*IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*
*Volume 4, Issue 3, Ver. I (May-Jun. 2014), PP 84-87*
*e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197*
*www.iosrjournals.org*

# Designing Of Pipelined Architecture of Arithmetic Core and Analysis of Area and Timing Performance

Hemraj Sharma[1], Abhilasha[2]
[1](VLSI Design, JECRC University, India)
[2](VLSI Design, JECRC University, India)

***Abstract:*** *The aim of this paper is designing of pipelined architecture of arithmetic core and analysis of area and timing performance of that arithmetic core consisting of fixed point as well as floating point arithmetic cores. The basic concept behind designing such a core is to optimally utilize the algorithms of fixed point as well as floating point arithmetic operations, i.e., addition, subtraction division and multiplication and to enhance the operational speed of these calculations along with comparing a better technique out of fixed and floating point techniques to choose one of them for implementing in future. For this purpose, the arithmetic core is divided into two parts, namely, fixed point arithmetic core and floating point arithmetic core. Both the fixed point as well as floating point cores are sub-divided into four parts which are basically the mathematical operations, i.e., addition, subtraction, multiplication and division. In this Research paper, we discuss the fixed point arithmetic core. The simulation has been carried out on Modelsim (Student edition) EDA tool 10.0c.*
***Keywords:*** *Carry Look Ahead Adder, N.M format, Urdhva - tiryakbhyam*

## I. 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, an arithmetic core is proposed.

Arithmetic core is nothing but a combinational unit consisting of two major types of arithmetic cores, namely, fixed point arithmetic core and floating point arithmetic core. In case of fixed 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, as well as the reduced software complexity for emulated integer multiply and divide. But this speed improvement does come at the cost of reduced range and accuracy of the algorithm variables. So in order to increase the range of variables and accuracy of operations, arithmetic core consisting of both fixed as well as floating point techniques is being used in parallel execution. In this paper, we will discuss about fixed point arithmetic core.

## II. Proposed Design

A 32-bit pipelined architecture of arithmatic core is proposed. In this proposed model, we are creating a pipelined architecture which is named as arithmetic core. It consists of fixed point and floating point arithmetic cores which are proposed to operate in parallel, i.e., mathematical operations, namely, addition, subtraction, multiplication and division in both the types of cores can be executed simultaneously using some specific methodologies for fixed point and floating point arithmetic core operations. The arithmatic core is of two types:

**a)** Fixed point arithmatic core
**b)** Floating point arithmatic core

This whole calculation is 32-bit calculation in which pipelined architecture of both the types of cores is proposed where arithmetical calculations based on fixed as well as floating point techniques will be executed at a very fast speed taking very less time.
The design flow diagram of the complete arithmetic core is drawn in Fig.1 below-

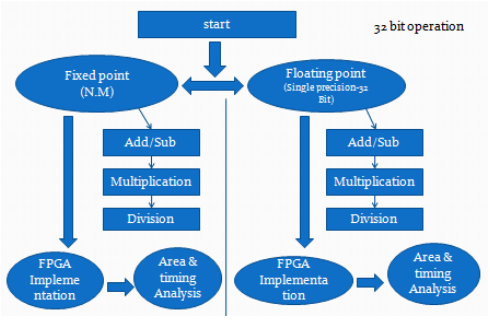


**Fig.1: Proposed Design Flow**

## III. Fixed Point Arithmetic Core

In fixed point arithmatic core based design we use N.M or Qn.m format. This format is commonly known as "Q" format. Its theory is as under-

***Qn.m*:** The uncertain form off the "Q" notation. Since the complete word is a 2's complement integer, a sign bit is implied. For instance, Q1.30 describes a number with 1 integer bit and 30 fractional bits stored as a 32-bit 2's complement integer.[1,2]

