# FPGA Implementation Of Pipelined Architecture Of Floating Point Airthmetic Unit

**Dharna Awasthi, Ajay Kumar Yadav**

***Abstract*— point numbers are widely adopted in many applications due their dynamic representation capabilities. Floating –point representation is able to retain its resolution and accuracy compared to fixed–point representations. Unfortunately, floating-point operators require excessive area for conventional implementations. High speed floating point arithmetic unit using least area is an important considerations for system designers. Here, we are aiming to fulfill major requirements of DSP Processors such as high speed and least area utilization. In the given architecture, floating point numbers are represented in single precision IEEE 754 floating point format. The basic building blocks of arithmetic unit are addition, subtraction, multiplication and division units. The algorithms for ADDSUB unit (for addition and subtraction operation), multiplier unit and division unit have been coded in VHDL language and simulated using Model Sim10.1c PE, and synthesized using Precision Synthesis RTL 2011 individually. An additional algorithm has been written in VHDL language for the floating arithmetic unit to integrate all the units like add sub, multiplier and division. The complete architecture of arithmetic unit using pipelined technique has been simulated using Model Sim10.1c PE and Synthesized using Precision Synthesis RTL 2011. We used XILINX Spartan3 xcs400pq208 for placement, routing, to generate plan overhead and implementation. The complete architecture of arithmetic unit uses 100 no. of IO's and 2368 no. of LUT's. It is successfully running with clock frequency of 6.93 MHz frequency. The simulation result has been verified by downloaded the complete VHDL code of arithmetic unit into the field programmable gate array trainer kit of XILINX Spartan3 xcs400pq208.**

***Index Terms*—FPGA, ADDSUB, VHDL, LUT's.**

## I. INTRODUCTION

In the complex and high speed designs requires the high speed floating point arithmetic units. To meet such high speed requirement it is also necessary that the Arithmetic Unit has to be area efficient. Several design approaches has been already developed and implemented like Single cycle Architecture, Parallel/ Pipelined Architecture, Fused Architecture and. A very popular method to achieve high speed is the pipelining. This architecture provides the desired speed to work fast with consuming less area. The main objective was to implement the floating point arithmetic algorithms using pipelining architecture with reduced Area. [1]

So the initial problem was to choose the appropriate floating point arithmetic algorithm suitable for pipelining architecture. After the selection of Algorithm I decided to implement this using VHDL because it is very much suitable Hardware Description Language for system level design. The implementation of floating point algorithm has been done using top~ down approach. Each sub modules has been designed, simulated and verified to make the Addition/Subtraction, Multiplication and Division module. These modules then combined to make the complete floating point unit. [2]

The complete unit is simulated and verified using Model Sim 10.1c PE. After the successful completion of simulation phase the main focus was to synthesized the code to achieve optimum Results such as Area, Timing and Power. [3]The Synthesis is done by Precision Synthesis. The timing analysis is then carried out also the Critical Path Analysis is performed. After the optimization the code is ready to implement on the target device which is XILINX Spartan 3 xcs400pq208 FPGA.

**Dharna Awasthi** M.Tech, EC Department, Mewar University Chittorgarh, Rajasthan, INDIA.

**Ajay Kumar Yadav** Asst. Professor in C.E.R.T. Meerut India

## II. FLOATING POINT ARITHMETIC

### Floating Point Addition and Subtraction

The floating point addition and subtraction is quite complex compared to multiplication and division. In the addition process the exponent of the two numbers must be same if not so we need to align it by shifting it to the right or left and adjusting the mantissa accordingly. Then we add or subtract the two numbers.[4]

Approach 1

1-Find exponent difference d = e1-e2. If e1 < e2, swap position of mantissas. Set larger exponent as tentative exponent of result.
2-Pre-align mantissas by shifting smaller mantissa right by d bits.
3-Add or subtract mantissas to get tentative result for mantissa.
4-Normalization. If there are leading-zeros in the tentative result, shift result left and decrement exponent by the number of leading zeros. If tentative result overflows, shift right and increment exponent by 1-bit.
5-Round mantissa result. If it overflows due to rounding, shift right and increment exponent by 1-bit.

