VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DESIGN AND IMPLEMENTATION OF HIGH SPEED MULTIPLIER USING VEDIC MATHEMATICS

Murugesan G. and Lavanya S.
Department of Computer Science and Engineering, St.Joseph's College of Engineering, Chennai, Tamil Nadu, India
E-Mail: murugesh02@gmail.com

ABSTRACT

High speed computing systems have been very much demand in recent years, because of the fast growing technologies in scientific computing applications. Designing a high speed multiplier will have a large impact on applications like Image Processing, Convolution, Fast Fourier Transform, and Filtering and in microprocessors. For this, we aggregate into the multiplication process, a sutra named Urdhva-Triyagbhyam from the Ancient Indian Vedic Mathematics since it has a unique way of calculations. Also, building an ALU using Vedic Multiplier is less complex when compared to other multipliers. In this paper we have proposed an algorithm for multiplying 16 bit value as Vedic Multiplier. While implementing this algorithm we studied that the speed of the computation process is increased and the computing time is reduced due to decrease of path delay compared to the existing multipliers. The design of the Vedic Multiplier is performed in Verilog language and the tool used for simulation is Xilinx 9.1 ISE, Spartan-3E

Keywords: Vedic Multiplier, Urdhvatirvagbhyham, Nikhilam Navatashcaramam, Fast Fourier Transformation, ALU design, Vedic Mathematics.

INTRODUCTION

Digital Multipliers are the very significant part in ALU and are important in performing tasks such as convolutions and Fast Fourier Transforms.These are the main components of all the digital signal processors (DSPs) and the speed of the DSP is largely found by the speed of its multipliers (Babulu, 2011). In all the digital circuit design the multiplier is the primary unit. They are fast and reliable components that are utilized for implementing any operation. Depending upon the components' arrangement, there are various types of multipliers are available and the type of multiplier architecture is selected based upon the application requirement.

In most of the DSP algorithms, the performance of the algorithm is based on the path delay of the multiplier. The speed of multiplication is very important in DSP as well as in general processors. In the early period, multiplications were implemented generally with a sequence of shift and add operations. There have been many algorithms proposed in literature to perform multiplication, each providing various advantages and having tradeoff in terms of speed, circuit complexity and area.Also, multiplication dominates the execution time of most DSP applications and hence there is a need of high speed multiplier for designing an efficient ALU (Himanshu 2004). For this, an ancient system of calculation which was rediscovered from Vedas by Sri Bharati Krushna Tirthaji Maharaj known as "Vedic Mathematics" is used. The peculiarity of Vedic Mathematics is because of its simplicity and flexibility in carrying out the calculations mentally (Jayaprakash*et al.* 2014). This gives us the liberty to choose the technique most suitable for us. According to Tirthaji, all of Vedic mathematics is based on sixteen Sutras , which are actually word formulae describing natural ways of solving a whole range of complex mathematical problems such as Algebra, trigonometry, factorizations, partial fractions, coordinate geometry, and higher order equations.

Two main Sutras for multiplication are

a) Urdhva- Tiryagbhyham - vertically and crosswise

b) Nikhilam Navatashcaramam Dashatah - All from 9 and the last from 10

VEDIC MULTIPLICATION

Nikhilam Navatashcaramam Dashatah

Although Nikhilam Navatashcaramam Dashatah sutra can be applied to all cases of multiplication, it is more suitable when the numbers involved in multiplication are large and this formula can be very effectively applied in multiplication of numbers, which are nearer to bases like 10, 100, 1000.i.e. to the powers of 10 (Sunitha*et al.*, 2013). The power of 10 from which the difference is calculated is called the Base. These numbers are considered to be references to find out whether given number is less or more than the Base. These numbers are considered to be references to find out whether given number is less or more than the Base. This sutra can be explained for (96x93) as in Figure-1 and when this is compared with the conventional multiplication method, the result is obtained easily and quickly. Since it identifies the complement of the large number from its nearest base to do the multiplication on it, larger the original number, lesser the complexity of the multiplication (Deepali*et al.*, 2013). The algorithm of this sutra is explained as follows.

