Novel High Speed Vedic Mathematics Multiplier Using Mathematics

T.Sowmya[1], K. Anitha[2], G.Vinatha[3]
[1, 2, 3] Dept of ECE
[1, 2, 3] St. Martin's Engineering College, Dhulapally (v), Kompally, Secunderabad-500100, Telangana State, India.

Abstract- *With the advent of new technology in the fields of VLSI and communication, there is also an ever-growing demand for high-speed processing and low area design. It is also a well-known fact that the multiplier unit forms an integral part of processor design. Due to this regard, high speed multiplier architectures become the need of the day. In this project, we introduce a novel architecture to perform high speed multiplication using ancient Vedic maths techniques. A new high-speed approach utilizing 4:2 compressors and novel 7:2 compressors for addition has also been incorporated in the same and has been explored. Upon comparison, the compressor-based multiplier introduced in this project, is almost two times faster than the popular methods of multiplication. With regards to area, a 1% reduction is seen. The design and experiments were carried out on a Xilinx 14.7i.*

Keywords- 4:2 Compressors, 7:2 Compressor, Booth's Multiplier, High Speed Multiplier, Modified Booth's Multiplier, UrdhwaTiryakbhyam Sutra, Vedic Mathematics

I. INTRODUCTION

The speed of a processor greatly depends on its multiplier's performance. This in turn increases the demand for high speed multipliers, at the same time keeping in mind low area and moderate power consumption [2]. Over the past few decades, several new architectures of multipliers have been designed and explored. Multipliers based on the Booth's and modified Booth's algorithm is quite popular in modern VLSI design but come along with their own set of disadvantages. In these algorithms, the multiplication process, involves several intermediate operations before arriving at the final answer. The intermediate stages include several comparisons, additions and subtractions which reduce the speed exponentially with the total number of bits present in the multiplier and the multiplicand [5]. Since speed is our major concern, utilizing such type of architectures is not a feasible approach since it involves several time consuming operations. In order to address the disadvantages with respect to speed of the above mentioned methods, and explored a new approach to multiplier design based on ancient Vedic Mathematics. Vedic Mathematics is an ancient and eminent approach which acts as a foundation to solve several mathematical challenges encountered in the current day scenario.

II. VEDIC MULTIPLICATION TECHNIQUE

The use of Vedic mathematics lies in the fact that it reduces the typical calculations in conventional mathematics to very simple one. This is so because the Vedic formulae are claimed to be based on the natural principles on which the human mind works. Vedic Mathematics is a methodology of arithmetic rules that allow more efficient speed implementation. It also provides some effective algorithms which can be applied to various branches of engineering such as computing.

A. Urdhva Tiryakbhyam Sutra

The proposed Vedic multiplier is based on the "Urdhva Tiryakbhyam" sutra (algorithm). These Sutras have been traditionally used for the multiplication of two numbers in the decimal number system. In this work, we apply the same ideas to the binary number system to make the proposed algorithm compatible with the digital hardware. It is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and crosswise". It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products. The algorithm can be generalized for n x n bit number. Since the partial products and their sums are calculated in parallel, the multiplier is independent of the clock frequency of the processor. Due to its regular structure, it can be easily layout in microprocessors and designers can easily circumvent these problems to avoid catastrophic device failures. The processing power of multiplier can easily be increased by increasing the input and output data bus widths since it has a quite a regular structure. Due to its regular structure, it can be easily layout in a silicon chip. The Multiplier based on this sutra has the advantage that as the number of bits increases, gate delay and area increases very slowly as compared to other conventional multipliers.

B. Multiplication of Two Decimal Numbers 252 x 846

To illustrate this scheme, let us consider the multiplication of two decimal numbers 252x846 by Urdhva-Tiryakbhyam method as shown in Fig. 1. The digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and hence the process goes on. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and all other bits act as carry for the next step. Initially the carry is taken to be zero.

