Design & Analysis of Floating Point Arithmetic Adder

Neelam Sharma[1] Anil Khandelwal[2]
[1]M.Tech Scholar
[1,2]VNS Group of Institution Bhopal, India

Abstract— Floating-point addition is the most frequent floating-point operation and accounts for almost half of the scientific operation. Therefore, 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 research has investigated optimization of floating-point units for area, delay and power consumption in hardware. While the others will be described shortly the result carried out on Xilinx in term of area and delay. This paper presents the highly efficient floating adder for effective way to present fast addition.

Key words: VHDL, Comparator, Add/Sub, Floating Point Adder

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 [1-4-11] over the span of last two decades. Field Programmable Gate Array (FPGA) is a silicon chip with unconnected logic blocks, these logic blocks can be defined and redefined by user at any time. 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. In FPGAs, the bottleneck for designing efficient floating-point units has mostly been area. With advancement in FPGA architecture, however, there is a significant increase in FPGA densities. Devices with millions of gates and frequencies reaching up to 300 MHz are becoming more suitable for floating-point arithmetic reliant applications.[10][7]

II. RELATED WORK

Rohita Watpade et.al [1].BSD Adder for Floating Point Arithmetic: A Review Now a days, modification in floating point architectures is very demanding area for researchers so there is always a challenge to design an efficient Floating Point Architecture. Out of any other operation, Floating Point Addition has been very popularly used. This paper presents the highly efficient BSD adder for effective way to present fast addition. Using this BSD adder in the computation algorithm of FFT Butterfly Architecture, can enhance the efficiency and at the same time can reduce the complexity, area, power consumption and delay. Starting from the design of 2 bit BSD adder we went up to design of 58 bit BSD adder as presented in this paper. BSD adder is implemented in Verilog HDL and targeted to three different families FPGA: Spartan6, Virtex5and Virtex6 in Xilinx 13.2 ISE software. The main aim is to speed up the circuit by reducing.

Amir Kaivani, et.al, [2] in their paper, "Floating Point Butterfly Architecture Based On Binary Signed Digit Representation" have proposed a fast FP (floating point) butterfly architecture, which is relatively quick than the previous work. The reason for the speed improvement is due to the FDPA unit suggested in this paper. Thus, by eliminating extra LZD, normalization and rounding units this high speed is achieved [1].delay have been achieved

Jr., Hani et.al [3] in their paper, "FFT Implementation with Fused Floating Point Operations" have used the contrived idea of two new fused floating-point arithmetic units. Both the fused dot product unit and the fused add-subtract unit are smaller than collateral executions constructed with discrete floating point adders and multipliers [2]

III. PROPOSED ALGORITHM

The addition of floating point operation is difficult to perform because the signs are expressed in the sign magnitude format. Depending on sign of two operand addition or subtraction operation is performed. In case of adding a carry-out may be generated in such a case the result will be de-normalized. In subtraction, a negative result may be obtained which signifies that both the sign bit and the signs need to be inverted Floating point addition/subtraction algorithm:

IV. ALGORITHM

The two operands, N1 and N2 are read in and compared for de normalization and infinity.

1) If numbers are de normalized, 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 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.

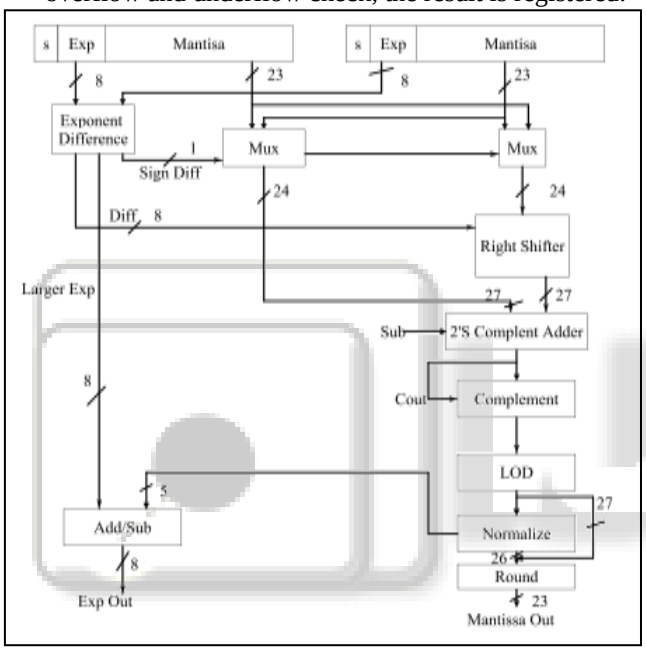

Fig. 1: Proposed Architecture

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 significant 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 significant. 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. Executed to compute the lowest path delay for varied adders. This part also involves the comparative result d designed adders and subtractors. The Table I (shows the area usage and delay and power of adders.