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

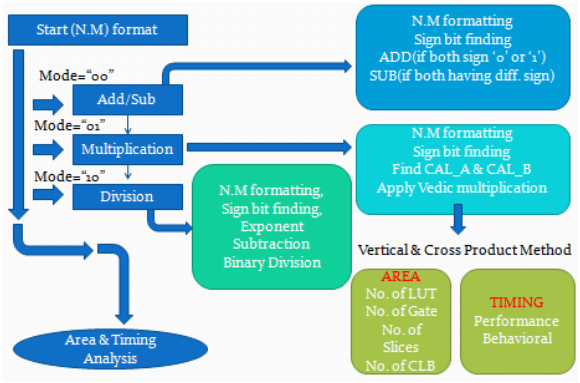


**Fig.2: Fixed Point Design Flow**

In the fixed point design architecture, codes of 32-bit addition/subtraction, multiplication and division have been designed and implemented. Coding is implemented using N.M format. 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 and multiplication code is designed using vedic multiplication technique named Urdhva - tiryakbhyam, i.e., "Vertically and Crosswise" technique.[3] In division code, binary division and exponent subtraction are performed based on N.M format.

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.[4,5] 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.[6,7]

## IV. Timing And Area Analysis

a) **Add/Subtract:**

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

**Table 1: Add/Subtract Code Timing Parameters**

| Parameters | Fixed |
|---|---|
| Min. input arrival time before clock (ns) | 6.655 |
| Max. output required time after clock (ns) | 7.078 |
| Max. combinational path delay (ns) | 50.956 |

**Table 2: Add/Subtract Code Area Parameters**

| Parameters | Fixed |
|---|---|
| Total Number of 4-input LUTs | 202 out of 7168 (2%) |
| Number of occupied Slices | 106 out of 3584 (2%) |
| Total Gate Count | 1550 |

b) **Multiplication:**

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

**Table 3: Multiplication Code Timing Parameters**

| Parameters | Fixed |
|---|---|
| Min. input arrival time before clock (ns) | 6.968 |
| Max. output required time after clock (ns) | 4.368 |
| Max. combinational path delay (ns) | Not Found |

**Table 4: Multiplication Code Area Parameters**

| Parameters | Fixed |
|---|---|
| Total Number of 4-input LUTs | 2549 out of 7168 (35%) |
| Number of occupied Slices | 1362 out of 3584 (38%) |
| Total Gate Count | 20,278 |

c) **Division:**

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

**Table 5: Division Code Timing Parameters**

| Parameters | Fixed |
|---|---|
| Min. Period (ns) | 87.539 |
| Min. input arrival time before clock (ns) | 95.813 |
| Max. output required time after clock (ns) | 90.892 |
| Max. combinational path delay (ns) | 104.369 |

**Table 6: Division Code Area Parameters**

| Parameters | Fixed |
|---|---|
| Total Number of 4-input LUTs | 8852 out of 7168 (123%) |
| Number of occupied Slices | 4504 out of 3584 (125%) |
| Total Gate Count | 2,94,062 |

## V. Results and Conclusion

From the above tables, we find that the timing and area performances of fixed point add/subtract and multiplication codes are **better** than the timing and area performances of fixed point division code. Hence, we can conclude that for less delay, less area consumption and high speed computation, fixed point add/subtract code using CLA and fixed point multiplication code using Vedic (Urdhva - Tiryakbhyam) multiplier can be used while in case of division, we should look for some alternate code since fixed point division code is not suitable for high speed and quality performance.

The simulation waveforms of add/subtract, multiplication and division codes of fixed point arithmetic core are shown in Fig.6, Fig.7 and Fig.8, respectively, below-

a) Add/subtract:

