International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878
Volume 2 Issue 5, May 2014

FPGA Implementation of Pipelined Architecture of Floating Point Arithmetic Core and Analysis of Area and Timing Performances

Hemraj Sharma[1], Abhilasha[2]

[1]JECRC University, M.Tech VLSI Design, Rajasthan, India
[2]JECRC University, Rajasthan, India

Abstract: ***The aim of this paper is FPGA implementation of architecture of floating point arithmetic core and analysis of area and timing performances of that arithmetic core. The basic concept behind designing such a core is to optimally utilize the algorithms of floating point arithmetic operations, i.e., addition, subtraction, multiplication and division and to enhance the operational speed of these calculations in order to determine the better code amongst them in order to use it in future to increase the processor efficiency. The simulation has been carried out on Modelsim (Student edition) EDA tool 10.0c and synthesis has been carried out on ISE Design Suit EDA tool 14.4 .***

Keywords: Binary Division, Carry Look Ahead Adder, Exponent Subtraction, Floating Point, FPGA, Single Precision technique, Urdhva – tiryakbhyam

1. Introduction

Real world is full of different types of mathematical calculations. Today people have shortage of time and they want calculations to be performed at a very fast speed. Some of the common applications of mathematical calculations are in determining the exponential values, logarithmic calculations, etc. where it is essential to eliminate the time consumed or in other words, we can call it as delay in performing high speed calculations. Therefore, some kind of electronic calculation technique is highly essential to be used to perform this calculation at a very fast speed. Mathematical calculations including addition, subtraction, multiplication and division are very important fundamental functions in arithmetic calculative operations. Computational performance of a DSP system is limited by the performance of these mathematical operations. So, in order to improve its performance, a floating point arithmetic core is proposed.

In case of floating point arithmetic calculations, a significant improvement can be observed in execution speed using its algorithms because of inherent integer math hardware support in a large number of processors but this speed improvement does come at the cost of reduced range and accuracy of the algorithm variables. So to increase the range of variables and accuracy of operations, floating point arithmetic core is being used for addition, subtraction, multiplication and division.

2. Proposed Design

In floating point arithmetic core based design we use single precision (or 4 byte) technique. This technique is commonly known as “float” in the C language family and “real” or “real*4” in FORTRAN. This binary format occupies 32 bits, i.e., 4 bytes and its significant and has a precision of 24 bits (about 7 decimal digits).[1] Format is as shown below-

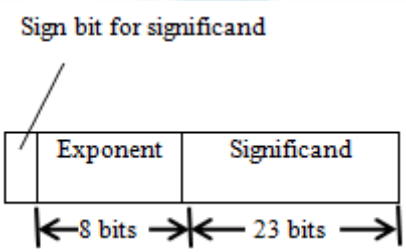

Figure 1: Single Precision Format

Using this format, we create codes of mathematical computations, namely, addition/subtraction, multiplication and division. After implementing the codes, we execute them on FPGA and then analyze their area and timing performances. The proposed floating point design flow is as drawn in figure below:

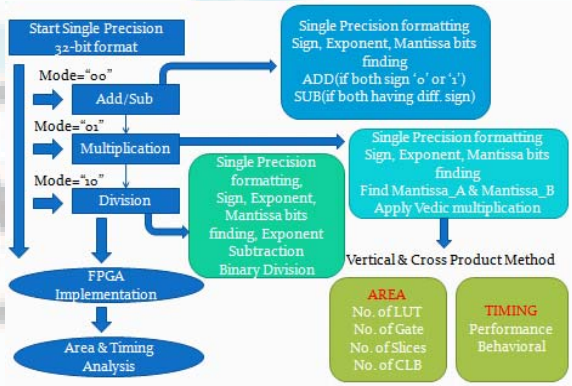

Figure 2: Floating Point Design Flow

In the floating point design architecture, codes of 32-bit addition/subtraction, multiplication and division codes have been designed and implemented. Complete coding is implemented using single precision technique.[2] For this purpose, different modes have been considered as mode “00” for add/subtract, mode “01” for multiplication and mode “10” for division. Add/subtract code is being

designed using conventional add/subtract methods using Carry Look Ahead Adder. Multiplication code is designed using vedic multiplication technique named Urdhva - tiryakbhyam, i.e., "Vertically and Crosswise" technique.[3,4] Division code is implemented using binary division and exponent subtraction techniques.