Approach 2

The floating point addition and subtraction algorithm is Consider the two floating point numbers x1 and x2.

$$y_1 = (-1)^{s_1}.F_1.2^{E_1} \quad and \quad y_2 = (-1)^{s_2}.F_2.2^{E_2}$$

Where $s_1, s_2$ are the sign, $F_1, F_2$ are the significant (mantissa) and $E_1, E_2$ are the exponents of the two numbers respectively. The addition/subtraction of two numbers will be

$$y_1 \pm y_2 = [(-1)^{s_1}.F_1.2^{E_1}] \pm [(-1)^{s_2}.F_2.2^{E_2}]$$

$$= [(-1)^{s_1} . F_1 \pm F_2 . 2^{E_1 - E_2}]$$

1. First we unpack the number and segregate the sign, exponent and significant bits. Add the implied bit in significant. Also we check for the special conditions and special numbers.
2. To perform the addition exponent must be same. For this we subtract the two exponents and shift the significant by the subtraction result. In subtraction one of the numbers is inverted.
3. After aligning the significant we add the two significant if both signs are equal otherwise subtract the two numbers.
4. The result of addition might fall into the either overflow or underflow category so we need to have normalization. If there are leading-zeros in the tentative result, shift result left and decrement exponent by the number of leading zeros. If tentative result overflows, shift right and increment exponent by 1-bit.
5. Round significant result. If it overflows due to rounding, shift right and increment exponent by 1-bit.

4- **Floating Point Multiplication**

1. First unpack the two numbers, separate the sign, exponent and mantissa bits. Also check for the special condition specifies by the IEEE 754 standard such as weather the no. is NaN or infinity or zero.
2. Then multiply the two mantissas and add the exponent. Also the bias value needs to be subtracted with the exponent addition result.
3. Calculate the sign of the result by XORing of the two sign bits.

4- Normalize and round the result, if necessary

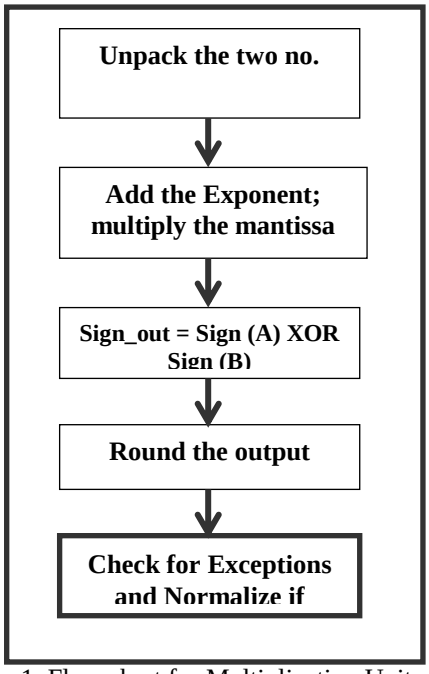


Figure 1. Flow chart for Multiplication Unit

Approach 2

The second approach is quite similar to the earlier one but the only difference is that this algorithm performs the floating point multiplication process in two stages, the figure shows the floating point multiplication process[a].[5]

Consider the two floating point numbers $y_1 = (-1)^{s_1} . F_1 . 2^{E_1}$ and $y_2 = (-1)^{s_2} . F_2 . 2^{E_2}$

Where $s_1, s_2$ are the sign, $F_1, F_2$ are the significant (mantissa) and $E_1, E_2$ are the exponents of the two numbers respectively. The multiplication result of the two can be written as

$$y_1 \times y_2 = [(-1)^{s_1} . F_1 . 2^{E_1}] \times [(-1)^{s_2} . F_2 . 2^{E_2}]$$
$$= [(-1)^{s_1 \oplus s_2} . (F_1 \times F_2) . 2^{E_1 + E_2}]$$

## III. IMPLEMENTATION OF FLAOTING POINT ARITHMETIC UNIT

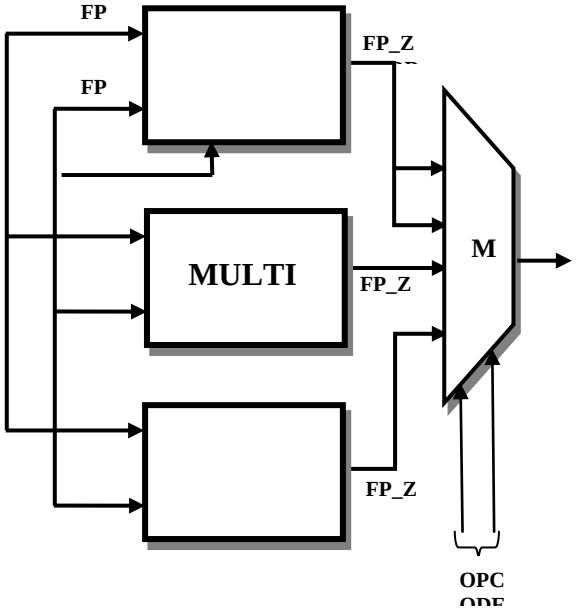


Figure 2. Block diagram of Floating Point AU

*1) 6-The Addition/Subtraction Unit*

The Addition and Subtraction unit is implemented with the pipelined architecture. The complete design consists of no. of sub-modules. These sub modules then arranged in six stages. The detailed description and block diagram is given below.