Inputs-> a_dec => 00000F
a_fraction => 30
b_dec => 00000F
b_fraction => 30

Output-> sum => 00001E60

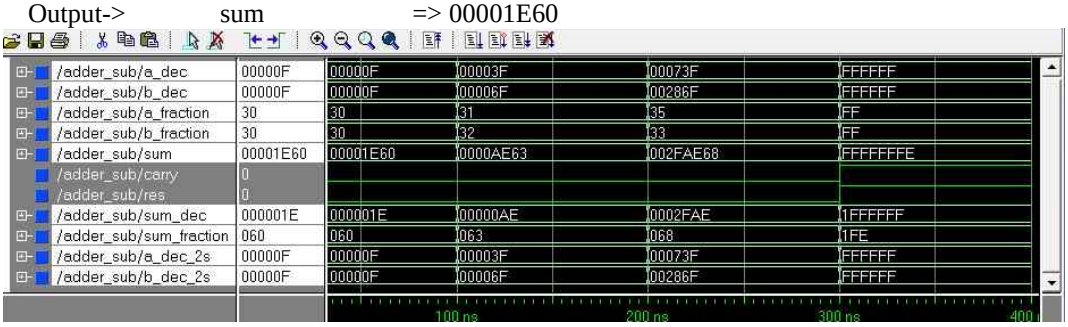


**Figure 6: Simulation Waveform of Fixed Point Add/Subtract Code**

b) Multiplication:

Inputs-> a_dec => 00001F
a_fraction => 03
b_dec => 000007
b_fraction => 0F

Output-> multiplication => 0000000000DAE62D

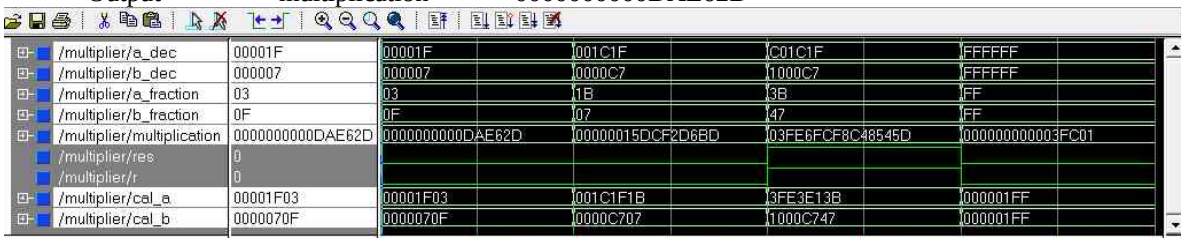


**Figure 7: Simulation Waveform of Fixed Point Multiplication Code**

c) Division:

Inputs-> a_dec => 000000
a_fraction => 0D
a_comb => 0000000D
b_dec => 000000
b_fraction => 03
b_comb => 00000003

Outputs-> remind => 00000001
quotient => 00000004

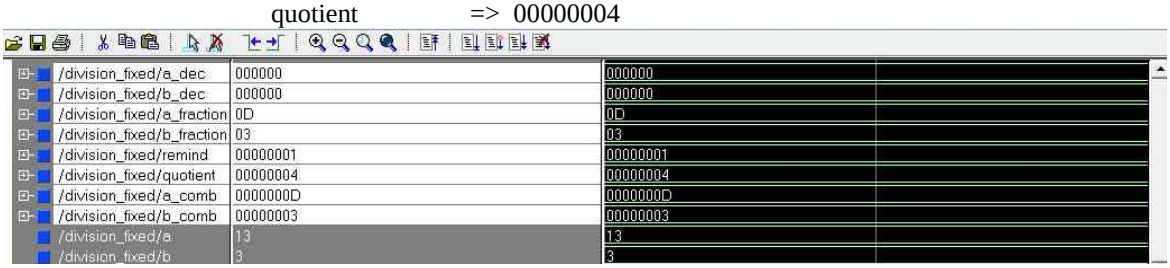


**Figure 8: Simulation Waveform of Fixed Point Division Code**

## Acknowledgement

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

## References

[1]. a b Texas Instruments, TMS320C64x DSP Library Programmer's Reference, Appendix A.2.
[2]. Math Works Fixed-Point Toolbox Documentation Glossary.
[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]. Basavaraj B., Comparison of Vedic Multipliers With Conventional Hierarchical Array of Multipliers, International Journal of Engineering Research & Technology, 2(10), 2013.
[5]. Nicholas A.P., Williams K.R. and Pickles J., Application of Urdhava Sutra, Spiritual Study Group, Roorkee, India, 1984.
[6]. Sree Nivas A, Kayalvizhi N, Implementation of Power Efficient Vedic Multiplier, International Journal of Computer Applications, 43(16), 2012.
[7]. 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.