For addition/subtraction coding, we use Carry Look Ahead Adder instead of ripple carry adder and any other kind of adder because this adder is a practical design with reduced delay. In case of multiplication, we use Vedic multiplier instead of any other conventional or array multipliers.[5,6] In Vedic multiplier, there are Nikhilam sutra which literally means "All from 9 and last from 10" and Urdhva – tiryakbhyam sutra which literally means "Vertically and Crosswise" out of which we proceed with Urdhva - tiryakbhyam sutra [7,8].

3. Timing and Area Analysis

The timing and area analysis of codes are as under:

3.1 Add/Subtract

The timing and area analysis, respectively, of add/subtract code are as shown under.

Table 1: Add/Subtract Code Timing Parameters

Parameters	*Floating*
Min. i/p arrival time before clock (ns)	7.382
Max. o/p required time after clock (ns)	4.368
Max. combinational path delay (ns)	28.726

Table 2: Add/Subtract Code Area Parameters

Parameters	*Floating*
Total Number of 4-input LUTs	447 out of 7168 (6%)
Number of occupied Slices	234 out of 3584 (6%)
Total Gate Count	3179

3.2 Multiplication

The timing and area analysis, respectively, of multiplication code are as shown under.

Table 3: Multiplication Code Timing Parameters

Parameters	*Floating*
Min. i/p arrival time before clock (ns)	Not Found
Max. o/p required time after clock (ns)	Not Found
Max. combinational path delay (ns)	107.379

Table 4: Multiplication Code Area Parameters

Parameters	*Floating*
Total Number of 4-input LUTs	3412 out of 7176 (47%)
Number of occupied Slices	1751 out of 3584 (48%)
Total Gate Count	23,597

3.3 Division

The timing and area analysis, respectively, of division code are as shown under.

Table 5: Division Code Timing Parameters

Parameters	*Floating*
Min. Period (ns)	Not Found
Min. i/p arrival time before clock (ns)	Not Found
Max. o/p required time after clock (ns)	Not Found
Max. combinational path delay (ns)	177.221

Table 6: Division Code Area Parameters

Parameters	*Floating*
Total Number of 4-input LUTs	1352 out of 7168 (18%)
Number of occupied Slices	700 out of 3584 (19%)
Total Gate Count	11,991

4. Results

The simulation waveforms of add/subtract, multiplication and division codes of floating point arithmetic core are shown in Fig.3, Fig.5 and Fig.7, respectively, along with their respective RTL top level schematics in Fig.4, Fig.6 and Fig.8 below-

4.1 Add/subtract

Inputs-> mantissa_a => 000007
exponent_a => 05
opa => 02800007
mantissa_b => 000038
exponent_b => 05
opb => 02800038
Output-> add_out => 0280003F