III. PROPOSED SYSTEM

I. Vedic Multiplier for 8x8 bit Module:

The 8x8 bit Vedic multiplier module as shown in the block diagram in Fig. 6 can be easily implemented by using four 4x4 bit Vedic multiplier modules as discussed in the previous section. Let's analyze 8x8 multiplications, say A= A7 A6 A5 A4 A3 A2 A1 A0 and B= B7 B6 B5B4 B3 B2 B1B0. The output line for the multiplication result will be of 16 bits as – S15 S14 S13 S12 S11 S10 S9 S8 S7 S6S5S4 S3 S2 S1 S0. Let's divide A and B into two parts, say the 8-bit multiplicand A can be decomposed into pair of 4 bits AH-AL. Similarly, multiplicand B can be decomposed into BH-BL. The 16-bit product can be written as:

Using the fundamental of Vedic multiplication, taking four bits at a time and using 4-bit multiplier block as discussed, we can perform the multiplication. The outputs of 4x4 bit multipliers are added accordingly to obtain the final product. Here total three 8-bit Ripple-Carry Adders are required as shown in Fig.1

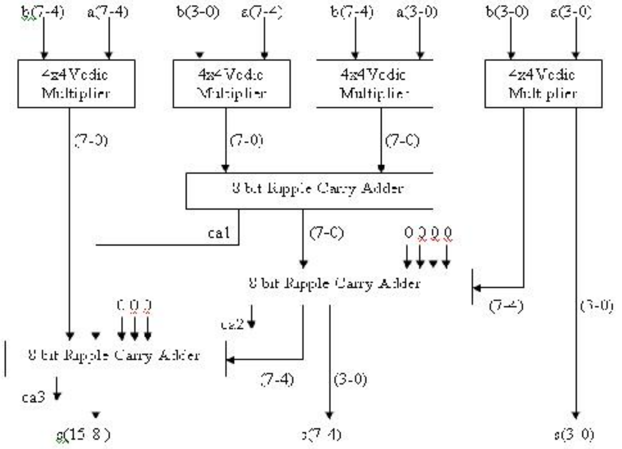

Fig: 1.1 Block Diagram of 8x8 bit Vedic Multiplier

II. Vedic Multiplier for 4X4 bit Module:

Urdhava Tiryakbhyam is a Sanskrit word which means vertically and crosswire in English. The method is a general multiplication formula applicable to all cases of multiplication. It is based on a novel concept through which all partial products are generated concurrently. Fig 3.2demonstrates a 4 x 4 binary multiplication using this method. The method can be generalized for any N x N bit multiplication. This type of multiplier is independent of the clock frequency of the processor because the partial products and their sums are calculated in parallel. The net advantage is that it reduces the need of microprocessors to operate at increasingly higher clock frequencies. As the operating frequency of a processor increases the number of switching instances also increases. This results in more power consumption and also dissipation in the form of heat which results in higher device operating temperatures. Another advantage of Urdhva Tiryakbhyam multiplier is its scalability.
The processing power can easily be increased by increasing the input and output data bus widths since it has a regular structure, Due to its regular structure, it can be easily layout in a silicon chip and also consumes optimum area. As the number of input bits increase, gate delay and area increase very slowly as compared to other multipliers. Therefore, Urdhava Tiryakbhyam multiplier is time, space and power efficient. The line diagram in fig 3.2 illustrates the algorithm for multiplying two 4-bitbinary numbers and. The procedure is divided into 7 steps and each step generates partial products. Initially as shown in step 1 of fig 3.2, the least significant bit (LSB) of the multiplier is multiplied with least significant bit of the multiplicand vertical multiplication). This result forms the LSB of the product. Instep 2 next higher bit of the multiplier is multiplied with the LSB of the multiplicand and the LSB of the multiplier is multiplied with the next higher bit of the multiplicand(crosswire multiplication).

These two partial products are added and the LSB of the sum is the next higher bit of the final product and the remaining bits are carried to the next step. For example, if in some intermediate step, we get the result as 1101, then 1 will act as the result bit (referred as rn) and 110 as the carried out as indicated by the line diagram. The important feature is that all the partial products and their sums for every step can be calculated in parallel. Thus, every step has a corresponding expression as follows:

r0=a0b0. (1)
c1r1=a1b0+a0b1. (2)
c2r2=c1+a2b0+a1b1 + a0b2. (3)
c3r3=c2+a3b0+a2b1 + a1b2 + a0b3. (4)
c4r4=c3+a3b1+a2b2 + a1b3. (5)
c5r5=c4+a3b2+a2b3. (6)
c6r6=c5+a3b3 (7)

With c6r6r5r4r3r2r1r0 being the final product. Hence this is the general mathematical formula applicable to all cases of multiplication and its hardware architecture is shown in fig3.1. In order to multiply two 8-bit numbers using 4-bit multiplier we proceed as follows. Consider two 8-bitnumbers denoted as AHAL and BHBL where AH and BH corresponds to the most significant 4 bits, AL and BL are the least significant 4 bits of an 8-bit number. When the numbers are multiplied according to Urdhava Tiryakbhyam (vertically and crosswire) method, we get,

AH AL
BH BL
____
(AH x BH) + (AH x BL + BH x AL) + (AL x BL).