Algorithm

Step 1. The base to be chosen is 100 as it is nearest to and greater than both these two numbers. i.e.,(100-96 = 4 and 100-93 = 7).

Step 2. The right hand side (RHS) of the result is found by finding the product of numbers of Column2. i.e., 7*4 = 28).
The left hand side (LHS) of the product can be found by cross subtracting the second number of Column 2 from the first number of Column 1 or vice versa.

i.e., (93 -4) = 89 or (96-7) = 89.

Step 3. The final result is obtained by concatenating RHS and LHS which is 8928.

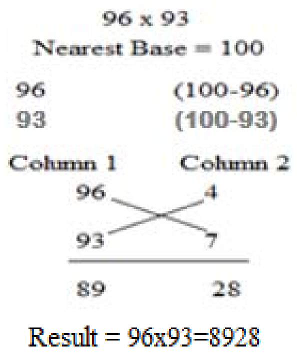


Figure-1. Multiplication using Nikhilam Sutra.

Urdhva-Tiryagbyham Sutra

Urdhva-Tiryagbyhamis a general multiplication formula among 16 sutras which is applicable to all cases of multiplication. In our proposed work, Urdhva-Tiryakbhyam Sutra is applied to the binary number system and is used to develop digital multiplier architecture since it is extremely simple and powerful (Rajesh 2013). Unlike Nikhilam sutra, this method can be applied for any binary number (Shivangi 2013). It relies on a concept that the generation of all partial products can be performed with the concurrent addition of these partial products whereas in shift and add method, four partial products have to be added to get the product. Thus, by using Urdhva-Tiryagbhyam Sutra in binary multiplication, the number of steps required for calculating the final product is reduced (Dayanand*et al.* 2011). Following steps give the brief description and diagrammatic illustration of the working principles of Urdhva-Tiryagbyham sutra.

Step by step procedure for Tiryagbyham sutra for the multiplication operation of 24 by 32.

Step 1. Multiply vertically on the right to find the units digit as in Figure-2.

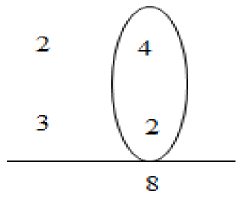


Figure-2. Unit's place Multiplication.

Step 2. Multiply vertically on the right to find the units digit as in Figure-2.

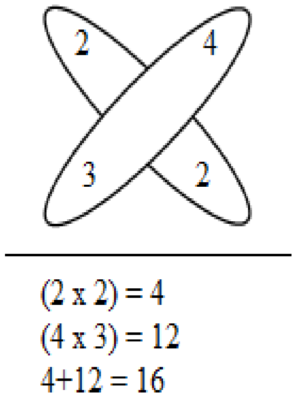
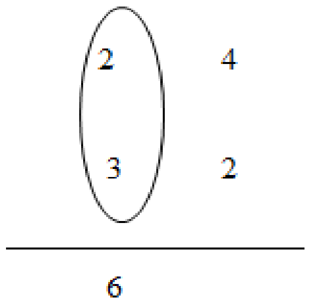


Figure-3. Crosswise Multiplication.

Step 3. Multiply vertically on the right to find the units digit as in Figure-2.



6+1(carry) = 7

Figure-4. Ten's place Multiplication.

Therefore, final product is 24 × 32 = 768.

For 4x4 bit binary multiplication using Vedic method, the algorithm is explained as follows and is illustrated with 4x4 bit binary number as in Figure-5 and the final result is obtained correctly (Jayaprakash*et al.* 2012).The same procedure can be extended for higher order bits and the steps followed are simple compared to other conventional multiplication methods. The 4 bit numbers for which Vedic method is applied are 1101 and 1010 and the result is obtained easily (Sriraman 2012).