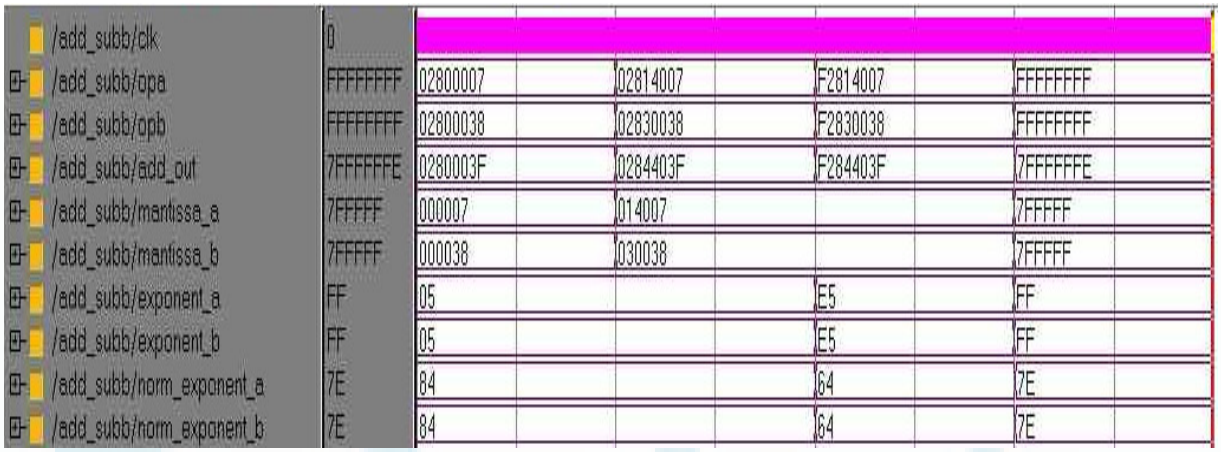

Figure 3: Simulation Waveform of Add/Subtract Code

RTL Top Level Schematic->

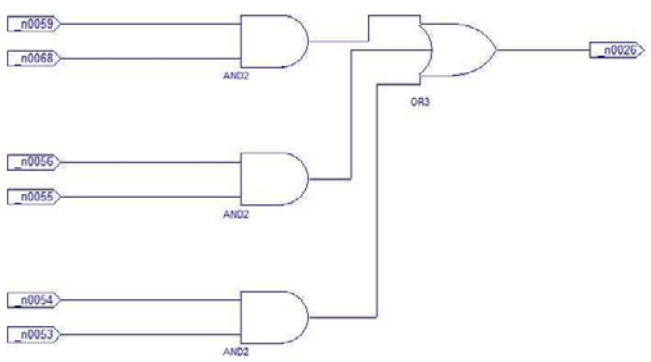

Figure 4: RTL Top Level Schematic of Add/Subtract Code

4.2 Multiplication

Inputs-> mantissa_a => 00001F
exponent_a => 06
opa => 0300001F
mantissa_b => 000007
exponent_b => 1E
opb => 0F000007
Output-> multiply_out => 120000D9