Thus, we need four 4-bit multipliers and two adders to add the partial products and 4-bit intermediate carry generated. Since product of a 4 x 4 multiplier is8 bits long, in every step the least significant 4 bits correspond to the product and there maining 4 bits are carried to the next step. This process continues for 3 steps in this case. Similarly, 16-bit multiplier has four 8 x 8 multiplier and two 16-bit adders with 8bit carry. Therefore, we see that the multiplier is highly modular in nature. Hence it leads to regularity and scalability of the multiplier layout.

Each block as shown above is 2x2 bit Vedic multiplier. First 2x2 bit multiplier inputs are A1A0 and B1B0. The last block is 2x2 bit multiplier with inputs A3 A2 and B3 B2. The middle one shows two 2x2 bit multiplier with inputs A3 A2 & B1B0 and A1A0 & B3 B2. So, the final result of multiplication, which is of 8 bit, S7 S6S5S4 S3 S2 S1 S0. To understand the concept, the Block diagram of 4x4 bit Vedic multiplier is shown in Fig. 5. To get final product (S7 S6 S5 S4 S3 S2 S1 S0), four 2x2 bit Vedic multiplier (Fig. 3) and three 4-bit Ripple-Carry (RC) Adders are required. The proposed Vedic multiplier can be used to reduce delay. Early literature speaks about Vedic multipliers based on array multiplier structures. On the other hand, we proposed a new architecture, which is efficient in terms of speed. The arrangements of RC Adders shown in Fig. 5, helps us to reduce delay. Interestingly, 8x8 Vedic multiplier modules are implemented easily by using four 4x4 multiplier modules.

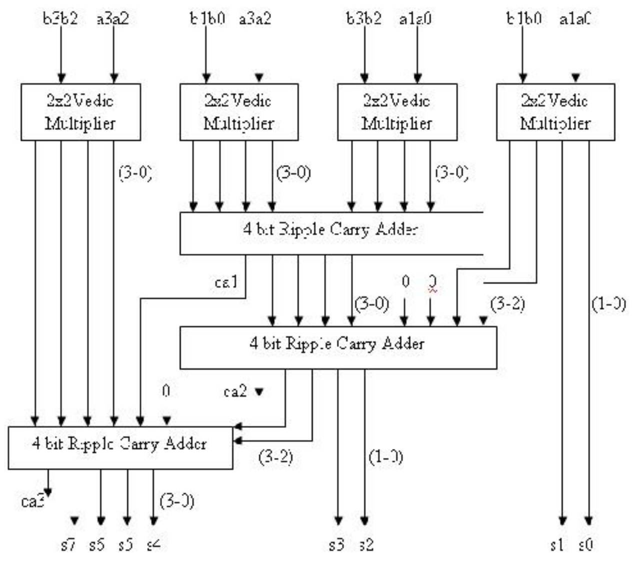

Fig: 1.2 Block Diagram of 4x4 bit Vedic Multiplier

IV. **WORKING OF PROPOSED SYSTEM:**

Project Navigator Overview

Project Navigator organizes your design files and runs processes to move the design from design entry through implementation to programming the targeted Xilinx device. Project Navigator is the high-level manager for your Xilinx FPGA and CPLD designs, which allows you to do the following:

1. Add and create design source files, which appear in the Sources window
2. Modify your source files in the Workspace
3. Run processes on your source files in the Processes window
4. View output from the processes in the Transcript window

The following Fig 4.1 shows the Project Navigator main window, which allows you to manage your design starting with design entry through device configuration