Algorithm

Step 1. Initially, the lowest bit are multiplied which results in the least significant bit of the product (vertical).

Step 2. The lowest bit of the multiplicand is now multiplied with the next higher bit of the multiplier and is added with the product of LSB of multiplier and next higher bit of the multiplicand (crosswise).

Step 3. The sum obtained gives the second bit of the product and the carry is added with the output of next stage sum obtained by the crosswise and vertical multiplication and addition of three bits of the two numbers from least significant position.

Step 4. All the four bits are processed with crosswise multiplication and addition to produce the sum and carry. The sum obtained is the corresponding bit of the product and the carry is again added to the next stage multiplication and addition of three bits except the LSB.

Step 5. The same procedure (Step 4.) is continued until the multiplication of the two MSBs gives the MSB of the product.

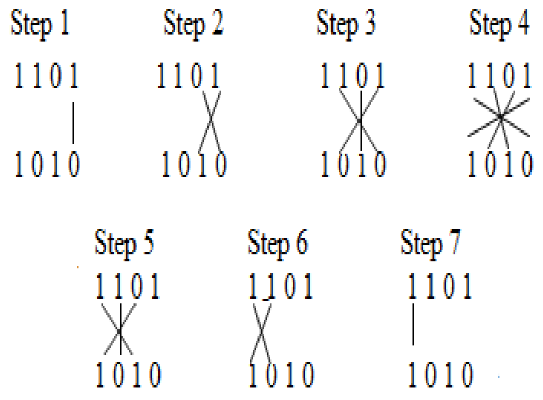


Figure-5. General rule for 4 bit by 4 bit Multiplication.

The intermediate results obtained at each step of Figure-5 can be represented as S [7:0] and c [7:0] (carry) as follows.

S_0: (1x0) = 0; c_0 = 0;
S_1: (0x0) + (1x1) = 1; c_1 = 0;
S_2: (1x0) + (0x1) + (1x0) = 0; c_2 = 0;
S_3: (1x0) + (1x1) + (0x0) + (1x1) =0; c_3 = 1;
S_4: (1x1) + (1x0) + (0x1) + 1 = 0; c_4 = 1;
S_5: (1x1) + (1x0) + 1 = 0; c_5 = 1;
S_6: (1x1) + 1 = 0; c_6 = 1;

The final product is $c_6s_6s_5s_4s_3s_2s_1s_0$. i.e.,10000010. The same approach is followed for higher order bit multiplication and this method is found to be faster when compared to the conventional shift and add multiplication (Baljinder 2014).

PROPOSED VEDIC MUTLIPLIER

2x2 Bit Vedic Multiplier

Consider two 2 bit binary numbers which are represented as below to produce a 4 bit product.

a[1:0]: a_1a_0
b[1:0]: b_1b_0

By applying Urdhva-Tiryagbhyam method to a and b, the product of a_0b_0, a_1b_0, a_0b_1 and a_1b_1 is found. Now a_0b_0 appears as the LSB (s_0) of final result. a_0b_1 and a_1b_0 uses half adder and the result is s_1 as in Figure-6.The obtained carry (c_1) and a_1b_1 is sent through another half adder (Alex et al 2004). The resulting bit is s_2 and its carry (c_2) is the left-most bit of the final result.

