ISSN XXXX XXXX © 2016 IJESC

Research Article Volume 6 Issue No. 12

FPGA Based High Speed, Low Power Matrix Multiplier using Urdva Tiryakbhyam Algorithm

Pranjali J. Rathod[1], S. S. Mungona[2]
ME student[1], Professor[2]
Sipna College of Engineering and Technology, Amravati, India

Abstract:
Matrix multiplication is a computation intensive operation and plays an important role in many scientific and engineering applications such as image processing, discrete signal processing. This paper presents architecture for the multiplication of two matrices using Field Programmable Gate Array (FPGA).This paper presents unsigned two 3x3 High-Speed matrix multiplier The hierarchical structuring has been used to optimize for multipliers using "Urdhava Trigyagbhyam" sutra (vertically and crosswise)which is one of the sutra for Vedic mathematics. Each element of matrix is represented by 4-bit, output is of 8 bit. The coding has been done using VHDL and synthesized using Altera Quartus II. A concept of design is hierarchical structuring; this gives less computation time for calculating the multiplication result. We will synthesize the proposed designs and the existing design using Altera Quartus II tools. The proposed structure consumes less energy.

Key words: FPGA, Matrix multiplier, vedic, Urdhava Trigyagbhyam, VHDL

I. INTRODUCTION

Matrix multiplication is frequently used operation in a wide variety of graphics, image processing, robotics, and signal processing applications. The increases in the density and speed of field-programmable gate arrays (FPGAs) make them attractive as flexible and high-speed alternatives to DSPs and ASICs. It is a highly procedure oriented computation, there is only one way to multiply two matrices and it involves lots of multiplications and additions. But the simple part of matrix multiplication is that the evaluation of elements of the resultant elements can be done independent of the other, this point to distributed memory approach. In this paper, we propose an architecture that is capable of handling matrices of variable sizes our designs minimize the gate count, area, improvements in latency, computational time, and throughput for performing matrix multiplication and reduces the number of multiplication and additions hardware required to get the matrices multiplied on commercially available FPGA devices. Matrix Multiplication is a frequently used kernel operation in a wide variety of graphics, image processing, robotics, and signal processing applications. Several signal and image processing operations can be reduced to matrix multiplication. Most of the previous work on matrix multiplication on FPGAs focuses on latency optimization. The proposed architecture realized on FPGA which is based on Vedic Multiplication sutra (algorithm) "Urdhava Trigyagbhyam" .The implementation architecture is also use of IP CORE (Intellectual Property) for adders which allows us to be optimized for speed and space. The objective of this paper is to propose a low area, speed, energy efficient, maximum running frequency, low power, matrix multiplier.

II. LITERATURE REVIEW

A Number of experimental investigations have been reported for matrix multiplication and their blends to improve speed as well as reduce power consumption. The findings of few such studies are presented below. This paper presented the design methods and Field Programmable Gate Array (FPGA) implementation of matrix multiplier in image and signal processing applications. This paper gives the detailed review of matrix product implementation on FPGA where area, energy, speed, power dissipation, latency and time efficient methods are compared for different Xilinx families. Here also presented parallel architecture and parameterized system for matrix product used in FPGA. In this way, this literature will help in making performance characterization to implement high performance matrix multiplier on FPGA. [1] Paper presented new algorithms and architectures for matrix multiplication on configurable devices. These have reduced energy dissipation and latency compared with the state-of-the-art field-programmable gate array (FPGA)-based designs. By profiling well-known designs, they identify "energy hot spots," which are responsible for most of the energy dissipation. Based on this, they develop algorithms and architectures that offer tradeoffs among the number of I/O ports, the number of registers, and the number of PEs. To avoid time-consuming low-level simulations for energy profiling and performance prediction of many alternate designs, they derived functions to represent the impact of algorithm design choices on the system-wide energy dissipation, area, and latency. These functions are used to either optimize the energy performance or provide tradeoffs for a family of candidate algorithms and architectures.. [4] This paper present computations involving matrices form the kernel of a large spectrum of computationally demanding applications for which FPGAs have been utilized as accelerators. Their performances related to their underlying architectural and system parameters such as computational resources, memory and I/O bandwidth. A simple analytic model that gives an estimate of the performance of FPGA-based sparse matrix vector and matrix-matrix multiplication is presented, dense matrix multiplication being a special case. The efficiency of existing implementations is compared to the model and performance trends for future technologies examined. [6] Paper presented 2x2 High-Speed matrix multiplier using Virtex 5 ML 507 Evaluation Platform (xc5vfx70t-1ff1136) FPGA. The