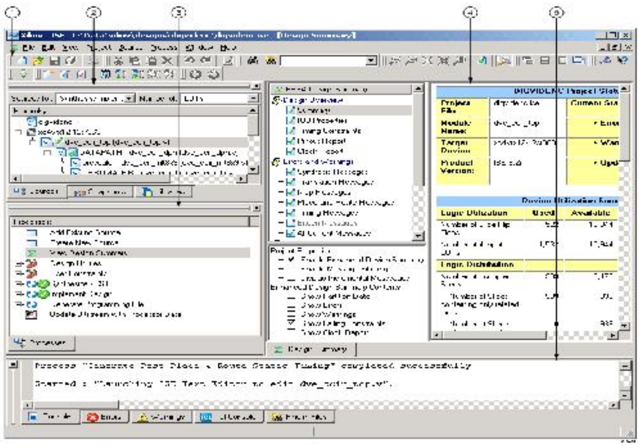

Fig: 1.3 Project Navigator window

1. Toolbar
2. Sources window
3. Processes window
4. Workspace
5. Transcript window

The first step in implementing your design for a Xilinx FPGA or CPLD is to assemble the design source files into a project. The Sources tab in the Sources window shows the source files you create and add to your project, as shown in the following fig. For information on creating projects and source files, see Creating a Project and Creating a Source File.

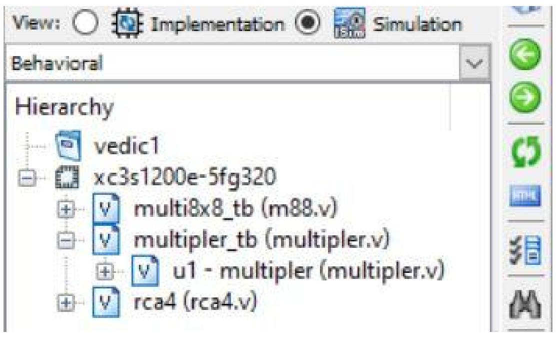

Fig: 1.4 Design view drop down list

BLOCK DIAGRAM (8X8)

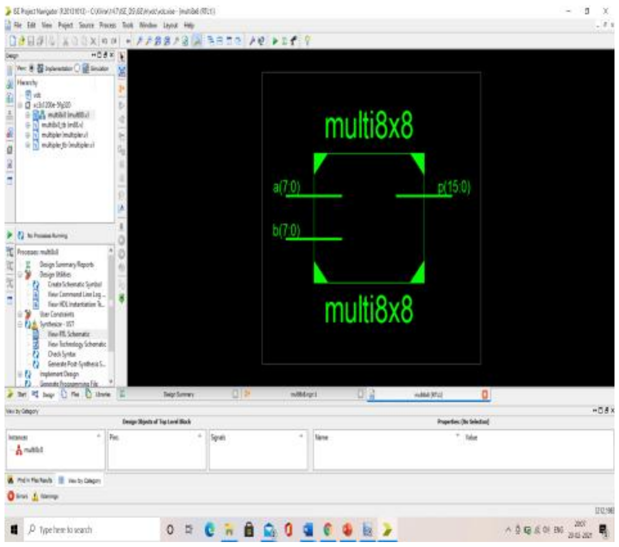

Fig: 1.5 block diagram of top level model

After the HDL synthesis phase of the synthesis process , we can display the schematic representation by selecting implementation in the source tab from the design view drop-down list. Now select the top module in the processes tab expand synthesize-XST. Double click view schematic, the source file is displayed in schematic form in the work space. It is nothing but register transfer level (RTL) viewer ,this view displays gates and elements.(The fig 5.1 show the RTL schematic diagram of 8X8 multiplier).

Technology Schematic

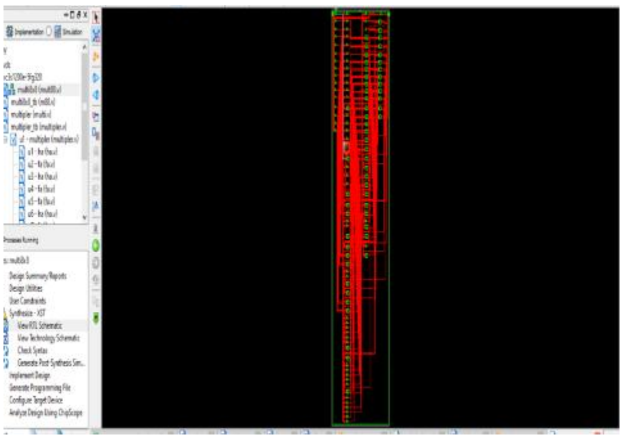

Fig: 1.6 Technological schematic of top level model

On double clicking the fig5.1 RTL schematic, the top-level model as shown in fig5.2 is obtained.