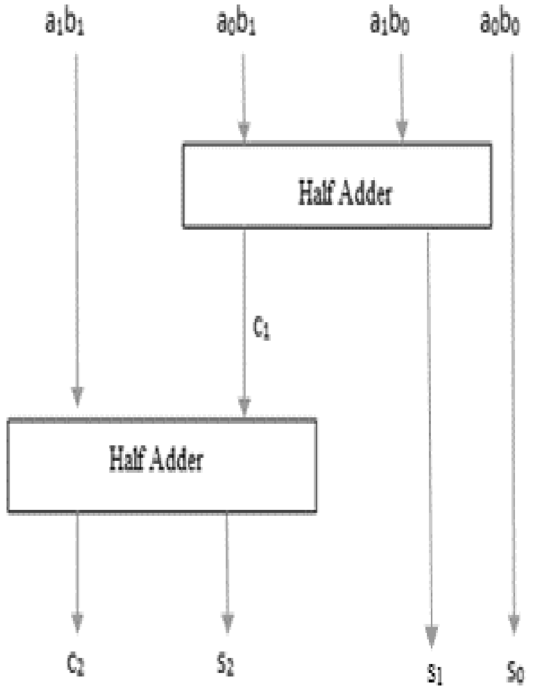


Figure-6. Block diagram of 2x2 bit Vedic Multiplier.

VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

The resulting product is of 4 bit binary number represented as $c_2s_2s_1s_0$.

4x4 Bit Vedic Multiplier

The next higher level of 2x2 multiplier is the 4x4 Vedic Multiplier. Using four 2x2 multiplier blocks, and with three adders (one 4 bit adder and two 6 bit adder), 4x4 bit multiplier is built as shown in Figure-7. Here a and b are 4 bit binary numbers that is, n =4 (bit size of the multiplicands).

a[3:0]: $a_3a_2a_1a_0$
b[3:0]: $b_3b_2b_1b_0$

The inputs are broken into tiny chunks of n/2 = 2, for both inputs, that is a and b. These newly generated chunks of 2 bits, that is a_1a_0 and b_1b_0, a_3a_2 and b_1b_0, $a_1a_0$and b_3b_2, a_3a_2 and b_3b_2 are given as input to 2x2 multiplier blocks and the result produced 4 bits, which are the output produced from 2x2 multiplier block are sent into the adder. The two lower bits of q_0 pass directly to output, while the upper bits of q_0 are fed into addition tree. The bits being fed to addition tree and finally the result are found. The resulting product is of 8 bit as $q_7q_6q_5q_4q_3q_2q_1q_0$ and the delay produced in nibble multiplier is much less than in other multipliers.

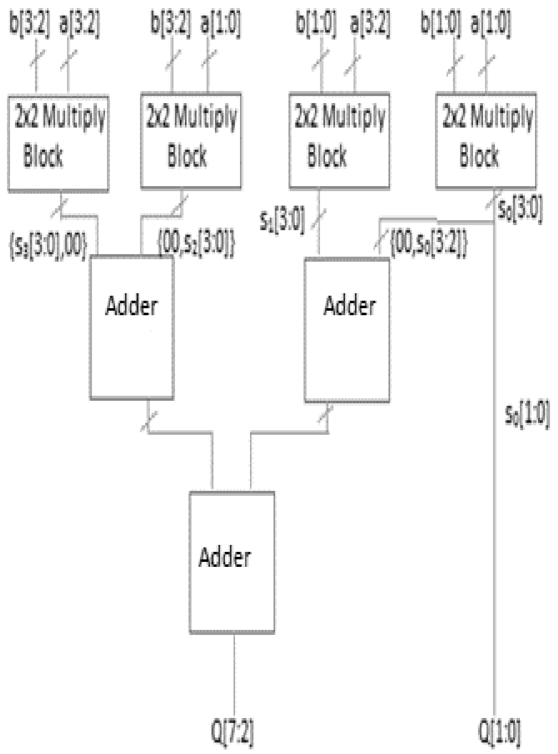


Figure-7. Block diagram of 4x4 Vedic Multiplier.