hierarchical structuring has been used to optimize for multipliers using "Urdhava Trigyagbhyam" sutra (vertically and crosswise) which is one of the sutra for Vedic mathematics. Each element of matrix is represented by 16-bit.The coding has been done using VHDL and synthesized using Xilinx 13.2. A concept of design is hierarchical structuring; This gives less computation time for calculating the multiplication result.**[7]** Paper presented Matrix multiplication is the kernel operation used in many transform, image and discrete signal processing application. They developed new algorithms and new techniques for matrix multiplication on configurable devices. In this paper, they have proposed three designs for matrix-matrix multiplication. These design reduced hardware complexity, throughput rate and different input/output data format to match different application needs. These techniques have been designed implementation on Virtex-4 FPGA. They have synthesized the proposed designs and the existing design using Synopsys tools.[8] Paper presented FPGA-based hardware realization of matrix multiplication based on distributed memory approach architecture. They propose an architecture that is capable of handling matrices of variable sizes our designs minimize the gate count, area, improvements in latency, computational time, and throughput for performing matrix multiplication and reduces the number of multiplication and additions hardware required to get the matrices multiplied on commercially available FPGA devices. [9]

III. SYSTEM ARCHITECTURE

The proposed work is aimed at finding high speed and low power consumption of matrix multiplication base on FPGA. Conventional method take more time for matrix multiplication calculation so we study different Vedic math's sutras for multiplication and use "Urdhva Triyagbhyam Algorithm" which take less timing.

3.1 Conventional System

The common multiplication method is adding and shift algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm [3] is one of the most popular algorithms. To achieve speed improvements Wallace Tree algorithm can be used, here the number of sequential adding stages can be reduced. However with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand, serial-parallel multipliers compromise speed to achieve better performance for area and power consumption. The selection of a parallel or serial multiplier actually depends on the nature of application. In this lecture we introduce the multiplication algorithms and architecture and compare them in terms of speed, power and combination of these metrics.

Method for 3x3 matrix multiplication:

Two 3x3 matrix multiplies requires 27 multiplications and18 additions. Here, in 3x3 matrix multiplication, each bit is of 4-bit where 4-bitmultiplication and 8-bit addition is takes place. For32-bit adder uses 32 bitIP core (Intellectual Property).

Matrix A and Matrix B 3x3 matrices

$$A=\begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix} \qquad B=\begin{pmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{pmatrix}$$

MATRIX A X MATRIX B = MATRIX C

$$C=\begin{pmatrix} C11 & C12 & C13 \\ C21 & C22 & C23 \\ C31 & C32 & C33 \end{pmatrix}$$

C11=a11xb11+a12xb21+a13xb31
C12=a11xb12+a12xb22+a13xb32
C13a11xb31+a12xb23+a13xb33

C21=a21xb11+a22xb21+a23xb31
C22=a21xb12+a22xb22+a23x32
C23=a21xb31+a22xb23+a23xb33

C31=a31xb11+a32x21+a33xb31
C32=a31xb12+a32xb22+a33xb32
C33=a31xb31+a32xb23+a33

in this case 3x3 matrix multiplication each element in multiplication and addition takes place so it take more time for calculation time delay is occurred.

3.1.1Array Multiplier

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length. In this paper conventional array multiplier is synthesized using 16T full adder cell. Consider the multiplication of two unsigned n-bit numbers, where A=An-1, An-2,….A0 is the multiplicand and B=Bn-1, Bn-2,….B0 is the multiplier. The product of these two bits can be written as S7S6S5S4S3S2S1S0, where S0 is the LSB and S7 is the MSB. Figure shows multiplication of a 4x4 array multiplier using carry save adders

A x B				A3	A2	A1	A0
				B3	B2	B1	B0
				A3B0	A2B0	A1B0	A0B0
			A3B1	A2B1	A1B1	A0B1	
		A3B2	A2B2	A1B2	A0B2		
	A3B3	A2B3	A1B3	A0B3			
S7	S6	S5	S4	S3	S2	S1	S0

Figure.1. 4x4 Array Multiplication.