V. SYNTHESIS RESULTS

HDL Synthesis Report

Macro Statistics

# Xors	: 55

1-bit xor3 : 55

Device utilization summary:

```
Selected Device : 3s1200efg320-5
Number of Slices: 80  out of   8672     0%
Number of 4 input LUTs: 139  out of  17344     0%
Number of IOs: 32
Number of bonded IOBs: 31  out of    250    12%
```

Summary

```
Delay                    :   31.663ns (Levels of Logic = 26)
Source                   :  a<1> (PAD)
Destination              :  p<15> (PAD)
Data Path                :  a<1> to p<15>
```

```
Gate       Net
Cell: in->out fanout Delay Delay Logical Name (Net Name)
-------------------------------------------------------------------------------
IBUF:I->O      14   1.106  1.002 a_1_IBUF (a_1_IBUF)
LUT4:I0->O      2   0.612   0.532  u1/co_and00001 (c1)
LUT4:I0->O      3   0.612   0.603  u3/co_and00001 (c3)
LUT4:I0->O      2   0.612   0.449u5/co1 (c5)
LUT3:I1->O      2  0.612   0.532      u7/Mxor_s_xo<0>1 (s4)
LUT4:I0->O      30.612   0.454        u9/Mxor_s_xo<0>1 (s6)
LUT4:I3->O      3  0.612  0.454       u12/Mxor_s_xo<0>1 (s8)
LUT4:I3->O      2   0.6120.532        u15/co_and00001 (c15)
LUT4:I0->O      3   0.612 0.603       u17/Mxor_s_xo<0>1 (s12)
LUT3:I0->O      2   0.612   0.449  u20/co1 (c20)
LUT3:I1->O      3   0.612   0.481  u23/Mxor_s_xo<0>1 (s17)
LUT3:I2->O      2   0.612   0.449    u27/co1 (c27)
LUT4:I1->O      2   0.612   0.532  u31/Mxor_s_xo<0>1 (s24)
LUT3:I0->O      2   0.612   0.532  u34/Mxor_s_xo<0>1 (s27)
LUT4:I0->O      2   0.612   0.532  u35/co_and00001 (c35)
LUT4:I0->O      2   0.612   0.410  u38/co1 (c38)
LUT3:I2->O      2   0.612   0.532  u42/Mxor_s_xo<0>1 (s33)
LUT4:I0->O      2   0.612   0.449  u45/Mxor_s_xo<0>1 (s36)
LUT3:I1->O      2   0.612   0.532  u46/co1 (c46)
LUT4:I0->O      2   0.612   0.532  u48/Mxor_s_xo<0>1 (s38)
LUT3:I0->O      2   0.612   0.449  u50/co1 (c50)
LUT4:I1->O     2   0.612   0.383  u52/Mxor_s_xo<0>1 (s41)
LUT4:I3->O      1  0.612   0.387  u54/co44_SW0 (N91)
LUT3:I2->O      2   0.612   0.532  u54/co44 (c54)
LUT3:I0->O           1       0.612       0.357     u56/co_and00001 (p_15_OBUF)
OBUF:I->O             3.169       p_15_OBUF (p<15>)
----------------------------------------
Total                  31.663ns (18.963ns logic, 12.700ns route)
(59.9% logic, 40.1% route)
```

Simulation is performed to verify RTL code and to confirm that the design is functioning as intended .So behavioral simulation is performed by following the below steps

1. Compiling the Verilog simulation library.
2. In the design panel ,Select behavioral simulation .
3. In the hierarchy pane , Select the test bench file to simulate.
4. Click on the simulate behavioral model .