8x8 Vedic Multiplier

Similar to the previous design, four such 4x4 multipliers is used and is also necessary to design an 8 bit adder and two 12 bit adders to obtain a complete 8x8 Vedic Multiplier. Here a and b are 8 bit binary numbers.

a[7:0]: $a_7a_6a_5a_4a_3a_2a_1a_0$
b[7:0]: $b_7b_6b_5b_4b_3b_2b_1b_0$

Here, the bit size of the multiplicand is 8 whereas the resulting output is of 16 bit size. The input is divided into smaller chunks size of n/2 = 4, for both inputs, that is a[7:0] and b[7:0], as in case of 4x4 multiply block. These newly produced chunks of 4 bits are given as input to 4x4 multiplier block, where again these new chunks are broken into still tiny chunks of size n/4 = 2 and fed to 2x2 multiply block. The result produced, from output of 4x4 bit multiply block which is of 8 bits, and are sent to an adder for performing addition. In 8x8 Multiplier, lower 4 bits of q_0 are passed directly to output and the remaining bits are fed for addition operation as in Figure-8. The resulting product is of 8 bit as Q [15:0] and the efficiency of this 8 bit Vedic Multiplier architecture is far better than the existing multipliers such as Booth and Array multiplier.

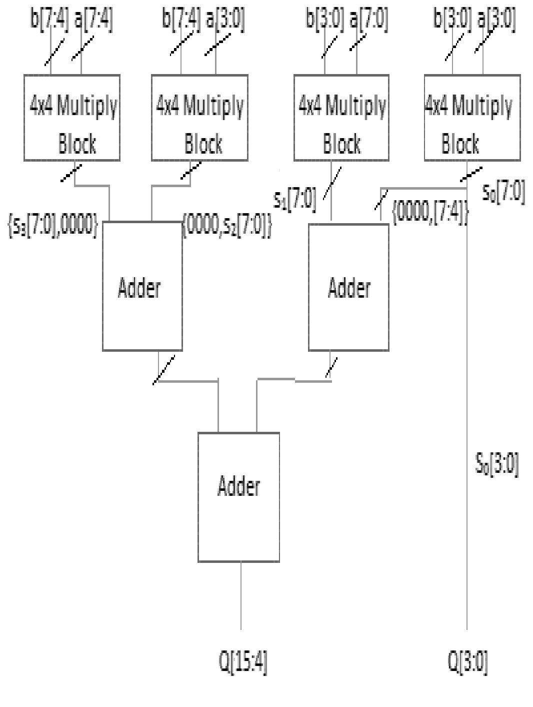


Figure-8. Block diagram of 8x8 Vedic Multiplier.

16x16 Vedic Multiplier

With four 8x8 multipliers as in Figure-7, one 12 bit adder and two 24 bit adders, a 16x16 Vedic Multiplier is designed. Here a and b are 16 bit binary numbers.

a[15:0]: $a_{15}a_{14}a_{13}a_{12}a_{11}a_{10}a_9a_8a_7a_6a_5a_4a_3a_2a_1a_0$
b[15:0]: $b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0$

The Multiplier for 16x16 is built with four 8x8 multiplier blocks. Here, the bit size of the multiplicands is n=16 whereas the resulting output is of 32 bit size. The input i.e., a[15:0] and b[15:0] is divided into tiny chunks with size of 8 (n/2=8), for both binary inputs, that is a and b. These newly produced chunks of 8 bits are brought as input to the 8x8 multiplier, where again these new chunks are splitted into still tiny chunks of size 4 and are sent to 4x4 multiply block, just as in case of 8x8 multiply block.

Again, the new chunks are splitted into half, to get chunks of size 2, which are sent to 2x2 multiply block. The result produced, from output of 8x8 bit multiply block which is of 16 bit, and are sent for addition to an adder. Here, as shown in Figure-9, the lower 8 bits of q_0 directly pass on to the result, while the higher bits are fed for addition into the addition tree. The resulting product is found to be of size 32 bit as Q [31:0] (Jagadeshwar 2012).