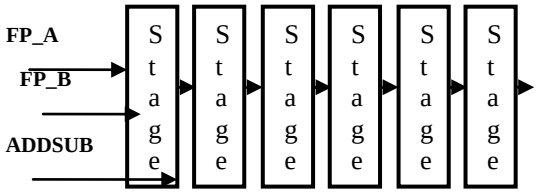


Figure 3. Block diagram of pipelined Add/Sub unit

a) Stage 1

In this stage we take the 32-bit floating point no. as an input and separate the sign bit, 8-bit of Exponent and the remaining Significant. The block we use for this operation is the **Unpack block.** This block used for unpack the floating point no. used as input. This blocks separates the mantissa, exponent and sign of the input number along with it also checks the special condition such as weather the no. is infinity[6]

**Stage 2**

After the separation of input no. we need to align the mantissa as mentioned in the algorithm and swap the no. if necessary. So this stage contains this part of algorithm which deals with aligning and swapping of the two numbers. The Align and Swap block are used for the respective operations

**Stage 3**

The different types of circuit techniques used follow a In the third stage we perform the addition operation. The subtraction operation is done by 2's complement method. So for this we need to invert one of the numbers. This is done by the invert unit.

b) Stage 4

u After the addition operation we need to normalize the result. Also in the subtraction process the result needs to be complemented again. This is done by the Selcomplement Unit. The result of addition is normalized by Add_normalize Unit.

The normalization unit uses the idea of checking the leading zeros in the result.For this a leading zero detector unit is also used. The Selcomplement unit performs the inversion of the result which may be produced by subtraction using 2's compliment addition method.[7]