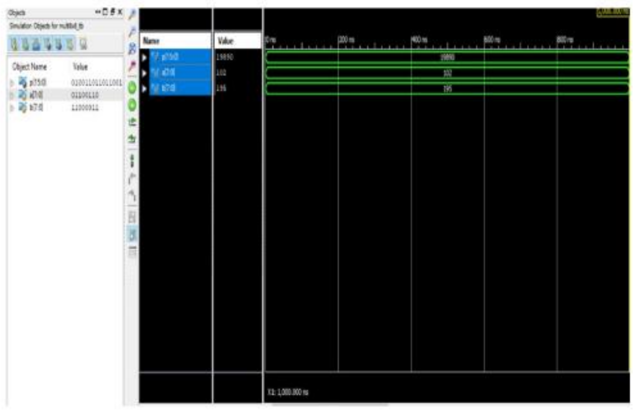

Fig: 1.7 8X8 bit multipler output waveform

a[7:0]=01100110 (the decimal representation for a is 102) and b[7:0]=11000011(the decimal representation for a is 195).
On multiplying the above two inputs the result obtained is p[15:0]=010011011011001(the decimal representation for a is 19890).

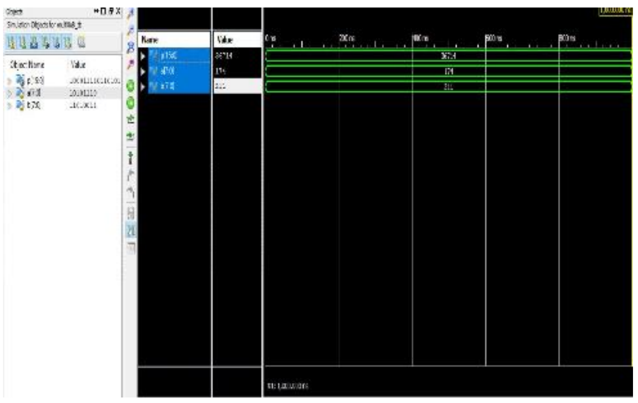

Fig: 1.8 8X8 bit multipler output waveform

a[7:0]=10101110 (the decimal representation for a is 174) and b[7:0]=11010011(the decimal representation for a is 211).
On multiplying the above two inputs the result obtained isp[15:0]=100011110110101 (the decimal representation for a is 36714).

BLOCK DIAGRAM(4X4)

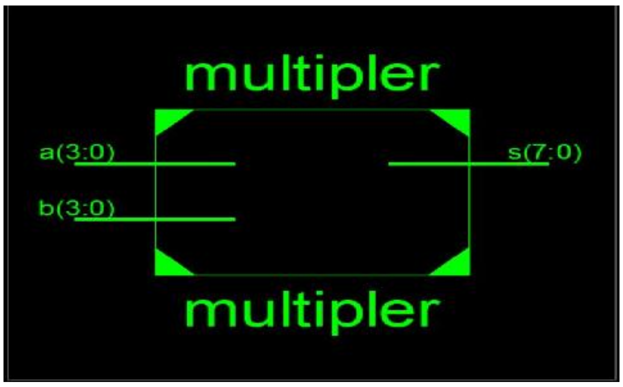

Fig: 1.9 block diagram of top level model

After the HDL synthesis phase of the synthesis process , we can display the schematic representation by selecting implementation in the source tab from the design view drop-down list. Now select the top module in the processes tab expand synthesize-XST. Double click view schematic, the source file is displayed in schematic form in the workspace. It is nothing but register transfer level (RTL) viewer ,this view displays gates and elements.(The fig 5.5 show the RTL schematic diagram of 4X4 multiplier).

Technology Schematic(4X4)

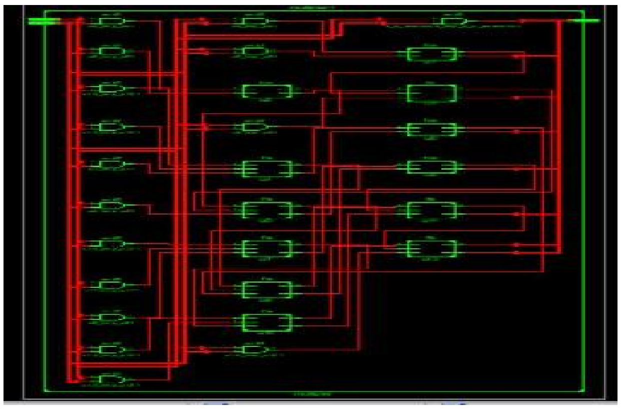

Fig: 1.10 Technological schematic of top level mo

```
HDL Synthesis Report
Macro Statistics
# Xors                          : 12
1-bit xor2                      : 4
1-bit xor3                      : 8
```