V. EXPERIMENTAL RESULTS

THE SIMULATION OF THE PROPOSE ADDER HAS BEEN SHOWN

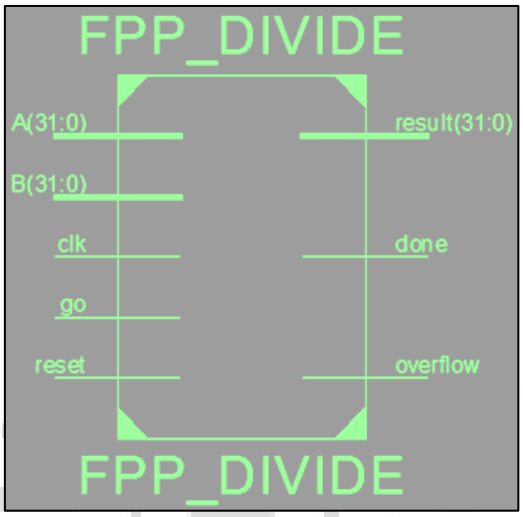

Fig. 2: Block of Floating Point Adder

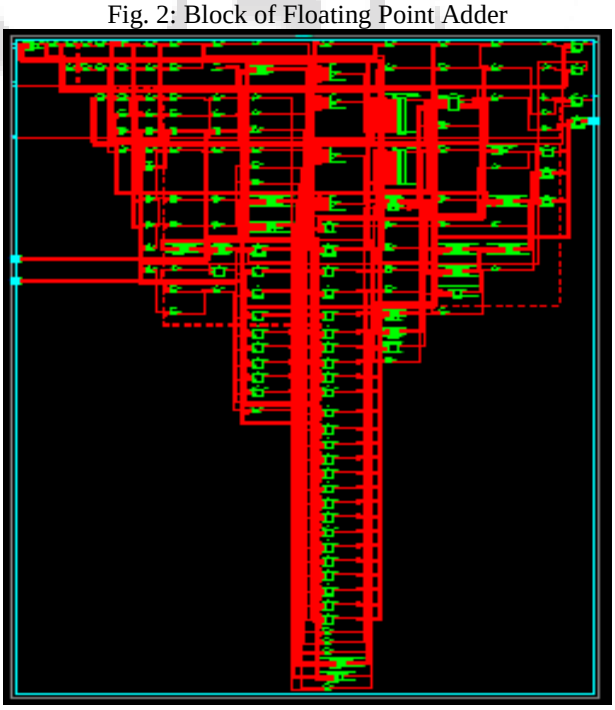

Fig. 3: RTL View Floating Point Adder

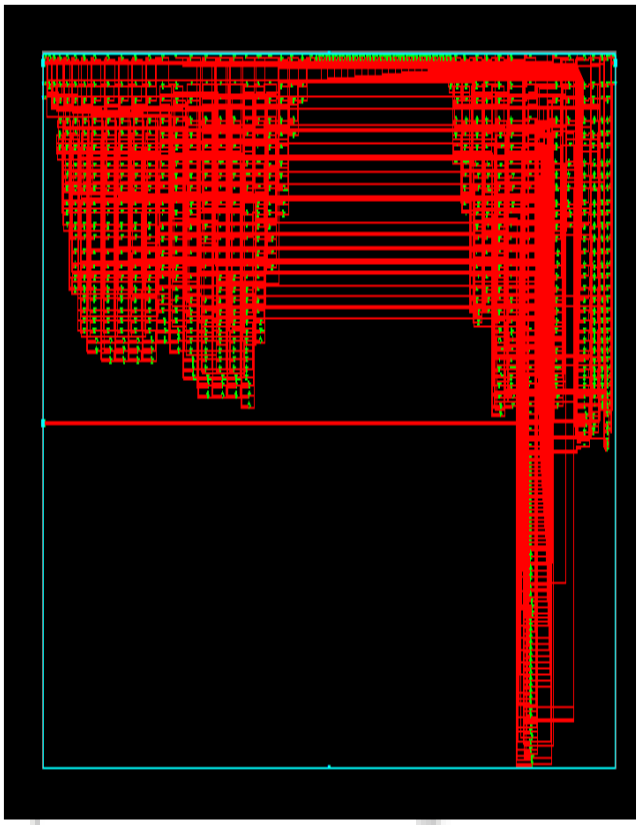

Fig. 4: LUT Views Floating Point Adder

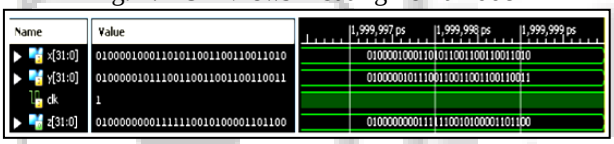

Fig. 5: Floating Point Adder

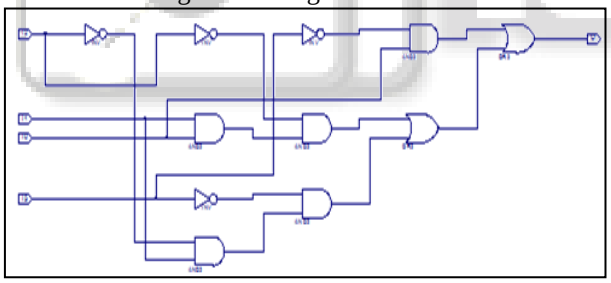

Fig. 6: Logic Presentation of Floating Point Adder

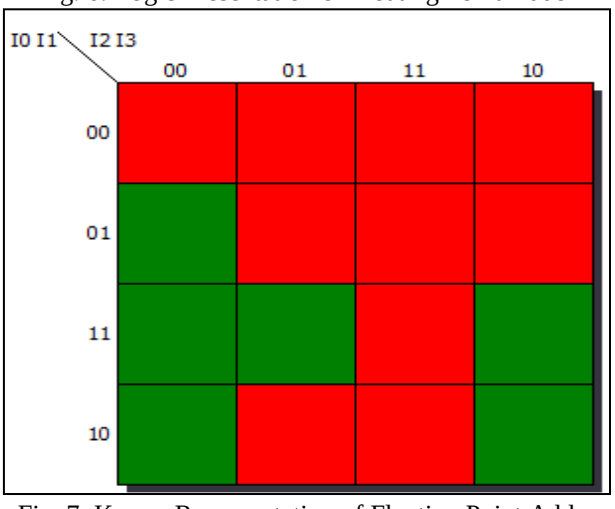

Fig. 7: K-map Representation of Floating Point Adder

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	271	126,800	1%
Number used as Flip Flops	271		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	254	63,400	1%
Number used as logic	246	63,400	1%
Number using O6 output only	131		
Number using O5 output only	1		
Number using O5 and O6	114		

Table 1: Summary of Floating Point Adder

Design	Proposed work	Rohita Watpade et.al.[1]
Delay	: 1.165ns	3.813
power	0.42	-
Number of slice LUT's	32 used out of 9112	212 out of 126800
Number of bonded IOB's	110 Out of 210	11 out 232
Number of fully used LUT's-FF pairs	137 out of 342	0 out of 1

Table 2: Comparison Table

REFERENCES

[1] Rohita Watpade et.al.BSD Adder for Floating Point Arithmetic: A Review International Conference on Communication and Signal Processing, April 6-8, 2017, India, 978-1-5090-3800-8/17/$31.00 ©2017 IEEE

[2] Amir Kaivani and Seokbum Ko "Floating-Point Architecture Based on Binary Signed-Digit Representation" IEEE Transaction on VLSI Systems, volume 24., no.3, pp.1208-1211, March 2016