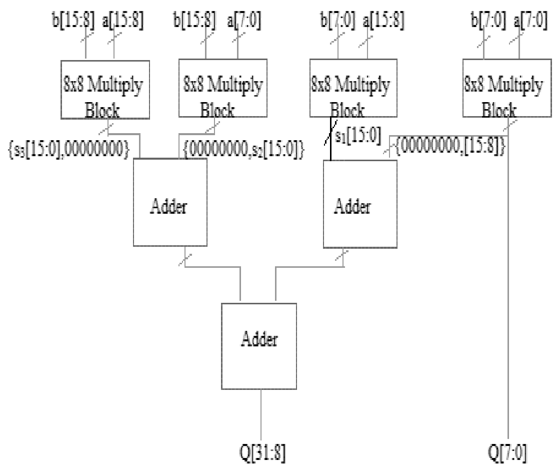


Figure-9. Block diagram of 16x16 Vedic multiplier

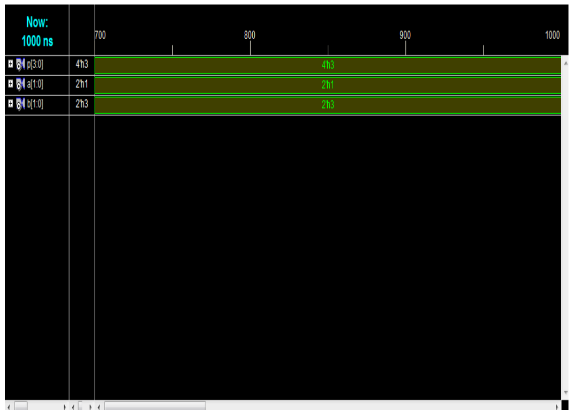
Simulation results

The simulation is modelled in ISE simulator and coding is done using Verilog language in Xilinx ISE 9.1 Spartan-3E.The simulation results in terms of path delay which is the delay between a source (input) pin and destination (output) pin on a module for booth, array and Vedic multiplier (Hasan 2008) is presented in Table-1.

For 2x2 multiplier, the inputs is as below.

$a[1:0]=01_2$
$b[1:0]=11_2$

The output is Q [3:0] = 0011_2.In the simulation, the last two lines represents a[1:0] and b[1:0] in hexadecimal format as 1_h and 3_h respectively which produces Q [3:0] as 3_h.

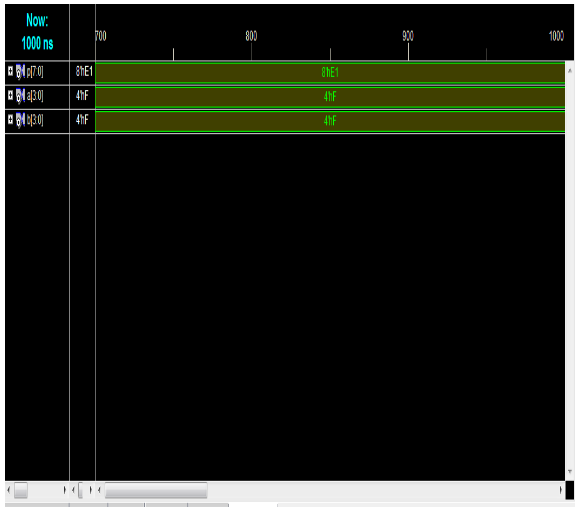


p[3:0] 4'h3 4'h3
a[1:0] 2'h1 2'h1
b[1:0] 2'h3 2'h3

In 4x4 multiplier, the inputs are as follows.

$a[3:0] = 1111_2$
$b[3:0] = 1111_2$

The output is Q[7:0] = 11100001_2.In this simulation, the last two lines represents a[3:0] and b[3:0] in hexadecimal format as F_h and F_h respectively which produces Q[7:0] as $E1_h$.