The conventional array multiplier uses carry save addition to add the products. In the carry save addition method, the first

row will be either half adders or full adders. If the first row of the partial products is implemented with full adders, Cin will be considered "0". Then the carries of each full adder can be diagonally forwarded to the next row of the adder. The resulting multiplier is said to be carry save array multiplier as the carry bits are not immediately added but rather saved for the next stage of addition. Hence the names carry save multiplier. In the design if the full adders have two input data the third input is considered as zero. The final adder which is used to add carries and sums of the multiplier is removed. Then the carries of the multiplier at the final stage is carefully added to the inputs of the multiplier. The carry of the fourth column of the multiplier is given to the input of the fifth column instead of zero. Then the carry of the fifth column is forwarded to the input of the sixth column and so on.

3.2 Urdhva Triyagbhyam Algorithm:

The Sanskrit word 'Veda' means 'knowledge'. Vedic Mathematics is the name given to the ancient system of Indian Mathematics which was rediscovered from the Vedas between 1911 and 1918 by Sri Bharati KrsnaTirthaji (1884-1960). According to his research all of mathematics is based on sixteen Sutras, or word-formulae. Vedic Mathematics is much simpler and easy to understand than conventional mathematics .Urdhva Triyagbhyam is a multiplication Sutra (Algorithm) in Vedic Mathematics for the multiplication of decimal numbers. We used same idea to the binary numbers to make it compatible with the digital hardware. This is the general formula applicable to all cases of multiplication. The sixteen sutras and their corollaries are as follows:

- Ekadhikina Purvena -By one more than the previous one (Cor: Anurupyena)
- Nikhilam Navatashcaramam Dashatah -All from 9 and the last from 10 (Cor: Sisyate Sesasamjnah)
- Urdhva-Tiryagbyham-Vertically and crosswise (Cor: Adyamadyenantyamantyena)
- Paraavartya Yojayet-Transpose and adjust (Cor: Kevalaih Saptakam Gunyat)
- Shunyam Saamyasamuccaye-When the sum is the same, that sum is zero. (Cor: Vestanam)
- (Anurupye) Shunyamanyat-If one is in ratio, the other is zero (Cor: Yavadunam Tavadunam)
- Sankalana-vyavakalanabhyam-By addition and by subtraction (Cor:Yavadunam Tavadunikritya Varga Yojayet)
- Puranapuranabyham-By the completion or non-completion (Cor: Antyayordashake)
- Chalana-Kalanabyham-Differences and Similarities
- Yaavadunam-Whatever the extent of its deficiency (Cor: Samuccayagunitah)
- Vyashtisamanstih-Part and Whole (Cor: Lopanastha panabhyam)
- Shesanyankena Charamena-The remainders by the last digit (Cor: Vilokanam)
- Sopaantyadvayamantyam-The ultimate and twice the penultimate (Cor: Gunitasamuccayah Samuccayagunitah)
- Ekanyunena Purvena-By one less than the previous one (Cor: Dhvajanka)
- Gunitasamuchyah-The product of the sum is equal to the sum of the product (Cor: Dwandwa Yoga)
- Gunakasamuchyah-The factors of the sum is equal to the sum of the factors

Urdhva means Vertical and Triyagbhyam means crosswise.
Steps:

a) Vertical Multiplication (LSB):
Multiply LSBs (Least Significant Bits) '1' and '0'; place product '0' as LSB of the result.

b) Crosswise Multiplication:
Multiply Crosswise '1' and '1'; '0' and '1'; add the product terms and place obtained sum as middle term of the result.

c) Vertical Multiplication (MSB):
Multiply the MSBs (Most Significant Bits) '1' and '1'; place product '1' as MSB of the result:
Example: a1 a0 * b1 b0
Explanation for 2-bit Multiplication: a1 a0 * a1 a0
a1 a0
X b1 b0
---
a1 X b1 ; a1 X b0 + a0 X b1 ; a0 X b0

Similarly multiplication can be done for any digital numbers using Urdhva Triyagbhyam algorithm of ancient Indian Vedic mathematics.

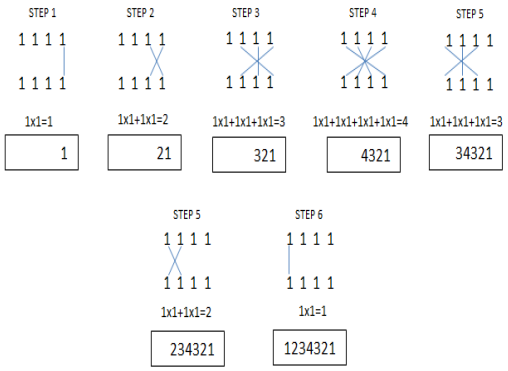

Figure.2. Illustration of Vertical and Crosswise algorithm.