[3] E. E. Swartzlander, Jr. and H. H. Saleh "FFT Implementation with Fused Floating-Point Operations" IEEE Transaction On Computers,volume 61., no. 2, pp. 284-288, Feb. 2012

[4] J. Sohn and E. E. Swartzlander, Jr. "Improved Architectures for a Floating-Point Fused Dot Product Unit" in Proc. IEEE 21st,Symposium On Computer Arithmetic, pp.41-48 , Apr. 2013

[5] Y.Tao, G. Deyuan, F. Xiaoya, and R. Xianglong "Three-operand floating-point adder" in Proc. 12th IEEE International Conference of Computer Inf. Technology, October 2012,pp. 192-196.

[6] J. H. Min, S.-W. Kim and E. E. Swartzlander, Jr. "A Floating-Point Fused FFT Butterfly Arithmetic Unit with Merged Multiple-ConstantMultipliers" in Proc. 45th Asilomar Conference on Signals, Systems and Computers, pp. 520–524, Nov. 2011

[7] Hani Saleh and Earl E. Swartzlander, Jr., "A Floating-Point FusedAdd- Subtract Unit" Proc. IEEE Midwest Symp. Circuits and Systems (MWSCAS), pp. 519-522, 2008

[8] F. Tenca, "Multi-operand floating-point addition" in Proc. 19thIEEE Symp. Comput. Arithmetic, Jun. 2009, pp. 161-168

[9] H. H. Saleh, "Fused Floating-Point Arithmetic for DSP", PhD dissertation, Univ. of Texas, 2008

[10] G. Jaberipur, B. Parhami, "Efficient realisation of arithmetic algorithms with weighted collection of posibits and negabits"IET Comput. Digit. Tech. 2012, Vol. 6, ISS. 5, pp. 259-268, Jan. 2012
[11] Ishan A. Patil, Prasanna Palsodkar, Ajay Gurjar, "Floating Point-based Universal Fused Add-Subtract Unit" Proc. Of Second International Conference on Soft Computing for Problem Solving, pp. 259-270, December 28-30 2012.