## IV. Simulation Results

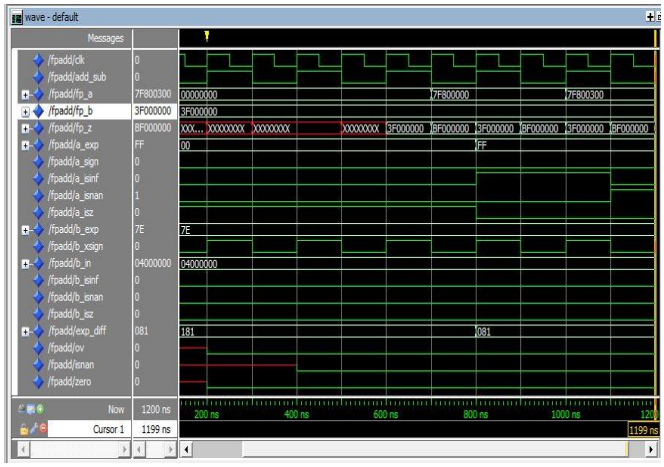

Figure 4 waveform for special condition check for add/sub unit

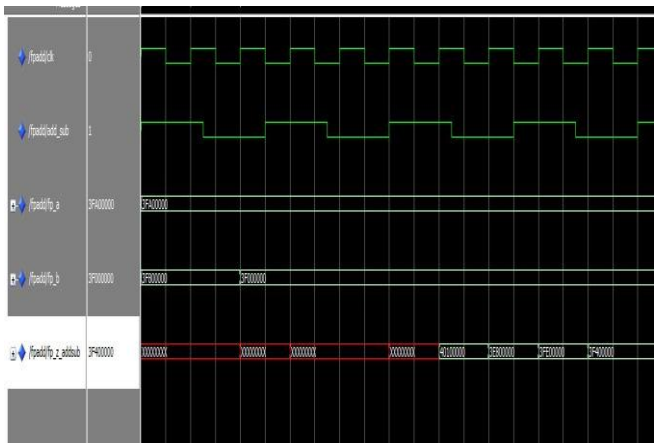

Figure 5 waveform for add/sub unit under normal operation

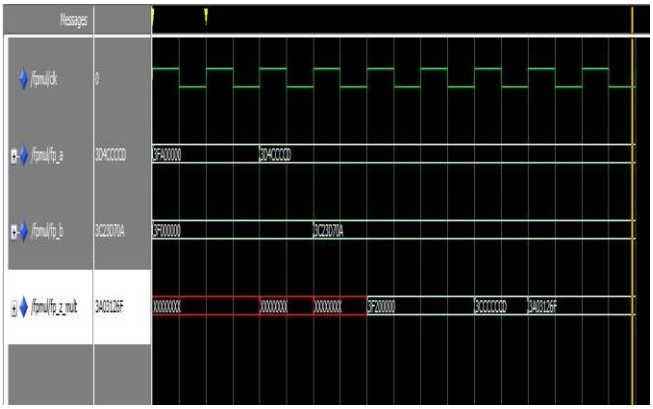

Figure 6 waveform for MULTIPLICATION unit

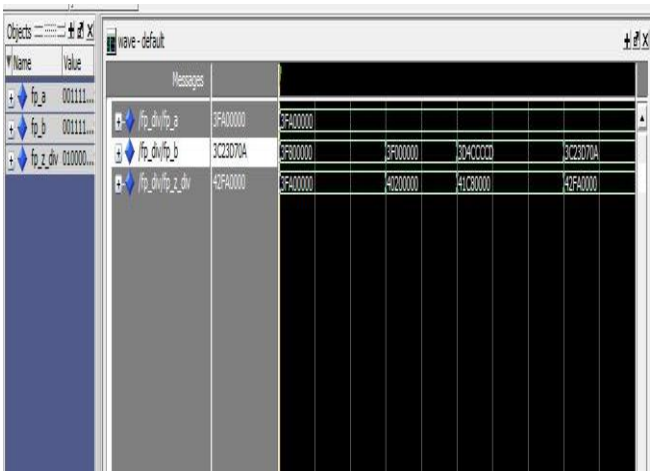

Figure 7 waveform for DIVISION unit

## V. Conclusion

I have presented some floating point arithmetic algorithms (addition, subtraction, multiplication and division) that are suited for implementing the Floating Point Arithmetic Logic Unit with the Pipelined Architecture. The algorithms have been coded in VHDL. The simulation of VHDL implementation of floating point arithmetic algorithms have been carried out using ModelSim 10.1c PE. The simulated code has been synthesized with Precision Synthesis using XILINX Spartan 3 xcx400pq208 FPGA as targeted device. Area Utilization has been optimized and Timing Report along with Power Report has also been analyzed. Timing Violations and Critical Path analysis has been carried out too. Placement, Floor Planning and Routing is then performed. Finally the code downloaded in the Field Programmable Gate Array and verified manually with the simulation result generated by the tools. With the help of comparison table it has been verified that this design has less area compared to its older counterpart

## REFERENCES

[1]Sahin Suhap, Kavak Adnan, Becerikli Yasar and Demiray H. Engin. Implementation of Floating Point Arithmetic using an FPGA.

[2]Khare Kavita, Singh R.P. and Khare Nilay. Comparison of pipelined IEEE-754 standard floating point adder with unpipelined adder. Journal of Scientific and Industrial Research.

[3]Al-Asharfy Mohamed, Salen Ashraf and Anis Wagdy. An Efficient Implementation of Floating Point Multiplier. 978-1-4577-0069/11/$26.00 ©2011 IEEE.

[4]Sharma Subhash Kuamr, Pandey Himanshu, Sahni Shailendra and Srivastava Vishal Kumar. Implementation of IEEE 754 Addition and

Subtraction for floating point unit. International Transactions in Mathematical Science and Computer.
[5] Saini Deepa and M'dia Bijendar. Floating point Unit Implementation on FPGA. International Journal of computational Engineering Research.
[6] Reaz Mamun Bin Ibne, Islam Md. Shabiul and Sulaiman Mohd. S. Pipelined floating point ALU design using VHDL. ICSE 2002, Periag, Malaysia.
[7] Tiwari Asish, Singh Rajit Ram, Singh Vinay Kumar and Tomar Geetan S. VHDL Envoirnment for floating point Arithmetic Logic Unit ALU design and Simulation. International Conference on Communication system and Network Technology.
[8] Brant Richard P. and Zimmermann. Modern Computer Arithmetic
[9] Parhami Behrooz. Computer Arithmetic Algorithms and Hardware Designs, Published by Oxford University press, 2000.
[10] Riya saini, R.D.Daruwala/ international journal of Engineering Research and Applications

**Dharna Awasthi** M.Tech, EC Department, Mewar University Chittorgarh, Rajasthan, INDIA
**Ajay Kumar Yadav** Asst. Professor in C.E.R.T. Meerut India