3.2.1. 4-bit multiplier:

The 4-bit multiplier is made by using 4, 2-bitmultiplier blocks and 3, 4-bit adders. Here the
Multiplicands are of bit size (n=4) and the output is of bit size 8. As per the sutra, the 4-bit (n) input is divided into small block of size n/2=2 which reduces the complexity. The newly formed blocks of 2 bits are given as input to 2-bit Vedic multiplier blocks. It will leads to high efficiency multiplier architecture. The output obtained from output of 2-bit Vedic

multiplier block which is of 4 bits is given to the 4-bit adder. For 4-bit adder uses 4-bit IP core (Intellectual Property). The block diagram is shown in

a3 a2 | a1 a0 (4-bit multiplicand)
X b3 b2 | b1 b0 (4-bit multiplier)

P7 P6 P5 P4 P3 P2 P1 P0 (8-bit Result)

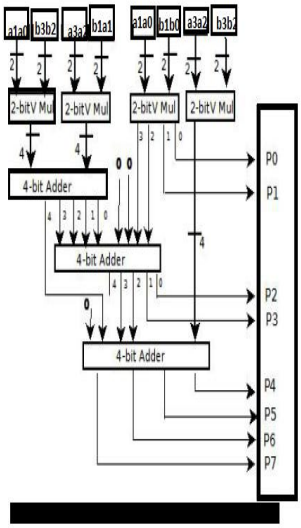

Figure.3: 4 bit multiplier block

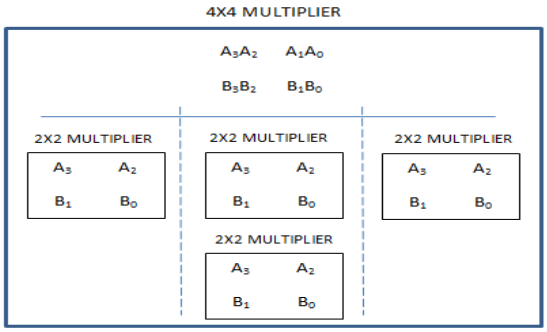

Figure.4: 4x4 bit multiplication.

In this case, A1A0 forms AL, A3A2 forms AM, and similarly B1B0 forms BL,B3B2 forms BM of the two 4-bit inputs. Firstly, the vertical result of AL and BL is computed by passing it through a 2x2 vedic multiplier. Then, the crosswise result of (AMxBL + ALxBM) is computed using 2x2 multipliers and a 4-bit adder[8]. Finally the vertical result of AM and BM are computed using a 2x2 multiplier.

3.3. FPGA

A **field-programmable gate array** (**FPGA**) is an integrated circuit created to be configured by the customer after manufacturing—hence "field-programmable". The FPGA configuration is generally defined using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC can perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design, offer advantages for many applications.

ADVANTAGES OF FPGA

FPGAs have become very popular in the recent years owing to the following advantages that they offer:

1) **Fast prototyping and turn-around time-**
2) **NRE cost is zero**
3) **High-Speed**
4) **Low cost**

3.4. VHDL

The **VHSIC Hardware Description Language** (VHDL) is a formal notation intended for use in all phases of the creation of electronic systems. Because it is both machine readable and human readable, it supports the development, verification, synthesis, and testing of hardware designs; the communication of hardware design data; and the
maintenance, modification, and procurement of hardware

- **BENEFITS OF USING VHDL**

- Executable specification
- Validate spec in system context (Subcontract)
- Functionality separated from implementation
- Simulate early and fast (Manage complexity)
- Explore design alternatives
- Get feedback (Produce better designs)
- Automatic synthesis and test generation (ATPG for ASICs)
- Increase productivity (Shorten time-to-market)
- Technology and tool independence (though FPGA features may be unexploited)
- Portable design data (Protect investment)

IV. RESULT

In this chapter we present the test environment and the experimental results of our design modules. Our objectives of this project are to study and design a new method for high speed and low power model of Matrix Multiplication using Urdhava Trigyagbhyam Algorithm to improve speed performance and time, reduction area. The design is implemented in VHDL, simulated using modelsim and synthesized by ALTERA QUARTUS II.