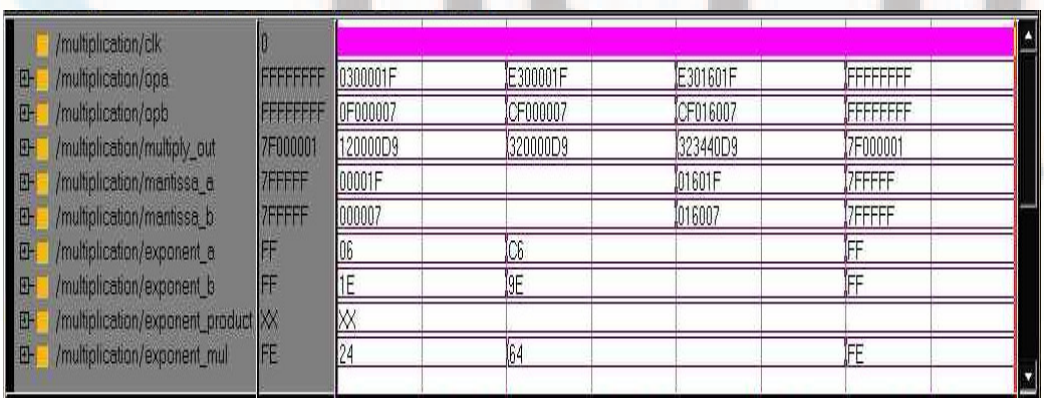

Figure 5: Simulation Waveform of Multiplication Code

RTL Top Level Schematic->

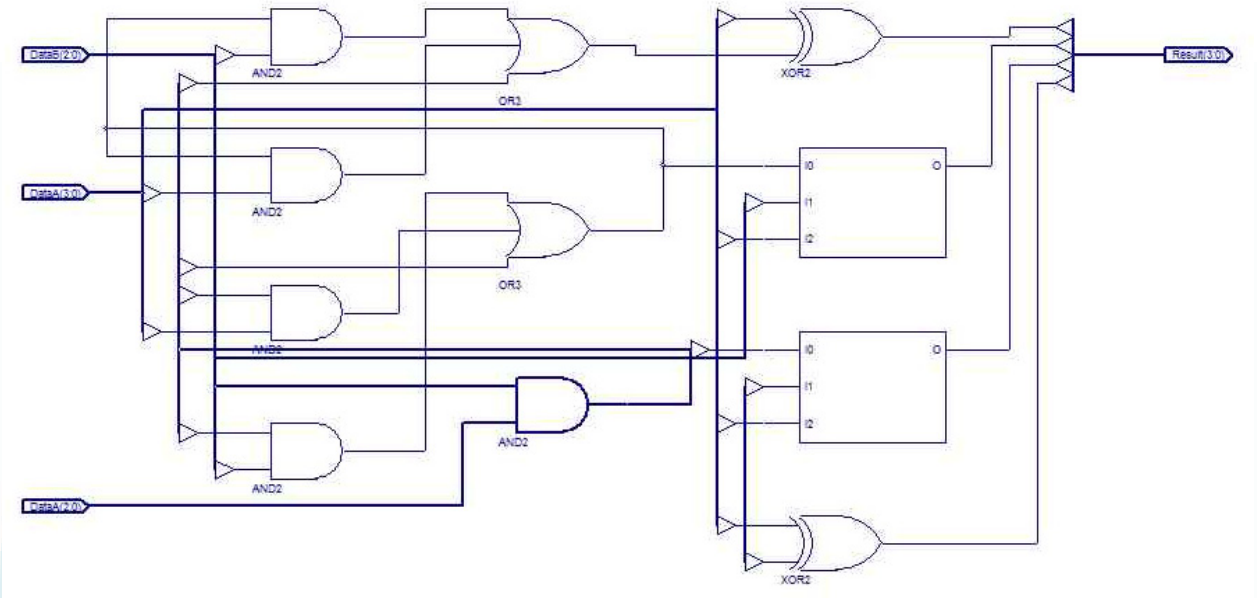

Figure 6: RTL Top Level Schematic of Multiplication Code

4.3 Division

Inputs-> x => 0000000D
y => 00000003
Output-> z => 7F800000

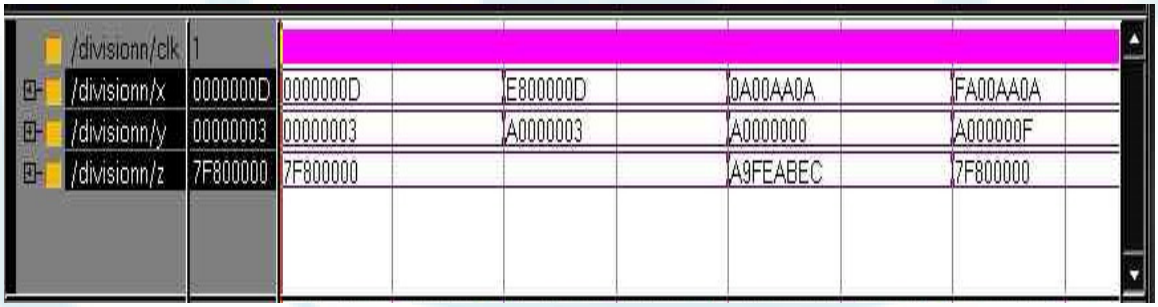

Figure 7: Simulation Waveform of Division Code

RTL Top Level Schematic->

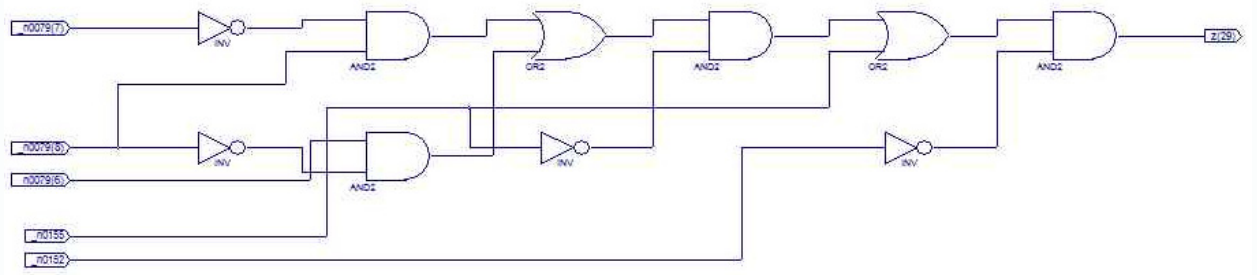

Figure 8: RTL Top Level Schematic of Division Code

5. Conclusion

From the timing and area analysis tables, we find that the timing and area performances of floating point add/subtract code are **better** than the timing and area performances of floating point multiplication and division codes. But amongst multiplication and division codes, multiplication code is having an upper hand over division

code in case of less delay, while division code is **better** than multiplication code in case of less area consumption.

Hence, we can conclude that for less delay, less area consumption and high speed computation, floating point add/subtract code using CLA is the **best** amongst the three while floating point division code using binary division and exponent subtraction techniques is **better** in area than multiplication code while in case of timing performance, multiplication code is **better** to use than division code.

Acknowledgement

I would like to acknowledge my mentor Gaurav Jindal Sir and Kanchan Sengar Ma'am who supported me during the period in calculating my results and verifying codes.

References

[1] Kahan W., "On the Cost of Floating-Point Computation Without Extra-Precise Arithmetic", 2012
