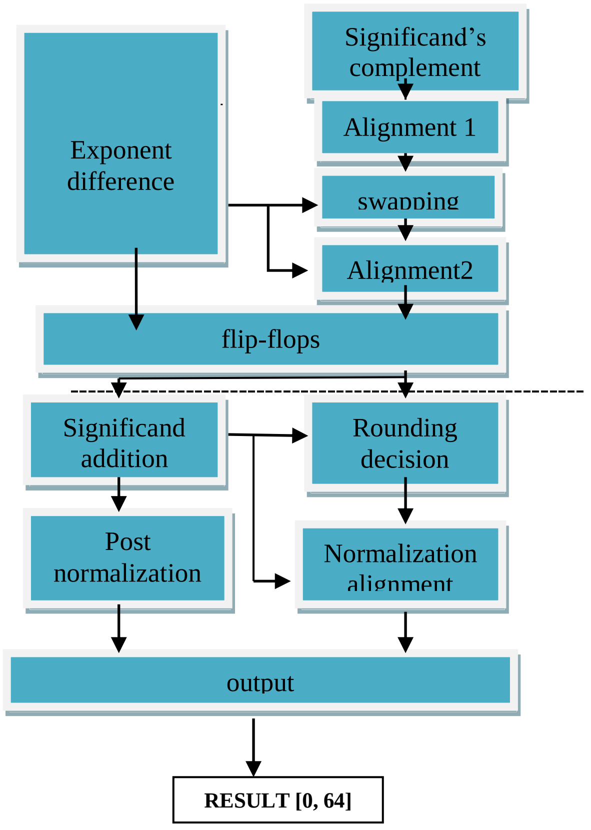
II. ADDITION OF TWO IEEE 754 DOUBLE PRECISION FLOATING POINT NUMBERS

In order to add to given numbers into double precision the two numbers primarily converted into double precision format and the exponents of two numbers are compared if it is equal then we can directly add the numbers and normalize the answer but if the exponent is not equal then we need to rewrite smaller number such that its exponent matches with the larger number and then addition has been performed. After the result of addition it must be normalized.



III. PROPOSED ALGORITHM

Proposed algorithm is similar to as used in [1].The floating point arithmetic operation uses two clock cycles here. In [1] it is also a two staged pipeline which is divided into two paths. The two paths (“R path” and “N path”) are selected on the basis of the exponent difference. The two pipeline stages execute in two different clock cycles. In very first step for this paper, algorithm is written for determining the exponent



difference , then big number is determined out of two exponents by implementing conditions when both the exponents have same sign and both the exponents have different sign. Then greater mantissa is selected for shifting operation and shifting is done for equalization of exponents. All of this operation is done in 1st clock cycle. In next clock cycle two mantissa are added and then normalization has been done.

IV. IMPLEMENTATION DETAILS

Somsubhra Ghosh implemented the proposed algorithm with XC2V6000 and XC3S1500 xilinx FPGA devices[1].Similar algorithm when implemented with 6slx45tfgg484-3 improvement in results has been noticed and tabulated as follows.

Table1: Estimation of usage of resources in device 6slx45tfgg484-3

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Slice Logic Utilization			
Number of Slice Registers	63	54576	1%
Number of Slice LUTs	645	27288	2%
Number used as Logic	645	27288	2%
Slice Logic Distribution:			
Number of LUT Flip Flop pairs used	645		
Number with an unused Flip Flop	582	645	90%
Number with an unused LUT	0	645	0%
Number of fully used LUT-FF pairs	63	645	7%
IO Utilization			
Number of IOs	193		
Number of bonded IOBs	192	296	64%

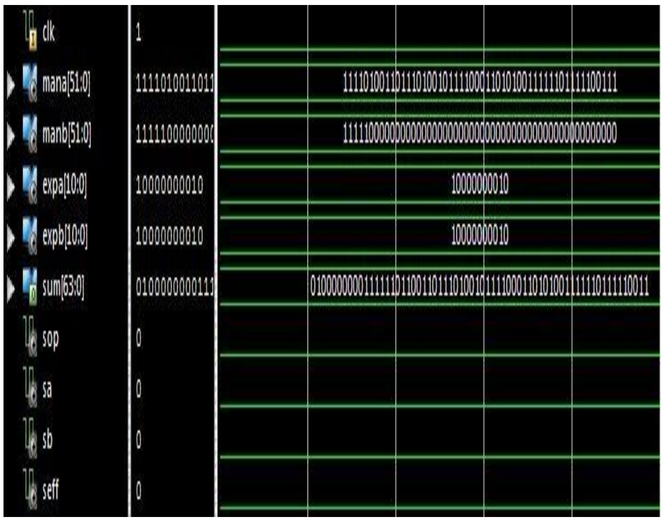
Minimum input arrival time before clock is: 27.424ns
Maximum output required time after clock: 5.463ns
Maximum combinational path delay: 39.551ns
Latency: 2 clock cycles

Table 2: Estimation of usage of resources in device 5vfx70tff136-1

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Slice Logic Utilization			
Number of Slice Registers	63	44800	0%
Number of Slice LUTs	602	44800	1%
Number used as Logic	602	44800	1%
Slice Logic Distribution:			
Number of LUT Flip Flop pairs used	602		
Number with an unused Flip Flop	539	602	89%
Number with an unused LUT	0	602	0%
Number of fully used LUT-FF pairs	63	602	10%
IO Utilization			
Number of IOs	193		
Number of bonded IOBs	192	640	30%

Table 2 give results of same algorithm on virtex 5.Number of resources are increased from Spartan 6 to virtex 5 and same results are arrived this simply proves that when same algorithm implemented on virtex 2 we get improved results.

V. SIMULATION RESULTS


clk 1
mana[51:0] 111101001101 1111010011011101001011110001101010011111101111100111
manb[51:0] 111110000000 1111100000000000000000000000000000000000000000000000
expa[10:0] 10000000010 10000000010
expb[10:0] 10000000010 10000000010
sum[63:0] 010000000111 0100000000111111011001101110100101111000110101001111110111110011
sop 0
sa 0
sb 0
seff 0

VI. CONCLUSION

The double precision floating point adder is successfully implemented .If the algorithm is implemented on the advanced versions then better results will be derived in future.

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] Purna Ramesh Addanki,Venkata Nagaratna Tilak Alapati And Mallikarjun 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 Pp 61-74 Vol 52, March 2013.

[3] Patrick J.Eibl, Andrew D. Cook ,Daniel J.Sorin “reduced Precision checking for a floating point adder”,24th International symposium on defect and fault tolerance In VLSI systems.Chicago,Illinois,oct 2000.

[4] 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.

[5] 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. Computer Applications, vol. 46, no. 9, pp. 1-5, May 2012.

[6] Chi.Huang,Xingu Wu, " Design of High Speed Double Precision Floating point adder using macromodules” Design automation conference 2005, Proceedings of the ASP-DAC 2005.pp.11-12 18-21,Jan 2005.

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

[8] Meenu Talwar,Karan Gumber, Sharmele 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 201

[9] 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.

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

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

[12] 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.

[13] We. Park, T.D. Han, S.D. Kim, S.B. Yang, "Floating Point Adder/Subtractor Performing IEEE Rounding and addition/ Substraction in parallel”IEICE Trans. On information and Systems, vol.4, pp.297-305,1996.

[14] S.Oberman, H. A1-Twaijry, and M. Flynn, Project “Design of Floating point arithmetic units”,Proceeding of 13th IEEE Symposium on computer arithmetic,pp.156-165,1997.

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

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

[17] 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.