Simulation result as follows:

1) consider a = 15 b =12
hence its product is 15x12=180

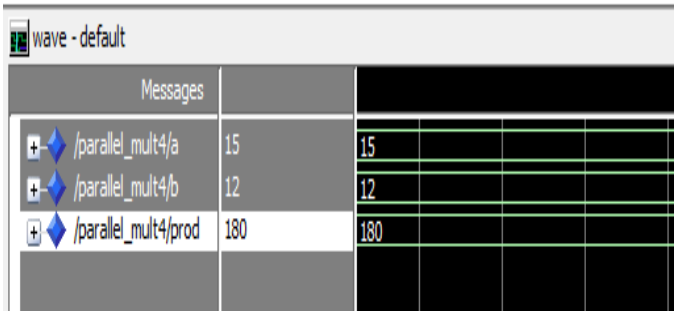

Figure. 5: Simulation result conventional method multiplication.

2) consider a = 2 b =3
hence its product is 2x 3= 6

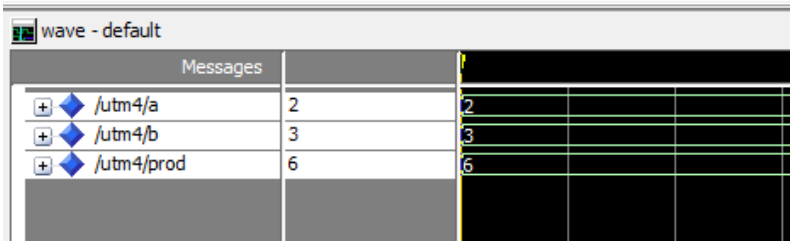

Figure. 6 : Simulation result Urdhava Trigyagbhyam multiplication algorithm.

3) Consider matrix x x matrix y = matrix z

$$X = \begin{pmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{pmatrix} \qquad Y = \begin{pmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{pmatrix}$$

$$Z = \begin{pmatrix} 23 & 36 & 28 \\ 25 & 39 & 34 \\ 28 & 32 & 37 \end{pmatrix}$$

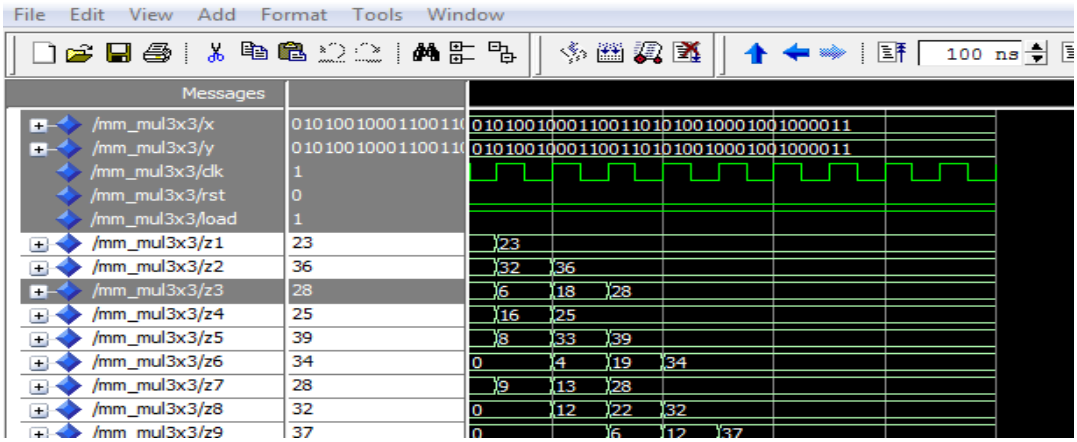

Figure .7 : Simulation result Matrix multiplication using conventional method.

4) Consider matrix x x matrix y = matrix z

$$X = \begin{pmatrix} 3\,4\,2 \\ 2\;\;5\,3 \\ 3\,2\,5 \end{pmatrix} \qquad Y = \begin{pmatrix} 3 & 4 & 2 \\ 2 & 5 & 3 \\ 3 & 2 & 5 \end{pmatrix}$$