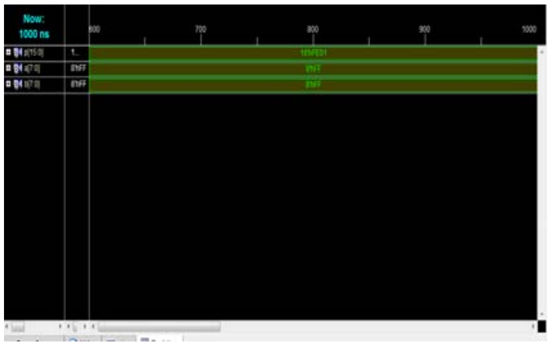


p[7:0] 8'hE1 8'hE1
a[3:0] 4'hF 4'hF
b[3:0] 4'hF 4'hF

The inputs of 8x8 multiplier are

$a[7:0] = 11111111_2$
$b[7:0] = 11111111_2$

The output is Q [15:0] = 1111111000000001_2.In this simulation, the last two lines represents a [7:0] and b [7:0] in hexadecimal format as FF_h and FF_h respectively which produces Q [15:0] as $FE01_h$.

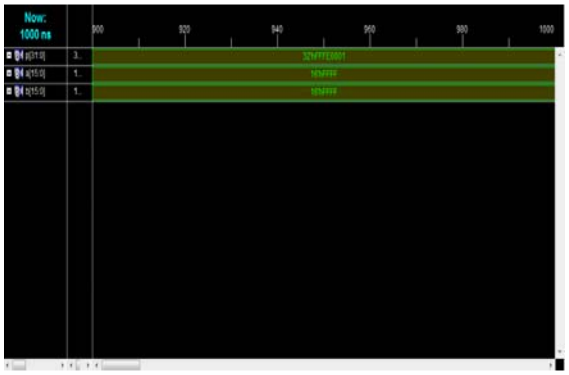


The inputs of 16x16 multiplier are as follows.

a[15:0]=1111111111111111_2
b[15:0]=1111111111111111_2

Q[31:0]=$11111111111111100000000000000001_2$ is found as the output.

In this simulation, the last two lines represents a[15:0] and b[15:0] in hexadecimal format as $FFFF_h$ and $FFFF_h$ respectively which produces Q[15:0] as $FFFE0001_h$.



The path delay of the proposed 16x16 bit Vedic multiplier has an increase in speed of 52.5% when compared with Booth and Array Multipliers. Also the logic levels used are less than the other multipliers. This proves that application of Vedic mathematics in multiplier can drastically decrease the delay thereby increasing the speed of the multiplier and making it suitable for its use in many areas.

Table-1. Simulation Results of 16x16 Vedic Multiplier.

Attribute	Booth multiplier	Array multiplier	Proposed Vedic multiplier
Path delay	46.740 ns	45.917 ns	22.201 ns
No. of logiclevels	73	59	41

From the analysis and comparison of proposed Vedic Multiplier with array and booth multipliers, the maximum combinational path delay is 22.201 ns whereas in booth multiplier (Ganesh *et al*., 2012)it is 46.740 ns and for array multiplier (Vamsi 2011) it is 45.917 ns. Also in the proposed method the number of logic levels has been decreased to 41 whereas in booth and array multipliers, the logic levels are 73 and 59 respectively.

CONCLUSIONS

A 16 x16 high speed multiplier is constructed, which is very efficient. The multiplier architecture is based on Urdhva-Tiryakbhyam Sutra of Vedic Mathematics and accumulation is done using adder, which gives better performance when compared with other multipliers such as Booth, Array and Wallace Tree multipliers. With this proposed design, it is found that our design works with much less delay of 22.201 ns.

As a future enhancement process, compressors or adders such as carry save adder can be aggregated to the proposed Vedic Multiplier so that the delay can be still more reduced thereby increasing the speed. This also results in reduction of logic levels to a large extent.

REFERENCES

Alex Panato, Sandro Silva, Flávio Wagner, Marcelo Johann, Ricardo Reis and Sergio Bampi. 2004. Design of Very Deep Pipelined Multipliers for FPGAs. Proceedings of Design, Automation and Test in Europe Conference and Exhibition, Brazil.3: 28-33.