Device utilization summary:

```
Selected Device : 3s1200efg320-5
Number of Slices               :18  out of   8672     0%
Number of 4 input LUTs:31  out of  17344     0%
Number of IOs          :16
Number of bonded IOBs          :16  out of    250     6%
```

```
Delay                          :12.673ns (Levels of
Logic = 9)
Source:a<1> (PAD)
Destination  :p<7> (PAD)
Data Path: a<1> to p<7>
```

Gate Net
Cell:in->out fanout Delay Delay Logical Name (Net Name)

```
---------------------------------------------------------------------------
-------
IBUF:I->O        6   1.106   0.721  a_1_IBUF (a_1_IBUF)
LUT4:I0->O       2   0.612   0.532  u1/co1 (c1)
LUT4:I0->O       3   0.612   0.603  u3/co1 (c3)
LUT4:I0->O       3   0.612   0.603  u5/co1 (c5)
LUT3:I0->O       2  0.612   0.532  u7/co1 (c7)
LUT3:I0->O       2   0.612   0.383  u11/Mxor_s_xo<0>1 (s7)
LUT4:I3->O       2   0.612   0.383  u12/co1 (c12)
LUT4:I3->O       1   0.612 0.357  u13/co1 (p_7_OBUF)
OBUF: I->O              3.169       p_7_OBUF (p<7>)
------------------------------------------------------------------------
Total                12.673ns (8.559ns logic, 4.114ns route)
(67.5% logic, 32.5% route)
```

Simulation is performed to verify RTL code and to confirm that the design is functioning as intended .So behavioral simulation is performed by following the below steps

1. Compiling the Verilog simulation library.
2. In the design panel ,Select behavioral simulation .
3. In the hierarchy pane , Select the test bench file to simulate.
4. Click on the simulate behavioral model .

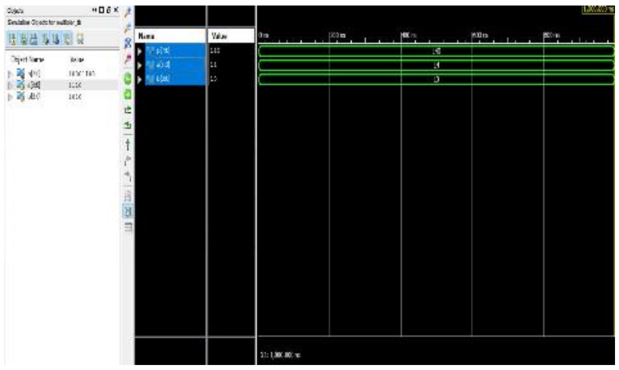

Fig: 1.11 4X4 bit multipler output waveform

a[3:0]= 1110(the decimal representation for a is 14) and b[3:0]= 1010(the decimal representation for a is 10).

On multiplying the above two inputs the result obtained is p[7:0]=10001100(the decimal representation for p is 140).

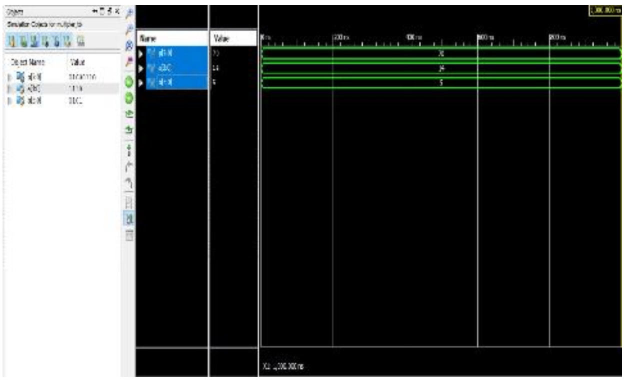
Fig: 1.12 4X4 bit multipler output waveform

a[3:0]=1110 (the decimal representation for a is 14) and
b[3:0]=0101(the decimal representation for a is 5).
On multiplying the above two inputs the result obtained is p[7:0]=01000110(the decimal representation for p is 70).