$$Z = \begin{pmatrix} 23\;36\;28 \\ 25\;39\;34 \\ 28\;32\;37 \end{pmatrix}$$

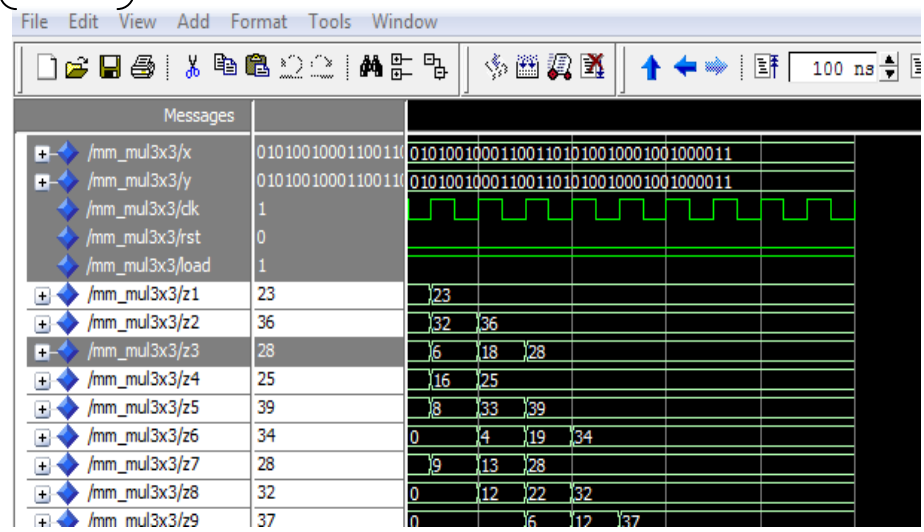

Figure. 8 : Simulation result Matrix multiplication using Urdhava Trigyagbhyam algorithm.

Analysis and synthesis results

All the result has shown below based on ALTERA cyclone II EP2C20F484C7 FPGA platform. From above result also we get the High speed and low power matrix multiplier using Urdhava Trigyagbhyam algorithm.

Table 1: compare between multiplier using Conventional method and Urdhava Trigyagbhyam algorithm

Sr. no.	Parameter	Conventional multiplier	Multiplier using Urdhava Trigyagbhyam algorithm
1	Area	35	33
2	Time(ns)	70.678	20.166
3	Frequency(MHz)	14.148	49.588
4	Power(mw)	68.53	68.52

Table 2: compare between matrix multiplier using Conventional method and Urdhava Trigyagbhyam algorithm

Sr. no.	Parameter	Matrix multiplication using Conventional multiplier	Matrix multiplication using Urdva Triyagbhyam sutra
1	Area	563	542
2	Time(ns)	9.342	8.854
3	Frequency (MHz)	107.043	112.943
4	Power(mw)	84	83.97

Area: Area can be defined as number of logic resources utilized by the FPGA circuit. Area is directly proportional to number of Logical Elements (LE) utilized by the architecture. Mathematically, it can be written as

Area = Total Number of logic elements utilized by the architecture

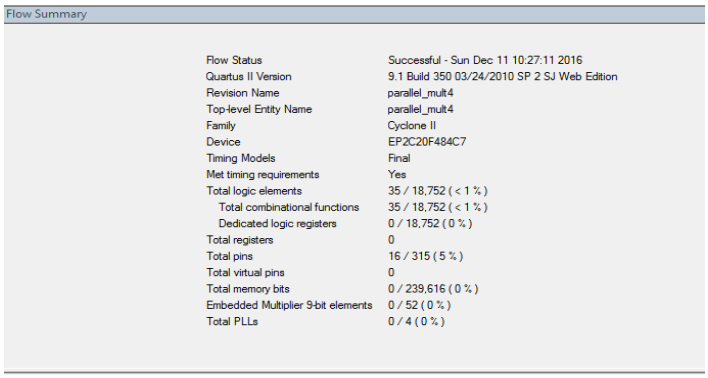

Figure . 9: Area for conventional method multiplication.

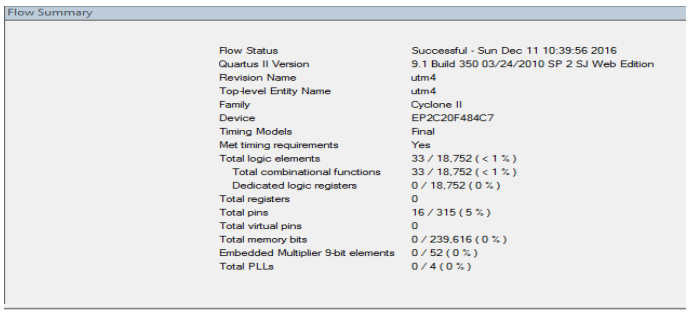

Figure .10: Area for Urdhava Trigyagbhyam algorithm.

Time: The propagation delay is the execution time required by the architecture after application of input to produce output. It ultimately called as process time required by the device.