Babulu K. and Parasuram G. 2011.FPGA Realization of Radix-4 Booth Multiplication Algorithm for High Speed Arithmetic Logics.International Journal of Computer Science and Information Technologies. 2(5): 2102-2107.

BaljinderKaur and Vipasha Thakur. 2014. Review of Booth Algorithm for Design of Multiplier.International Journal of Emerging Technology and Advanced Engineering.4(4): 134-137.

Dayanand A.K., Sudheer B.D., Sateesh C.M., NageswaraRaoe D.V., Avinash V. and Vamsikrishna F.P. 2011. High Performance Pipelined Signed 64x64 - Bit Multiplier using Radix-32 Modified Booth Algorithm and Wallace Structure.International Journal of Advanced Computer Technology. 3(2): 22-26.

DeepaliChandel, GaganKumawat, PranayLahoty, VidhiVartChandrodaya and Shailendra Sharma. 2013. Booth Multiplier: Ease of Multiplication.International Journal of Emerging Technology and Advanced Engineering. 3(3): 326-330.

Ganesh K.V., Sudha Rani T., VenkateswaraRao P.N. and Venkatesh. K. 2012. Constructing a low power multiplier using Modified Booth Encoding Algorithm in redundant binary number system.International Journal of Engineering and Research Applications. 2(3): 2734-2740.

HasanKrad and AwsYousif Al-Taie. 2008.Performance Analysis of a 32-Bit Multiplier with a Carry-Look-Ahead Adder and a 32-bit Multiplier with a Ripple Adder using VHDL.International Journal of Recent Trends in Engineering. 2(6): 83-86.

HimanshuThapliyal and Srinivas M.B. 2004.High Speed Efficient N x N Bit Parallel Hierarchical Overlay Multiplier Architecture Based On Ancient Indian Vedic Mathematics. Transactions on Engineering, Computing and Technology. 3(4): 225-228.

JagadeshwarRao M. and Sanjay Dubey.2012.A High Speed Wallace Tree Multiplier Using Modified Booth Algorithm for Fast Arithmetic Circuits. Journal of Electronics and Communication Engineering. 3(2): 07-11.

Jayaprakash M., Peer Mohamed M. and Dr.Shanmugam A. 2014.Design and Analysis of Low Power and Area Efficient Multiplier.International Journal of Electrical, Electronics, Mechanical Controls. 3(1): 2011-2017.

Jeevitha.M, Muthaiah.R and Swaminathan.P. 2012.Review Article: Efficient Multiplier Architecture in VLSI Design. Journal of Theoretical and Applied Information Technology. 38(2): 196-201.

Rajesh Pidugu and P. Mahesh Kannan. 2013. Design of 64 Bit Low Power ALU for DSP Applications.International Journal of Advanced Research in Electrical, Electronics and Insrumentation Engineering. 2(4): 1526-1531.

ShivangiTiwari and NitinMeena. 2013. Efficient Hardware Design for Implementation of Matrix Multiplication by using PPI-SO.International Journal of Innovative Research in Computer and Communication Engineering.1(4): 1020-1024.

Sriraman L. and Prabakar T.N. 2012.Fpga Implementation ofHigh Performance Multiplier using Squarer.International Journal of Advanced Computer Engineering and Architectures.2(2): 121-128.

Sunitha M. S., Bharat G. Hegde and Deepakakumar N Hegde. 2013. Design and Comparison of RISC Processors Using Different ALU Architectures. International Journal of Innovative Research in Science, Engineering and Technology.2(7): 2951-2960.

Vamsi Krishna Rongali and Srinivas B. 2011.Design of Area Efficient High Speed Parallel Multiplier Using Low Power Technique on 0.18um technology.International Journal of Advanced Research in Computer Engineering and Technology.2(7): 2357-2362.