[2] Serene Jose, Sonali Agarwal, "Single Precision Floating Point Divider Design", International Journal Of Computational Engineering Research, 2(3), 955-958, 2012
[3] Ganesh Kumar G. and Charishma V., Design of high Speed Vedic Multiplier using Vedic Mathematic Techniques, International Journal of Scientific and Research Publication, 2(3), 2012
[4] Nicholas A.P., Williams K.R. and Pickles J., Application of Urdhava Sutra, Spiritual Study Group, Roorkee, India, 1984.
[5] Hemraj Sharma, Gaurav K. Jindal and Abhilasha Choudhary, "Comparison Between Array Multiplier And Vedic Multiplier", International Journal of Computer Science information And Engg.,Technologies, 4(1), 2014
[6] Basavaraj B., Comparison of Vedic Multipliers With Conventional Hierarchical Array of Multipliers, International Journal of Engineering Research & Technology, 2(10), 2013
[7] Sree Nivas A, Kayalvizhi N, Implementation of Power Efficient Vedic Multiplier, International Journal of Computer Applications, 43(16), 2012
[8] Verma Pushpalata, Mehta K. K., Implementation of an Efficient Multiplier based on Vedic Mathematics Using EDA Tool, International Journal of Engineering and Advanced Technology (IJEAT), 1(5), 2012

Author Profile

Hemraj Sharma received the B. Tech. degree in Electronics and Communication Engineering from Rajasthan Technical University, Kota in the year 2012 and pursuing M. Tech. degree in VLSI Design from JECRC University, Jaipur, Rajasthan, India

Abhilasha received the B.Tech. and M.Tech. degrees in Electeonics and communication Engineering and VLSI Design respectively from Mody Institute of Technology and science, India in 2010 and 2012, respectively. Currently she associate with JECRC University, Jaipur, Rajasthan, India