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	9.720 ns	load	x9[3]	--	clk	0
2	Worst-case tco	N/A	None	9.342 ns	tz9[7]	z9[7]	clk	--	0
3	Worst-case th	N/A	None	0.659 ns	y[8]	y3[0]	--	clk	0
4	Clock Setup: 'clk'	N/A	None	155.57 MHz (period = 6.428 ns)	temp1[0]	tz1[7]	clk	clk	0
5	Total number of failed paths								0

Figure.11: Time for matrix multiplication using conventional method.

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	9.124 ns	load	temp26[3]	--	clk	0
2	Worst-case tco	N/A	None	8.854 ns	tz1[1]	z1[1]	clk	--	0
3	Worst-case th	N/A	None	-0.106 ns	y[8]	y3[0]	--	clk	0
4	Clock Setup: 'clk'	N/A	None	88.68 MHz (period = 11.276 ns)	temp17[0]	tz6[7]	clk	clk	0
5	Total number of failed paths								0

Figure. 12: Time for Matrix multiplication using Urdhava Trigyagbhyam algorithm.

Here, the propagation delay for both the system is as shown in table where Time for matrix multiplication using conventional method is9.342 ns and Time for Matrix multiplication using Urdva Triyagbhyam algorithm is 8.854 ns . By comparing this above value we found that the propagation delay required for conventional method is more than the using Urdhava Trigyagbhyam algorithm.

Power:

The power required by the device is the summation of core static thermal power dissipation and I/O thermal power dissipation. Mathematically it can be written as,
Total Power = Core Static thermal power dissipation + I/O Thermal Power dissipation

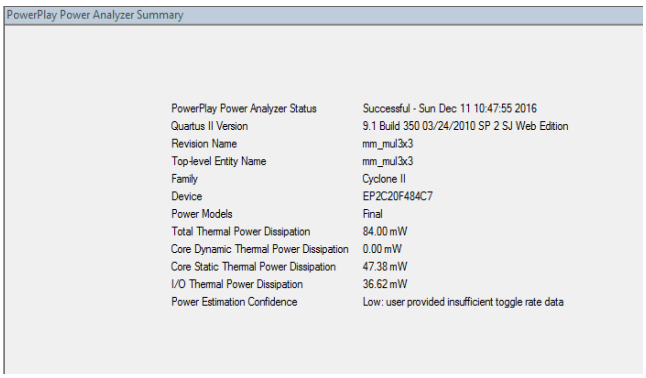

Figure.13: Power for Matrix multiplication using Conventional method.

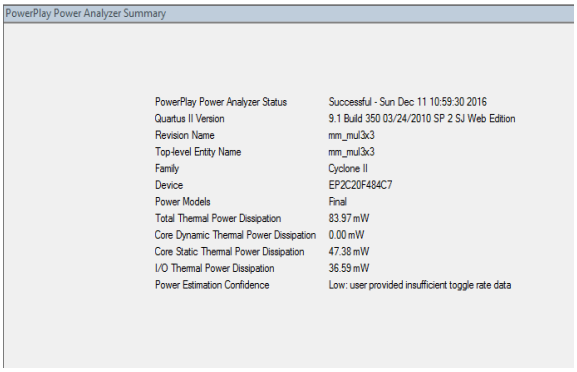

Figure. 14: Power for Matrix multiplication using Urdhava Trigyagbhyam algorithm.

Here, power for both the system is as shown in table where power for matrix multiplication using conventional method isc84 maw and power for Matrix multiplication using Urdva Triyagbhyam algorithm is 83.97 mW . By comparing this above value we found that the power required for conventional method is more than the using Urdhava Trigyagbhyam algorithm.

RTLVIEW:

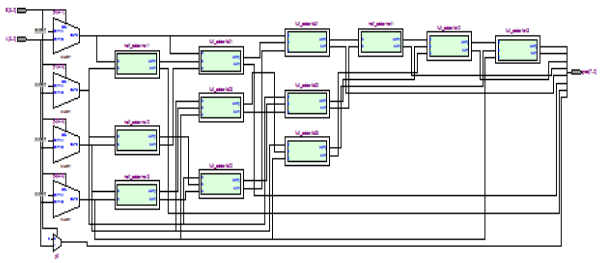

Figure 15: RTL view for multiplication using Conventional method.