VI. CONCLUSION

This paper presents a highly efficient method of multiplication – "Urdhva Tiryakbhyam Sutra" based on Vedic mathematics. It gives us method for hierarchical multiplier design and clearly indicates the computational advantages offered by Vedic methods. Hence our motivation to reduce delay is finely fulfilled. Therefore, we observed that the Vedic multiplier is much more efficient than Array and Booth multiplier in terms of execution time (speed). An awareness of Vedic mathematics can be effectively increased if it is included in engineering education. In future, all the major universities may set up appropriate research centers to promote research works in Vedic mathematics. Vedic multiplier has regular and coherent in its structure so that it can be easily laid on silicon and also it takes less power compared to normal booth multiplier. Future work lies in the direction of introducing pipeline stages in the multiplier architecture for maximizing throughput and also, we can implement for 64 – bit also.

REFERENCES

[1] Jagadguru Swami, Sri Bharati Krishna, Tirthaji Maharaja, "Vedic Mathematics or Sixteen Simple Mathematical Formulae from the Veda, Delhi (1965)", Motilal Banarsidas, Varanasi, India, 1986.

[2] M. Morris Mano, "Computer System Architecture", 3rd edition, Prientice-Hall, New Jersey, USA, 1993, pp. 346-348.

[3] H. Thapliyal and H.R Arbania. "A Time-Area-Power Efficient Multiplier and Square Architecture Based on Ancient Indian Vedic Mathematics", Proceedings of the 2004 International Conference on VLSI (VLSI'04), Las Vegas, Nevada, June 2004, pp. 434-439.

[4] P. D. Chidgupkar and M. T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", Global J. of Engg. Edu, Vol.8, No.2, 2004, UICEE Published in Australia.

[5] Thapliyal H. and Srinivas M.B, "High Speed Efficient NxN Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics", Transactions on Engineering, Computing and Technology, 2004, Vol.2.

[6] Harpreet Singh Dhillon and Abhijit Mitra, "A Reduced–Bit Multiplication Algorithm for Digital Arithmetics", International Journal of Computational and Mathematical Sciences 2.2 @ www.waset.orgSpring2008.

[7] Honey Durga Tiwari, Ganzorig Gankhuyag, Chan Mo Kim and Yong Beom Cho, "Multiplier design based on ancient Indian Vedic Mathematician", International SoC Design Conference, pp. 65- 68, 2008.

[8] Parth Mehta and Dhanashri Gawali, "Conventional versus Vedic mathematics method for Hardware implementation of a multiplier", International conference on Advances in Computing, Control, and Telecommunication Technologies, pp. 640-642, 2009.

[9] Ramalatha, M.Dayalan, K D Dharani, P Priya, and S Deoborah, "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques", International Conference on Advances in Computational Tools for Engineering Applications (ACTEA) IEEE, pp. 600-603, July 15-17, 2009.

[10] Sumita Vaidya and Deepak Dandekar, "Delay-Power Performance comparison of Multipliers in VLSI Circuit Design", International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, pp. 47-56, July 2020.

[11] S.S.Kerur, Prakash Narchi, Jayashree C N, Harish MKittur and Girish V A "Implementation of Vedic Multiplier For Digital Signal Processing" International conference on VLSI communication & instrumentation (ICVCI), 2011.

[12] Asmita Haveliya, "A Novel Design for High-Speed Multiplier for Digital Signal Processing Applications (Ancient Indian Vedic mathematics approach)", International Journal of Technology and Engineering System (IJTES), Vol.2, No.1, pp. 27-31, Jan-March, 2011.

[13] Prabha S., Kasliwal, B.P. Patil and D.K. Gautam, "Performance Evaluation of Squaring Operation by Vedic

Mathematics", IETE Journal of Research, vol.57, Issue 1, Jan-Feb 2018.

[14] Aniruddha Kanhe, Shishir Kumar Das and Ankit Kumar Singh, "Design and Implementation of Low Power Multiplier Using Vedic Multiplication Technique", (IJCSC) International Journal of Computer Science and Communication Vol. 3, No. 1, pp. 131-132, January-June 2012.

[15] Umesh Akare, T.V. More and R.S. Lonkar, "Performance Evaluation and Synthesis of Vedic Multiplier", National Conference on Innovative Paradigms in Engineering & Technology (NCIPET-2012), Proceedings published by International Journal of Computer Applications (IJCA), pp. 20-23, 2012.