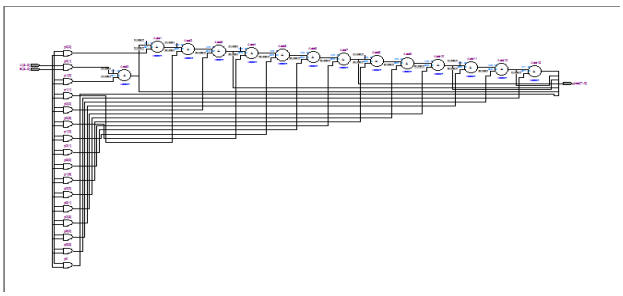

Figure .16: RTL view for multiplication using Urdhava Trigyagbhyam algorithm.

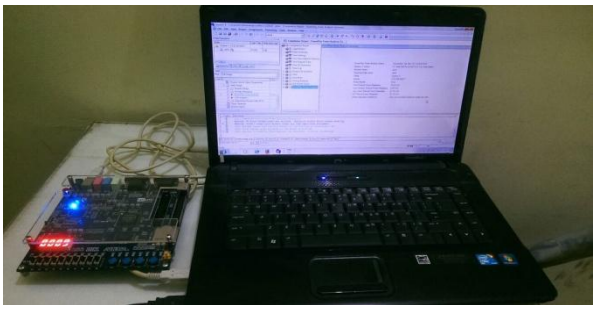

Figure. 17: ALTERA DE1 FPGA board and Quartus II

V. CONCLUSION

Proposed work presents architecture for the multiplication of two matrices using Field Programmable Gate Array (FPGA).This paper presents unsigned two 3x3 High-Speed matrix multiplier The hierarchical structuring has been used to optimize for multipliers using “Urdhava Trigyagbhyam” sutra (vertically and crosswise)which is one of the sutra for Vedic mathematics. Each element of matrix is represented by 4-bit, output is of 8 bit. The coding has been done using VHDL and synthesized using Altera Quartus II. A concept of design is hierarchical structuring; This gives less computation time for calculating the multiplication result. We will synthesized the proposed designs and the existing design using Altera Quartus II tools.

VI. RFERENCE

[1]. Ankita Mishra, Hemika Yadav, Sarita Rani, Shivani Saxena “A Review of Different Methods for Matrix Multiplication Based on FPGA” International Journal of VLSI and Embedded Systems-IJVES ISSN: 2249 – 6556 Vol 05, Article 02225; February 2014

[2]. Shriyashi Jain “FPGA Implementation of Latency, Computational Time Improvements in Matrix Multiplication” International Journal of Computer Applications (0975 – 8887) Volume 86 – No 8, January 2014

[3]. Qasim, Abbasi and Almashary, “A proposed FPGA-based parallel architecture for matrix multiplication” Circuits andSystems, 2008.APCCAS 2008. IEEE Asia PacificConference,pp. 1763-1766, Nov. 30-Dec. 3 2008

[4]. J. Jang, S. Choi, and V. K. Prasanna, “Energy-efficient matrix multiplication on FPGAs,” in Proc. Int. Conf. Field Programmable Logic Appl., 2002, pp. 534–544.

[5]. J. Jang, S. Choi, and V. K. Prasanna, “Energy and Time Efficient Matrix Multiplication on FPGAs,” IEEE Trans.on VLSI Systems, Vol. 13, No. 11, pp. 1305–1319, Nov.2005.

[6]. Colin Yu Lin, Philip H.W. Leong “A MODEL FOR MATRIX MULTIPLICATION PERFORMANCE ON FPGAS” 2011 21st International Conference on Field Programmable Logic and Applications

[7]. Ms.S.V.Mogre1,Mr. D. G.Bhalke2 “Implementation of High Speed Matrix Multiplier using Vedic Mathematics on FPGA” 2015 International Conference on Computing Communication Control and Automation

[8]. Shivangi Tiwari, Shweta Singh, Nitin Meena “FPGA Design and Implementation of Matrix Multiplication Architecture by PPI-MO Techniques” *International Journal of Computer Applications (0975 – 8887) Volume 80 – No1, October 2013*

[9]. Shriyashi Jain, Neeraj Kumar, Jaikaran Singh, Mukesh Tiwari “FPGA Implementation of Latency, Computational Time Improvements in Matrix Multiplication” International *Journal of Computer Applications (0975 – 8887) Volume 86 – No 8, January 2014*

[10]. Tai-Chi Lee, Michael Guddy “Matrix Multiplication on FPGA-Based Platform” Proceedings of the World Congress on Engineering and Computer Science 2013 Vol IWCECS 2013, 23-25 October, 2013, San Francisco, USA.